Rover telecontrolado mediante protocolos de internet (WEB/HTTP)

Antonio Pezuela García
Grado tecnologías de telecomunicación
Sistemas encastados

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

Junio 2018

FICHA DEL TRABAJO FINAL

Título del trabajo:	Rover telecontrolado mediante protocolos de internet (WEB/HTTP)
Nombre del autor:	*Antonio Pezuela García*
Nombre del consultor/a:	Jordi Bécares Ferrés
Nombre del PRA:	*Pere Tuset Peiró*
Fecha de entrega (mm/aaaa):	06/2018
Titulación::	*Grado tecnología de telecomunicación*
Área del Trabajo Final:	*Sistemas encastados*
Idioma del trabajo:	*Castellano*
Palabras clave	*Microcontroladores*

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

El objecto de este proyecto ha consistido en el desarrollo de un sistema de control para un vehículo tele-controlando o *rover.* El vehículo cuenta con un sistema de comunicaciones Wifi y mediante una panel de control web embebido es capaz de recibir instrucciones para controlar sus movimientos.

El sistema desarrollado cuenta con la capacidad de de evitar obstáculos (frontales) y además cuenta con una serie de sensores para captar información del entorno. El sistema ademas cuenta con una aplicacion cliente para permitir el control del sistema desde un teléfono móvil inteligente y otra para analizar la información de los sensores.

Como base para el desarrollo del trabajo se ha utilizado la plataforma Simplelink MSP432P401R de Texas Instruments, junto con el sistema operativo FreeRTOS y las herramientas proporcionadas por el SDK del fabricante.

El trabajo se ha orientado como una prueba de concepto para evaluar la posibilidad de producir una plataforma universal móvil personalizable, donde el usuario pueda elegir entre varios tipos de módulos de comunicaciones y de sensores.

Abstract (in English, 250 words or less):

The goal of this project is to develop a system to control a rover. This vehicle has a wireless link to access to the user network and a embebed control panel based on HTTP protocol.

The system can avoid obstacle when is moving forward and it has positional and some environmental sensors. In adition the system counts with an application

for smartphones that allows to control the rover and a web application for analyzing sensors' telemetry.

The base platform for developing the system has been used Simplelink MSP432P401R Launchpad, FreeRTOS operating system and SDK's which was provided by the manufacturer.

The work was focused on developing a proof of concept to demonstrate the posibility of building an universal mobile platform for customers, where the customer is able to choose between different communication modules and sensors.

Índice

Índice de ilustraciones

1 Introducción

1.1 Contexto y justificación del trabajo

Uno de los posibles campos de aplicación de los sistemas empotrados es el desarrollo de drones y vehículos tele-controlados. En la actualidad los drones son el centro de atención para el gran público dentro de este tipo de artefactos, pero los vehículos terrestres tele-controlados o *rover* pueden ser de utilidad en muchos campos, como por ejemplo: la revisión del interior canalizaciones (gasoductos, oleoductos, alcantarillado, etc.), la desactivación de minas terrestres (en zonas de conflicto), la exploración de zonas con ambientes peligrosos para las personas (incendios, accidentes químicos, …), toma de muestras y monitorización del entorno, ...

Otro de los campos de aplicación de los sistemas encastrados son las redes de sensores y el denominado internet de las cosas (IoT), en parte gracias al importante desarrollo de los sistemas de telecomunicaciones durante los últimos años. En determinados entornos desplegar redes de sensores fijos puede ser costoso tanto en su montaje como en su operación y mantenimiento. En dichos entornos puede ser más útil contar con un vehículos de exploración que cuenten con los sensores necesarios .

1.2 Descripción del trabajo

A lo largo de este proyecto se ha desarrollado un sistema de control web para un vehículo tele-controlado. El vehículo es capaz de establecer un enlace de comunicaciones Wifi con una red wifi en su radio de acción mediante protocolos basados en el estándar 802.11. Una vez se ha establecido el enlace de comunicaciones el usuario es capaz de acceder mediante un navegador web al servidor del sistema, donde se le presenta el panel de control del vehículo (embebido en el procesador de red del sistema microcontrolador).

Mediante el panel de control el usuario es capaz de enviar instrucciones al vehículo y visualizar información sobre el estado del mismo junto con información procedente de los sensores con lo que cuenta. El sistema también es capaz de enviar la información del sistema o telemetría a un servidor de syslog o similar con la finalidad de realizar estudios sobre los datos históricos usando herramientas de Big Data.

Los sensores con los que cuenta proporcionan información sobre el entorno -temperatura, presión atmosférica, humedad en el ambiente y luminosidad- y sobre el estado del vehículo -compás, acelero-metro y giroscopio-.

Se ha desarrollado una pequeña aplicación para teléfonos inteligentes basados en el sistema operativo Android que permite el control del rover y además un caso de uso sobre los datos históricos obtenidos de la telemetría del rover mediante la herramienta Splunk.

1.3 Objetivos del TFC.

Los objetivos del proyecto se han agrupado según su importancia entre objetivos primarios y objetivos secundarios -funcionalidades adicionales o mejoras-.

Los objetivos primarios del proyecto son:

- Dotar al sistema de conexión inalámbrica mediante protocolos WIFI.
- Implementar un panel de control para el vehículo basado en el protocolo HTTP.
- Dotar al sistema de la capacidad de ejecutar secuencias de instrucciones.
- Dotar al sistema de la capacidad de evitar las colisiones frontales.
- Contar con la capacidad de monitorizar las variables ambientales.

Por otro lado son objetivos secundarios del proyecto:

- Dotar al sistema de una aplicación de control para dispositivos móviles.
- Implementar un caso de uso sobre el aprovechamiento de la telemetría del vehículo.
- Dotar al sistema de seguridad mediante la implementación del control de acceso (usuario y contraseña) y cifrado de las comunicaciones (https).

1.4 Enfoque y método seguido.

El planteamiento dado al proyecto ha sido el de la creación de un prototipo que sirva de prueba de concepto, que de valide la idea y sirva de aprendizaje a el desarrollo de futuros productos basados en la tecnología utilizada.

Para el desarrollo del proyecto se ha seguido una serie de etapas:

- La primera etapa ha consistido en el aprendizaje, tanto de conceptos como de las tecnologías implicadas.
- A continuación se ha creado una diseño o visión de alto nivel del proyecto.
- La siguiente etapa ha consisto en la validación de los conceptos aprendidos mediante la realización de pruebas individuales sobre la plataforma elegida para el desarrollo del proyecto.
- El siguiente paso ha sido la integración de cada una de las funcionalidades individuales para obtener el resultado final.

- El últimos paso ha sido la validación del sistema en su totalidad y la implementación de mejoras.

Todas estas etapas se han incorporado a la planificación como actividades y distribuido a lo largo del tiempo de realización del trabajo.

1.5 PLANIFICACIÓN DEL TRABAJO

1.5.1 Listado de tareas del proyecto

Las tareas del proyecto se han agrupado en cuatro grandes bloques:

- Actividades relativas al diseño, tanto hardware como software. Donde se han incluidos las actividades relacionadas con el diseño físico, la elección de los materiales, los conexionados necesarios, módulos de software a implementar, etc.
- Actividades relacionadas con la generación del código de programa.
- Actividad de testeo y verificación del sistema.
- Actividades relacionadas con la documentación generada para el proyecto.
- Actividades relacionadas con la propia gestión del proyecto. Seguimiento y replanificación de las actividades realizadas.

1.5.2 Planificación temporal

Una vez obtenido el listado de tareas se ha procedido a la realización de la planificación temporal del proyecto, en dicha planificación el trabajo se ha divido en tres grandes fases que coinciden con las tres PEC (pruebas de evaluación continua) que se proponen en la asignatura.

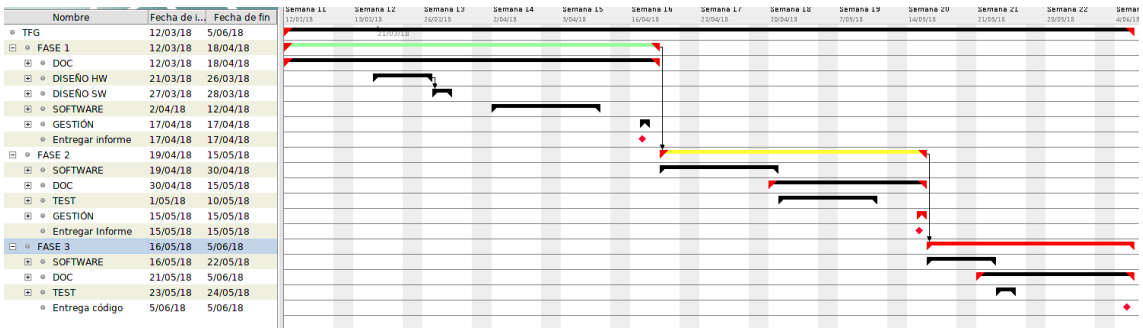


Ilustración 1: Vista completa de la planificación

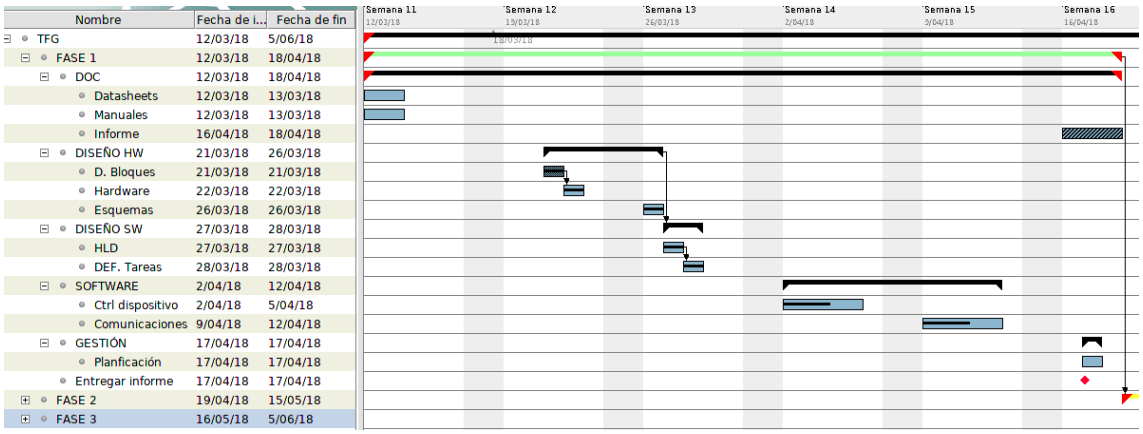


Ilustración 2: Planificación fase I

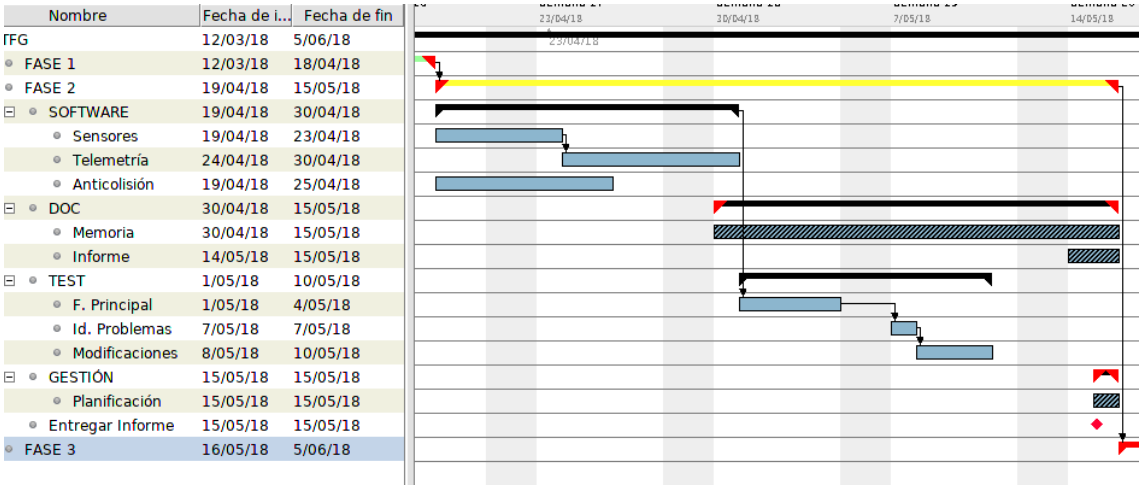


Ilustración 3: Planificación fase II

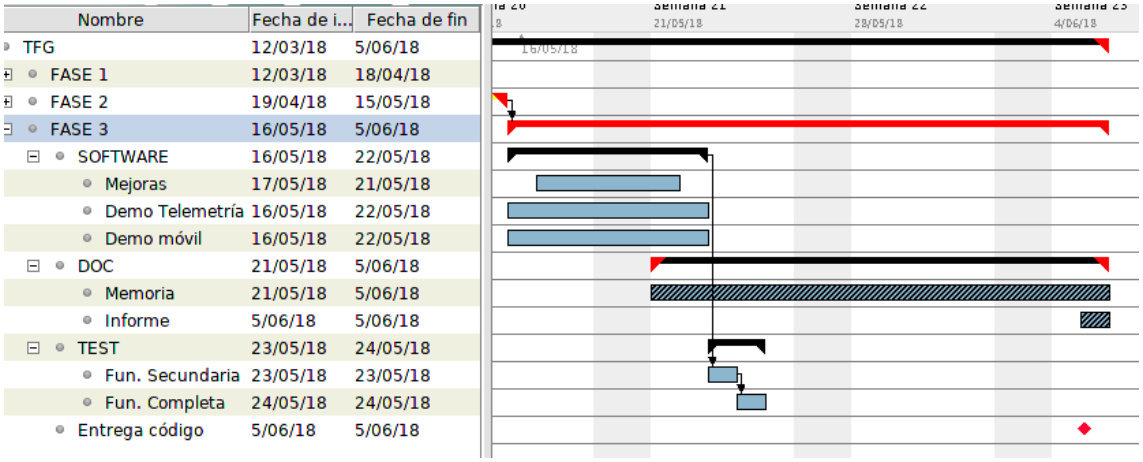


Ilustración 4: Planificación fase III

1.5.3 Identificación de riesgos

Durante la fase de planificación se identificaron una serie de riesgos asociados a la realización del proyecto y que se resumen en la siguiente tabla:

Riesgo	Probabilidad	Impacto	Implicaciones	Medidas correctivas
Incompatibilidad entre los componentes hardware del proyecto	Media	Medio	Modificar el diseño hardware y la sustitución de componentes incompatibles.	Utilizar en la medida de lo posible Boosterpack de TI. Validar compatibilidad mediante las hojas de características proporcionadas por los fabricantes.
Falta de tiempo para completar las actividades del proyecto.	Medio	Alto	Imposibilidad de completar el proyecto en la fecha programada.	Acotar el alcance del proyecto. Revisar la planificación para detectar las desviaciones. Priorizar las tareas principales del proyecto sobre las secundarias.
Complejidad en el montaje del chasis	Baja	Medio	Retraso en otras actividades del proyecto.	Realizar el montaje del chasis con anterioridad. Elegir entre modelos sencillos de montar.
Documentación del proyecto	Alta	Alto	La imposibilidad de que se valore el trabajo realizado para la realización del proyecto.	Dedicar el tiempo suficiente a la la realización de la memoria y la presentación. Revisar asignaturas donde se trabajan estos aspectos.

Tabla 1: Riesgos del proyecto

1.6 RECURSOS EMPLEADOS

A continuación se detallan los recursos utilizados para la realización del TFG agrupados según su tipo (hardware o software).

1.6.1 Recursos hardware

Texas Instruments SimpleLink™ MSP432P401R LaunchPad

Esta placa de desarrollo integra los sistemas necesarios para desarrollar aplicaciones para el microcontrolador Simplelink MSP432P401R e incluye las herramientas necesarias para la depuración y control del consumo de energía durante el desarrollo del sistema. Las principales características de esta placa de desarrollo son las siguientes:

- Sistema microcontrolador TI Simplelink MSP432P401R basado en el procesador ARM Cortex M4F de 32 bits capaz de operar a frecuencias de hasta 48 Mhz. Cuenta con una memoria flash de 256KB y 64KB de memoria RAM.
- Cuenta con 4 temporizadores de 16 bits y 2 de 32 bits.
- Soporta hasta 4 enlaces de comunicaciones I2C, 8 enlaces utilizando SPI y 4 enlaces UART.
- Cuenta con un conector universal de 40 pin compatible con los Boosterpack de TI.
- Integra el depurador XDS-110ET y un puerto UART (sobre USB) que permite la programación del sistema.

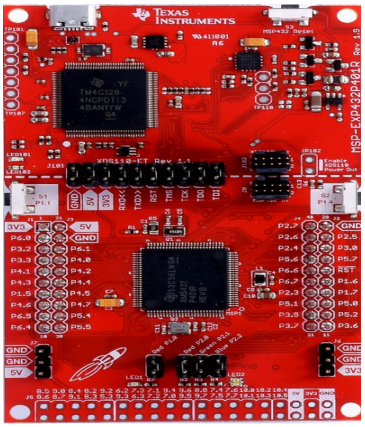


Ilustración 5: Kit de desarrollo MSP432P401R Launchpad

Texas Instruments SimpleLink™ Wi-Fi® CC3120 Wireless Network Processor BoosterPack Plug-In Module

Es un procesador de red WiFi (WNP) encargado de descargar a la MCU de las tareas relacionadas con la gestión de los enlaces inalámbricos basados en el estándar 802.11 b/g/n y de los protocolos de internet (IP) tanto en versión 4 (IPv4) como en su versión 6 (IPv6). Además implementa alguno servicios de red embebidos como son un servidor HTTP/S, servidor DHCP, etc.

Texas Instruments Advanced Emulation Kit for SimpleLink™ Wi-Fi® CC31xx BoosterPack™ plug-in module

Es un placa de emulación destinada a la procesadores de red de la serie SimpleLink™ Wi-Fi® CC31xx. Esta placa permite actualizar la memoria flash de estos dispositivos -tanto con e contenido de usuario como con el firmware- y evaluar el rendimiento de estas placas mediante el software SimpleLink™ Studio.

TI BOOSTXL-SENSORS Sensors BoosterPack Plug-in Module

Es una placa accesoria para las tarjetas de desarrollo de los microcontroladores Simplelink, que proporciona al sistema los siguientes sensores digitales: sensor inercial BOSCH BMI160 que proporciona un acelerómetro y giroscopio digitales de 6 ejes, un sensor Bosch BMM150 que mide la intensidad del campo magnético en 3 ejes, un sensor digital Bosch BME280 que proporciona la presión atmosférica, la temperatura y la humedad relativa del aire, un sensor TI OP3001 que mide la luz ambiental y un sensor de temperatura infrarrojo TI TMP007 que es capaz de medir la temperatura de un objecto a distancia.

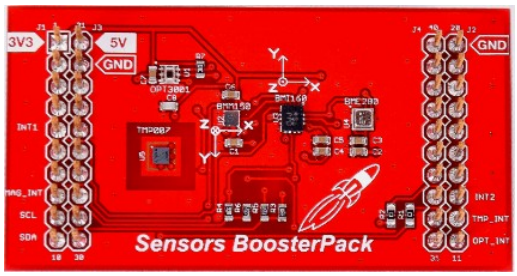


Ilustración 6: TI Sensor BoosterPack Plug-in Module

Sensor de ultrasonidos HC-SR04

Es sensor HC-SR04 es un circuito orientado a obtener medidas -distancias entre los 2 cm y los 4 metros- utilizando para ellos pulsos de ultrasonidos.

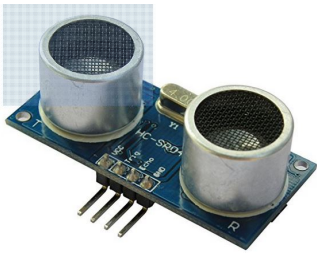


Ilustración 7: Módulo ultrasónico HC-SR04

Conversor de niveles lógicos 3.3/5v BSS138

Este circuito permite adaptar circuitos que trabajan con tecnología CMOS (3.3 voltios) a circuitos que funcionan con tecnología TTL (5 voltios) y viceversa. Este circuito esta basado en el transistor FET(transistor de efecto campo) BSS138.

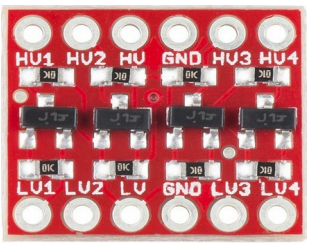


Ilustración 8: Conversor de niveles lógicos

L298N Dual H-Bridge DC Motor Controller

Este circuito permite controlar la entrega de potencia a cargar inductivas como motores de corriente continua, es capaz de entregar hasta 3A de corriente a cada una de las dos salidas con las que cuenta.

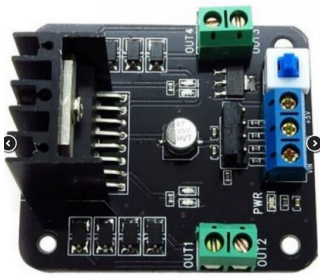


Ilustración 9: Circuito L298N H-Bridge

Devastator Tank Mobile Robot Platform de DF Robot

Es un chasis para robot móviles que cuenta con una plataforma de aluminio y utiliza orugas para desplazarse. El chasis viene equipado con dos motores eléctricos de corriente continua que se pueden alimentar con tensiones de entre 3 y 8 voltios.

Ilustración 10: Chasis del robot utilizado para la realización del proyecto

Equipo de desarrollo

Se ha utilizado un portátil Toshiba Satellite Pro A50-D-1FZ y un equipo de sobremesa basado en un procesador AMD Phenom X6 con 8 GB de memoria RAM.

Osciloscopio y polímetro

Para realizar medidas y comprobaciones se ha utilizando un osciloscopio portátil SainSmart DS203 y un polímetro KAISE MAS830L.

1.6.2 Recursos software

TI Code Composer Studio 8

Code Composer Studio es un entorno de desarrollo o IDE indicado para el desarrollo de software en lenguaje C/C++ sobre microcontroladores y sistemas embebidos de TI. Cuenta con un compilador, un editor de código y las herramientas necesarias para la depuración de código.

TI Uniflash Standalone Flash Tool for TI Microcontrollers, Sitara Processors & Simplelink devices 4.2.2

Esta herramienta permite programar la memoria flash disponible en las MCU y dispositivos Simplelink de Texas Instruments.

FreeRTOS v10

Es el sistema operativo en tiempo real de libre uso (licencia BSD) y propiedad de Amazon.

Simplelink SDK

Los Simplekink SDK consisten en una serie de paquetes de software proporcionado por Texas Instruments para el desarrollo de software para la plataforma MSP432. Esta compuesto por drivers, API's (application programming interface) y ejemplos de uso que permiten la potabilidad del código entre todos los dispositivos de la familia SimpleLink.

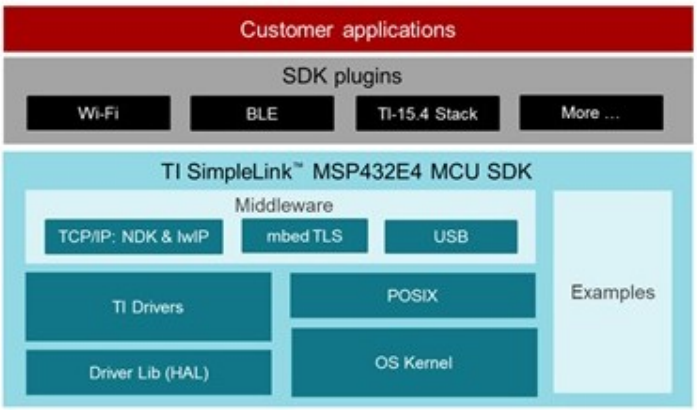


Ilustración 11: Arquitectura de los Simplelink SDK

Bluefish Web Development Studio

Bluefish es una herramienta de código libre orientada al desarrollo de aplicaciones web basadas en HTLM.

MIT App Inventor 2

App Inventor 2 es un entorno de desarrollo visual Online para aplicaciones móviles basadas en el sistema operativo Android desarrollada por MIT (Massachusetts Institute of Technology) y disponible a través de la siguiente URL http://appinventor.mit.edu/explore/index-2.html.

Splunk

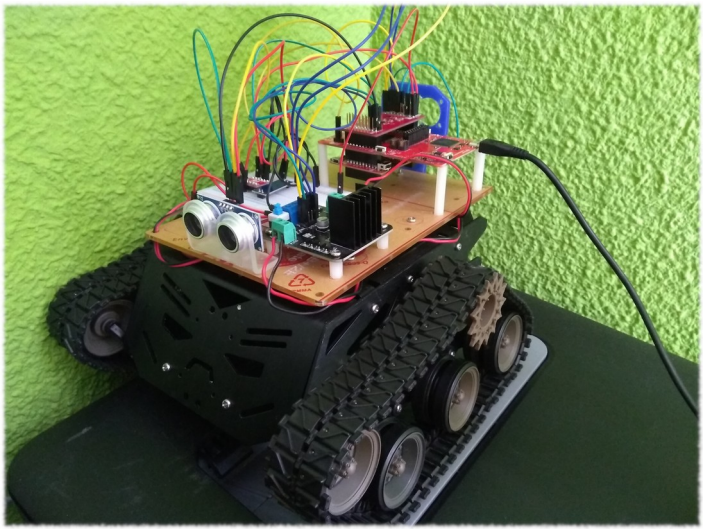
Splunk[1] es un software comercial que permite analizar los datos generados por maquinas y extraer información de los mismos. Cuenta con una versión libre aunque con ciertas limitaciones.

Sistema operativo Ubuntu

Es el sistema operativo con el que trabajan los equipos informáticos utilizados durante el desarrollo del proyecto. Incluye la suite ofimática LibreOffice.

1.7 PRODUCTOS OBTENIDOS

Un prototipo de vehículo tele-controlado mediante protocolos Wifi que implementa un panel de control web y es capaz de obtener datos del entorno y del estado del mismo a través de los sensores que incorpora.



Una aplicación móvil que permite la gestión del vehículo a través de un teléfono inteligente basado en el sistema operativo Android.

Una aplicación web sobre la plataforma Splunk para visualizar y analizar la telemetría del vehículo a lo largo del tiempo.

1 Es posible encontrar más información en el portal web del producto www.splunk.com

1.8 Breve descripción de los otros capítulos de la memoria.

En el capitulo antecedentes se describe la evolución de los sistemas empotrados, su relación con otras tecnologías actuales y las opciones de desarrollo de la plataforma Simplelink de Texas intruments.

A continuación en los siguientes dos capítulos se describe el sistema propuesto. En el primero se hace una descripción funcional del sistema, donde se muestra la arquitectura del sistema desarrollado y las aplicaciones clientes que complementan el sistema propuesto. A continuación, en el siguiente capitulo, se describe en detalle el sistema desde el diseño hardware para a continuación pasar a describir cada uno de los módulos software.

Los últimos capítulos se dedican a analizar la viabilidad, los costes del proyecto y por último reflejar las conclusiones obtenidas a la realización del proyecto. Finalmente la memoria incluye varios anexos con detalles técnicos de varias de las tecnologías que se han utilizado en la realizado del trabajo.

2 Antecedentes

2.1 Estado del arte

2.1.1 Evolución sistemas empotrados.

Durante los últimos años los sistemas empotrados han sufrido una tremenda evolución. Han pasado de ser sistemas prácticamente aislados del exterior a ser sistemas conectados, esto ha sido gracias por un lado a la amplia difusión de Internet y por otro a la propia evolución de la capacidad de los microcontroladores que les ha permitido incorporar módulos de comunicaciones avanzados basados en tecnologías como Bluetooth, Wi-Fi, LoRa,…

Esta evolución de los sistemas esta ligada al concepto Internet de las Cosas (IoT), donde se podría decir que detrás del termino “cosas” en la mayoría de las ocasiones vamos a encontrar un sistema empotrado. El hecho que los sistemas empotrados hayan pasado a ser dispositivos conectados -dispositivos IoT-, ha supuesto un enorme reto tanto para los desarrolladores de estos sistemas como para los fabricantes de hardware.

Los fabricantes hardware han tenido que dotar a sus dispositivos de mecanismos de seguridad -como la autentificación y el cifrado- para lo ha sido necesario incorporar a sus diseños aceleradores criptográficos, mecanismos para proteger el firmware de los equipos, mecanismos ágiles para las actualizaciones tanto firmware como para el software, etc.

Internet ha sido el punto donde han convergido los sistemas embebidos que controlan nuestras “cosas” con los sistemas de propósito general desde donde el usuario quiere tener acceso a las mismas. Los desarrolladores se han visto obligado a dotar a sus desarrollos de mecanismos para que el usuario pueda acceder a la información de los dispositivos tanto desde su red local como a través de Internet, para ello han tenido que adaptar sus diseños a nuevos protocolos -protocolos estándar de Internet- como puede ser el ejemplo del protocolo HTTP.

Otro importante reto que ha surgido ha raíz de la ingente cantidad de dispositivos conectados a la red -sistemas embebidos o dispositivos *IoT*- es el tratamiento y aprovechamiento de la gran cantidad de información que generan -información útil tanto como para los desarrolladores como para los consumidores de estos productos-. Aquí es donde el conjunto de tecnologías conocidas como *Big Data*[2] ha pasado a tener un papel relevante.

2 *Big Data* es un conjunto de tecnologías orientados al tratamiento masivo de información y a la obtención de valor de la misma.

2.1.2 Análisis de plataformas microcontrolador: Arduino y Simplelink MCU.

Arduino es una plataforma para el desarrollo de sistemas microcontrolador de amplia difusión debido a su carácter abierto que cuenta con amplia de microcontroladores. Este gama de microcontroladores abarca desde los más sencillos basados en microcontroladores Atmel ATmega328 de 8 bits hasta sistemas microcontrolador basados en ARM Cortex-M0+ . Además cuenta con una inmensa gama de periféricos desarrollados o compatibles para estos sistemas. Por último cuenta con un amplio soporte software que incluye un entorno de desarrollo y un lenguaje de programación de alto nivel.

La plataforma Simplelink es la actualidad esta formada por placas microcontrolador basadas en el microcontrolador MSP432 de Texas Instruments basadas en ARM Cortex-M4 de bajo consumo, placas de comunicaciones y otro módulos periféricos compatibles. Desde el punto de vista del software Simplelink soporta varios IDE tanto libres como comerciales, incluido un entorno clon del utilizado por Arduino, también soporta incorporar varios sistemas operativos en tiempo real tanto libres como comerciales.

Arduino es ideal para principiantes pues cuenta con multitud de recursos de aprendizaje y una enorme base de usuarios. Por contra Simplelink tiene una curva de aprendizaje más brusca y además cuenta con menos recursos para su aprendizaje. También hay que tener en cuenta que Simplelink cuenta con un SDK que simplifica el desarrollo para sus placas.

Sin embargo desde el punto de vista del coste, la placa Arduino más sencilla conocida como Arduino Uno tiene un coste aproximado de 20€ frente los 13€ del Launchpad MSP432P401R, como el utilizado en este proyecto, que es un sistema mucho más potente.

Por otro lado hay que tener en cuenta que Simplelink cuenta con entornos de desarrollo y depuración profesionales que cuentan con funcionalidades avanzas que pueden ser necesarias en proyectos de cierta complejidad.

2.1.3 Sistemas operativos en tiempo real en la plataforma TI MSP432

Aunque para el desarrollo de un sistemas embebido no es un requisito indispensable hacer uso de un sistemas operativo, cuando nos enfrentamos a un desarrollo de cierta complejidad si es altamente recomendable la utilización de un sistema operativo en tiempo real debido a la flexibilidad, modularidad y facilidades para la resolución de problemas que nos ofrece.

Cuando nos referimos a la plataforma Simplelink MCU de TI podemos optar por cuatro sistemas operativos:

- FreeRTOS es un sistema operativo en tiempo real recientemente adquirido por Amazon pero que se ofrece de forma totalmente abierta (bajo licencia BSD).

- TI-RTOS es un sistema operativo en tiempo real desarrollado por Texas Instruments, evolución del sistema SYS/BIOS, que se ofrece de forma abierta (bajo licencia BSD).
- MICRIUM-OS en un sistema operativo en tiempo real portado a más de 50 arquitecturas desarrollado por Micrium Software Embebed y que se ofrece bajo licencia comercial.
- RTK es un sistema operativo en tiempo real desarrollado para la plataforma ARM Cortex ofrecido por la empresa Arm Keil. Es un software libre de royalties pero tiene algunas restricciones para su uso.

A continuación se comparan las ventajas que aportan tanto FreeRTOS como TI-RTOS para el desarrollo de sistemas en la plataforma TI MSP432 -se ha descartado MICRIUM OS y RTK debido a su baja cuota de mercado y las restricciones que tienen para su utilización-:

- FreeRTOS tiene la ventaja de ser el sistema operativo líder del mercado junto con Embebed Linux[3] por lo que es ampliamente conocido por los profesionales del sector.
- FreeRTOS es más versátil al poder trabajar en modo colaborativo y anticipativo. TI-RTOS solo opera en modo anticipativo.
- FreeRTOS soporta diversas arquitecturas y plataformas por lo que una posible migración del desarrollo a otra plataforma (no TI) es menos traumático que en el caso de TI-RTOS.
- TI-RTOS tiene una mejor integración con la plataforma MSP432 y con la ecosistema Simplelink (SDK, TI drivers,…).

En definitiva FreeRTOS ofrece portabilidad mientras que TI-ROS ofrece una mejor integración con el ecosistema Simplelink.

3 De acuerdo al estudio “2017 Embedded Markets Study” realizado por Embebbed.com

2.1.4 Plataformas de desarrollo para microcontroladores TI MSP432P

A la hora de desarrollar código para la plataforma MSP432 existen principalmente dos opciones: Energia y Code Composer Studio (aunque también existe la posibilidad de usar otros IDE como IAR Embedded Workbench®, Keil, etc.)

Energia es un entorno de desarrollo integrado de código abierto basado en Wiring[4] y Procesing[5], por lo que es muy similar al entorno de desarrollo que suele utilizarse para Arduino. Energia puede ejecutarse tanto en la computadora local como en la nube a través de la web para desarrolladores de TI (Cloud CCS) y tiene la posibilidad de exportar el código generado a CCS.

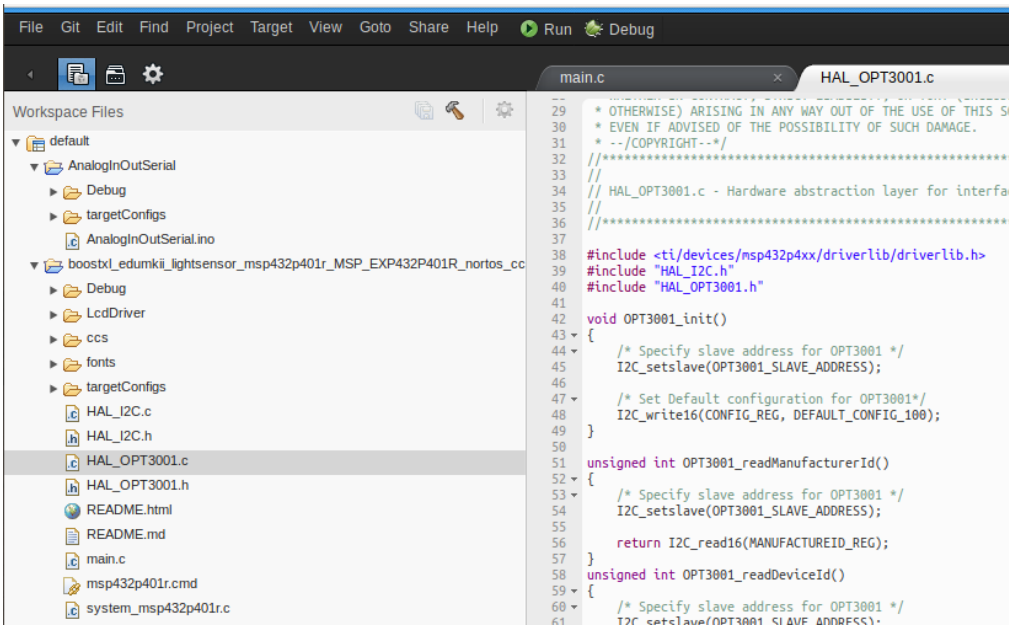


Ilustración 12: Energia ejecutandose desde la Cloud CCS

Code Composer Estudio (CCS) es un IDE profesional basado a su vez en el entorno Eclipse. CCS cuenta con un editor de código, compilador del lenguaje C/C++, depurador, herramientas para analizar el consumo de energía, … CCS ofrece soporte para un mayor numero de plataformas comparado con Energia.

Podemos resumir que Energia es un entorno ideal para iniciarse en el desarrollo de software para lo ecosistema Simplelink mientras que CCS es un entorno profesional de desarrollo que cuenta con capacidades avanzadas de depuración.

4 Es posible encontrar más información sobre Wiring en la web del proyecto disponible en: http://wiring.org.co/
5 Es posible encontrar más información sobre Procesing en la página web del proyecto: https://processing.org/

2.2 ESTUDIO DE MERCADO

2.2.1 Orientación del producto

La orientación de de este proyecto hacia el mercado sería la de ofrecer una plataforma y sistema base universal (chasis robot que incluyera la MCU principal y los sensores de posicionamiento) que el cliente pudiera personalizar añadiendo los módulos necesarios en función de la finalidad a la que el vehículo fuera a ser destinado (vigilancia, monitorización de variables ambientales, exploración de ambientes tóxicos, etc.). Los módulos personalizables incluirían sensores, módulos de comunicaciones diferentes, he incluso algún sistema para que el robot pudiera captar muestras del entorno.

El ofrecer una plataforma universal con diversas utilidades permitiría ampliar la base de futuros clientes y además reducir los costes de desarrollo y mantenimiento. No sería un producto orientado al gran publico y a la producción en masa si no que estaría más orientado al mundo profesional con requerimientos específicos.

2.2.2 Propuestas similares en el mercado

Fuera del ámbito universitario y de investigación no ha sido posible identificar una plataforma con una orientación similar a la propuesta.

Sin embargo dentro del ámbito de la monitorización con sensores ambientales diversas empresas ofrecen estaciones fijas. Este sería el caso Sigfox a través de su plataforma Sens'it[6] que ofrece un dispositivo IoT portable que es capaz de monitorizar algunas variables del entorno (luz, vibraciones, temperatura, humedad, …), la plataforma integra un sistema de comunicaciones desarrollador por la misma empresa y una plataforma en la nube donde el usuario puede acceder a los datos y personalizar el funcionamiento del dispositivo.

6 Se puede encontrar más información del producto Sens'it en la web del fabricante https://www.sensit.io/

3 Descripción Funcional

3.1 Rover telecontrolado mediante protocolos de internet (WEB/HTTP)

3.1.1 Descripción del sistema

El sistema desarrollado esta compuesto por el chasis robot donde se alojan todos los componentes del sistema: el sistema microcontrolador principal (MCU), el procesador de red inalámbrico (WNP), el módulo de sensores, el sensor de proximidad y sistema de alimentación.

El usuario interactuá con el sistema a través del portal de administración web embebido en el el sistema para ello puede hacer uso de un navegador, una aplicación para móviles basados en el sistema operativo Android o mediante la linea de comandos utilizado utilidades que soporten el protocolo HTTP (por ejemplo cURL).

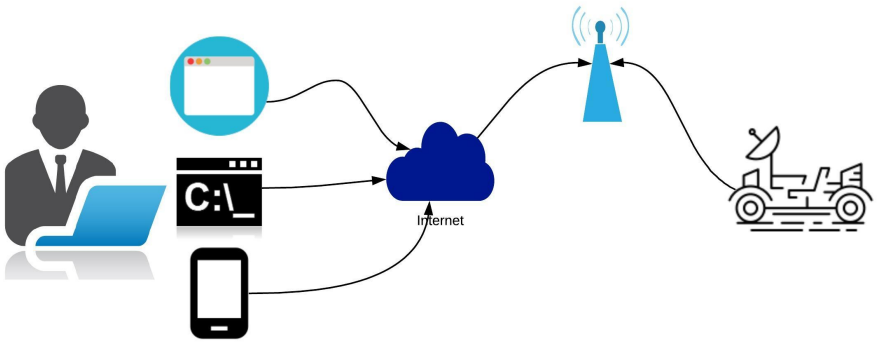


Ilustración 13: Diagrama conceptual del sistema de control del rover

El sistema cuenta con varios modos de funcionamiento:

- Modo interactivo. En este modo de funcionamiento el usuarios introduce las instrucciones en el sistema y esta se ejecutan inmediatamente.
- Modo secuencia. En este modo el usuario introduce instrucciones secuencialmente en el sistema y asigna una duración temporal a cada una de ellas. El sistema encola cada una de las instrucciones recibidas y las ejecuta por orden de llegada.
- Modo bucle. En este modo el sistema espera recibir una secuencia de comandos que repetirá de forma secuencial el número de veces indicado por el usuario.

Durante la operación del vehículo el sistema capta de forma periódica la información de los sensores instalados, parte de esta información se visualiza sobre el panel de control y simultáneamente se enviá para su tratamiento al servidor configurado a tal efecto.

3.1.2 Arquitectura del sistemas

A continuación se presenta la arquitectura hardware del sistema a partir del diagrama de bloques. En el diagrama anterior además podemos apreciar los protocolos de comunicaciones utilizados de forma interna y con el exterior.

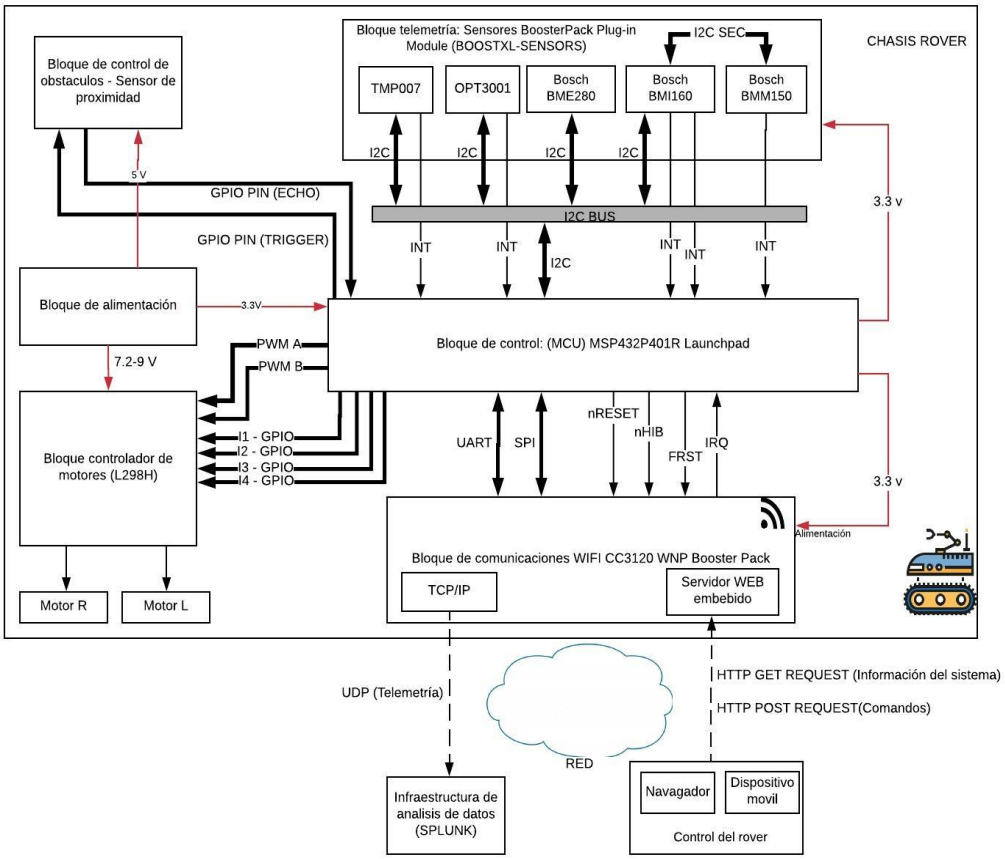


Ilustración 14: Diagrama de bloques del sistema

La arquitectura del sistema a nivel software quedaría definida mediante el diagrama de módulos que se muestra a continuación donde además se puede apreciar las relaciones entre los diferentes módulos.

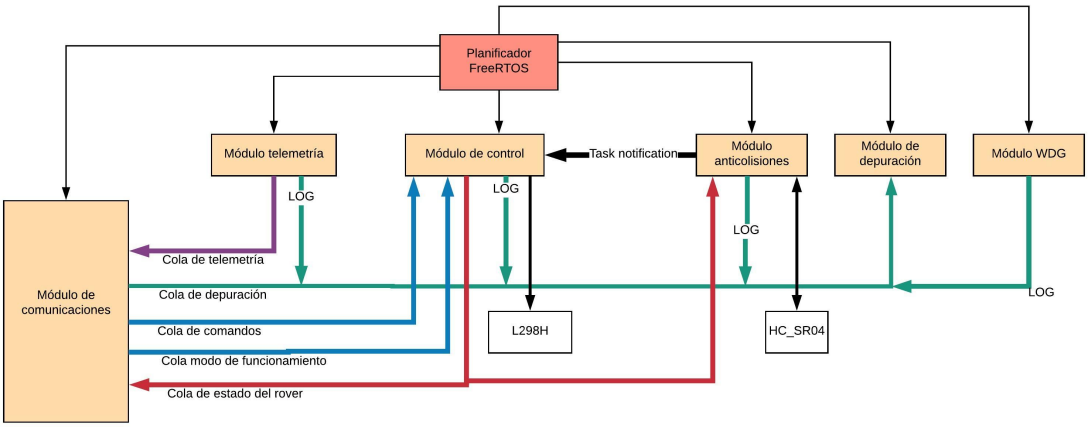


Ilustración 15: Diagrama de módulos software

3.1.3 Arquitectura de comunicaciones

La arquitectura de comunicaciones esta basada en el modelo TCP/IP. A través de la figura se detalla la distribución de las capas del modelo y los correspondiente protocolos utilizados en la implementación del sistema en cada una de las capas.

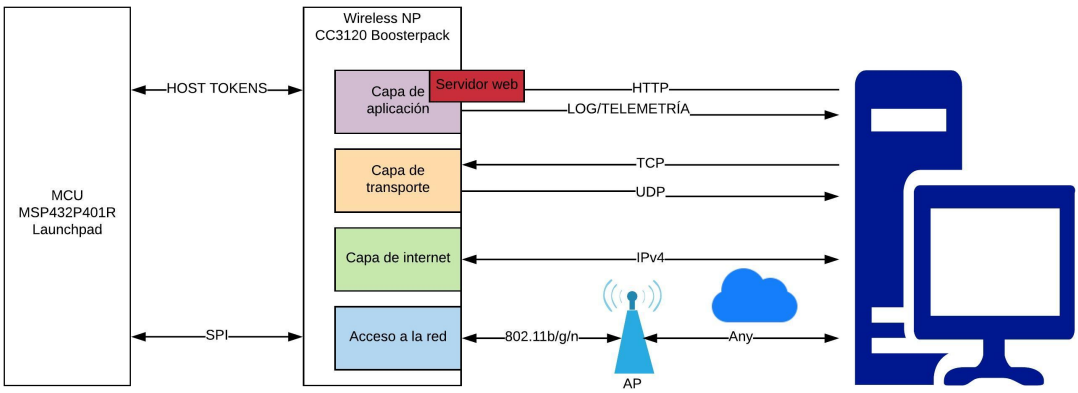


Ilustración 16: Arquitectura de comunicaciones

Lo primero que podemos observar es que el procesador de red implementa la mayor parte de los procesos de red liberando de esta forma a la MCU de estos tareas. Este hecho se puede apreciar con más detalla a a través de la siguiente ilustración:

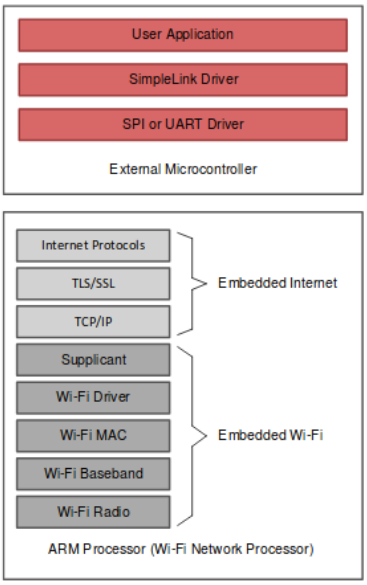


Ilustración 17: CC3120R Software Overview. Fuente: Texas Instruments Incorporated

A continuación de detallan los aspectos más importantes de los protocolos utilizados por cada una de las capas del modelo TCP/IP:

- En la capa de acceso se establece un enlace inalámbrico basado en el protocolo 802.11b/g/n sobre la banda de los 2.4 Ghz donde el WNP se comporta como cliente del punto de acceso. Desde el punto de vista de las funciones de seguridad se utiliza WPA2 (Wifi Protected Access 2) con clave pre-compartida.
- En la capa de red se hace uso del protocolo IPv4 con asignación de los parámetros de red de forma estática (dirección IP, máscara de subred, …).
- En la capa de transporte se hace uso de TCP o UDP en función de la aplicación de capa superior que estemos usando.
- En la capa de aplicación, por una lado el sistema utiliza HTTP mediante el servidor web embebido implementado en la WNP y por otro se encuentra la aplicación que gestiona el envió de telemetría (basado en UDP).

3.2 APLICACIONES CLIENTE

3.2.1 Panel de control web

El panel de control web de control del sistema esta compuesto por una pagina web principal, donde se encuentran las imágenes que representan las acciones a ejecutar (avanzar, acelerar, …), y un *frame* inferior que presenta la información del sistema. Ambas paginas se han desarrollado utilizando HTML y Javascript. El panel de control permite el control del vehículo unicamente en modo interactivo.

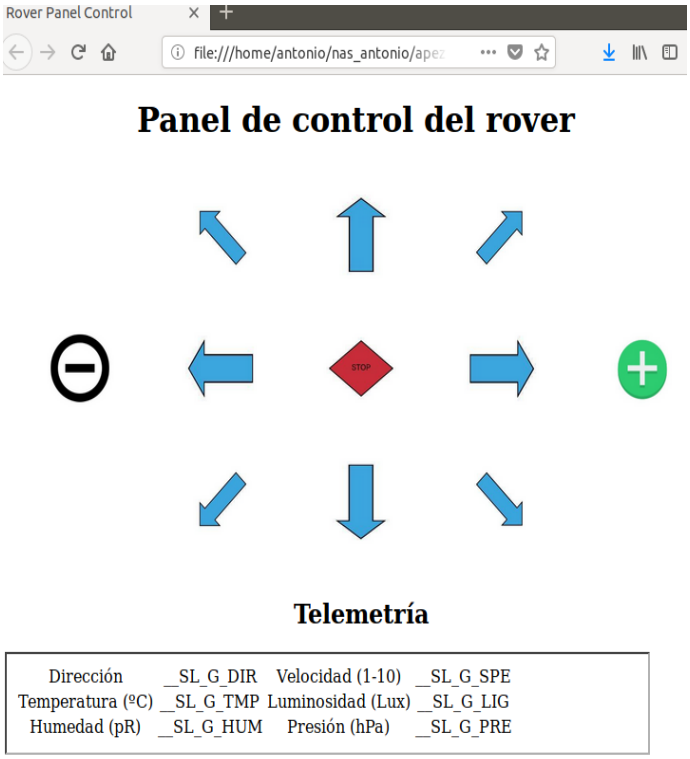


Ilustración 18: Aspecto de la página web desarrollada para el panel de control del sistema

El código HTML de la pagina implementa el contenido estático de la web. Dentro de este código se encuentran insertados una serie de identificadores especiales o "*host token*"[7] que al ser procesados por el servidor web en tiempo de ejecución del sistema son procesados por la MCU y sustituidos por los valores del parámetros correspondientes.

7 Estos identificadores son conocidos como Host Token y su funcionamiento se detalla en el anexo adjunto.

Mediante código JavaScript se generan las peticiones HTTP POST -desde el lado del cliente- que contienen codificadas mediante los parámetros de la petición las instrucciones a ejecutar (en forma también de *host token*). Estas peticiones se originan cada vez que el usuario pulsa sobre el icono que representa la instrucción o movimiento a ejecutar.

También se utiliza JavaScript para forzar el refresco del marco inferior y de esa forzar a que se actualicen los valores con la telemtría.

3.2.2 Aplicación de control para móviles

La aplicación desarrollada para móviles que utilizan el sistema operativo Android permite el control del vehículo en modo interactivo. La aplicación se ha desarrollo como demostración de las facilidades que ofrece trabajar con un protocolo estándar y abierto como HTTP a la de hora de integrar diferentes sistemas y tecnologías.

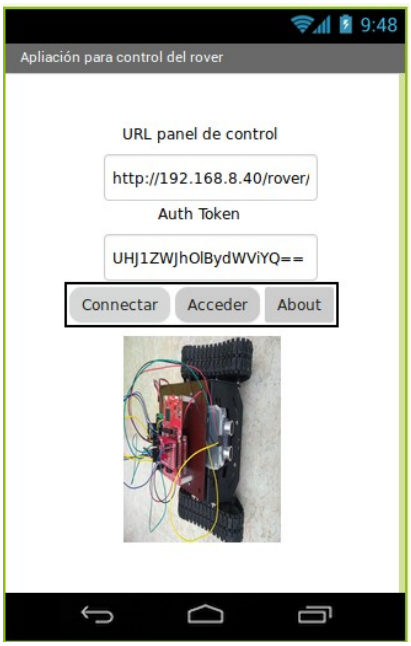


Ilustración 20: Aplicación móvil configuración

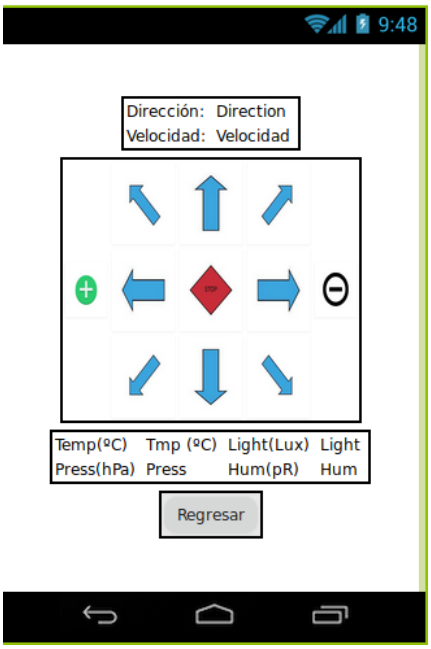


Ilustración 19: Aplicación móvil control del vehículo

Como se aprecia en la las figuras la aplicación cuenta con dos paneles, uno donde se realiza la configuración del sistema y otro de control del vehículo que imita el aspecto del panel de control web. Este sistema ha sido desarrollado utilizando en entorno de desarrollo visual MIT App Inventor 2 debido a las facilidades que ofrece para el desarrollo de aplicaciones móviles.

App Inventor permite desarrollar una aplicación para móviles de una forma visual, por un lado permite modelar el aspecto de las distintas pantallas de nuestra aplicación y por otro modelar su comportamiento mediante “bloques”.

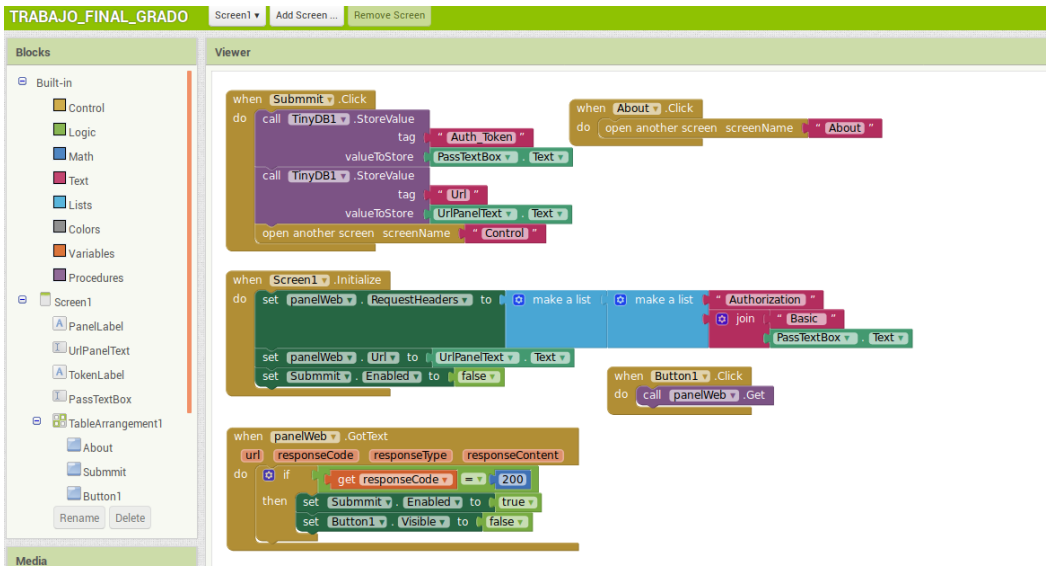


Ilustración 21: IDE Online App Inventor 2

3.2.3 Aplicación Splunk para visualización de la telemetría

La aplicación desarrollada para Splunk[8] sirve de ejemplo de la explotación de la información que generan los sensores. La aplicación esta dividida en dos paneles para diferenciar la información de estado del vehículos (giroscopio, acelerómetro, …) de la información medioambiental (temperatura, humedad, …).

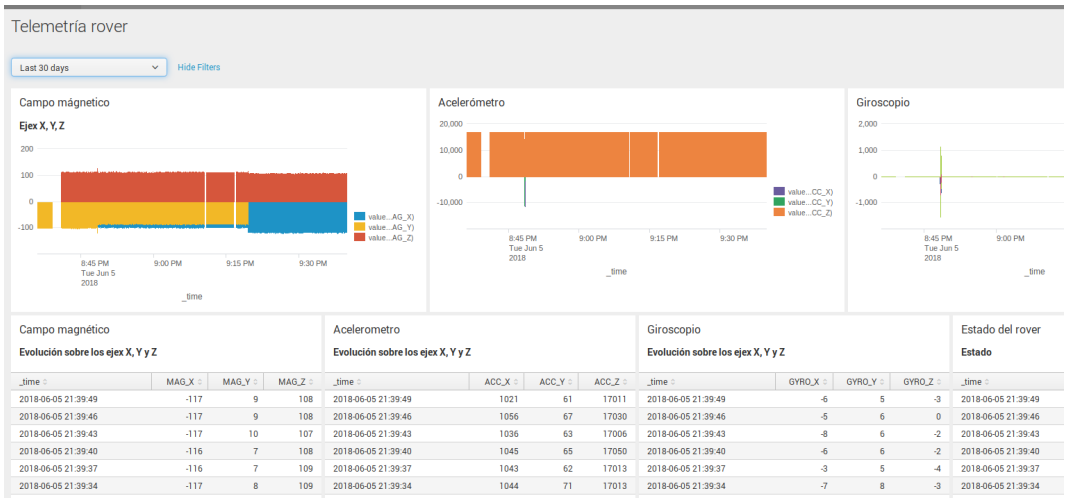


Ilustración 22: Vista del panel con la telemetría del rover

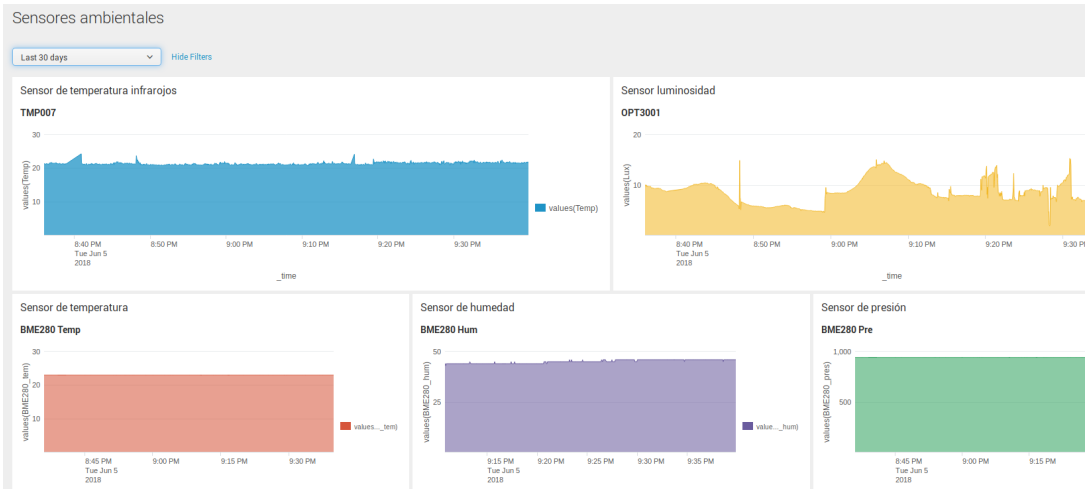


Ilustración 23: Vista con la información de los sensores ambientales

8 Es posible obtener más información sobre Splunk en la web del fabricante www.splunk.com

4 Descripción Detallada

4.1 DESCRIPCIÓN HARDWARE

4.1.1 Alimentación eléctrica del sistema

Para diseñar el sistema de alimentación se ha tenido en cuenta los requerimientos de alimentación de cada uno de los componentes que lo forman. A continuación se enumera los características de la alimentación de los elementos principales del sistema:

- El MSP432P401R Launchpad requiere una tensión de alimentación de 3.3 voltios de corriente continua (CC) y un consumo de potencia de 80uA/Mhz en modo activo[9].
- El procesador de red CC3120 Boosterpack puede alimentarse con una tensión de 3.3 voltios (tiene un regulador DC-DC interno) y un consumo de energía de hasta 350mA[10].
- La placa de sensores Sensors BoosterPack Plug-In Module requiere una tensión de alimentación de 3.3 voltios[11].
- Cada uno de los motores del chasis del vehiculo requiere una alimentación de entre 3.5 y 7 voltios , y puede tener un consumo de hasta 2.8A.
- El sensor de ultrasonidos HC-SR04 opera a una tensión de 5 voltios y un consumo entorno a los 15mA.
- El circuito adaptador de niveles lógicos requiere estar alimentado a 5 voltios y a 3.3 voltios para poder funcionar.

A partir de los características anteriores se ha procedido a diseñar el sistema de alimentación. El diseño ha aprovechado el bloque de baterías proporcionado con el kit del chasis del vehículo. Este bloque de baterías esta compuesto por 6 baterías del tipo AA -donde cada una de las baterías que lo forman proporcionan hasta 1.5 voltios de CC(corriente continua) y cuenta con una capacidad de 2500-3000 mAh-. Este sistema es capaz de proporcionar la suficiente potencia a los motores eléctricos del chasis y alimentar al resto de componentes del sistema.

El bloque de baterías se conecta al circuito de control de potencia “L298H Dual H-Bridge” que se encarga de entregar la potencia los motores del vehículo. Esta conexión además permite aprovechar la salida regulada de 5 voltios que proporcionar el circuito de control de potencia cuando lo alimentamos con tensiones inferiores a 12 voltios.

9 Apartado 2.4. “MSP432P401R SimpleLink Microcontroller LaunchPad Development Kit (MSP-EXP432P401R). User's Guide.” disponible en http://www.ti.com/lit/ug/slau597f/slau597f.pdf.
10 Apartadado 2.4 “CC3120 SimpleLink Wi-Fi BoosterPack Plug-in Module and IoT Solution. User Guide” disponible en http://www.ti.com/lit/ug/swru457/swru457.pdf.
11 Apartado 2.2.1 ”BOOSTLXL-SENSOR BoosterPack Plug-in Module User’s Guide (Rev. b)” disponible en http://www.ti.com/lit/ug/slau666b/slau666b.pdf.

Finalmente para obtener los 3.3 voltios de tensión necesarios para alimentar la MCU y los BoosterPack se ha recurrido a un circuito regulador de voltaje AMS1117. Este circuito es capaz de obtener una salida regulada de 3.3 voltios a partir de una tensión de 5 voltios.

En la figura se puede apreciar un esquema con el sistema de alimentación y las diferentes conexiones con los componentes a los que alimenta.

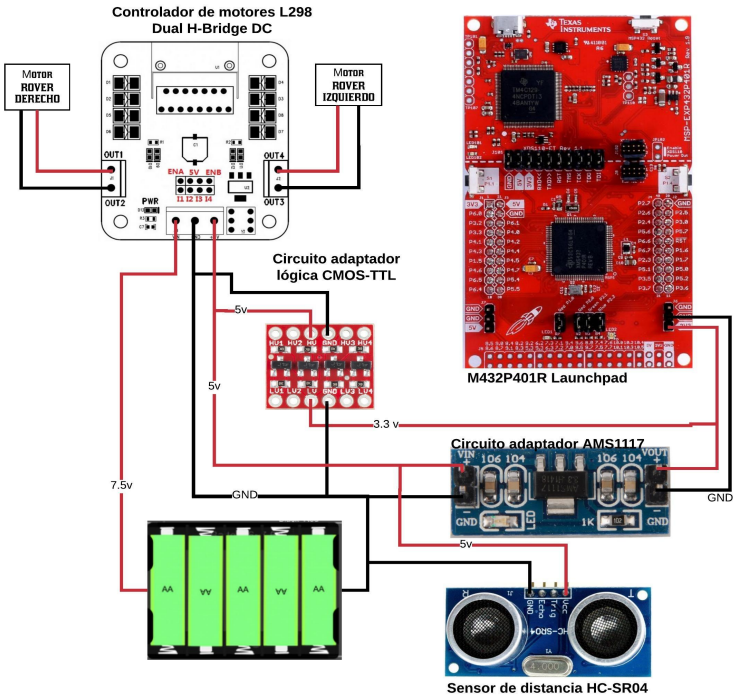


Ilustración 24: Diagrama de alimentación del sistema

4.1.2 Conexionado del los diferentes módulos del sistemas

El primer paso que se ha sido verificar la compatibilidad entre Launchap y BoosterPack utilizados, para ello se ha recurrido a la aplicación web Booster Checker[12]. Mediante esta aplicación ha sido posible detectar el pin 4.1 (en el Launchapad) es utilizado tanto por el BoosterPack CC3120 (nHIB) como por el BoosterPack de sensores (INT1).

Para mitigar dicha incompatibilidad entre las placas se ha recurrido a la siguiente solución técnica:

- Conectar el BoosterPack CC3120 a la parte inferior de la MCU princpal (Launchpad) y el BoosterPack de sensores a la parte superior.

12 La aplicación Booster Checker esta disponible en la web de desarrolladores de TI a través del enlace https://dev.ti.com/bpchecker/

- Eliminar el pin 4.1 del LaunchPad en su parte superior.
- Redirigir el puerto 4.1 en el Sensors BoosterPack Plug-In Module al puerto 6.1[13].

El siguiente paso la hora de diseñar el conexionado entre componentes ha consistido en verificar que la MCU dispone de los suficientes puertos de entrada-salida (GPIO) para acomodar el resto de periféricos del sistema. Para ello los requerimientos que han tenido en cuenta son los siguientes:

- El circuito HC-SR04 requiere un puerto de salida (trigger) y un puerto de entrada (echo). Además su lógica de funcionamiento es TTL.
- El circuito de control de potencia requiere 4 puertos de salida para controlar el estado de los dos motores y dos puertos adicionales para controlar la potencia de los mismos mediante PWM. Además su lógica de funcionamiento es TTL.

A partir de los requerimientos y las modificaciones introducidas se ha construido una tabla de asignación de puertos (donde J1, J2, J3 y J4 corresponden a cada una de lineas de pins en el Launchpad MSP432P401R).

J1	MSP432	SENSOR	CC3120	GPIO
1	3.3V	3.3V	3.3V	
2	P6.0			
3	P3.2		UART1_TX	
4	P3.3		UART1_RX	
5	P4.1	INT1	nHIB	
6	P4.3			
7	P1.5		SPI_CLK	
8	P4.6	MAG_INT		
9	P6.5	I2C SCL		
10	P6.4	I2C SDA		

J3	MSP432	SENSOR	CC3120	GPIO
1	5V		5V	
2	GND	GND	GND	
3	P6.1	INT1		
4	4.0			HC-SR04 ECHO INPUT
5	P4.2			
6	P4.4			HC-SR04 TRIG OUTPUT
7	P4.5			L298H I1 OUTPUT
8	P4.7			L298H I2 OUTPUT
9	P5.4			L298H I3 OUTPUT
10	P5.5			L298H I4 OUTPUT

J2	MSP432P401R	SENSOR	CC3120	GPIO
1	P2.7			L298H ENA OUTPUT
2	P2.6			
3	P2.4			L298H ENB OUTPUT
4	P5.6		UART1_CTS	
5	P6.6		UART1_RTS	
6	P6.7			
7	P2.3		NWP_LOG_TX	
8	P5.1		WLAN_LOG_TX	
9	P3.5			
10	P3.7			

J4	MSP432P	SENSOR	CC3120	GPIO
1	GND	GND	GND	
2	P2.5		IRQ	
3	P3.0		SPI_CS	
4	P5.7			
5	RESET		nRESET	
6	P1.6		SPI_MOSI	
7	P1.7		SPI_MISO	
8	P5.0	INT2		
9	P5.2	TMP_INT		
10	P3.6	OPT_INT		

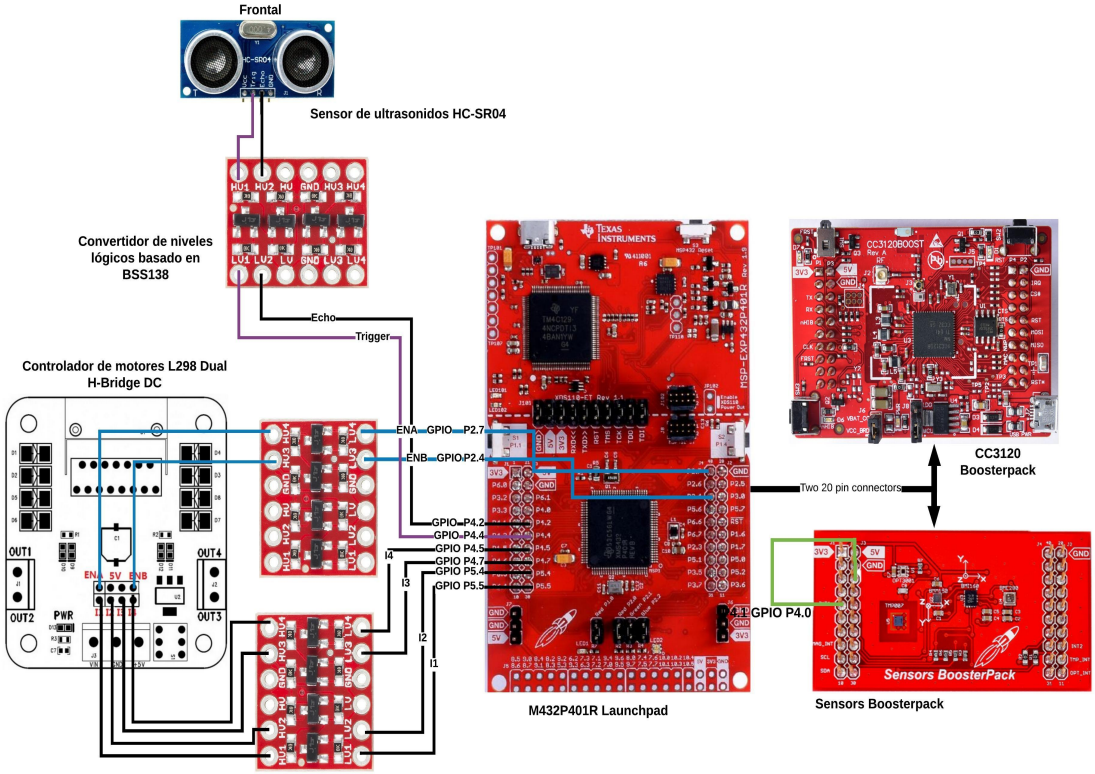
* En puerto 4.1 se recablea en el SENSORXL-BOOSTERPACK al puerto 6.1 (INT1)

Ilustración 25: Tabla de asignación de pins

La última etapa de este proceso ha consistido en generar un esquema visual con todas las conexiones del sistema. Aunque en el tabla de asignación se ha omitido por claridad es necesario incluir adaptadores entre los integrados que funcionan con lógica TTL y los que lo hacen con lógica CMOS. Sin embargo en el prototipo cuando se trata de conectar lógica CMOS

13 En el proyecto software desarrollado en CCS es necesario adaptar la configuración al nuevo mapa de puertos.

con lógica TTL (sensor de ultrasonidos y circuito de control de motores) se ha realizado la conexión directa para facilitar el cableado, esto es debido a que el niveles de tensión CMOS están dentro de los umbrales validos de los tensiones admitidas por TTL.



Los pin utilizados para la conexión entre la tarjeta M432P401R Launchpad y el Sensors Booster Pack se encuentran definidos en la guía slau666a.pdf apartado 2.1.1

Los pin utilizados para la conexión entre la tarjeta M432P401R Launchpad y el CC3120 Booster Pack se encuentran definidos en la guía swru457.pdf apartado 2.3.3

Ilustración 26: Conexión entre los diferentes componentes del sistema

4.2 DESCRIPCIÓN SOFTWARE

El diseño del software del sistema se ha dividido en bloques funcionales que faciliten la modularidad del sistema -tal como se puede apreciar en el diagrama de módulos en la arquitectura software-, cada uno de estos bloques funcionales tiene su correspondencia con las tareas del sistema operativo FreeRTOS.

El criterio utilizado a la hora de implementar el sistema ha sido la de utilizar las funcionalidades proporcionadas por el sistema operativo FreeRTOS y en segundo lugar utilizar las funcionalidades proporcionadas por el Simplelink SDK -y los conectores o plugins correspondiente para operar con cada Boosterpack-. Este decisión hace que el código por un lado se portable entre diferentes plataformas y por otro facilita la codificación al ocultar al desarrollador detalles de bajo nivel de cada plataforma.

A continuación se presenta el diagrama de flujo de inicio del sistema donde básicamente se inician los procesos básicos, se cargan las tareas que implementan cada uno de los módulos de software y por último inicia el planificador de FreeRTOS.

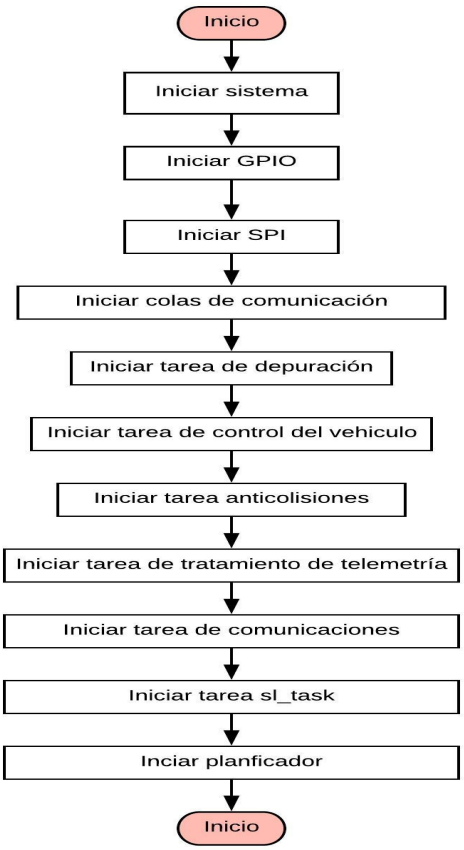


Ilustración 27: Diagrama de flujo del inicio del sistema

4.2.1 Comunicación entre tareas

El principal método utilizado para el paso de mensajes entre tareas ha sido las colas de FreeRTOS, existiendo la singularidad de que algunas de ellas se han implementado en la forma de *mailbox* -el contenido de la cola se sobrescribe cuando se añade un dato y al realizar una lectura no se elimina el contenido de la misma-.

Las colas que se han sido necesarias para cumplir los requerimientos del sistemas son las siguientes:

- **Cola de comandos**. Contiene los comandos recibidos, cada comando esta compuesto por una dirección, la velocidad y la duración mínima de la ejecución del mismo.
- **Cola de modo** (*mailbox*). Mantiene el modo de ejecución activo actualmente para el vehiculo (modo interactivo o modo bucle).
- **Cola de estado del vehículo** (*mailbox*). Mantiene el estado del vehículo (dirección y velocidad).
- **Cola de telemetría** (*mailbox*). Mantiene los últimos datos disponibles procedentes de los sensores y del estado del rover.
- **Cola de depuración**. Contiene mensajes sobre el estado del sistema generados por los diferentes módulos del mismo.

Además se utiliza el mecanismo de notificación entre tareas[14] para la comunicación de la detección de un obstáculo entre el modulo de detección y el modulo de control del vehículo. La notificaciones son un método más rápido y eficiente que las colas en los casos donde no es necesario transmitir información.

4.2.2 Módulo de comunicaciones

La actividades realizadas por este modulo son las relacionadas con la inicialización del procesador de red, la gestión del del enlace de comunicaciones Wi-FI, el mantenimiento del servidor web embebido en el procesador de red y la gestión de las transmisión de datos de red (recepción de comandos y envió de la telemetría). La implementación de este modulo se apoya en el uso de las funcionalidades proporcionadas por el Simplelink Wifi Plug-in SDK.

En el diseño de este módulo se han tomado las siguientes decisiones:

- Establecer el enlace de comunicaciones Wi-Fi en modo estación (ROLE_STA), este hecho se debe a que el prototipo se ha diseñado para funcionar en interiores y lugares que cuenten con varios puntos de acceso inalámbricos que irradien el mismo SSID (Service Set Identifier), de esta forma el vehículo al desplazarse ira cambiando de

14 Se puede encontrar información adicional sobre esta funcionalidad en la documentación Online de FreeRTOS disponible en https://www.freertos.org/RTOS-task-notifications.html.

punto de acceso en función de la zona en la que se encuentre de forma transparente para el usuario.

- Utilizar IPv4 y establecer parámetros de red fijos (dirección IP, gateway y dsn) para facilitar el acceso al panel de control. En el caso de establecer los parámetros de forma dinámica habría que implementar algún sistema para que el cliente conociera la información necesaria para conectarse.
- Incluir mecanismos de seguridad como son el uso de autentificación -mediante usuario y contraseña-, y la utilización de cifrado activando HTTPS en el servidor web. En la actualizada la ciberseguridad es un aspecto a tener en cuenta sobre todo en dispositivos que pueden tener acceso a una red pública como Internet.
- Establecer el mecanismo de Host Token[15] como método para el intercambio de información entre el sistema microcontrolador principal y el procesador de red, de esta forma se libera a la MCU principal de tener que procesar el tráfico HTTP.
- En cuanto al protocolo de comunicaciones entre la MCU el procesador de red se utilizara el protocolo por defecto, que en este caso es SPI (Serial Peripheral Interface).

La funcionalidad de este modulo se ha divido en dos tareas diferenciadas, una dedicada a los eventos síncronos del sistema y otra parte dedicada a los eventos asíncronos. Esto se debe al hecho de que la arquitectura Simplelink impone la utilización de una tarea especifica para manejar los eventos asíncronos.

Funcionalidad síncrona

Las actividades llevadas a cabo por este modulo son las siguientes:

- Inicio del sistema: establecimiento y gestión el enlace WiFi, establecimiento de la configuración IP del sistema, activación y configuración el servidor Web embebido.
- Tareas periódicas de red: comprobaciones sobre el estado de red, envió en intervalos periódicos de la información referente a la telemetría del sistema.

Se puede apreciar el diagrama de flujo de esta tarea que en función del estado de la red -el estado de la red es establecido en función de los eventos asíncronos recibidos por la tarea sl_task- se procede a intentar recuperar el sistema de comunicaciones o bien se opera en condiciones normales -reenviando la telemetría al servidor configurado-.

El envió de telemetría se realiza mediante el protocolo UDP debido a que no se requieren las características adicionales proporcionadas por TCP. La principal razón es que el procesado de los paquetes UDP es más sencillo y rápido, además para este uso no es importante una perdida puntual de esta información o que algún paquete pueda llegar desordenado.

15 El mecanismo Host Token se explica en detalle en el anexo 9.2

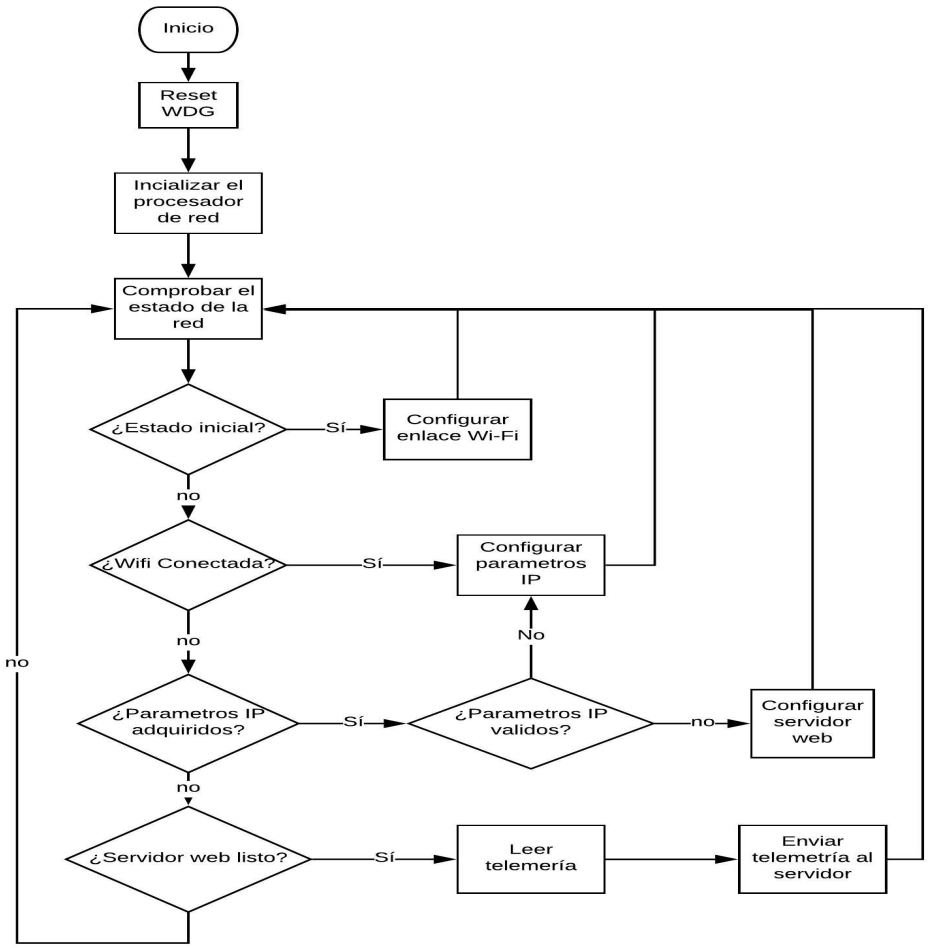


Ilustración 28: Diagrama de flujo

Funcionalidad asíncrona

Esta funcionalidad es responsable de gestionar los eventos de red asíncronos del sistema. La arquitectura Simplelink define una serie de rutinas (funciones callback) cada una de las cuales es responsables de la gestión de un tipo de evento de red y el programador es responsable de codificar estas rutinas de acuerdo a las necesidades del sistema a desarrollar.

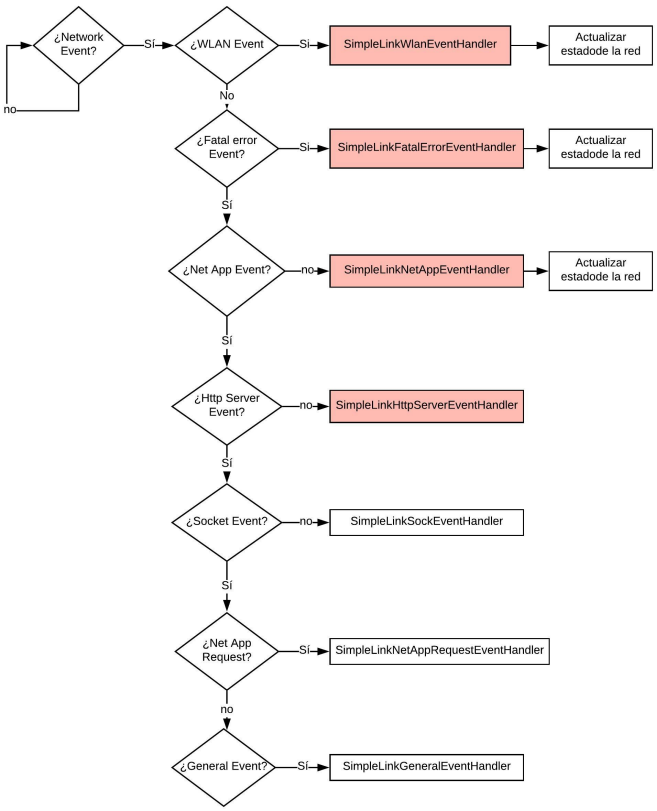


Ilustración 29: Eventos asíncronos generados por la librería Simplelink

En este caso, el diseño de la aplicación ha requerido implementar las rutinas de retorno responsables de los siguientes eventos:

- Eventos relacionados con enlace Wi-Fi (WLAN Events): conexiones y desconexiones del enlace inalámbrico, cambio de modo de funcionamiento en el enlace, … En este caso es conveniente actualizar el estado de la red para que el sistema pueda reaccionar al evento.
- Eventos relacionados con los parámetros de red (Net App Events): adquisición de la dirección IP, cambio de los parámetros DHCP,… Al igual que en el caso anterior es necesario actualizar el estado de red.
- Errores generales del sistema (Fatal Error Events). Al igual que en los casos anteriores es necesario actualizar el estado de red para que se pueda producir un reinicio del procesador de red.
- Eventos relacionados con el tratamiento eventos del servidor HTTP (HTTP Server Events): en concreto esta rutina se encarga de procesar los *host token* recibido por el servidor web a través de peticiones HTTP mediante el método POST o GET. Es la rutina más importante y por ello, a continuación, se detalla su funcionamiento.

La rutina *SimpleLinkHttpServetEventHandler* es la responsable del tratamiento de los *host token* que el servidor de web embedido delega para su tratamiento en la MCU principal. Estos *token* contienen los parámetros de los comandos que se tienen que procesar por el modulo de control del vehículo, como son las indicaciones con el modo de funcionamiento seleccionado, y también las peticiones de información sobre alguno de los parámetros disponibles del estado del vehículo o sus sensores. En la siguiente tabla se definen los *host token* definidos para el sistema, su significado y los posibles valores que pueden contener.

Token	**Método HTTP**	**Valores aceptados**	**Significado**
__SL_P_DIR	Post	FORWARD, BACKWARD, LEFT, RIGHT, STOP, L_FORWARD, L_BACKWARD, R_BACKWARD, L_BACKWARD	Dirección que debe adoptar el vehículo. Forma parte la estructura de un comando.
__SL_P_SPE	Post	Entre 1 y 10.	Velocidad que debe adoptar el vehículo. Forma parte la estructura de un comando.
__SL_P_SPE	Post	Entre 0 y 255.	Tiempo mínimo que debe ejecutarse el comando. Forma parte la estructura comando. 0 indica indefinidamente.
__SL_P_MOD	Post	0 Modo interactivo 1 Modo buchle	Indica el modo de funcionamiento del bucle.
__SL_P_LOP	Post	Entre 0 y 255.	Indica el número de bucles de instrucciones a realizar cuando el modo de operación es en bucle.
__SL_G_DIR	Get	Es una cadena	Indica la dirección que esta siguiendo el vehículo.
__SL_G_SPE	Get	Entre 1 y 10.	Indica la velocidad establecida para el movimiento del vehículo.
__SL_G_TMP	Get	Valor en grados Celsius.	Temperatura procedente del sensor de temperatura infrarrojo TPM007
__SL_G_LIG	Get	Valor de la luminosidad en Lux.	Luminosidad procedente del sensor de luz OPT3001
__SL_G_HUM	Get	Valor de la humedad relativa del aire.	Humedad procedente del sensor Bosch BME280
__SL_G_PRE	Get	Valor de la presión atmosférica en hectopascales (milibares).	Presión atmosférica del sensor del sensor Bosch BME280

Como se puede observar la nomenclatura es los *token* difiere en función de si el *token* se va utilizar en una petición GET (solicitando información) o una petición POST (se quiere establecer el valor de un parámetro).

En el diagrama de flujo se puede observar los pasos seguidos por esta rutina para gestionar la recepción de los *host token* recibos desde el servidor web embebido en el WNP. Donde se puede observar que el tratamiento del *token* difiere según el tipo de método HTTP que lo incluye.

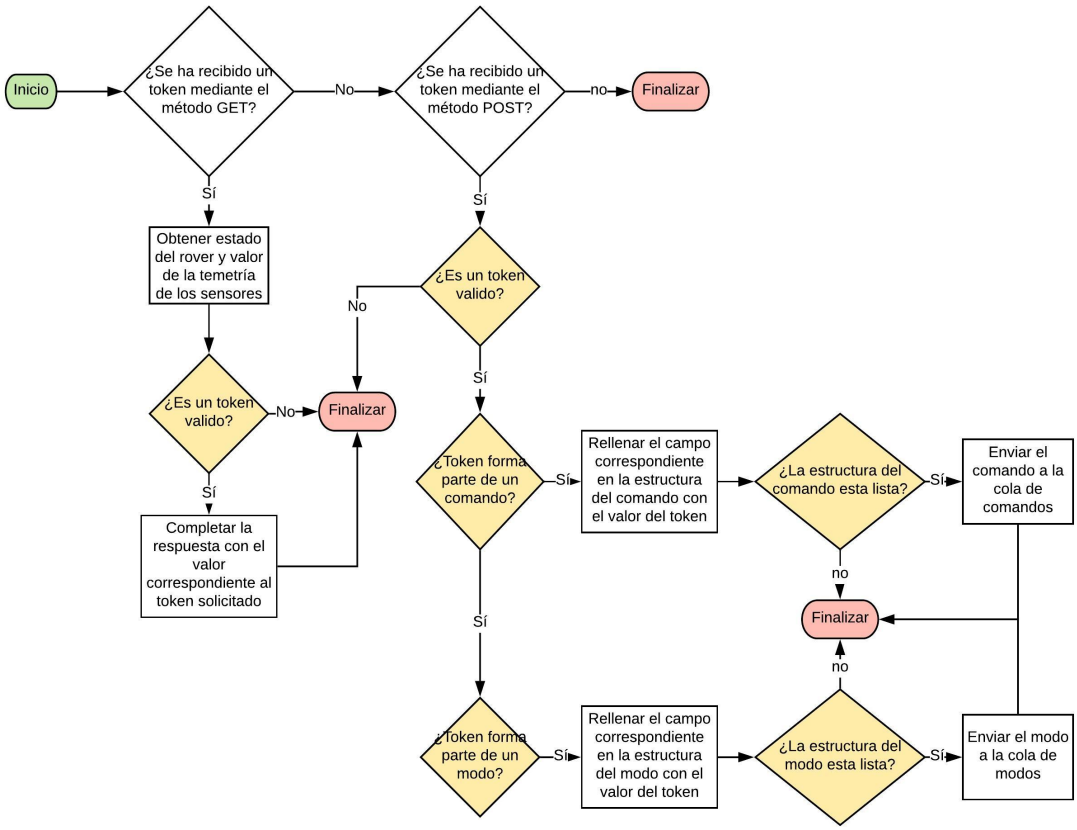


Ilustración 30: Diagrama de flujo de la rutina SimpleLinkHttpServetEventHandler

En importante hacer notar que en este diseño se considera que un “comando” esta listo para su envió a la cola de comandos cuando el valor de cada unos de sus parámetros (dirección, velocidad y tiempo) tiene un valor valido. De forma similar ocurre con el modo de funcionamiento y sus correspondientes parámetros (modo y número de bucles).

El hecho anterior tiene un gran impacto en la implementación de la rutina *SimpleLinkHttpServerEventHandler,* esto es debido a la forma en que las peticiones de este tipo son tratadas dentro de la librería *Simplelink*. La librería *Simplelink* trata de forma independiente cada uno de los parámetros incluidos en la petición HTTP (ya sea mediante el método HTTP POST o GET) cuando son transferidos desde el procesador de red a la MCU.

Por ejemplo la siguiente petición realizada mediante el comando cURL *“curl -d "__SL_P_SPE=3&__SL_P_DIR=STOP&__SL_P_TIM=5" -X POST "https://192.168.8.40/rover/control.html" -v -u Prueba:Prueba -k”* contiene todos los parámetros necesarios para ejecutar un comando (velocidad, duración y tiempo de ejecución), sin embargo al procesarse por la librería *Simplelink* se generan tres llamadas independientes a la rutina *SimpleLinkHttpServetEventHandler* -una por cada tupla de valores enviados en la petición-. Es decir de cara a esta rutina el comportamiento sería igual al de recibir tres peticiones HTTP independientes cada una actuando sobre un solo parámetro de un comando.

4.2.3 Módulo de control del vehículo

La función principal de este modulo es la de ejecutar los comandos recibidos en la cola de comandos en función del modo de funcionamiento establecido en cada momento (modo interactivo o modo bucle). Para ello la primera actividad que se realiza en el módulo es la de verificar el modo de funcionamiento establecido en la cola de modo (definida como un *maibox*) y en función de su valor procesar la recepción de comandos de una forma u otra.

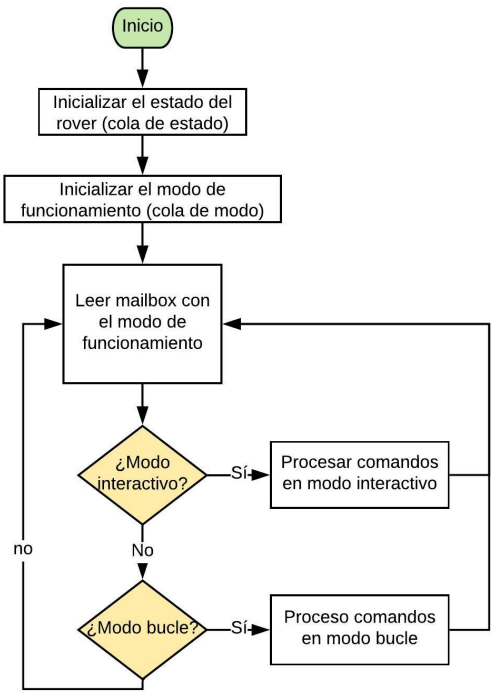


Ilustración 31: Diagrama de flujo del módulo de control

Operación en modo interactivo

En el modo interactivo los comandos son ejecutados por orden de llegada a la cola de comandos. Si un comando no tiene asignado una duración temporal (el campo duración del comando es igual a cero) sera ejecutado hasta la llegada de un nuevo comando y en el caso de tener una duración asignada el comando siguiente en la cola no se ejecutara hasta que el tiempo de ejecución del anterior haya vencido.

Para establecer la duración de un comando se hace uso de las siguientes funcionalidades proporcionadas por el sistema operativo FreeRTOS:

- Un temporizador software[16] cuyo valor de disparo se establece con el mismo valor de la duración del comando.
- Un semáforo binario[17] que bloquea la lectura de nuevos comandos hasta que el tiempo del temporizador anterior haya vencido.
- Las notificaciones entre tareas[18] que permite recibir los avisos sobre la presencia de obstáculos (procedente del modulo de detección de obstáculos).

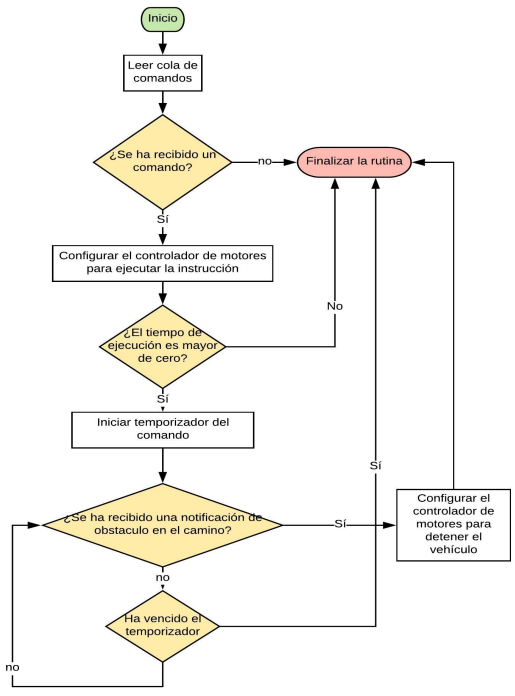


Ilustración 32: Diagrama de flujo del funcionamiento en modo interactivo

16 Una descripción detallada de los temporizadores software puede encontrarse en la documentación Online de freeRTOS accesible en el siguiente enlace https://www.freertos.org/RTOS-software-timer.html
17 La descripción del funcionamiento de los semáforos binarios puede encontrarse en la documentación Online de freeRTOS accesible en el siguiente enlace https://www.freertos.org/Embedded-RTOS-Binary-Semaphores.html
18 La descripción del funcionamientos de las notificaciones entre tareas pueden encontrase en la documentación Online de freeRTOS accesible en el siguiente enlace https://www.freertos.org/RTOS-task-notifications.html

El proceso identificado como “Configurar el controlador de motores” en el diagrama de flujo de esta rutina consistente en establecer los valores necesarios en las salidas de la MCU (puertos GPIO y PWM) para instruir al modulo de control de motores (L298H Dual Bridge) a ejecutar la instrucción[19].

Operación en modo bucle

Cuando el sistema opera en modo bucle el sistema ejecuta un secuencia de comandos el número de veces indicado por el parámetro número de bucles[20] (loops) de la cola del modo de funcionamiento.

El funcionamiento es simple, tal como se observa en el diagrama de flujo, el sistema lee la secuencia de comandos (que finaliza cuando llega un comando cuya duración es cero). A continuación vuelca esta secuencia sobre la cola de comandos y hace una llamada a la misma rutina defina para el proceso interactivo para que proceda a ejecutar los comandos en la cola.

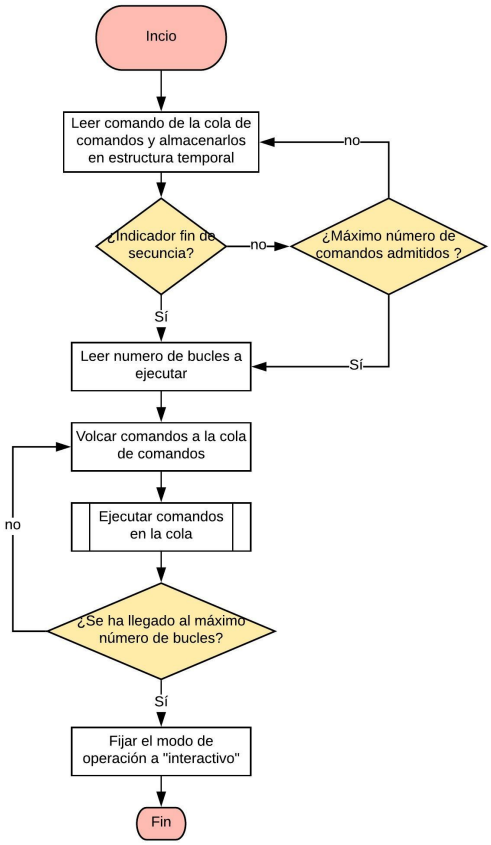


Ilustración 33: Diagrame de flujo de la operación en bucle

19 El funcionamiento de este integrado se detalla en el anexo 9.3 de la memoria del proyecto.
20 Existe un número máximo de comandos admitidos definido mediante la constante MAX_LOOP_COMMANDS.

4.2.4 Módulo de telemetría

Este módulo se encarga de interrogar a los diferentes sensores del sistema . Una vez obtenida la lectura de los sensores pone a disposición del resto de subsistencias los valores obtenidos a través de la cola de telemetría (implementada en modo *mailbox*).

Para el desarrollo de este modulo se hecho uso de las funcionalidades que proporciona el SDK SimpleLink™ Sensor and Actuator Plugin[21] proporcionado por el fabricante TI. La comunicación con los sensores se realiza mediante el protocolo I^2C (Inter-Integrated Circuit).

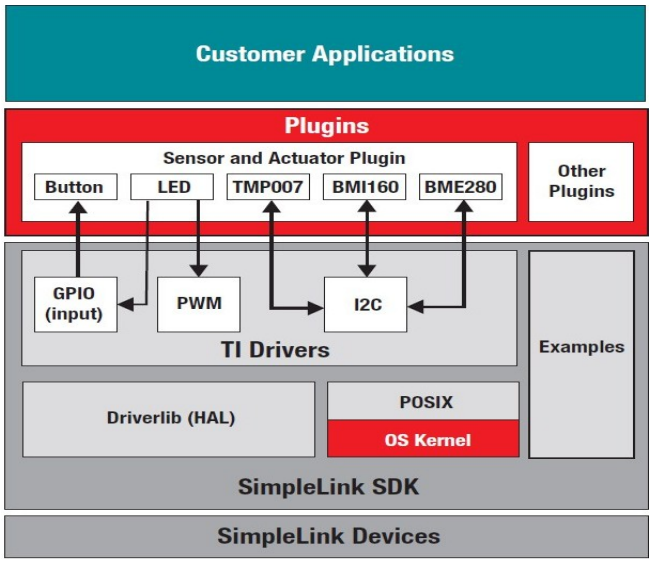


Ilustración 34: Sensor and Actuator Plugin dentro de la arquitectura Simplelink. Fuente: SAIL API Guide

En el diagrama de flujo se puede observar el funcionamiento de este modulo. Indicar que los valores obtenidos de los sensores son tratados como cadenas por simplicidad ya que no se utilizan para realizar ningún tratamiento posterior y las conversiones necesarias se realizan en las propias rutinas de comunicación con los sensores.

21 En el caso de los sensores Bosch BMI160 y Bosch BME280, a su vez el SDK, hace uso de los módulos distribuidos por el fabricante de forma abierta y disponible a través del repositorio https://github.com/BoschSensortec/

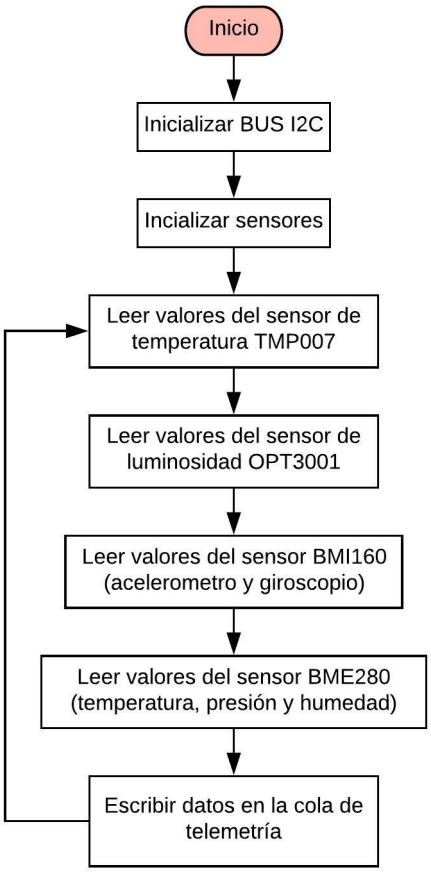


Ilustración 35: Diagrama de flujo del módulo de telemetría

4.2.5 Módulo anti-colisiones

Este modulo se encarga de detectar obstáculos en el sentido de avance del vehículo en el caso de detectar un obstáculo realiza una notificación a la tarea de control para detener el vehículo. Para calcular la distancia a un obstáculo este modulo hace uso de la funcionalidad proporcionada el sensor de ultrasonidos HC-SR04[22]. Aunque el prototipo desarrollado solo cuenta con un sensor frontal, el diseño realizado permite incorporar nuevos sensores, mediante sencillas modificaciones el código, que puedan detectar obstáculos cuando el vehículo se desplaza en otra dirección (hacia atrás, lateralmente, etc.).

La distancia de seguridad que se ha considerado durante la realización del proyecto es de aproximadamente 15 cm, aunque este parámetro es configurable mediante un constante definida en el código del proyecto.

22 En el anexo se detalla la operación del sensor de ultrasonidos HC-SR04.

A partir del diagrama de flujo de este modulo podemos pasar a analizar las principales funcionalidades utilizadas en este modulo.

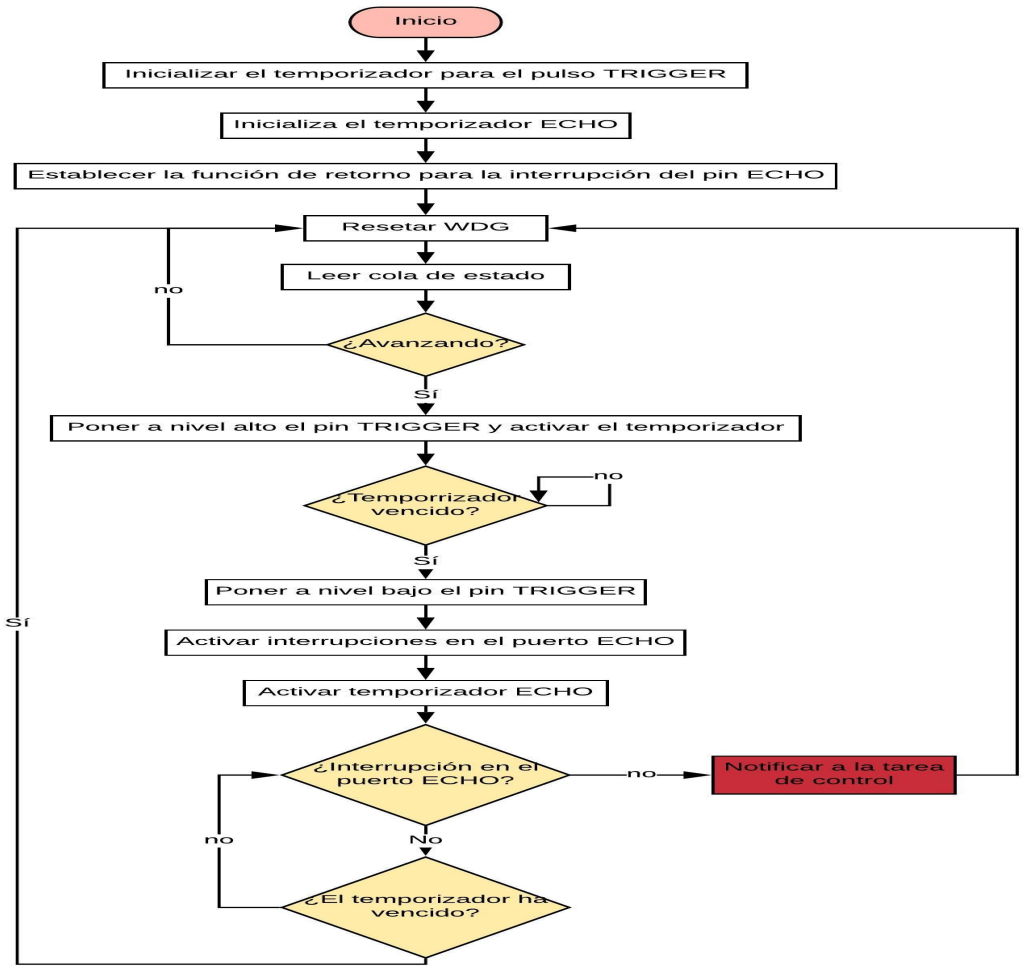


Ilustración 36: Diagrama de flujos del módulo anticolisiones

El diseño del modulo hace uso de dos temporizadores, el primero sirve para establecer la duración del pulso en el puerto TRIGGER -este pulso sirve para lanzar una medida del sensor-. Una vez que este temporizador se libera el semáforo que bloquea la tarea hasta ese momento y se activan las interrupciones en el puerto ECHO, donde simultáneamente se activa el segundo temporizador. Este segundo temporizador sirve como *timeout* y nos indica que no hay un obstáculo dentro de la distancia de seguridad establecida, en el caso de recibirse una una interrupción antes del vencimiento de este *timeout* (provocada por la conmutación del puerto ECHO), esta nos indicaría la presencia de un objeto a una distancia inferior a la de seguridad. Este último hecho provocaría la generación de una notificación a la tarea de control que a su vez que conlleva la detención del vehículo.

Para establecer el valor del temporizador es necesario tener en cuenta que la duración del pulso ECHO es equivalente al tiempo necesario para que la ráfaga de ultrasonidos llegue al

posible objeto y vuelva teniendo en cuenta que la velocidad de propagación del sonido es aproximadamente de 340 m/s.

Debido a que las requerimientos temporales necesarios para implementar los anteriores temporizadores -entre 10 microsegundos y 900 microsegundos- se ha recurrido a los temporizadores propios del microcontrolador en lugar de usar los temporizadores software proporcionados por FreeRTOS. Para la configuración de los mismos se ha usado la interface proporcionada por el SDK a través de los TI Drivers.

4.2.6 Modulo de depuración

La utilidad de este modulo es la de mostrar mensajes sobre el estado del sistema a través del puerto UART cuando este se encuentra conectado a un computador mediante el puerto USB. Se utiliza como mecanismo adicional para la depuración y mejora del software del sistema. Durante el desarrollo del software del sistema se ha utilizado ampliamente este mecanismo junto con el depurador activado desde Code Composer Studio. Este modulo se ha desarrollado sobre la base del ejemplo de este tipo proporcionado por el Simplelink SDK.

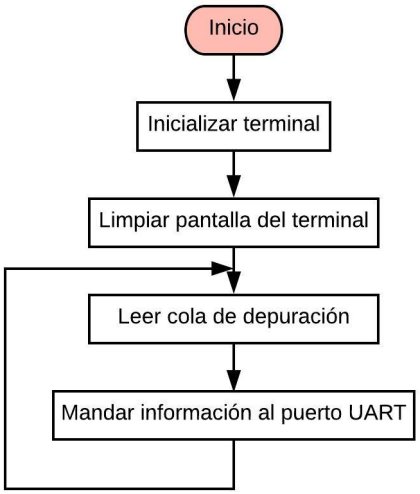


Ilustración 37: Diagrama de flujos del modulo de depuración

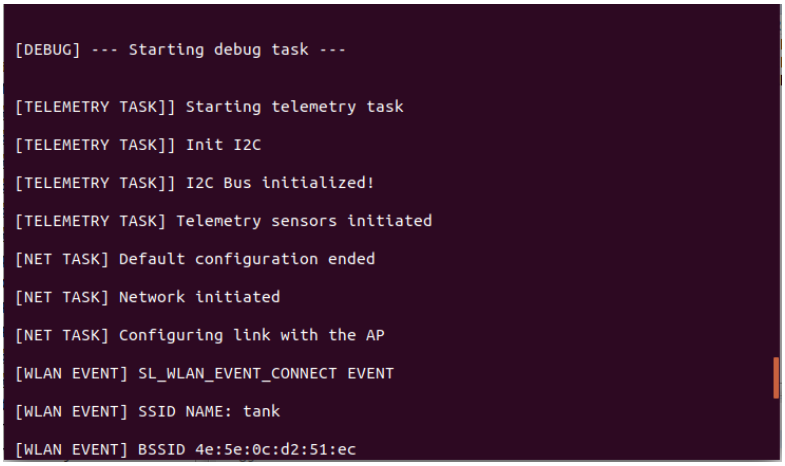


Ilustración 38: Información de depuración obtenida por el puerto UART

4.2.7 Módulo watchdog

El objecto de este módulo es el detectar cuando el sistema se ha congelado debido a un error (principalmente a errores el código del programa) y reiniciarlo. El *watchdog* (WDG) es un temporizador que incorpora el microcontrolador MSP432 que al vencer produce un reinicio del sistema. El resto de las tareas del sistema tiene la obligación de reiniciar el contador del WDG en cada ciclo, ya que este paso significa que el sistema sigue funcionando de acuerdo a las especificaciones.

4.3 VIABILIDAD TÉCNICA

En un primer momento se ha realizado una análisis teórico sobre las funcionalidades y capacidades de los productos incluidos en el kit de la asignatura para la realización del proyecto y de esta forma garantizar a priori la viabilidad de la realización del mismo. Durante este análisis se han analizado las hojas de características y documentación proporcionadas por los fabricantes, complementando dicha información recursos disponibles en Internet. A continuación se ha obtenido un listado de componentes hardware adicionales necesarios para llevar a cabo el proyecto.

Una vez seleccionada la plataforma y el hardware necesario para la realización del proyecto se ha procedido a realizar un prueba de concepto de cada una de las funcionalidades basándose en la documentación y ejemplos proporcionados por los diversos fabricantes. Esto ha permitido identificar los principales problemas y buscar alternativas en el caso de no poder llegar a la resolución de los mismos. Una vez probada la funcionalidad de cada uno de los módulos por separado se ha procedido a la integración de las partes.

El último paso para garantizar la viabilidad técnica han sido las pruebas de campo sobre el prototipo montando con todos sus componentes.

5 Valoración Económica

5.1.1 Costes de desarrollo del proyecto

A continuación se detallan mediante las siguientes tablas los costes asociados a la realización del proyecto, tanto en equipamiento como en horas de trabajo. Para el calculo de los costes del equipamiento se han utilizado los precios de adquisición a través de minoristas (principalmente a través de tiendas virtuales). Los precios de las horas de trabajo en cada perfil son aproximados debido a alta variabilidad en el mercado y que según el proveedor pueden o no incluir el coste del IVA.

Componente	Precio	Unidades	Total
MSP432P401R LaunchPad Development Kit	12 €	1	12€
Sensors BoosterPack Plug-In Module for LaunchPad Development Kit	23.25 €	1	23.25€
CC3120 Wireless Network Processor BoosterPack™ Plug-In Module	28 €	1	28€
Advanced Emulation BoosterPack for SimpleLink Wi-Fi CC3100 BoosterPack plug-in module	20 €	1	20€
Chasis robot Devastator	66 €	1	66€
Circuito L298H Dual Bridge	8 €	1	8€
Integrado AMS1117	2 €	1	2€
Sensor de ultrasonidos HC-SR04	3 €	1	3€
Conversor de niveles lógicos	2.90€	3	8,7€
Place de inserción y cables de inserción	5€	1	5€
Coste total de los componentes			175.5€

Tabla 2: Coste de los componentes utilizados

Role	Horas de dedicación	Coste hora	Total
Analista funcional	50 horas	80€	4000€
Analista programador	250 horas	50€	12500€
Costes asociados al desarrollo			16500€

Tabla 3: Costes de desarrollo

5.1.2 Costes de industrialización

El primer paso seria desarrollar la versión industrial de la electrónica del proyecto. Para ello se han identificado las siguientes acciones:

- Adaptación del prototipo basado en placas de desarrollo a un diseño de circuito impreso. Este proceso se simplifica por el hecho de que los diseños de las dispositivos Simplelink MCU están disponibles de forma abierta.
- Como el producto esta orientado de una forma modular, es necesario habilitar un sistema de conectorizado y encapsulado entre los diferentes módulos que sea lo suficiente robusto para el uso diario
- Utilizar un plataforma robot industrial a una escala mayor.
- Incorporar un sistema de alimentación a baterías de alta capacidad que dote al sistema de la suficiente autonomía.

Una vez se ha industrializado la plataforma, a continuación, habría que llevar el proceso de certificación para obtener el sello CE que permite su comercialización en Europa y donde el fabricante garantiza que cumple los requisitos de la directiva 2014/30/UE. Este es un proceso complejo que usualmente se gestiona a través de una empresa externa.

A los costes de adaptación necesarios habría que añadir los costes necesarios para publicitar y dar a conocer el producto al tipo de usuario al que va dirigido.

6 Conclusiones

Desde un punto de vista general podría decirse que se han cumplido los objetivos propuesto al inicio del proyecto. Se ha conseguido tener un vehículo telecontrolado con capacidad para seguir las instrucciones tanto en tiempo real como ejecutando bucles, con capacidad para evitar obstáculos en la dirección de avance y también se ha conseguido obtener y visualizar la información de los diferentes sensores con los que cuenta el rover.

6.1 AUTOEVALUACIÓN

Como se ha comentado con anterioridad podría decirse que se han cumplido los objetivos propuestos pero el autor no esta de acuerdo con los resultado en cuanto a la calidad y presentación de los resultados obtenidos.

Se han subestimado los problemas relacionados con la integración de diferentes elementos hardware: conexiones incorrectas, el deterioro de componentes decido a una incorrecta manipulación y conexión, etc. La falta de agilidad a la hora de identificar este tipo de problemas ha penalizado otras actividades como la mejora del código y la documentación de la memoria.

La planificación planteada al inicio del proyecto ha sido demasiado optimista en cuanto al número de tareas y el tiempo asignado a la mismas. Este hecho ha supuesto que en las etapas finales de la realización del proyecto el desarrollo del mismo no se haya ajustado a la planificación.

Desde el punto de vista de la realización personal el trabajo ha sido muy satisfactorio. Se ha trabajado en un área muy diferente al área en el que al autor desarrolla su trabajo habitual, lo que ha supuesto un reto. Se han adquirido nuevos conocimientos y soltura trabajando con este tipo de sistemas.

6.2 LINEAS DE TRABAJO FUTURO

De cara el futuro se pueden englobarse en las siguientes lineas de trabajo:

- Mejorar la robustez y modularidad del código de programa.
- Integración del sistema con sistemas de localización -tanto en interiores como en el exterior-,
- Explotar la información de los sensores que permita mejorar el producto y ofrecer al usuario una mejor experiencia.
- Incorporar nuevas capacidades y sensores.
- Desarrollar un modelo de negocio que permita la comercialización del producto obtenido.

7 Glosario De Términos Y Abreviaturas

API: Abreviatura de Application programming interface. Es un conjunto de funciones y rutinas ofrecidos por una biblioteca como capa de abstracción para el acceso a la funcionalidades ofrecidas por esta.

Host: la documentación de TI hace referencia al host cuando quiere diferenciar la MCU de propósito general de los microcontroladores que forman parte de los procesadores de red.

Host token: es un identificador especial insertado en código HTML que tiene un significado especial al ser procesado por el servidor web de los microcontroladores Simplelink MCU.

HTTP: abreviatura de Hypertext Transfer Protocol.

IDE: Abreviatura de Integrated development environment. En un entorno de programación grafico que cuenta con todos las funcionalidades para desarrollar código (compilar, editor, depurador,..).

I^2C: Es un bus y protocolo de comunicaciones en serie utilizados para la comunicación entre circuitos electrónicos. Se suele utilizar para comunicar la unidad microcontrolador con los periféricos dentro de un sistema basado en microcontrolador.

MCU: Abreviatura de Microcontroler Unit. Es el sistema compuesto por el propio microcontrolador junto con circuitería adicional como el sistema de alimentación, el sistema de depuración y de generación de reloj. En esta memoria se usa el termino para referirse al MSP432P4014 Launchpad.

PWM: Abreviatura de Pulse Width Modulation. PWM es una modulación digital que funciona modificando el ciclo de trabajo de una señal digital y de esta forma enviar información o controlar la energía sobre una carga (por ejemplo para poder controlar velocidad de rotación de un motor).

SDK: Abreviatura de Software Developer Kit. Conjunto de herramientas de programación que facilitan la programación para un determinado entorno.

Simplelink: Simplelink MCU es una plataforma de Texas Instruments compuesta por microcontroladores, placas accesorias y herramientas de desarrollo. A lo largo del texto se hace referencia como Simplelink a cualquiera de los componentes anteriores (en función del contexto).

SPI: Es un bus y protocolo orientando a las comunicaciones serie entre circuitos electrónicos al igual que I^2C.

TI: Abreviación de Texas Instruments. Texas Instruments en un fabricante norteamericano de semiconductores.

WNP: Abreviación de wireless network procesor. Suele tratarse de un sistema microcontrolador orientado unicamente a la gestión de las comunicaciones inlambricas.

WDG: Abreviación de watchdog. El watchdog o perro guardián es un temporizador que incorporan la mayoría de los microcontroladores cuya finalidad es la de reiniciar el sistema en el caso de que este quede bloqueado.

8 Bibliografía

(1) Texas Instruments (2018), "SimpleLink Microcontroller LaunchPad Development Kit. User's guide", disponible en línea http://www.ti.com/lit/ug/slau597f/slau597f.pdf y accedido en Abril de 2018.

(2) Texas Instruments (2018), “CC3120 SimpleLink Wi-FiBoosterPack Plug-In Module and IoT Solution. User’s Guide”. Disponible en línea http://www.ti.com/lit/ug/swru457/swru457.pdf y accedido en Abril de 2018.

(3) Texas Instruments (2017), "MSP432P4xx SimpleLink™ Microcontrollers Technical Reference. Manual", disponible en http://www.ti.com/lit/ug/slau356h/slau356h.pdf y accedido en Abril de 2018.

(4) Texas Instruments (2017), "CC3120, CC3220 SimpleLink™ Wi-Fi® and Internet of Things Network Processor. Programmer's Guide", Literature Number: SWRU455, disponible en http://www.ti.com/lit/ug/swru455e/swru455e.pdf y accedido en Abril de 2018.

(5) Richard Barry (2016), "Mastering the FreeRTOS™ Real Time Kernel, A Hands-On Tutorial Guide", disponible en https://www.freertos.org/Documentation/161204_Mastering_the_FreeRTOS_Real_Time_Kernel-A_Hands-On_Tutorial_Guide.pdf y accedido en Abril de 2018.

(6) Amazon (2017), "The FreeRTOS™ Reference Manual, API Functions and Configuration Options", disponible en https://www.freertos.org/Documentation/FreeRTOS_Reference_Manual_V10.0.0.pdf y accedido en Abril de 2018.

8.1 Recursos web

(1) Embeddded. Portal web destinado a desarrolladores de sistemas embebidos. Accesible a través del siguiente enlace https://www.embedded.com/

(2) TI Cloud Tools. Portal web de Texas Instruments destinado a desarrolladores de sistemas embebidos del fabricante, contiene desde herramientas para el desarrollo hasta foros de soporte al usuario. Disponible a través del siguiente enlace https://dev.ti.com/

9 Anexos

9.1 CONTROL DEL VEHÍCULO MEDIANTE LA UTILIDAD DE LINEA DE COMANDOS CURL

En este apartado se hace una demostración de los diferentes posibilidades de control del vehículo utilizando la utilidad cURL.

Control del vehículo en modo interactivo

- Ejemplo 1. Instruir al vehículo para que avance de forma indefinida hacia delante.

```
curl --basic -u Prueba:Prueba https://roverweb/rover/control.html -X POST -d "__SL_P_SPE=1&__SL_P_DIR=FORWARD&__SL_P_TIM=0" -k
```

la descripción de los parámetros de la petición sería:

__SL_P_SPE=1 esta parámetro (o host token) sirve para indicar la velocidad (en este caso el valor es 1).

__SL_P_DIR=FORWARD este parámetro establece la dirección de avance del vehiculo.

__SL_P_TIM=0 aquí se establece el tiempo de ejecución para el comando (en segundos), en este caso concreto el valor 0 indica que el comando se ejecute de forma indefinida.

Control del vehículo en modo bucle

- Ejemplo 2. Instruir al vehículo para que repita en dos ocasiones un bucle formado por dos comandos (avanzar y retroceder durante dos segundos)

```
curl --basic -u Prueba:Prueba https://roverweb/rover/control.html -X POST -d "__SL_P_MOD=LOOP&__SL_P_LOP=2" -k

curl --basic -u Prueba:Prueba https://roverweb/rover/control.html -X POST -d "__SL_P_SPE=1&__SL_P_DIR=FORWARD&__SL_P_TIM=2" -k

curl --basic -u Prueba:Prueba https://roverweb/rover/control.html -X POST -d "__SL_P_SPE=1&__SL_P_DIR=BACKWARD&__SL_P_TIM=2" -k

curl --basic -u Prueba:Prueba https://roverweb/rover/control.html -X POST -d "__SL_P_SPE=1&__SL_P_DIR=BACKWARD&__SL_P_TIM=0" -k
```

El primer comando sirve para cambiar el modo de funcionamiento (modo bucle) y ejecutar el numero de bucles indicados (2). A continuación se introducen los dos comandos a ejecutar en cada uno de los bucles. Finalmente el último comando con que cuenta con la peculiaridad de tener una duración igual a cero sirve para indicar el fin de la secuencia de comandos a reproducir.

9.2 Mecanismo "Host Token" de los procesadores de red SimpleLink CC3120/CC3200 de TI

El mecanismo Host Token es una funcionalidad disponible en los procesadores de red TI CC3120 y CC3200 que permite generar contenido web dinámico evitando que la MCU principal del sistema tenga que realizar por completo todo el procesado HTML. Los token consisten en una serie de identificadores especiales incluido en las peticiones web -ya sean peticiones HTTP por el método GET o por el método POST- que al llegar al servidor web embebido son enviados desde el procesador de red hacia la MCU principal para que sean tratados. Estas peticiones son manejadas por la tarea SimpleLink Wi-Fi (sl_Task()) mediante la rutina sl_HTTPServerCallBack(). Las ventajas que aporta esta funcionalidad es que permiten generar contenido HTTP dinámico con una mínima intervención del MCU del sistema.

Cuando el Host Token forma parte de una petición HTTP GET, la MCU responde al procesador de red con el valor asociado al token y el servidor web embebido en procesador de red sustituirá el token por su valor. En cambio, cuando la petición forma parte de una petición HTTP POST el token contiene además un valor un valor, esta tupla (token/valor) es la que la procesador de red enviá a la MCU. Es decir mediante peticiones GET queremos obtener datos de la MCU y mediante peticiones POST enviar datos a la MCU.

Para que el procesador de red pueda identificar correctamente los *host token* estos tienen que incluir un prefijo que en el caso de las peticiones por el método GET sera de la forma "__SL_G_XXX" y para el caso de las peticiones por el método POST sera del tipo "__SL_P_XXX" donde los dígitos identificados como XXX corresponden a un valor libre elegido por el usuario (siempre y cuando no coincida con un valor de un *token* del sistema).

El gráfico adjunto ilustra el funcionamiento de los Host Token, en este caso concreto para procesar una petición HTTP con el método GET que incluye dos tokens (token 1 y token2)

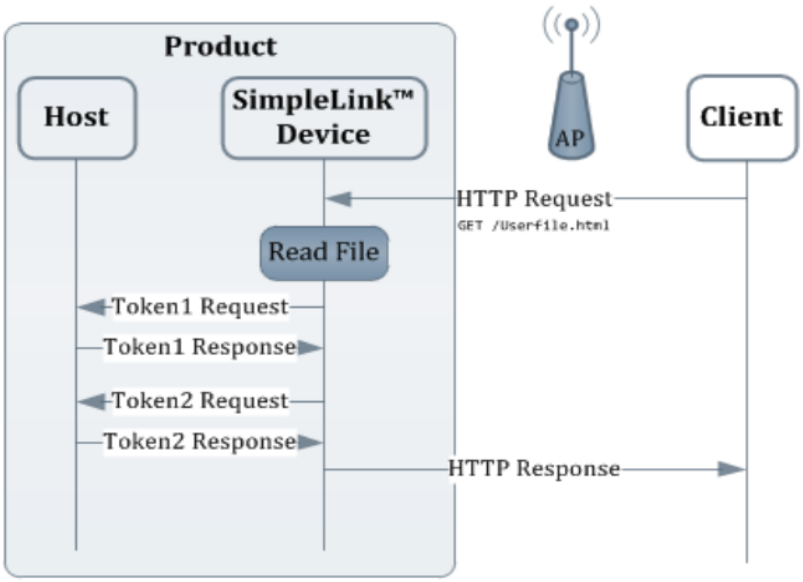


Ilustración 39: Funcionamiento de los host token. Fuente: “CC3120, CC3220 Simplelink Wi-Fi and Internet of Things Network Procesor. Progammer’s Guide”.

9.3 Funcionamiento del módulo de control de potencia basado en el integrado L298H Dual Bridge

La funcionalidad de este circuito, basado en el integrado L298N, es la de permitir a un circuito digital controlar una carga inductiva como puede ser un motor de corriente continua o un motor paso a paso. Adicionalmente este circuito cuenta con un regulador de tensión capaz de ofrecer una tensión de 5 voltios de corriente continua siempre y cuando la alimentación con la alimentemos el circuito sea inferior a 12 voltios.

La forma de controlar la velocidad de rotación de los motores es mediante pulsos modulados mediante PWM (pulse wave modulation), para ello el integrado cuenta con dos entradas digitales denominada ENA y ENB -una para cada motor-.

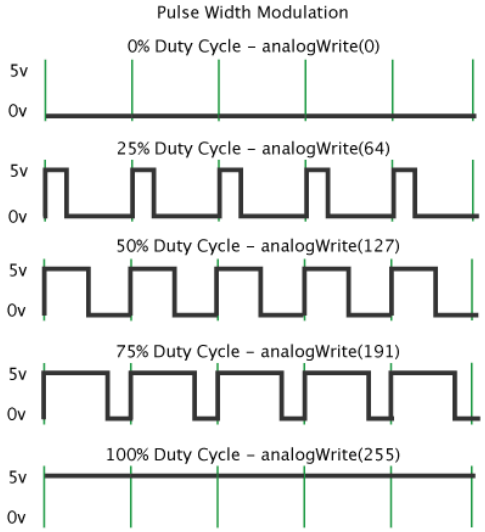


Ilustración 40: Señal PWM con diferentes ciclos de trabajo. Fuente: https://www.arduino.cc/

El circuito cuenta con cuatro entradas digitales denominadas I_1, I_2, I_3 e I_4 que permiten controlar la activación de cada motor y el sentido de rotación. Las entradas I_1 e I_2 permiten actuar sobre el primer motor mientras que las entrada I_3 e I_4 actuá sobre el otro.

En la siguiente tabla se resume el funcionamiento del circuito para una de las salidas (A) cuando queremos controlar el funcionamiento de montor de corriente continua:

ENA	I_1	I_2	ESTADO MOTOR A
PWM	0	0	Motor parado
PWM	0	1	Motor rotando en un sentido al porcentaje de potencia indicado por el ciclo de trabajo de la señal ENA
PWM	1	0	Motor rotando en un sentido contrario al porcentaje de potencia indicado por el ciclo de trabajo de la señal ENA
PWM	1	1	Motor parado
0	x	X	Motor parado

9.4 Funcionamiento del Sensor de ultrasonidos HC-SR04

El sensor de ultrasonidos HC-SR04 es un circuito cuya funcionalidad principal es la de medir la distancia a la que se encuentra un objecto, siempre que este se encuentre dentro de el rango de funcionamiento -entre 2 y 400 cm-. El principio de funcionamiento se basa en el tiempo que tarda una ráfaga de ultrasonidos(operando a una frecuencia de 40kHz) en llegar al obstáculo y volver.

La forma de funcionamiento del sensor consiste en generar un pulso de 10 microsegundos en el puerto TRIGGER del dispositivo. Este pulso de disparo activa el circuito, que lanza en ese momento un ráfaga de ultrasonidos compuesta por 8 pulsos a una frecuencia de 40kHz. Seguidamente el sensor pone la salida del pin Echo a nivel alto durante el tiempo la ráfaga de ultrasonidos tarda en llegar al objeto y volver de nuevo al sensor reflejada. En la figura adjunta se puede observar de forma grafica el funcionamiento del sensor.

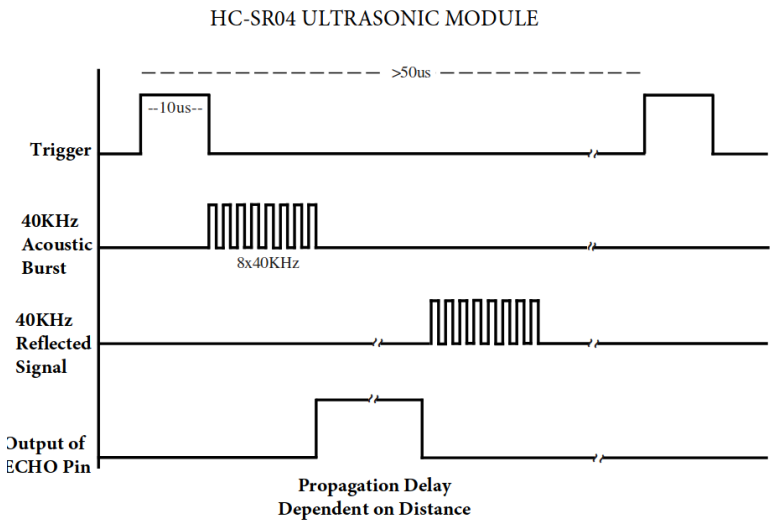


Ilustración 41: Funcionamiento del sensor. Fuente: datasheet del fabricante.

Para calcular la distancia al objeto simplemente hay que tener en cuenta que un pulso de ultrasonidos viaja a una velocidad aproximada de 340m/s (velocidad del sonido) y que el periodo de tiempo en que la señal echo esta activa corresponde al tiempo de ida y vuelta hasta el objecto.