Robot de Vigilancia Remota

Ingeniería Técnica de Telecomunicaciones, especialidad Telemática

Estudiante
José Manuel Rodríguez Rama

Consultor
Jordi Bécares Ferrés

17/01/2015

A todas las personas que han seguido a mi lado
a pesar de que muchas veces no tuviese tiempo para ellos.

Especialmente a mi pareja Patricia y a mi Familia, por apoyarme,
respetar mis decisiones, y darme ánimos en los momentos difíciles.

Sin ellos no habría sido posible.

Gracias.

AGRADECIMIENTOS

Quisiera agradecer a los buenos profesores, compañeros, y tutores que me han acompañado durante este tiempo y a la UOC por permitirme la flexibilidad necesaria para realizar esta carrera a distancia mientras trabajaba.

Agradecer especialmente al consultor de esta área de TFC, Jordi Bécares Ferrés, por su ayuda durante la realización de este proyecto, sus consejos e indicaciones, y sobre todo por su comprensión y apoyo durante el desarrollo del mismo.

RESUMEN

El presente proyecto forma parte del Trabajo Fin de Carrera, concretamente del área *Sistemas Empotrados*, que se había seleccionado como primera opción entre las distintas posibilidades, debido a mi interés por aprender acerca de este tipo de sistemas, cada vez más utilizados hoy en día.

Mediante este proyecto se pretende poner en práctica lo aprendido en diversas asignaturas como “Fundamentos tecnológicos” y “Fundamentos de computadores” y afianzar conocimientos de otras como “Programación orientada a objetos”, complementándolo además con mi experiencia profesional en electrónica de seguridad y sistemas de vigilancia.

En el proyecto se desarrollará, mediante el correspondiente *sistema embebido*, un robot de vigilancia, concretamente una base dotada de tracción mecánica controlada mediante la *mota LPC1769* dotada de un procesador *ARM* y el sistema operativo de tiempo real *FreeRTOS.*

La base mencionada sustentará una *cámara IP*, cuya *alimentación* se controlará igualmente con el sistema empotrado, y funcionará de manera completamente inalámbrica a través de *Wi-Fi,* concretamente utilizando un módulo de comunicación inalámbrico *Wifly*. Para el control de los motores que dotarán de tracción mecánica al robot se utilizarán dos *placas de potencia* del fabricante “Keyes” basadas en el *driver de motores L298N*.

El control del robot se realizará mediante una GUI basada en un *script Python* especialmente diseñada a tal efecto, que permitirá al usuario interactuar con la mota y enviar órdenes al robot, así como visualizar la cámara, todo ello a tiempo real.

El resultado del proyecto será un robot *teledirigido* con una gran capacidad de ampliación al basarse en un sistema embebido, destinado a la vigilancia no personal de recintos.

Índice de contenidos

Índice de figuras

1. INTRODUCCIÓN

1.1. Justificación

La creciente demanda en seguridad electrónica, como alternativa económica a los vigilantes de seguridad ha provocado que en los últimos años los sistemas de tele-vigilancia evolucionen y se conviertan en un elemento indispensable en prácticamente cualquier establecimiento, llegando incluso al ámbito doméstico, debido a la bajada de precios experimentada en los últimos años.

Hay dos tipos principales de sistemas de vigilancia, aquellos que son estáticos, y que graban un punto durante un período de tiempo determinado, y los móviles, principalmente cámaras Domo motorizadas, que si bien están fijadas a un sitio determinado, pueden girar sobre sí mismas y ampliar la imagen, ya sea de manera óptica o digital, lo que les confiere la posibilidad de vigilar grandes superficies desde un mismo punto.

Últimamente han ganado popularidad un tercer tipo de elementos de vigilancia, los drones, que se mueven ya sea por tierra, aire o agua, y realizan la misma función que las cámaras motorizadas, pero sin la limitación de permanecer ancladas a un punto determinado.

Tanto las cámaras motorizadas como los drones pueden moverse de forma autónoma o bien guiados por una persona a través de los elementos de control correspondientes, ya sea mediante un Joystick de control, o un software.

Tras haber trabajado varios años en el sector de sistemas de seguridad he llegado a la conclusión de que, si bien existen productos similares, las empresas no los conocen o no los suelen ofertar, y es un área que no parece estar muy evolucionada ni enfocada al público general, aun cuando serían la solución perfecta para las necesidades de vigilancia y seguridad que se plantean en muchos casos.

El presente proyecto pretende sentar una base en el desarrollo de este tipo de sistemas de vigilancia, debido a que el uso de un sistema embebido permite una ampliación y configuración bajo demanda, algo muy demandado en este sector. Hay clientes que necesitan poder grabar, detectar movimiento, incendios, fugas de gas, etc. todo esto en un entorno en el que probablemente no haya una conexión fija de internet, o alimentación eléctrica.

Si bien el sistema presentado en este proyecto es sencillo, y sólo se basa en la vigilancia por video, y con control externo de un usuario, cabe destacar que el uso de un sistema embebido permite la conexión de infinidad de sensores y elementos de vigilancia, manteniendo un consumo eléctrico y de datos reducido. Es un sistema portátil que se puede reaprovechar fácilmente en diferentes instalaciones, y que se puede configurar para cubrir exactamente las necesidades que se necesiten en cada caso.

1.2. Descripción del proyecto

El presente proyecto pretende ser un aporte a la investigación de uno de los más novedosos sistemas de vigilancia, los drones, mediante la realización de un robot de vigilancia, controlado por un sistema empotrado, y que utilizará una cámara IP para el visionado remoto de video a tiempo real. La función principal del robot será la vigilancia de recintos a través de la cámara instalada en el robot.

La recepción del video a tiempo real y el control del robot se harán a través de una aplicación que reciba la imagen mediante un Streaming RTSP, y que se comunique de forma inalámbrica con el sistema embebido a través de Socket TCP y Wi-Fi. El sistema se encargará de controlar el movimiento del robot, y de mantener la cámara apagada hasta que reciba la orden de activarse, con el fin de reducir el consumo, consiguiendo con todo esto un sistema inalámbrico y robusto, que puede ser controlado de forma remota a demanda.

1.3. Objetivos del TFC

Los objetivos principales del proyecto son:

- Dotar al sistema de tracción mecánica vía motores eléctricos.
- Dotar al sistema de una comunicación inalámbrica para conectarse con la aplicación de PC para recibir comandos (vía el módulo Wifly).
- Implementar un sistema de encendido de la cámara con el fin de reducir el consumo.
- Integrar una cámara comercial IP para el visionado de imágenes de forma remota.
- Integrar las funcionalidades de control de movimiento y activación de la cámara de forma remota.

1.4. Enfoque y método seguido

El proyecto se aborda de manera gradual, distinguiendo dos grandes etapas que se exponen a continuación.

Formación e investigación:

Formación y estudio de posibilidades del sistema: esta etapa, a pesar de no parecer la más compleja, es crítica, debido a que la formación sustenta el posterior desarrollo del sistema, y sin los conocimientos adecuados no se puede obtener un buen resultado.

Investigación de materiales necesarios y adecuados, para la posterior compra de materiales: de igual manera cualquier error en la compra y elección de materiales puede suponer retrasos, e incluso si los elementos no se seleccionan adecuadamente, podríamos, en el peor de los casos, provocar una avería irreversible en la mota, con graves consecuencias para el proyecto.

Desarrollo de productos:

Desarrollo de los módulos y librerías nuevos (motores, relé, pwm), reaprovechamiento de los existentes (wifly, printf): Se desarrollarán módulos específicos para el control de los motores a través de las placas de potencia. Para la comunicación se reaprovecharán, pudiendo realizar ciertas ampliaciones, los módulos creados en la fase de formación.

Creación de la aplicación para el sistema embebido. Diseño e implementación de la aplicación para PC: se realizará una aplicación basada en FreeRTOS que utilice los módulos creados para el control del sistema, dicha aplicación actuará como servidor, y se comunicará con otra aplicación cliente para PC que se conectará a la primera, para enviar comandos de control.

Diseño de conexionado y montaje de Hardware: realización del montaje final del hardware en la base del robot, cableado y conexionado definitivo de los elementos.

1.5. Planificación del proyecto

Viabilidad en función del tiempo

El tiempo destinado al proyecto será unas 150 horas, para lo que se pretenden dedicar 2 horas de media al día una vez finalizada la etapa de aprendizaje marcada por la asignatura, durante un total de 75 días. La planificación se realiza de forma lineal, si bien por cuestiones laborales se espera tener más tiempo en las últimas semanas de Diciembre, cuando se compensarán los retrasos que se produzcan hasta ese momento.

Se considera que el tiempo es suficiente para realizar la totalidad del proyecto, aunque las posibles fluctuaciones debidas a imprevistos podrán afectar a la última etapa, correspondiente a la investigación de la cámara OV7670, en la cual se avanzará hasta dónde alcance el tiempo disponible.

Fases

El proyecto se ha llevado a cabo en tres fases bien diferenciadas:

FASE 1, etapa de aprendizaje:

Esta primera etapa incluye la realización de las 4 primeras PECS, que sirven para familiarizarse con el sistema FreeRTOS, y los diversos elementos de Hardware. Durante esta etapa se deciden los elementos que van a conformar el sistema y se idea el funcionamiento del software.

FASE 2, etapa principal:

Esta es la etapa más importante, en la cual se realiza desde la compra de los materiales, hasta el montaje del Hardware y la implementación del código y Software utilizado en el proyecto. Esta etapa pretende alcanzar los objetivos principales del proyecto, dejando las mejoras y ampliaciones para etapas posteriores.

FASE 3, investigación, mejora y documentación:

Esta fase incluye la revisión y mejora del proyecto, estudio de alternativas, etc. Al mismo tiempo se realizará toda la documentación y la memoria del proyecto, añadiendo los resultados de esta fase.

Cambios en la planificación:

Como vemos a continuación en los cronogramas, durante la realización del proyecto ha habido un retraso importante (marcado en rojo en el cronograma de la planificación final) provocado fundamentalmente por falta de tiempo debida a motivos laborales, que no han permitido dedicar las horas suficientes al estudio en las últimas dos PEC, y esto ha provocado a su vez que la tarea de diseño del driver Wifly sufriese un retraso considerable.

Dicho retraso ha sido compensado, en parte, por el trabajo extra realizado a finales de Diciembre y principios de enero en el cual se ha podido recuperar parte del tiempo (marcado en verde), pero a pesar de esto se ha afectado a la planificación total del proyecto, al retrasar la finalización de la fase 2 en ocho días, y ha provocado que no se pudiese realizar la investigación del módulo de cámara OV7670 hasta el nivel que se quería (marcado en negro), forzando además un retraso de dos días en la fecha prevista inicialmente para la entrega de la memoria.

Destacar también que ha habido un retraso en la entrega de materiales (marcado en ámbar), y además estos se han ido pidiendo gradualmente, para evitar posibles problemas de aduana, y sobrecostes en los envíos, si bien este retraso no ha afectado en el desarrollo del proyecto ya que a pesar del retraso siempre se dispuso del hardware necesario en cada tarea.

Cronograma de la planificación inicial

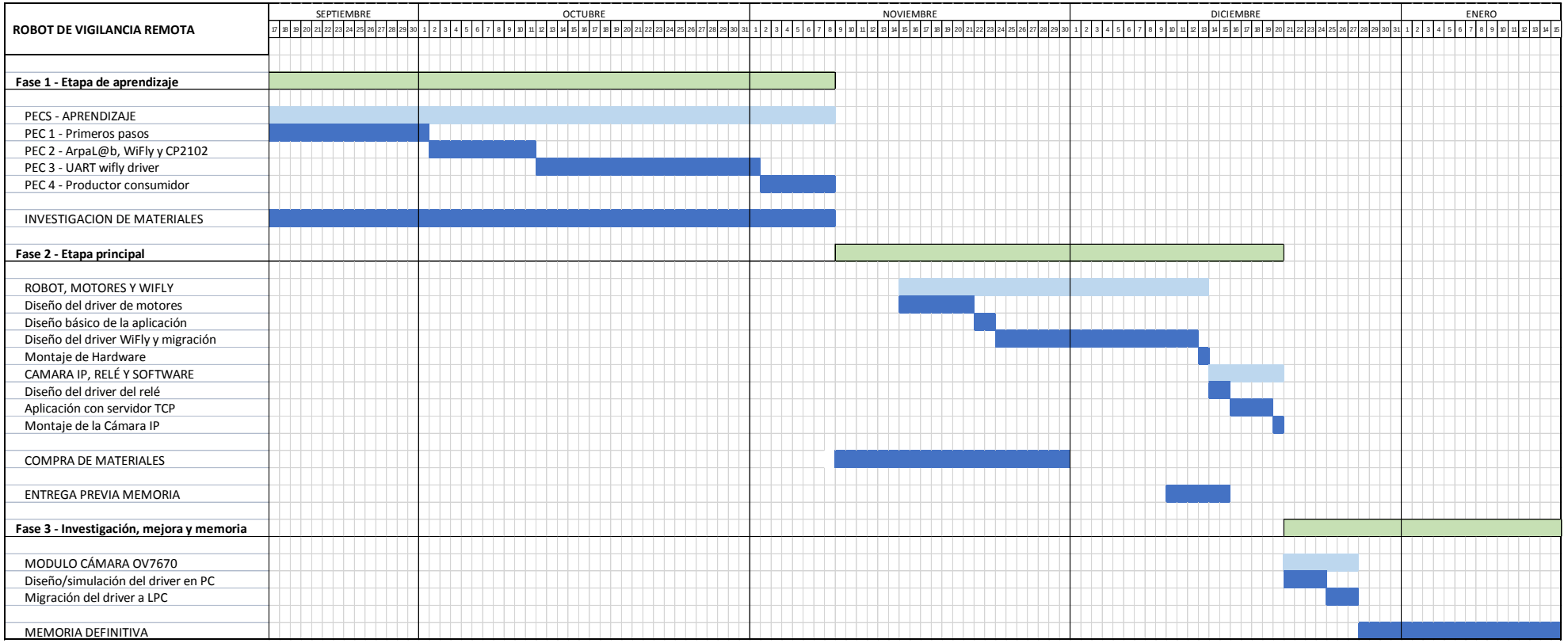


Figura 1 : Cronograma de la planificación inicial

Cronograma de la planificación final

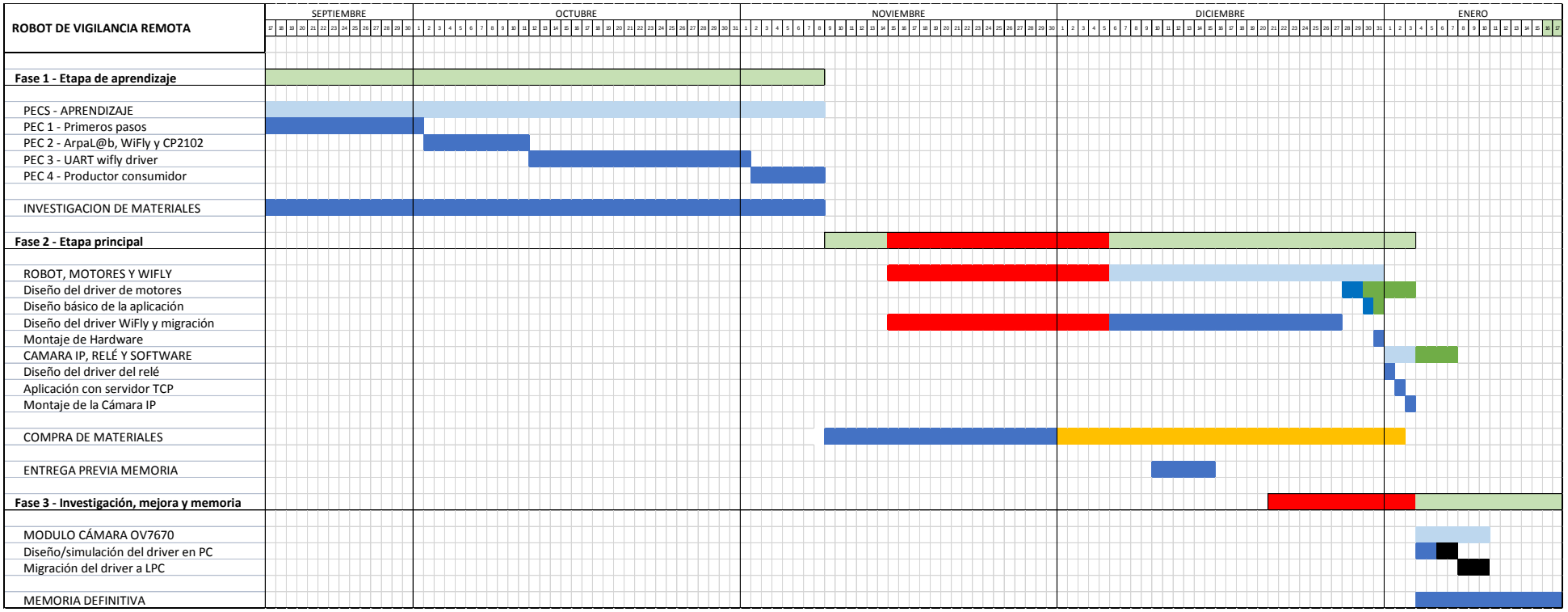


Figura 2: Cronograma de la planificación final

1.6. Recursos empleados

Recursos de Software:

Como veremos más adelante, como recursos de software se utilizará Python para programar la GUI de la aplicación en el PC así como el cliente TCP. Para la programación de la placa se utilizará el software LPCXpresso para realizar los módulos y el programa escrito en lenguaje C, y utilizando FreeRTOS.

Recursos de Hardware:

En cuanto al Hardware los materiales utilizados serán la mota LPC1769, el módulo Wifly, una cámara IP Wi-Fi y las placas de potencia para el control de los motores, así como el correspondiente sistema de alimentación para el robot y la cámara basado en baterías, y diferentes componentes electrónicos para el ensamblaje.

1.7. Productos obtenidos

Robot de Vigilancia

Este es el producto principal que se obtiene del proyecto una vez finalizado. Consiste, como se ha mencionado en la descripción del presente capítulo, en la realización de un robot, controlado por un sistema empotrado, y que utilizará una cámara IP para el visionado remoto de video a tiempo real. La función principal del robot será la vigilancia de recintos a través de la cámara instalada en el robot. La visualización del video y el control del robot se realizarán mediante el software expuesto en el siguiente punto.

Software de Vigilancia

Se trata de un software sencillo basado en un script escrito usando el lenguaje interpretado Python, que consta de una interfaz gráfica con botones para el control del sistema, un cliente TCP para la comunicación con el módulo Wifly, y un lanzador del programa de captura de video VLC para visualizar la cámara por Streaming.

Aplicación para la mota y Driver de motores

La aplicación cargada en la mota es el tercer producto. Dicha aplicación está compuesta por diversos módulos entre los que se destacan los de acceso a la UART, al módulo Wifly, y particularmente aquellos destinados al control de motores, ya que se busca que estos sean reaprovechados para otros proyectos que realicen el control de motores mediante placas de potencia que no funcionen a través de la UART.

1.8. Descripción de los otros capítulos de la memoria

A continuación describiremos brevemente el contenido de los demás capítulos de la memoria.

Antecedentes

En este capítulo se presenta la evolución y estado actual las tecnologías utilizadas para obtener nuestro producto, y se realiza un estudio de mercado.

Descripción funcional

En este capítulo se explica el diseño realizado, las decisiones tomadas y se describen los componentes generales del sistema.

Descripción detallada

En este capítulo se describen detalladamente los diferentes componentes del sistema, cómo se ha hecho, diagramas, etc.

Viabilidad técnica

En este capítulo se demuestra que el sistema es viable de implementar, y cuáles son sus puntos fuertes y débiles.

Valoración económica

En este capítulo se valora económicamente el proyecto, se realiza un presupuesto, y se explican los diferentes costes asociados.

Conclusiones

En este capítulo se estudia el grado de consecución de los objetivos, se proponen mejoras y se realiza una Autoevaluación.

2. ANTECEDENTES

Un sistema embebido o empotrado es un sistema de computación diseñado para realizar una o algunas pocas funciones dedicadas, frecuentemente en un sistema de computación en tiempo real. Al contrario de lo que ocurre con los ordenadores de propósito general (como por ejemplo un ordenador personal) que están diseñados para cubrir un amplio rango de necesidades, los sistemas embebidos se diseñan para cubrir necesidades específicas

En la parte central se encuentra el microprocesador, microcontrolador, DSP, etc. Es decir, la CPU o unidad que aporta capacidad de cómputo al sistema, pudiendo incluir memoria interna o externa, un micro con arquitectura específica según requisitos.

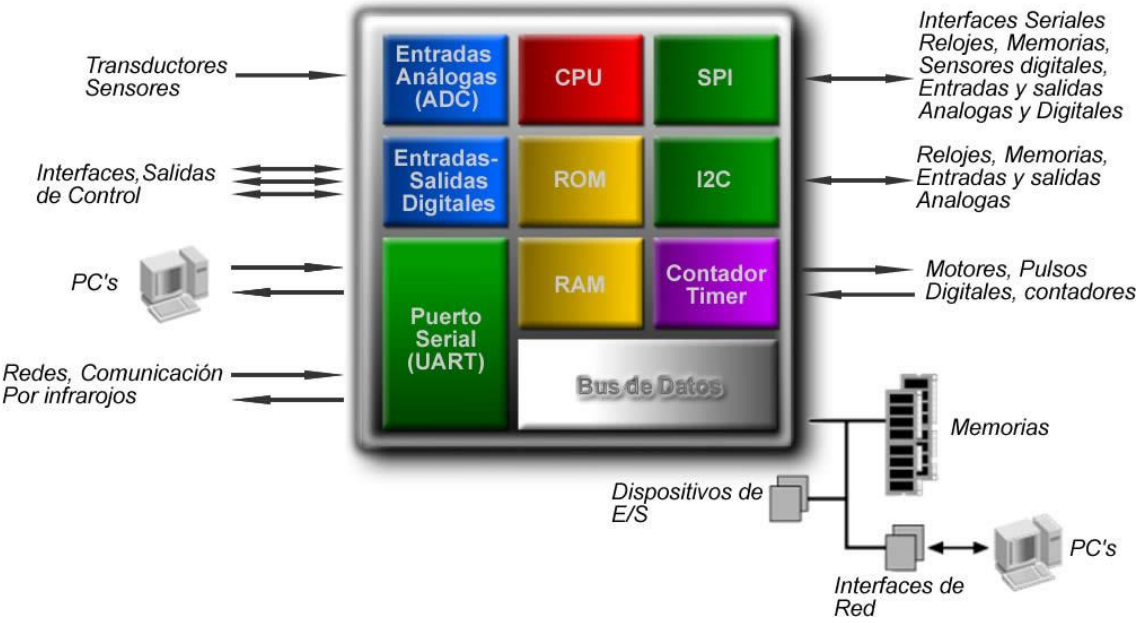


Figura 3: Diagrama de bloques de un sistema embebido

La comunicación adquiere gran importancia en los sistemas embebidos. Lo normal es que el sistema pueda comunicarse mediante interfaces estándar de cable o inalámbricas. Así normalmente incorporará puertos de comunicaciones del tipo RS-232, RS-485, SPI, I²C, CAN, USB, IP, Wi-Fi, GSM, GPRS, etc.

2.1. Estado del arte

Desde la aparición del primer sistema embebido reconocido, utilizado para el sistema de guía de las misiones Apolo hacia la luna (Advanced Guiding Computer), diseñado por el MIT Instrumentation Laboratory, hasta el día de hoy, ha habido un avance considerable. El equipo tenía 2048 palabras de memoria RAM y 36.864 de memoria ROM con una longitud de palabra de 16 bits. Si lo comparamos con un procesador 8088 de Intel contenido en un PC IBM de 1981, ya tenía 8 veces más memoria que el AGC. Un procesador actual de Smartphone de 1000 MHz y 512 MB de RAM, tiene 100.000 veces más RAM que el AGC, que además corría a alrededor de 1 MHz, lo que nos da una idea de la evolución de este tipo de sistemas desde su aparición hasta la actualidad.

Como se expuso en el apartado anterior, en la parte central de un sistema embebido se encuentra un microprocesador o microcontrolador. La siguiente tabla muestra los principales fabricantes y micros disponibles actualmente, entre los cuales se encuentra el contenido en la mota LPC1769 usada en el proyecto (Cortex-M3 de NXP):

Empresa	8 bits	16 bits	32 bits
Atmel	AVR (mega y tiny),89Sxxxx familia similar 8051		SAM7 (ARM7TDMI), SAM3 (ARM Cortex-M3), SAM9 (ARM926), AVR32
Freescale (antes Motorola)	68HC05, 68HC08,68HC11, HCS08	68HC12, 68HCS12, 68HCSX12,68HC16	683xx, PowerPC, ColdFire
Holtek	HT8		
Intel	MCS-48 (familia 8048) MCS51 (familia 8051) 8xC251	MCS96, MXS296	x
National Semiconductor	COP8	x	x
Microchip	Familia 10f2xx Familia 12Cxx Familia 12Fxx, 16Cxx y 16Fxx 18Cxx y 18Fxx	PIC24F, PIC24H y dsPIC30FXX,dsPIC33F con motor dsp integrado	PIC32
NXP Semiconductors (antes Philips)	80C51	XA	Cortex-M3, Cortex-M0, ARM7, ARM9
Renesas (antes Hitachi, Mitsubishi y NEC)	78K, H8	H8S, 78K0R, R8C, R32C/M32C/M16C	RX, V850, SuperH, SH-Mobile, H8SX
STMicroelectronics	ST 62, ST 7		STM32 (ARM7)
Texas Instruments	TMS370	MSP430	C2000, Cortex-M3 (ARM), TMS570 (ARM)
Zilog	Z8, Z86E02		

Figura 4: Tabla de microcontroladores actuales

Este tipo de micros también es uno de los componentes más importantes del otro elemento en el que se basa este proyecto, las cámaras IP.

Una cámara IP está constituida básicamente por dos partes, la "cámara" (lente, sensor de imagen, etc.) y un sistema embebido encargado de dotar a la misma del resto de funcionalidades que la distinguen de una cámara analógica, como son la conectividad Ethernet y Wi-Fi, la memoria interna, entradas y salidas de alarma, análisis de video, etc.

Actualmente existe una gama muy amplia de cámaras IP, que van desde las cámaras más profesionales con precios de hasta 9000 euros y resoluciones de hasta 29 Megapíxel, que cubren un área equivalente a 95 cámaras analógicas tradicionales, hasta cámaras económicas por debajo de los 100 euros, que a pesar de no tener resoluciones tan altas, son las más populares y para la mayoría de aplicaciones cumplen perfectamente su función.

A continuación presentamos una comparativa de cuatro cámaras IP de exterior con carcasa tipo bullet de AXIS, uno de los fabricantes con mayor reputación actualmente.

AXIS P1405-E
Select accessories »

AXIS P1425-E
Select accessories »

AXIS P1427-E
Select accessories »

AXIS P1428-E
Select accessories »

Cámara				
Sensor de imagen:	CMOS	CMOS	CMOS	CMOS
Tamaño del sensor de imagen:	1/2.8"	1/2.8"	1/3.2"	1/2.5"
Sensor megapíxel:	✓	✓	✓	✓
Modelo de sensor en megapíxeles:	2	2	5	8.6
Exploración progresiva:	✓	✓	✓	✓
Alcance amplio y dinámico:	✓	✓	✓	✓
Iluminación mínima/ sensibilidad a la luz (color):	0.35 lux	0.25 lux	0.35 lux	1.4 lux
Iluminación mínima/ sensibilidad a la luz (B/N):	0.07 lux	0.05 lux	0.07 lux	0.3 lux
Vídeo				
Funcionalidad día y noche:	✓	✓	✓	✓
Resolución máxima de vídeo:	1920x1080	1920x1080	2592x1944	3840x2160
N.º máx. de imágenes por segundo:	25/30 fps with power line frequency 50/60 Hz	25/30 fps with power line frequency 50/60 Hz	25/30 fps with power line frequency 50/60 Hz	25/30 fps with power line frequency 50/60 Hz

Figura 5: Comparativa cámaras IP Axis

2.2. Estudio de mercado

En cuanto al mercado actual de los sistemas empotrados, según la firma de análisis IDC en su último informe “Sistemas Embebidos e Inteligentes 2014-2019”, se prevé un gran crecimiento en la implantación de estos sistemas, y señala que el mercado para sistemas inteligentes –definido por microprocesadores, conectividad, sistemas operativos e interfaces de usuario de alta disponibilidad, excluyendo PC, teléfonos, servidores y tablets – crecerá en 800 millones de unidades en tan solo un año, superando los 2000 millones, con una facturación total de más de un billón de dólares.

Entre los sectores que experimentarán mayor crecimiento están los sistemas de “driver & fuel management” en el ámbito del transporte, los “smartwearables” y los sistemas inteligentes de iluminación en lo que se refiere a consumo, la “patología digital” y la “metrología virtual” en salud, y sistemas específicos para los procesos productivos de la industria.

Con lo que respecta a los robots, desde pequeños aspiradores, hasta drones helicóptero para fotografía aérea, nos muestran que la robótica está entrando con fuerza en un mercado más general durante los últimos años, y no tardará en ocurrir lo mismo en el sector de la vigilancia, si bien como ya se ha explicado en la justificación del capítulo 1, es un producto que no está muy afianzado todavía en este sector, debido a que para el público general este tipo de drones parece algo poco accesible y que solo utilizan las fuerzas de seguridad o empresas especializadas.

Realmente es un producto que se puede fabricar por un coste bajo en comparación con otros sistemas de seguridad, y que, basándolo en un sistema embebido, tendría una escalabilidad superior a la mayoría de sistemas comerciales.

Este tipo de sistemas ha evolucionado considerablemente en los últimos años, y se ha abaratado, pasando de ser un elemento de vigilancia exclusivo, a ser algo accesible a prácticamente cualquier sector.

Existen actualmente varios robots de vigilancia, de los cuales nombraremos 3, empezando por uno de los más básicos (SPY-C de Logicom), y terminando por el más avanzado (Knightscope K5).

Logicom SPY-C

Es un robot para vigilancia controlado por Wi-Fi y una aplicación móvil creado por la compañía Logicom. Su precio oscila entre los 100 y los 130 euros. Captura video a una calidad reducida (320x240), tiene una autonomía de 1 hora y 15 minutos en funcionamiento, y 30 metros de alcance en interiores.

Figura 6: Robot Logicom SPY-C

Rotundus GroundBot

Es un robot para vigilancia controlado por Wi-Fi creado por la compañía Rotundus. Alberga dos cámaras domo que capturan video de alta calidad en formato MPEG-4, y diversos sensores. Sus características físicas innovadoras lo hacen capaz de patrullar por cualquier terreno. Tiene una autonomía de entre 8 y 16 horas en funcionamiento, GPS, acelerómetro y giroscopio.

Figura 7: Robot Rotundus GroundBot

Knightscope K5

Es sin duda el robot más avanzado de los expuestos, creado por la compañía Knightscope. Dispone de 4 cámaras de 3 Megapíxeles, GPS, radar, lector de matrículas, diversos sensores de temperatura, humedad, C02, acústicos, etc. Es un robot autónomo e inteligente que pretende ser un sustituto real a los vigilantes de seguridad. Se ofrece en formato de alquiler de servicios con un coste de unos 2500 euros al mes.

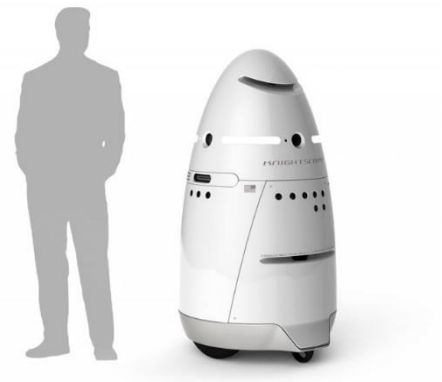


Figura 8: Robot Knightscope K5

3. DESCRIPCIÓN FUNCIONAL

En el presente capítulo expondremos la funcionalidad del sistema creado. Se pretende crear una imagen general presentando los diversos elementos que componen el sistema y su interacción, para después profundizar en ellos en el capítulo 4.

3.1. Robot de Vigilancia Remota

El sistema se compone de dos partes principales, la primera es el robot de vigilancia, conformado por la carcasa del robot, la electrónica con el sistema embebido, los motores y la cámara IP, y por otro lado tenemos la aplicación para PC escrita en Python, con una GUI compuesta por un joystick virtual, una consola para los mensajes de debug, y el lanzador para la recepción del Streaming de video mediante VLC.

Ambas partes se encuentran interconectadas mediante una red Wi-Fi local, si bien no es necesario que éstas se encuentren en el mismo emplazamiento, y la interconexión se podría realizar a través de internet de forma transparente.

Diagrama de bloques del sistema

En la siguiente figura se observa el diagrama general del sistema desarrollado:

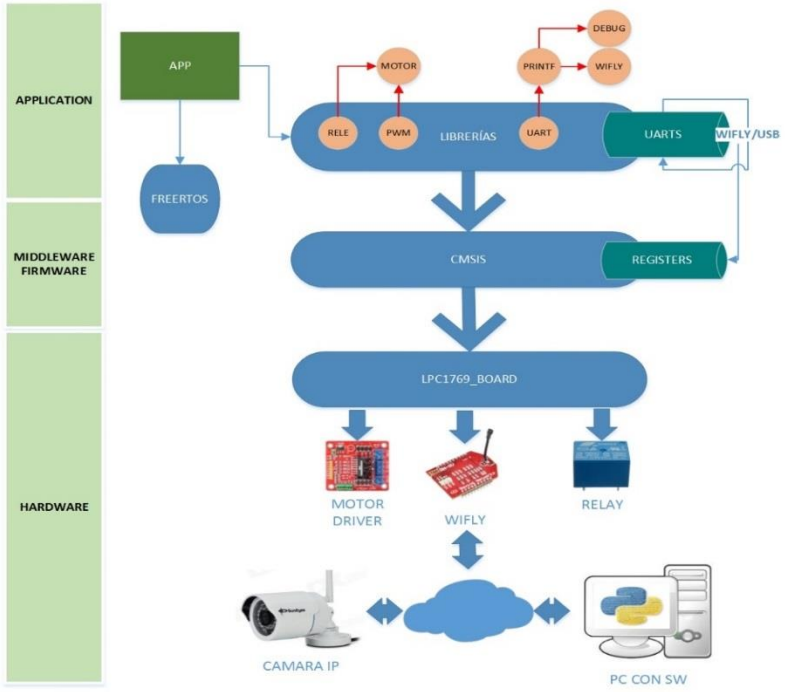


Figura 9: Diagrama general del sistema

Como podemos observar en la figura 9, la mota LPC1769 contiene una aplicación que es la base del sistema y se encarga de controlar los distintos dispositivos conectados a ella.

Por otro lado tenemos la cámara IP, cuya alimentación se controla también a través de un relé conectado a la mota, y el PC que contiene el software necesario para recibir la imagen, y enviar los comandos a la LPC1769.

Cómo es la red

El robot se conecta a una red Wi-Fi local mediante el módulo Wifly. El PC con el software de control, se puede conectar, o bien a la misma red, o bien en un emplazamiento remoto. El robot hace de servidor y el PC se conecta a este como cliente para enviar comandos a la mota.

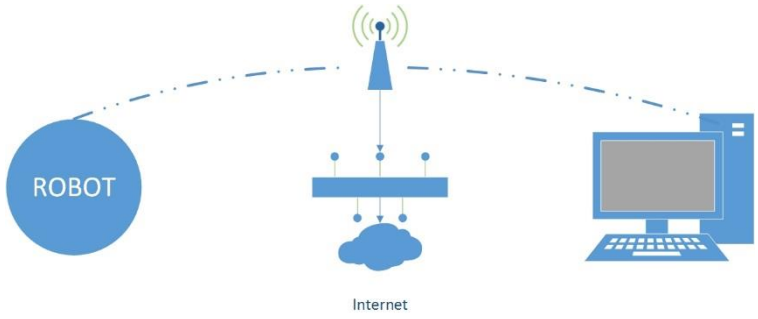


Figura 10: Esquema general de la red

Cómo interactúan los diferentes objetos en el sistema

En la siguiente figura se observa el diagrama de interacción de los distintos elementos, y a continuación una pequeña explicación de cada uno de ellos:

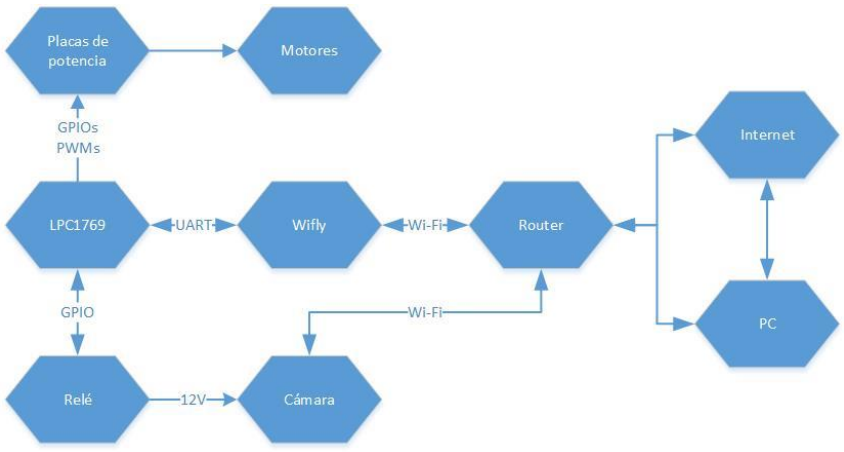


Figura 11: Interacción de los elementos del sistema

Los elementos principales mostrados en la figura anterior son los siguientes:

- **Mota LPC1769:** Es el elemento principal del sistema, almacena la aplicación que hace que interactúen todos los elementos.

- **Placas de potencia:** Se utilizan dos placas de potencia, por razones de consumo, para controlar los motores, el control se realizará mediante las salidas de PWM y GPIO de la mota.

- **Cámara IP:** Este es el otro elemento esencial del sistema. Se ha optado por una cámara IP como primera opción por sus características superiores en comparación con otras opciones. La cámara es independiente y no utiliza la LPC1769 para enviar la imagen.

- **Relé:** Controlará la alimentación a la cámara, reduciendo el consumo al encenderla sólo cuando esté en uso el robot.

- **Baterías:** El sistema de alimentación está conformado por 2 baterías de Li-Ion para la cámara y la mota, y dos de Ni-MH para cada uno de los conjuntos placa de potencia - motor. No se muestran en la figura anterior ya que son elementos que no interactúan.

- **PC:** Ordenador con el software de control, que se conecta ya sea a través de internet o de forma local a través de un router Wi-Fi.

3.2. Software de Vigilancia

Para el control de la mota desde el PC y la recepción de video se ha desarrollado un pequeño software basado en scripts Python.

El programa dispone de una GUI con un joystick virtual para el envío de los comandos de movimiento del robot y botones para conectarse a la mota a través del módulo Wifly, mediante socket TCP.

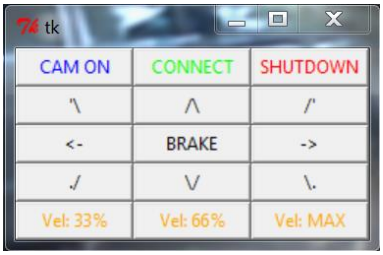


Figura 12: GUI Software de vigilancia

También permite encender la cámara, lo cual arrancará a su vez el programa VLC con el Streaming de video correspondiente, cambiar la velocidad de movimiento del robot y apagar el sistema (desconexión del módulo Wifly y apagado remoto de la cámara).

Diagrama de bloques

A continuación se muestra un diagrama de bloques y ejecución de la aplicación de control, y de los principales comandos:

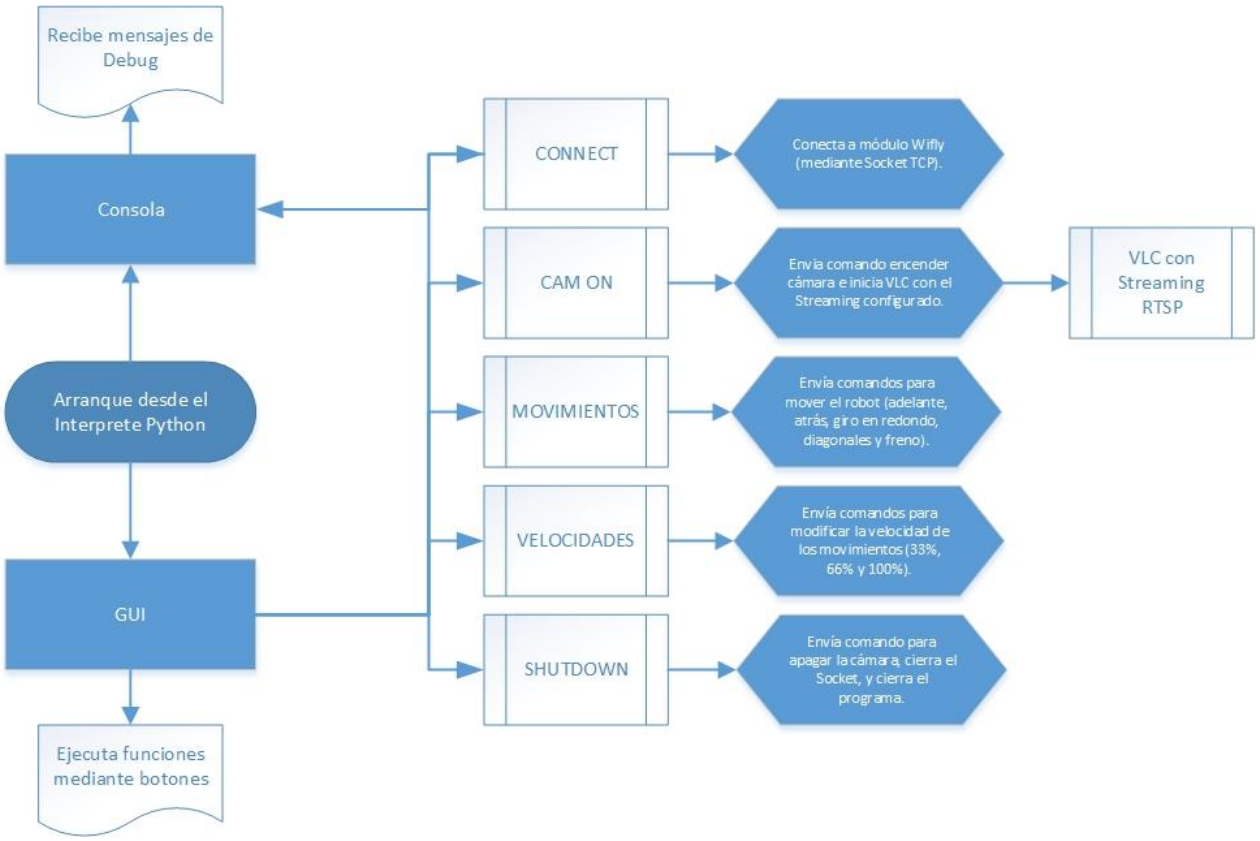


Figura 13: Diagrama de bloques de la aplicación de vigilancia

Partes

El programa se basa en 3 partes principales, la consola, la GUI, y el VLC.

Consola:

Se utiliza para recibir los mensajes de confirmación enviados por la mota la procesar los comandos enviados, así como para ver cualquier de mensaje de error enviado por el script.

GUI:

Es una sencilla interfaz gráfica basada en botones. Se ha creado con Python y Tkinter, y permite enviar todos los comandos necesarios para el control del sistema.

VLC:

Para la recepción de video, uno de los botones de la GUI lanza la aplicación de captura de video VLC con el Stream de la cámara pre configurado, que se reproduce automáticamente al abrir.

3.3. Aplicación de la Mota

Se trata de una aplicación creada utilizando el LPCXpresso, que se carga en la memoria de la placa e incluye diversos módulos necesarios para conseguir la funcionalidad deseada.

La aplicación, denominada RobotVigilancia, utiliza FreeRTOS y está conformada por dos tareas, wiflyTcpServer y controlMotor, comunicadas por una cola que almacena y envía los comandos recibidos desde el PC. La interacción de estas dos tareas es la base del funcionamiento del sistema.

Diagrama de bloques

La aplicación se basa en la interacción de dos tareas mediante una cola de 6 elementos. La primera tarea, de mayor prioridad "wiflyTcpServer" conecta el módulo Wifly, y lee de la UART a la que éste está conectado, si recibe algún comando desde la aplicación del PC (que previamente se ha conectado al módulo por Socket TPC) se la envía a la otra tarea "controlMotor" que procesará el comando recibido actuando sobre la electrónica del robot. Como es característico de FreeRTOS, este proceso se realiza de forma infinita.

En el capítulo 4 profundizaremos en el funcionamiento de las tareas y en las librerías asociadas. A continuación se muestra un sencillo diagrama de bloques de las tareas de la aplicación:

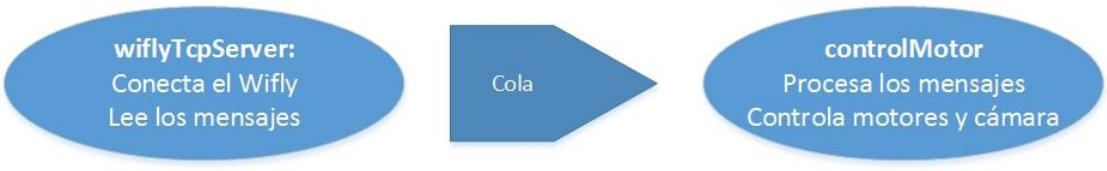


Figura 14: Tareas de la aplicación de la mota

4. DESCRIPCIÓN

A continuación explicaremos con detalle cada uno de los apartados del capítulo anterior. Se realiza una descripción detallada del sistema con sus componentes de hardware y los módulos asociados a la aplicación de la mota, para posteriormente profundizar en las herramientas de software usadas para programar tanto las aplicaciones de la mota como del PC, describiéndolas en cada caso.

4.1. HARDWARE

En este apartado describiremos con más detalle cada uno de los elementos de hardware utilizados, para posteriormente mostrar su interconexión detalladamente.

Placa LPCXpresso LPC1769 de NXP

Como comentamos en apartados anteriores, el target board LPC1769 es el "cerebro" del sistema, y se basa en un microcontrolador ARM de NXP, concretamente el Cortex-M3. Es una placa de bajo coste, bajo consumo y un alto nivel de integración, con todo lo necesario para encender, que incluye un programador y un depurador JTAG.

Las especificaciones de la placa son las siguientes:

- Microprocesador LPC1769FBD100
- 512KB de memoria Flash
- 64KB de memoria SRAM
- Reloj de 120 MHz
- 70 pines I/O digitales
- 4 UARTs
- 64 Canales de NVIC
- Comunicación SPI
- 3 I2C hasta 1 Mbit/s
- I2S
- 4 TIMERs de 16 bits
- 6 PWM de 16 bits
- Conversor ADC de 12 bits
- Conversor DAC de 10 bits
- Dos canales CAN
- Interfaz USB Device/Host/OTG
- QEI (Quadrature Encoder Interface)

Figura 15: LPC1769 Target board

La placa LPC1769 forma parte de la plataforma de desarrollo de bajo coste LPCXpresso, que comprende también un IDE simplificado basado en Eclipse, el cual se ha utilizado para el desarrollo de la aplicación de la mota y que presentaremos en este mismo capítulo en el apartado de Software.

Como podemos ver en las características, la mota dispone de 4 UART, de las cuales utilizaremos dos para el sistema diseñado. La interacción con dichas UART es esencial para el funcionamiento de los diferentes módulos de hardware es una parte principal de las librerías desarrolladas.

Para trabajar con la UART en nuestra aplicación tenemos tres módulos principales con las funciones utilizadas en el sistema, los describiremos a continuación.

Uart

Este módulo de NXP, proporcionado en una de las PECs de la asignatura, es la base para enviar y leer datos de la UART. Dispone de las siguientes funciones:

uint32_t UARTInit(uint32_t portNum, uint32_t Baudrate)

Inicia en la mota el puerto UART especificado por el parámetro portNum a la velocidad especificada por Baudrate, para poder enviar o recibir datos por esa UART.

*void UARTSend(uint32_t portNum, uint8_t *BufferPtr, uint32_t Length)*

Envía a la UART los datos por el puerto portNum, que estén almacenados en *BufferPtr, de longitud especificada en Length.

Printf

Este módulo es desarrollado por el estudiante para las PECs de la asignatura, y utiliza el módulo uart. Su funcionalidad debe ser análoga a la del printf de c, y escribirá los mensajes por la UART, pudiendo recibir parámetros. Dispone de las siguientes funciones:

uint32_t printfInit (uint32_t PortNum, uint32_t baudrate)

Esta función es la análoga a UARTInit, pero en este caso creamos un semáforo para asegurar la exclusión mutua cuando se internet acceder a la UART. Las funciones de printf harán uso de este semáforo, que se captura con la función xSemaphoreTake y se libera con xSemaphoreGive, de modo que no pueda haber dos funciones a la vez utilizándolo a la vez.

void printfUart (uint32_t PortNum, const char format , ...)*

Imprime por la UART especificada, recibe como parámetros el puerto, la cadena con el mensaje o formato del mensaje, y puede recibir parámetros variables.

PrintUtils

Este módulo contiene otras funciones muy útiles para nuestro proyecto, que son las siguientes:

uint32_t printfScanf (char cadena, uint32_t PortNum, uint32_t espera)*

Lee datos de la UART especificada en PortNum, durante el tiempo definido en “espera”, y guarda el resultado en "cadena”, devolviendo el número de caracteres leídos.

Esta función es de gran importancia ya que se usa en otras muchas funciones, y de hecho es la que utilizamos en una de las tareas de la aplicación principal para leer comandos recibidos mediante el módulo Wifly.

uint32_t printfCheck (uint32_t PortNum, char esperado, uint32_t tiempo Espera)*

Compara los datos recibidos por printfScanf comparándolos con los pasados por “esperado” devolviendo TRUE si coinciden.

uint32_t printfRespuesta (uint32_t PortNum, char esperado, uint32_t tiempo Espera, char* respuesta, uint32_t tamanoRespuesta)*

Esta función se encarga de leer la respuesta esperada de la UART y copiarla en la variable “respuesta”, en caso de no encontrar la respuesta devolverá FALSE.

Módulo conversor USB a UART CP2102

Se trata de un módulo que se encarga de convertir los mensajes de una UART TTL a USB, para que podamos verlos por pantalla desde el ordenador, a través de una aplicación de comunicaciones como Putty. Se trata de un módulo muy útil para recibir mensajes de la aplicación de la mota a modo debug, o para aprender a utilizar módulos que utilicen UART para la comunicación como el caso del Wifly.

Las principales características de este módulo son:

- Convierte USB 2.0 a UART
- Compatible con sistema TTL
- Soporta Baud Rates desde 300 bps hasta 1 Mbps
- 640 bytes de búfer de transmisión
- Compatible con los principales Sistemas Operativos

A continuación vemos el diagrama del chip CP2102 de Silicon Labs en el que se basa el módulo:

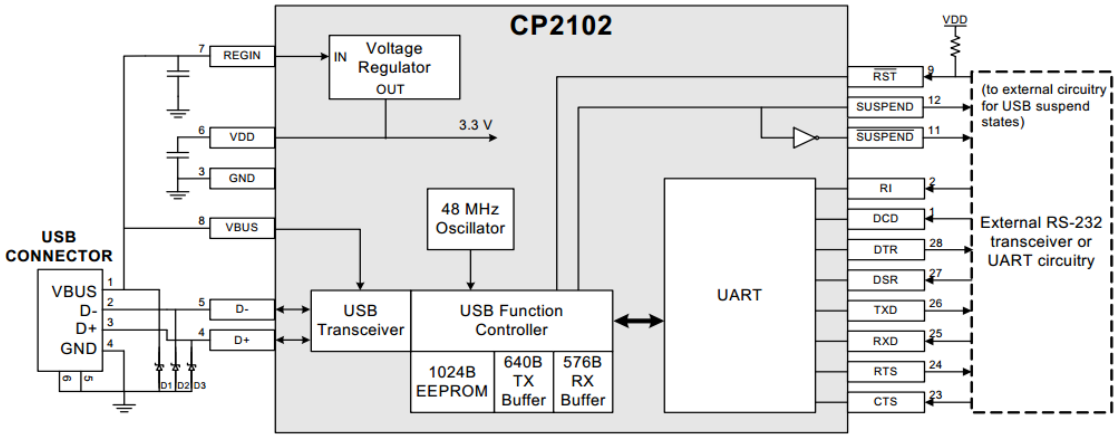


Figura 16: Diagrama de bloques del chip CP2102

Este módulo se usa en nuestro proyecto como debug para controlar la correcta ejecución de la aplicación de la mota, sin embargo como comentamos anteriormente su utilidad es mucho mayor, ya que este elemento de hardware se utiliza desde el principio en la fase de aprendizaje hasta prácticamente el final del proyecto, siendo también esencial para aprender a utilizar el módulo Wifly ya que puede conectarse directamente con este, y darle alimentación a través del pin de 3.3V que integra.

Para realizar las pruebas con el Wifly o ver los mensajes de debug enviados por la mota, se utiliza una aplicación que soporte comunicaciones por puerto serie, conectándose al puerto COM asociado al CP2102, y al baudrate que use la UART del dispositivo que queremos utilizar. En este caso hemos elegido Putty por ser un software sencillo e intuitivo, pero existen otras posibilidades como HyperTerminal.

Para el debug se ha implementado la siguiente librería:

Debug:

Se trata de una librería para enviar mensajes de debug por la UART 3. Estos mensajes se muestran con una cabecera indicando el tiempo actual de ejecución. Las funciones de esta librería son las siguientes:

uint32_t debugInit ()

Se encarga de iniciar la UART por la cual se enviarán los mensajes de debug. También inicia el RTC y el semáforo de exclusión mutua.

void debugUart (const char format , ...)*

Funciona de manera similar a la función printfUart, enviando un mensaje por la UART, pero añade una cabecera con el tiempo del RTC en formato hora.

A continuación se muestra la conexión física de este hardware con la mota:

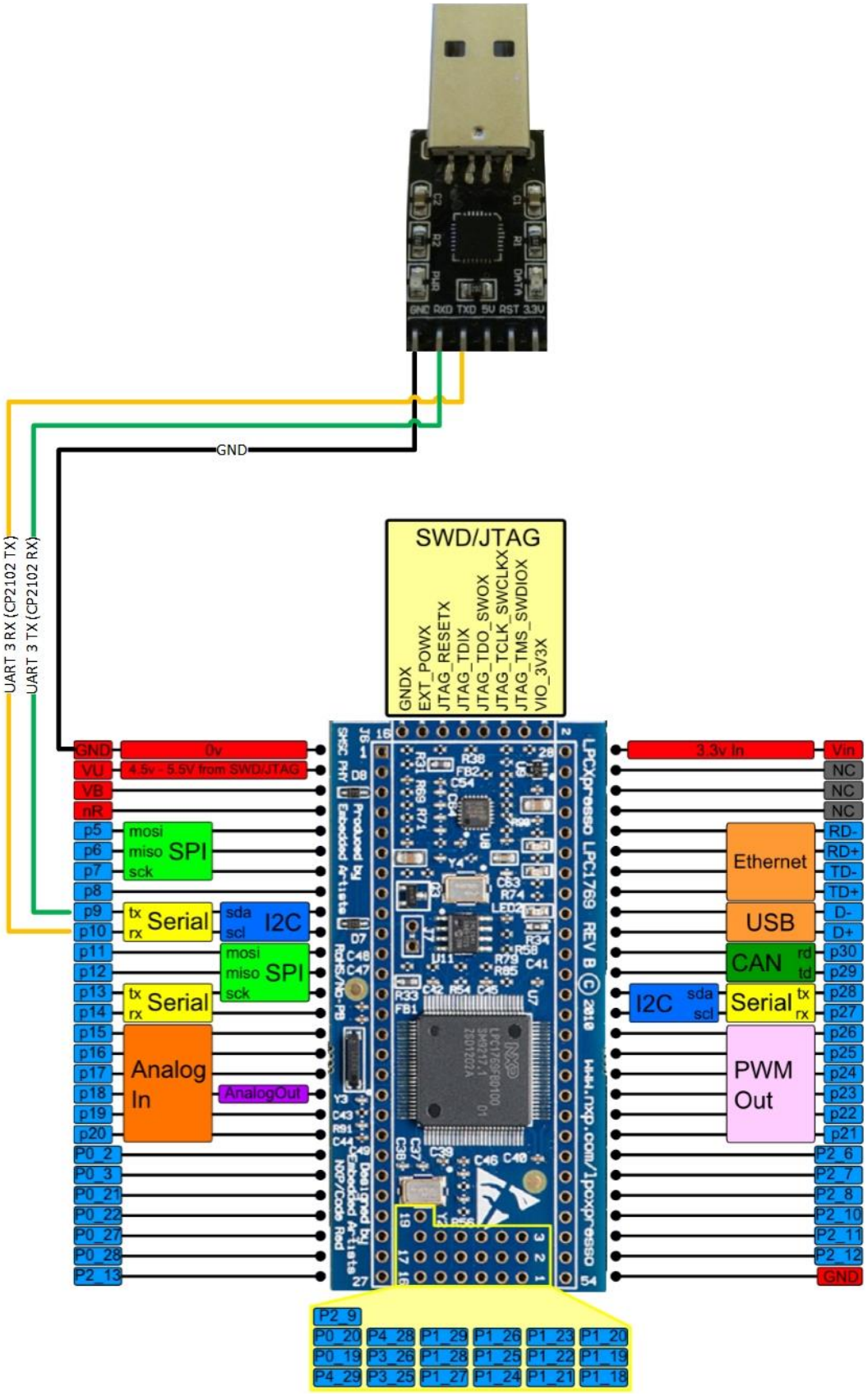


Figura 17: Conexión módulo CP2102 y mota

Placa de potencia Keyes L298

Es una placa de potencia para motores basada en el chip L298N que permite controlar dos motores de corriente continua o un motor paso a paso bipolar de hasta 2 amperios.

El módulo cuenta con todos los componentes necesarios para funcionar sin necesidad de elementos adicionales, entre ellos diodos de protección y un regulador LM7805 que suministra 5V a la parte lógica del integrado L298N.

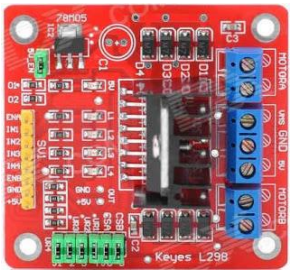


Figura 18: Placa de potencia Keyes L298

Este módulo se puede alimentar de 2 maneras gracias al regulador integrado LM7805. Cuando el jumper de selección de 5V se encuentra activo, el módulo permite una alimentación de entre 6V a 12V DC. Como el regulador se encuentra activo, el pin marcado como +5V tendrá un voltaje de 5V DC. Este voltaje se puede usar para alimentar la mota siempre que no sea superior a 500 mA.

Cuando el jumper de selección de 5V se encuentra inactivo, el módulo permite una alimentación de entre 12V a 35V DC. Como el regulador no está funcionando, tendremos que conectar el pin de +5V a una tensión de 5V para alimentar la parte lógica del L298N.

Para el presente proyecto, se han utilizado dos de estas placas, una para cada motor/rueda, debido a la sencillez y compatibilidad para controlar el sentido y la velocidad de una gran gama de motores, así como la relación funcionalidad/precio. Para el control de las placas se ha diseñado una API para motores controlados por placas de potencia, que puede ser reaprovechada para otros proyectos que utilicen placas de potencia controladas por PMW y entradas digitales.

A pesar de que esta placa ofrece la posibilidad de controlar dos motores DC de 2A, se ha instalado una para cada motor debido a razones de potencia, para evitar sobrecalentamientos y averías en la placa.

En la figura siguiente podemos ver los consumos de la serie RS-380.

Electrical Performance

MODEL		VOLTAGE		NO LOAD		AT MAXIMUM EFFICIENCY					STALL		
		OPERATING RANGE	NOMINAL V	SPEED r/min	CURRENT A	SPEED r/min	CURRENT A	TORQUE mNm	TORQUE g.cm	OUTPUT W	TORQUE mNm	TORQUE g.cm	CURRENT A
RS-380SA	2789R	3.0-12.0	6	7000	0.24	5837	1.21	7.75	79	4.73	46.67	476	6.11
	3133	6.0-15.0	14.2	27700	0.55	23909	3.48	11.76	120	29.44	85.98	877	22
RS-380SH	4530	6.0-9.0	7.2	25500	0.9	21960	5.57	11.57	118	26.59	83.33	850	34.5
	6017R	3.0-5.0	4.8	27800	2.05	23082	9.99	13.43	137	32.44	79.12	807	48.7

Outline:

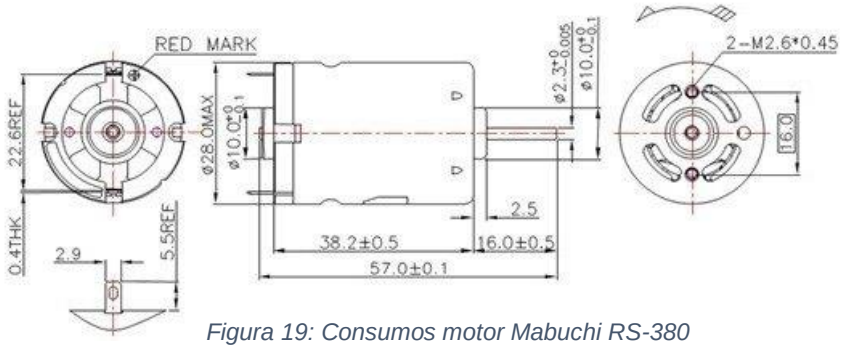


Figura 19: Consumos motor Mabuchi RS-380

A continuación se muestran los módulos usados. Para mayor detalle de la API y las librerías creadas, se puede consultar el anexo correspondiente a la API de motores.

Motor

Esta librería contiene las funciones necesarias para realizar todos los movimientos que permite la placa de potencia. Funciona a través de pwm.h y relé.h,

Pwm

Inicia y controla las salidas PWM de la LPC1769 donde están conectadas las placas de potencia.

Rele

Inicia y controla las salidas GPIO de la LPC1769 que modifican los movimientos del motor.

A continuación se muestra cómo se ha realizado el conexionado de los motores y las placas de potencia con la mota:

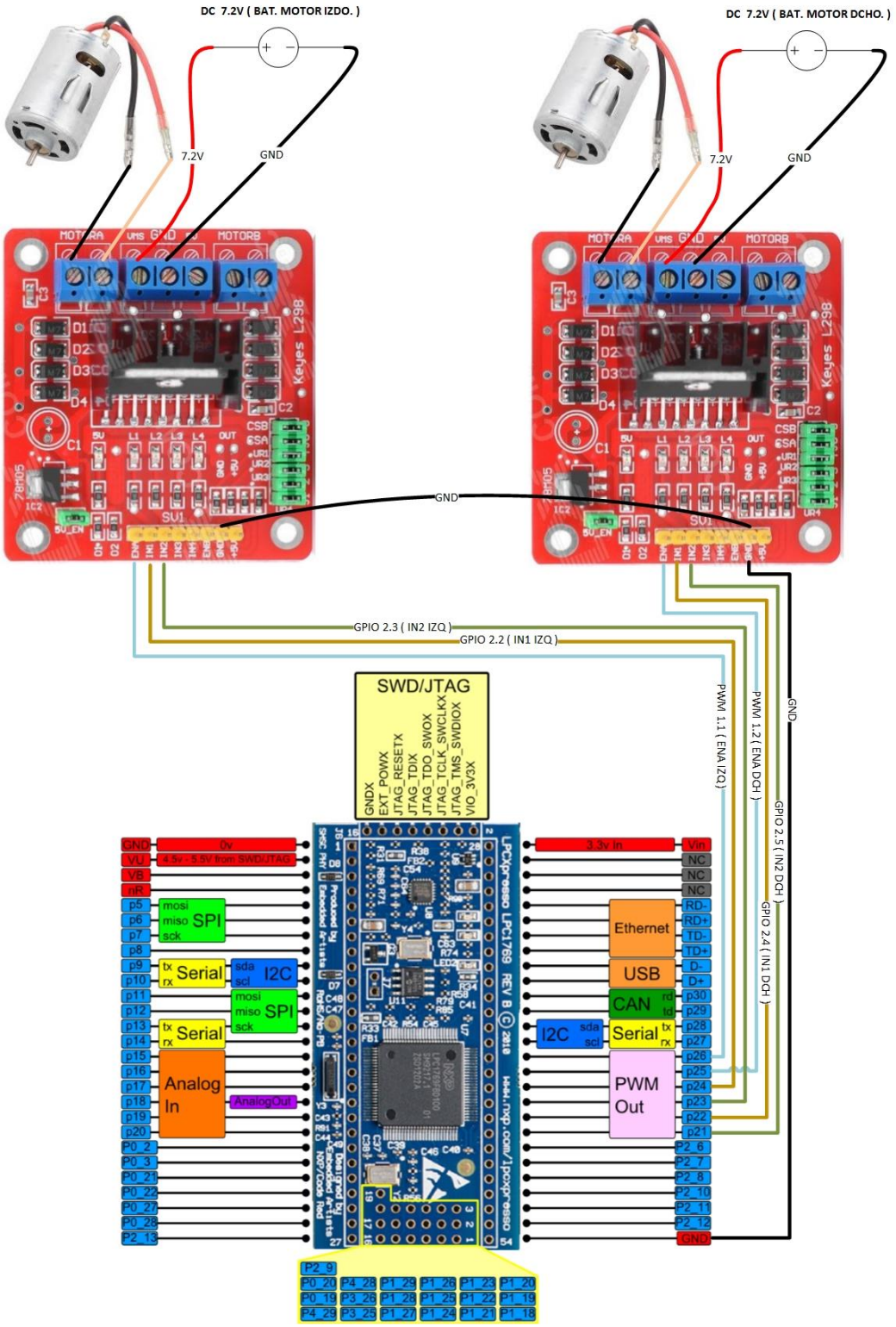


Figura 20: Conexionado placas de potencia y mota

Módulo Wifly de Roving Networks

El módulo RN-XV desarrollado por Roving Networks es un sistema de comunicación Wi-Fi certificado.

El módulo RN-XV integra un RN-171 Wi-Fi, un radio 802.11 b/g, un procesador de 32 bits, un stack TCP/IP, un reloj de tiempo real, un crypto acelerador, una unidad de manejo de potencia y una interfaz análoga. El módulo tiene pre-grabado un firmware que simplifica la integración y minimiza el tiempo de desarrollo teniendo una simple configuración de hardware donde solo se tienen cuatro conexiones PWR, TX, RX y GND.

Figura 21: Módulo Wifly

Las especificaciones del módulo son las siguientes:

- Tamaño y forma de un XBee 802.15.4
- Estándares: TCP/IP stack incluidos, DHCP, UDP, DNS, ARP, ICMP, HTTP cliente, FTP cliente y TCP.
- Potencia: 0dBm a 12dBm
- Hardware interfaces: TTL UART
- Host data rate: hasta 464Kbps (UART)
- Puertos: 8 GPIO y 3 análogos
- Voltaje: 3.3VDC
- Antena: tipo cable

A continuación podemos ver la descripción de los pines del módulo:

Pad Number	Signal Name	Description	Optional Function	Direction
1	VDD_3V3	3.3V regulated power input to the module		POWER
2	UART_TX	UART TX, 8mA drive, 3.3V tolerant		OUT →
3	UART_RX	UART RX, 3.3V tolerant		IN ←
4	GPIO 8	GPIO, 24mA drive, 3.3V tolerant		IN / OUT
5	RESET	**Optional** Module Reset Signal (active low), 100k Pull up, apply pulse of at least 160us, 3.3V Tolerant		INPUT
6	GPIO 5	GPIO, 24mA drive, 3.3V tolerant	Data TX/RX	OUT
7	GPIO 7	GPIO, 24mA drive, 3.3V tolerant		IN / OUT
8	GPIO 9	Enable Adhoc mode, Restore factory defaults, 8mA drive, 3.3V tolerant		IN / OUT
9	GPIO 1	GPIO, 8mA drive, 3.3V tolerant		IN / OUT
10	GND	Ground		GND
11	GPIO 14	GPIO, 8mA drive, 3.3V tolerant		IN / OUT
12	UART_RTS	UART RTS flow control, 8mA drive, 3.3V tolerant		OUT →
13	GPIO 4/SEN 6	GPIO, 24mA drive, 3.3V tolerant/ADC input , (**3.3V tolerant**). Defaults to GPIO 4	Association Status	IN / OUT
14	Not Used			No Connect
15	GPIO 6/SEN 7	GPIO, 24mA drive, 3.3V tolerant/ADC input , (**3.3V tolerant**). Defaults to GPIO 6	Connection Status	POWER
16	UART_CTS	UART CTS flow control, 3.3V tolerant		IN ←
17	SENSOR 5	Sensor Interface, Analog input to module, (**3.3V tolerant**)		INPUT
18	GPIO 3/SEN 4	GPIO, 8mA drive, 3.3V tolerant/ADC input (**3.3V tolerant**). Defaults to GPIO 3		IN / OUT
19	GPIO 2/SEN 3	GPIO, 8mA drive, 3.3V tolerant/ADC input (**3.3V tolerant**). Defaults to SEN 3		IN / OUT
20	SEN 2	Sensor Interface, Analog input to module, **3.3V tolerant**		INPUT

Figura 22: Pinout módulo Wifly

Para el control de este módulo se ha realizado una librería en la etapa de formación, concretamente para la PEC3, la describimos a continuación, así como las funciones que se han usado para el proyecto.

Wifly

Librería creada para interactuar con el módulo Wifly a través de una UART de la mota. Al ser un módulo muy potente, se implementaron bastantes funciones, aunque solo nombraremos las utilizadas en el proyecto:

void wiflyReset()

Esta función se encarga de resetear el modulo WiFly utilizando una GPIO de la mota, concretamente J6-24 (P0.22) del LPC1769 (al pin 5 de WiFly).

uint32_t wiflyInit(uint32_t portNum, uint32_t baudrate)

Inicializa el puerto de la UART para el Wifly a través de printfInit, a la velocidad indicada en el parámetro baudrate. Una vez iniciada la UART realiza un reset del módulo a través de wiflyReset, y crea el semáforo de exclusión mutua. Devuelve TRUE si se ha iniciado con éxito.

uint32_t wiflyConsola()

Entra en modo consola de comandos, leyendo la respuesta recibida por la UART para confirmar que se ha entrado correctamente.

uint32_t wiflyConnect(const char cifrado, const char* ssid, const char* clave)*

Configura y conecta el modulo a un punto de acceso. Se encarga de configurar el cifrado, la contraseña y el SSID del punto de acceso. Si la conexión se ha realizado con éxito devuelve TRUE. La aplicación de la mota, como se verá más adelante, comprueba este punto particular, y en caso de recibir FALSE trata de reconectar.

uint32_t wiflySave();

Guarda la configuración realizada al módulo, quedando esta como la nueva configuración por defecto.

uint32_t wiflyIpStaticParams();

Función encargada de configurar los parámetros IP necesarios para realizar una conexión estática, desactiva el DHCP, configura el puerto para la conexión del socket, DNS, etc.

void wiflyPruebas();

Corre pruebas para las funciones del módulo. Devuelve las respuestas por la UART3 a través de la librería debug. Se puede correr esta función para realizar una comprobación general del módulo y de sus funciones. Comprueba el funcionamiento de diversos comandos y configuraciones, haciendo ping, mostrando la ip actual, el tiempo desde el último reinicio, cambiando la potencia de transmisión y el baudrate, etc.

A continuación se presenta un diagrama de conexionado del módulo Wifly con la mota, tal y como se ha hecho en este proyecto.

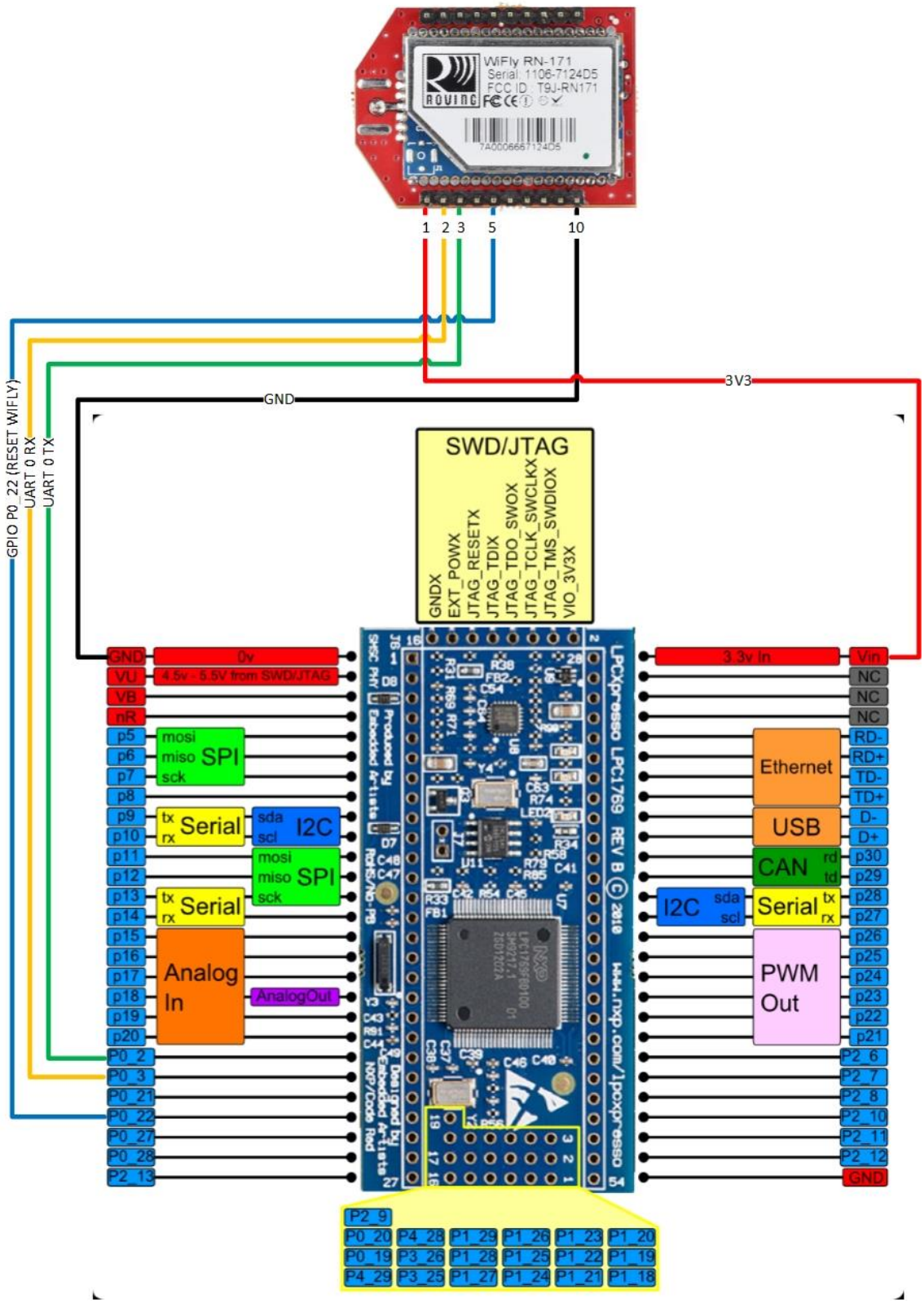


Figura 23: Conexionado módulo Wifly y mota

Cámara IP Wi-Fi Suneyes SP-Q701W

Como elemento principal de la parte de vigilancia se utiliza una cámara IP, de gama media-baja, pero que cumple las especificaciones ONVIF que asegura la compatibilidad con la mayoría de software de grabación, y que además posee características clave para conseguir el resultado deseado. Se trata de una cámara fija con carcasa de exterior tipo bullet y conectividad Wi-Fi para mantener la condición inalámbrica del sistema. Con el fin de reducir el consumo en espera se ha realizado un control del encendido y apagado de la cámara mediante un módulo de relé conectado a la mota.

Figura 24: Cámara IP SunEyes

Las principales características de la cámara son:

- Sensor CMOS de 1.4 pulgadas
- Resolución de 1.0 Megapixel
- Lente de 3.6mm y 70º de apertura
- Resolución 1280x720p y 25 FPS
- Wi-Fi b/g/n, TCP, UPnP, TFTP, HTTP, SMTP, DHCP, NTP, DDNS
- Filtro día/noche e iluminación IR 10m

Módulo relé con optoacoplador

Para el control de la alimentación de la cámara se ha instalado un relé montado en placa, con optoacoplador, testigo LED y jumpers de selección para el estado por defecto (low to high, o high to low). Se ha elegido este módulo porque permite activar el relé con voltajes de señal de 5 o 3.3 voltios.

Las especificaciones del módulo son las siguientes:

- Tensión (conmutación) de 250V.
- Corriente (conmutación) de 10A.
- Opto aislamiento.
- Jumper de selección de modo de trabajo.
- Disociación VCC – RelayVCC.
- Led indicador de estado.

Figura 25: Modulo relé con optoacoplador

Para el control del relé se ha utilizado la librería rele.h, que es utilizada también por motor.h, por lo que no la describiremos en detalle en este apartado. Se puede consultar su funcionamiento en detalle en el anexo de la API de motores.

A continuación se presenta un diagrama de conexionado del relé con la mota y la alimentación de la cámara:

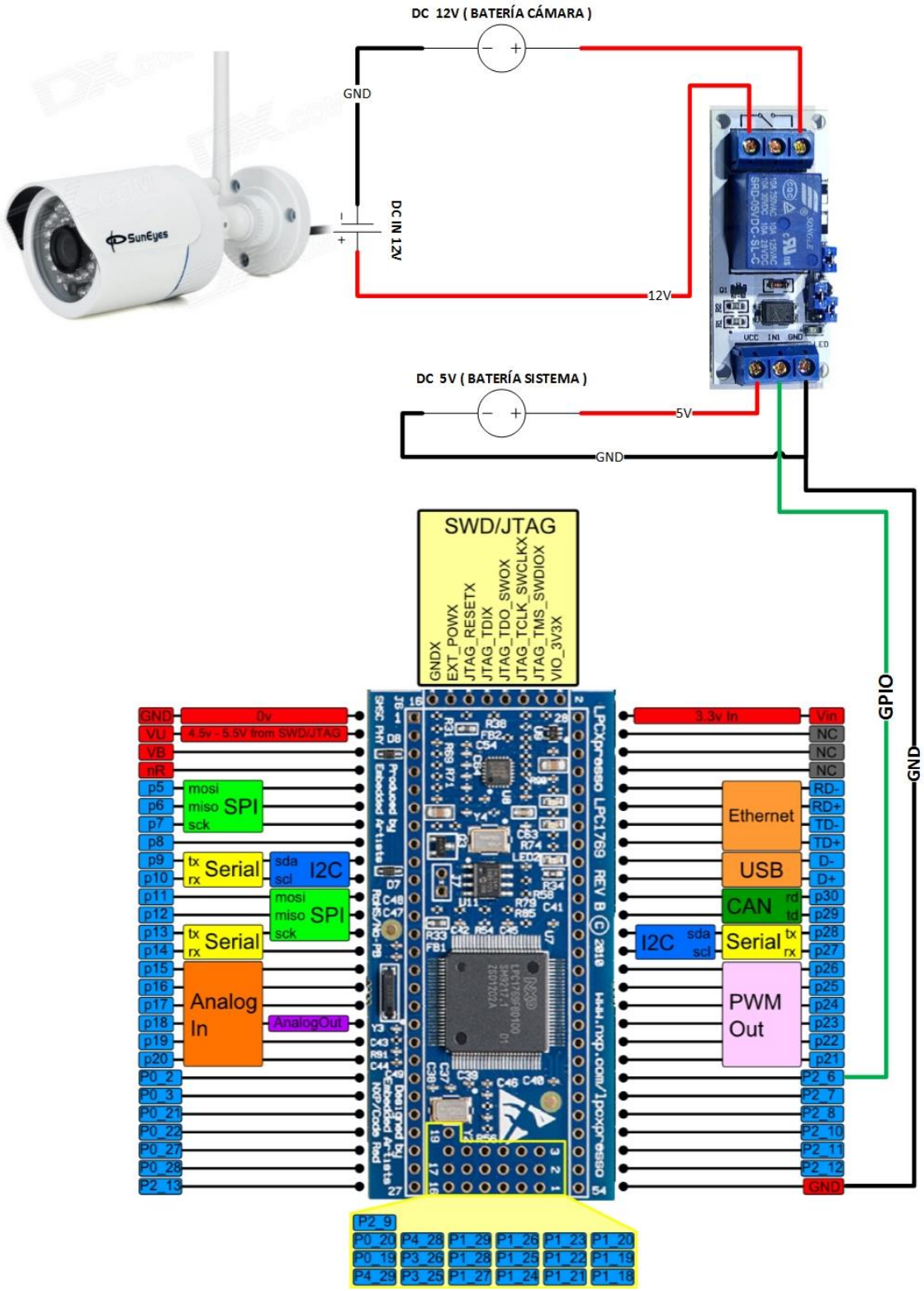


Figura 26: Conexionado módulo relé y mota

Cuerpo de Robot (aspirador Q7)

Como medio para soportar todos los elementos del sistema, y dotar al robot de tracción mecánica, se han reutilizado algunas partes de un robot aspirador averiado. Principalmente se utiliza toda la estructura plástica, los motores de las ruedas con su correspondiente tracción reductora, y las baterías de Ni-MH.

A pesar de tratarse de un modelo de bajo coste, tiene un buen sistema tractor, y una carcasa robusta y espaciosa, que permite anclar la cámara y albergar en su interior toda la electrónica que conforma el sistema.

Figura 27: Robot aspirador Q7

Para la tracción dispone de dos ruedas de goma con sus correspondientes reductoras, y sendos motores de escobillas Mabuchi modelo RS-380, con una buena relación de potencia suficiente y un buen agarre en diversas superficies.

Para la alimentación dispone de dos baterías de Ni-MH de 7,2V y 1500mAh, que proporcionan unos valores adecuados para alimentar las placas de potencia y los motores.

Baterías

Para alimentar el sistema, se utiliza un conjunto de baterías recargables. Además de las ya mencionadas en el apartado anterior (Ni-MH) para las placas de potencia y los motores, y a fin de aumentar la autonomía del sistema lo máximo posible, se utilizan también dos baterías de Li-Ion, una de 12V y 6800mAh para la cámara, y otra de 5V y 24000mAh para la mota, esta última pudiéndose recargar automáticamente mediante una placa solar integrada en la misma.

Figura 28: Baterías usadas para alimentar el robot

Dado que una de las principales características de la mota y del módulo Wifly es su consumo ultra-bajo, tendremos un sistema con una gran autonomía, y que nos permitirá mantener el sistema en espera durante mucho tiempo.

4.2. SOFTWARE

IDE LPCXpresso

Un entorno de desarrollo integrado o IDE (acrónimo en inglés de Integrated Development Environment), es un programa informático compuesto por un conjunto de herramientas de programación. Para el desarrollo de la aplicación para la mota se ha utilizado el IDE LPCXpresso, una completa herramienta para evaluación y desarrollo con microcontroladores de NXP.

El LPCXpresso IDE fue desarrollado por CodeRed junto a NXP. El mismo incluye un entorno de Eclipse específicamente adaptado para interactuar con el target board.

Conviene destacar dos elementos principales de este entorno:

Workspace

Es el contenedor de nuestros proyectos. Estos proyectos pueden ser aplicaciones y/o bibliotecas. También almacena todas las configuraciones del entorno por lo que se puede mover muy fácilmente de computadora en computadora.

Proyecto

Este puede ser de dos tipos. Biblioteca estática o una aplicación ejecutable. Contiene archivos de código fuente (.c), encabezados (.h) y cualquier otro archivo que se desee.

En general utilizaremos el workspace para intercambiar proyectos (en el sentido convencional de la palabra) ya que el mismo incluirá todas las bibliotecas necesarias.

FreeRTOS

Los sistemas operativos de tiempo real facilitan las herramientas para cumplir los compromisos temporales definidos por el programador. Se emplean principalmente cuando hay que administrar varias tareas simultáneamente, en un plazo de tiempo estricto.

Para el desarrollo de la aplicación de la mota hemos usado el FreeRTOS. Se trata de un potente sistema operativo de tiempo real para sistemas embebidos, que permite realizar multitarea. Las tareas actúan como funciones independientes con una memoria RAM asignada al contexto de ejecución de cada una. Existe un scheduler que va cambiando de contexto y asignando unos ciclos de reloj a unas tareas y otras, pudiendo trabajar con diferentes prioridades, colas semáforos, etc.

Aplicación de la mota:

Tras la fase de aprendizaje, y familiarizarse con el entorno de programación y el FreeRTOS, se ha procedido a realizar de la aplicación de la mota.

Se ha realizado un proyecto librería utilizando el LPCXpresso, que incluye diversos módulos necesarios para conseguir la funcionalidad deseada, unos se han reaprovechado de las PECs realizadas durante el aprendizaje, y otros se han creado con posterioridad, como los destinados al control de motores, que tratamos con detalle en el anexo II de esta memoria.

La aplicación, denominada RobotVigilancia, está conformada por dos tareas, comunicadas por una cola que almacena y envía los comandos recibidos desde la aplicación de control del PC. Estos comandos se reciben a través de la tarea llamada wiflyTcpServer y se procesan por la tarea controlMotor. La interacción de estas dos tareas es la base del funcionamiento del motor.

A continuación se muestra un diagrama de funcionamiento e interacción de ambas tareas y se detalla cada una de ellas:

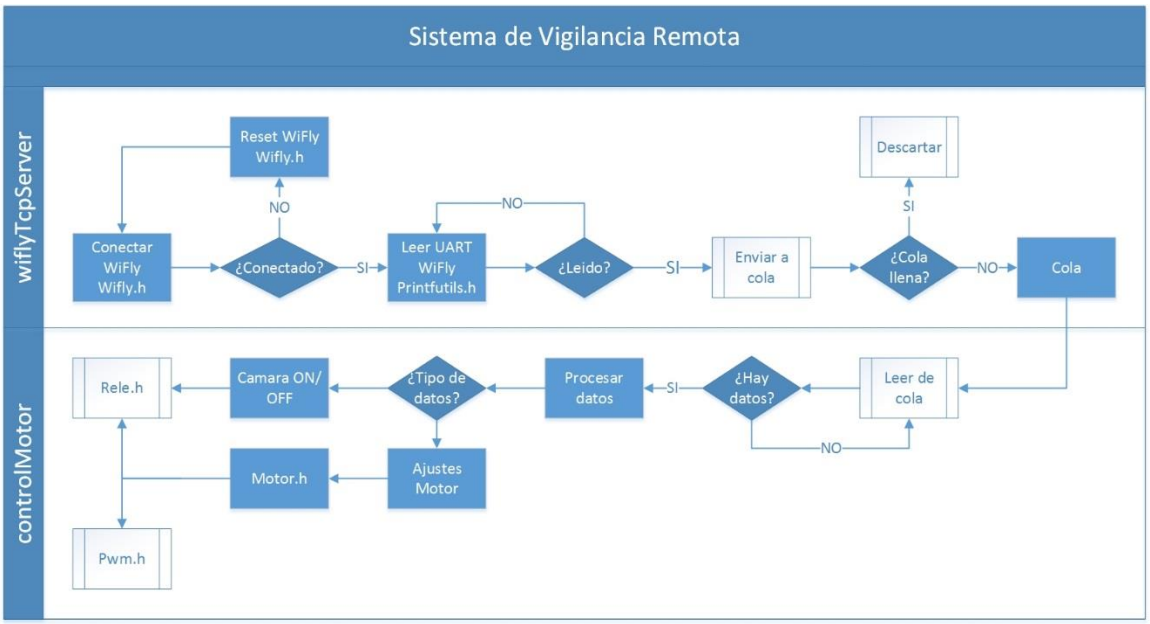


Figura 29: Diagrama de aplicación de la mota

wiflyTcpServer

Esta tarea es la destinada configurar y conectar el módulo Wifly, leer de la UART del Wifly los comandos enviados desde el PC y enviarlos a la cola de donde consumirá controlMotor.

Esta tarea tiene más prioridad que controlMotor ya que debe atender a la UART dónde se encuentra conectado el módulo Wifly y es necesario que pueda leer los comandos inmediatamente después de que el módulo los recibe. Para leer se utiliza la función printfScanf del módulo printfutils.

Para la conexión del módulo Wifly a la red Wi-Fi se ejecutan una serie de funciones de wifly.h que sirven para configurar los parámetros de la red inalámbrica, y conectarse a la misma. En caso de que la conexión no se haya realizado con éxito se realiza un reset del módulo y tras una espera de 3 segundos se vuelve a intentar la conexión.

Una vez conectado, tal y como vemos en el diagrama, se pasa a la siguiente etapa que es leer constantemente de la UART. Todo el proceso se puede monitorizar a través de la UART destinada a debug (UART3) a través del módulo CP2102, por la cual se envían constantemente mensajes de estado y confirmación.

La aplicación del PC tiene una característica en el Joystick virtual, que si se mantiene apretado un botón, este envía el comando constantemente, con un intervalo de repetición definido en el script de dicha aplicación. Dicha característica puede provocar que se envíen más mensajes de los que la tarea controlMotor puede procesar. Por esta razón se ha decidido crear una cola de 6 elementos, que almacenará los últimos 6 comandos recibidos, y en caso de que controlMotor no pueda recibir un comando nuevo, y la cola esté llena, el último comando recibido se descartará enviando un mensaje a la consola de la aplicación del PC. El tamaño de la cola se ha ajustado de manera que permita que el robot se pueda mover de manera fluida al mantener apretado un botón de movimiento en la aplicación del PC, pero a la vez sin que éste se mueva durante más tiempo del deseado.

Según la configuración considerada óptima, y tras realizar varios test de funcionamiento, la duración del movimiento de cada comando se estableció en 250ms, de manera que desde que se empieza a ejecutar el primer comando de la cola el robot pueda moverse como máximo durante 1,5s.

controlMotor

Esta tarea es la destinada a recibir a través de la cola los comandos. Si hay algún mensaje en la cola se procesa de la manera que corresponda.

Dependiendo del tipo de mensaje tenemos 2 posibilidades, la más probable debido a la cantidad de comandos relacionados es que se trate de algún control para el motor, ya sea un ajuste de velocidad, o una indicación de movimiento. La otra opción es que se trate de un comando para encender o apagar la cámara.

En el primer caso la tarea utiliza la librería motor.h, y en el de la cámara rele.h, y se encarga de enviar las órdenes correspondientes a las funciones contenidas en cada uno.

Cada vez que se ejecuta un comando se envía una confirmación a la aplicación del PC a través de la UART a la que está conectada el Wifly, mediante la función printfUart de printf.h, y se imprime por la UART de debug el mensaje leído de la cola, de forma que haya una doble vía para ver si se han recibido y procesado los comandos.

Intérprete Python

Para el desarrollo de la aplicación para PC hemos escogido Python, un lenguaje de scripting, interpretado, cuya filosofía hace hincapié en una sintaxis que favorezca un código legible. Se trata de un lenguaje de programación multiparadigma, ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es un lenguaje interpretado, usa tipado dinámico y es multiplataforma.

Todas estas características hacen que sea un lenguaje simple, versátil, y que favorece una gran rapidez de desarrollo, debido a que todo está modularizado, pero el hecho de ser un lenguaje interpretado hace que sea indispensable tener instalado el intérprete de Python en el ordenador o dispositivo desde el que se ejecutará el script, y que la ejecución sea más lenta en comparación con lenguajes no interpretados.

Aplicación del PC:

Como ya vimos en el capítulo 3, se ha diseñado, utilizando el lenguaje Python, una aplicación para el control de la aplicación de la mota desde el PC y la recepción de proveniente la cámara. Con el código del proyecto se aporta también la aplicación para Windows que consta de un lanzador llamado RVR.bat que abre el intérprete Python (debe estar previamente instalado en su versión 2.7) y carga el script RVR.py que es la aplicación del PC.

Una vez ejecutado RVR.bat se abrirán las dos ventanas que vemos en al figura, una es la consola para el debug, por la que podremos ver mensajes de confirmación, errores, etc. y la otra es la Interfaz Gráfica de Usuario, desde la cual interactuaremos con el la aplicación y enviaremos las órdenes correspondientes a la mota.

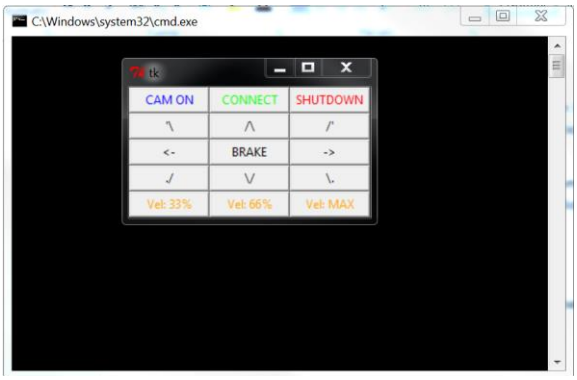


Figura 30: Resultado de ejecutar RVR.bat

Para la comunicación con la mota se ha desarrollado un Joystick virtual basado en una cuadrícula de botones del módulo Tkinter de Python. Cada botón tiene una función asociada y ejecuta uno o varios comandos.

A continuación se muestra el nombre de cada botón y las funciones o comandos asociados a cada uno. Los mensajes enviados a la mota tienen el siguiente formato: "____LPC1769:XXXXX LPC1769:XXXXX LPC1769:XXXXX ____\r\n" donde “XXXXX” es diferente para cada comando enviado.

Esto tiene una razón, y es que en ocasiones se pueden producir errores de lectura en la UART y perder algún caracter. Cuando la tarea controlMotor recibe el mensaje busca en todo el mensaje “LPC1769:XXXXX” y si encuentra una coincidencia ejecuta el comando correspondiente.

Cada botón enviará uno de estos mensajes a la mota como veremos a continuación. Los mensajes de movimiento se envían, una sola vez, y los críticos como el encendido y apagado de la cámara, se envían por duplicado.

CONNECT

Este es el primer botón que, según el manual de la aplicación, adjunto en los anexos, un usuario debería pulsar. Este botón se encarga de ejecutar la función wiflyConnect(), que conecta al socket TCP creado al arrancar el programa, y recibe el mensaje del módulo Wifly confirmando la conexión, o un mensaje de error de timeout del socket en caso de no recibir respuesta.

Como en el resto de funciones, las confirmaciones recibidas siempre se mostrarán por consola.

CAMON

Una vez conectados, y antes de mover el robot se debe presionar este botón que ejecuta la función camara(). Dicha función envía un mensaje a la mota ("…LPC1769:CAMON…”) con el formato antes comentado para encender la cámara, y pone en espera la aplicación durante 25 segundos que es el tiempo que tarda la cámara en iniciarse una vez que recibe alimentación.

Flechas y BRAKE

Los botones 9 centrales de la rejilla son los destinados a mover (o frenar en el caso de BRAKE) el robot. Cada botón ejecuta una función similar que envía un mensaje determinado a la mota, recibe la respuesta y la imprime por la consola de debug.

No tiene sentido que escribamos las 9 funciones al ser similares, pero podemos nombrar una a modo de ejemplo, forward() que enviará el mensaje "____LPC1769:FOWRD LPC1769:FOWRD LPC1769:FOWRD____\r\n", que será procesado por la tarea controlMotor y ejecutará las instrucciones necesarias para mover el robot hacia adelante.

Los movimientos posibles son adelante, atrás, giro en redondo, las 4 diagonales y freno, y como siempre se recibirá confirmación cada vez que uno de los mensajes sea procesado, como podemos observar en la imagen.

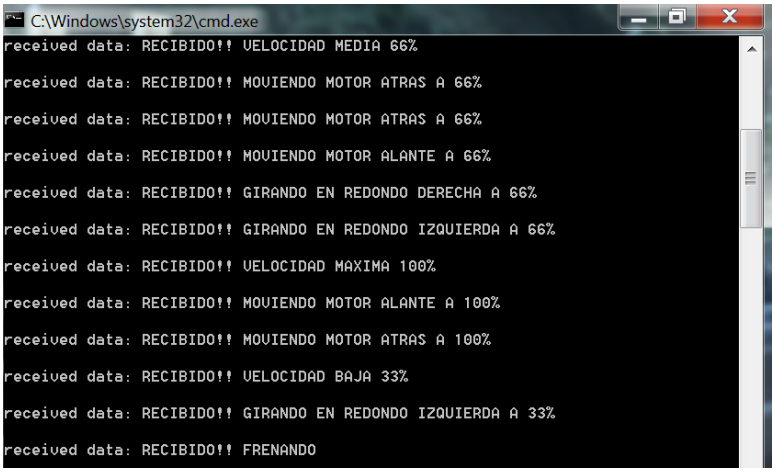


Figura 31: Confirmaciones en consola del software para PC

Velocidades

Los botones 3 botones inferiores de la rejilla son los destinados a cambiar la velocidad a la que se mueve el robot, y el funcionamiento es análogo al comentado para los botones de movimiento.

Como ejemplo nombramos la función velocidad_33() que enviará el mensaje ("…LPC1769:VEL33…"). Se han implementado tres botones, uno para la velocidad baja (33%) otro media (66%) y por último la máxima (100%). Estos porcentajes se corresponden con los que configurará la mota en cada caso para el ancho de pulso del PWM, como detallaremos en el anexo destinado a la API de motores.

SHUTDOWN

Este botón es el que hay que presionar para salir, envía el mensaje ("…LPC1769:CAMOF…") que provocará que la mota corte la alimentación de la cámara a través del relé, posteriormente cerrará el socket y por último la aplicación.

Cuando se sale del programa hay que tener en cuenta que este botón no cierra el VLC, que debe ser cerrado manualmente para que la conexión con la mota quede completamente liberada, y el módulo Wifly pase a modo de espera de conexión.

5. VIABILIDAD TÉCNICA

La principal funcionalidad del sistema es el control de los motores del robot a través de la mota y dos placas de potencia, enviando las órdenes vía Wi-Fi a través de un software para PC basado en Python y recibiendo el video por IP a través de VLC.

Esta distribución hace que el sistema sea portable, y pueda ser instalado en cualquier lugar de manera sencilla, siempre y cuando exista una conexión Wi-Fi con la que se pueda comunicar la máquina remota. Dicha comunicación se basa en una conexión Socket TPC por lo que no es necesario que la máquina dónde está instalado el software de control se encuentre en el mismo emplazamiento, siempre y cuando haya una ruta para transportar los datos.

Al ser sistemas que no necesitan alimentación eléctrica externa, y comunicarse a través de una conexión TCP/IP, se podrían tener cuantos robots se quisieran en una misma zona, y el área de actuación de los robots estaría limitada solamente por la extensión de la red Wi-Fi, pudiendo cubrir extensiones muy amplias.

Los robots se podrían controlar remotamente desde una CRA (Central Receptora de Alarmas) para realizar rondas de vigilancia, o en caso de que se detecte una intrusión por un sistema externo de alarmas, realizar una videoverificación de la zona, para poder confirmar si se trata de una alarma real, o una falsa alarma, evitando llamadas innecesarias a las fuerzas de seguridad, tal y como exige la ley actual.

A continuación enumeraremos los puntos clave que hacen funcional sistema, así como los puntos débiles que lo limitan técnicamente.

Puntos clave:

- Alta portabilidad y área de cobertura
- Capacidad de ampliación e instalación de diversos sensores
- Coste reducido en comparación con otros sistemas de vigilancia
- Gran autonomía en espera
- Hardware económico y de fácil sustitución en caso de avería
- Buena calidad de video, incluso en condiciones de oscuridad total

Puntos débiles a mejorar:

- Autonomía limitada en funcionamiento
- Pérdida total de conexión en caso de fallo o inhibición de la red Wi-Fi
- Pérdida puntual de paquetes
- Complejidad del sistema de carga (tres tipos de batería), carga presencial
- Configuración de Wi-Fi y actualizaciones se deben hacer presencialmente
- Se necesita el intérprete Python para ejecutar el Software para PC

6. VALORACIÓN ECONÓMICA

Para el desarrollo del proyecto se han utilizado los materiales de la asignatura, y se ha aprovechado el cuerpo y motores de un robot aspirador como base. A mayores, y partiendo de un presupuesto cerrado de 150€ para materiales, se han realizado las compras oportunas para dotar al sistema de todas las funcionalidades deseadas.

Tras valorar los materiales necesarios y realizar una consulta de las opciones disponibles se concluye que el presupuesto es suficiente, quedando un ligero margen para cubrir costes imprevistos quedando distribuido de la siguiente manera: Cuerpo y motores del robot reaprovechados 0€, Cámara hasta 65€, componentes electrónicos hasta 45€, y sistema de alimentación y cableado hasta 50€.

A continuación se presenta un presupuesto con los valores reales de todos los materiales utilizados (incluyendo los materiales de la asignatura y los reaprovechados del robot), así como de las horas empleadas en desarrollo e instalación.

Item	Descripción	Cantidad	Precio	Subtotal
001	Mota NXP LPCXpresso1769	1	39,30 €	39,30 €
002	Módulo RN-XV WiFly	1	27,20 €	27,20 €
003	Placa de potencia Keyes L298N	2	4,51 €	9,02 €
004	Conversor UART-USB CP1202	1	6,90 €	6,90 €
005	Motor Mabuchi RS-380	2	12,90 €	25,80 €
006	Estructura con reductoras y ruedas	1	30,00 €	30,00 €
007	Módulo Relé con optoacoplador	1	5,65 €	5,65 €
008	Cámara IP Wi-Fi Suneyes SPQ701W	1	53,23 €	53,23 €
009	Batería recargable 7.2V 1500mAh Ni-MH	2	7,45 €	14,90 €
010	Batería recargable solar 5V 24000mAh Li-Ion	1	17,23 €	17,23 €
011	Batería recargable 12V 6800mAh Li-Ion	1	17,30 €	17,30 €
012	Pequeño material	1	25,00 €	25,00 €
013	Programación, investigación y desarrollo (horas)	140	15,00 €	2.100,00 €
014	Instalación y puesta en marcha (horas)	10	20,00 €	200,00 €
TOTAL CON I.V.A.				**2.571,53 €**
I.V.A.				**540,02 €**

Figura 32: Presupuesto

7. CONCLUSIONES

7.1. Conclusiones

A continuación se analizan los objetivos planteados para el proyecto, y se valora el grado de consecución de los mismos:

- **Dotar al sistema de tracción mecánica vía motores eléctricos:** Se ha conseguido, el robot se mueve como se esperaba y sin ningún problema.

- **Dotar al sistema de una comunicación inalámbrica para conectarse con la aplicación de PC para recibir comandos (vía el módulo Wifly):** La creación de las librerías para controlar el módulo Wifly y parsear las respuestas ha sido la parte más complicada, pero finalmente se ha conseguido.

- **Implementar un sistema de encendido de la cámara con el fin de reducir el consumo:** Se ha conseguido mediante un relé con optoacoplador controlado por la mota.

- **Integrar una cámara comercial IP para el visionado de imágenes de forma remota:** Se ha integrado la cámara en el sistema, si bien no se ha podido visionar directamente desde el script y éste debe lanzar el software VLC. Esto se debe a que los tipos de compresión de video soportados por la cámara son limitados, y no se pudo programar un visor integrado en la GUI.

- **Integrar las funcionalidades de control de movimiento y activación de la cámara de forma remota:** Se ha realizado mediante el software de control para PC.

Además se ha desarrollado con éxito una API para el control de dos motores y se ha documentado en los anexos. No ha habido tiempo a hacer pruebas físicamente con el módulo de cámara OV7670, que se había planteado inicialmente como un objetivo secundario, pero se ha podido realizar un pequeño estudio y se ha concluido que es técnicamente posible su utilización como alternativa a una cámara IP, si bien sus características son insuficientes para el resultado buscado en el presente proyecto.

7.2. Propuesta de mejoras

Como ya se ha comentado en la viabilidad técnica, existen varios puntos a mejorar, por lo que comentaremos cada uno de ellos y propondremos cómo se podría mejorar.

- **Autonomía limitada en funcionamiento**: Se dispone de una gran autonomía en espera al disponer de una batería de mucha capacidad para el sistema, que incorpora carga solar, pero existe una limitación debido al consumo de los motores y la cámara. Se podría mejorar este punto, así como el de la complejidad del sistema de carga, debido a que existen tres tipos de batería y se necesita realizar la carga presencialmente, mediante la unificación de la alimentación, usando una misma batería de gran capacidad que alimente todo el sistema, y que igualmente posea carga solar o algún otro método que permita recargarse automáticamente sin que sea necesario asistir al lugar presencialmente. El robot puede transportar una batería de mayores dimensiones sin problema, y podría instalarse una placa solar que ocupara toda su superficie.

- **Pérdida total de conexión en caso de fallo o inhibición de la red Wi-Fi:** Este es un punto complejo, que se podría solucionar instalando un medio de comunicación alternativo, por ejemplo en caso de fallo se podría tener un modem GPRS con una tarifa especial para M2M (machine to machine), que tiene un coste mensual reducido, y evitaría una pérdida total de la conexión. La cámara necesitaría una conexión 4G para poder enviar video a la misma calidad que ya supondría un coste mayor.

- **Pérdida puntual de paquetes:** establecer en las aplicaciones un método de reenvío de paquetes en caso de que no se reciba respuesta por parte del robot, de forma que se compruebe si el comando enviado se ha ejecutado, y en caso contrario se vuelva a intentar.

- **Configuración de Wi-Fi y actualizaciones se deben hacer presencialmente:** Si el robot tuviese siempre una tarjeta M2M como la antes comentada, modificando la aplicación de la mota, se podría conseguir establecer los parámetros del Wi-Fi mediante comandos, ya que el robot siempre tendría conexión. En cuanto a actualizar la aplicación remotamente sería necesario buscar un método equivalente a la carga a través del LPC-Link, que permitiese conectar a través de M2M, o utilizar USB over Ethernet.

- **Se necesita el intérprete Python para ejecutar el Software para PC:** Se podría portar el código a un lenguaje no interpretado, o adaptar el script para que pueda compilarse mediante un software que lo permita, como py2exe.

7.3. Autoevaluación

Tanto la asignatura como el proyecto han resultado más difíciles de lo que me esperaba cuando me decidí por esta área de TFC, ya que no imaginaba tener que programar tanto, pero al mismo tiempo he adquirido muchos conocimientos acerca de sistemas embebidos y microcontroladores, placas de potencia y demás hardware usado, y estoy satisfecho con los resultados.

He aprendido acerca de los RTOS, y la gestión de tareas, colas, semáforos, etc., he realizado mi primera GUI en Python, y lo más importante he conseguido que el robot funcionara finalmente, y estoy seguro que seguiré trabajando en él ya que me gustaría seguir aprendiendo acerca de este mundo que conocía poco y ahora veo que tiene muchísimo potencial.

8. GLOSARIO

ARM: Arquitectura RISC de microcontroladores desarrollada por ARM Holdings.

Cortex-M3: Microcontrolador de 32-bit ARM.

Cámara analógica: Cámara tradicional de videovigilancia.

Cámara domo: Cámara con carcasa en forma de esfera.

Cámara IP: Cámara de videovigilancia con conectividad Ethernet.

Cámara bullet: Cámara con carcasa en forma de bala.

CAN: Controller Area Network.

CMSIS: ARM® Cortex™ Microcontroller Software Interface Standard.

DDNS: DNS dinámico.

Driver de motores: Placa de potencia para el control de motores.

Dron: Vehiculo de vigilancia no tripulado.

DSP: Procesador digital de señal.

E/S: Sistema de comunicación entre la CPU y el exterior.

FreeRTOS: Sistema operativo de tiempo real usado en el proyecto.

GPIO: Entradas y salidas de propósito general.

GPRS: Protocolo de comunicación de datos móvil de 64Kbps.

GUI: Interfaz Gráfica de Usuario, parte gráfica de una aplicación.

IP: Dirección que identifica a un elemento en la red.

JTAG: Arquitectura usada para testear PCBs utilizando escaneo de límites.

Microcontrolador: Circuito integrado capaz de ejecutar las órdenes grabadas en su memoria.

Microprocesador: Circuito central de un sistema informático.

Mota: Sistema embebido con conectividad inalámbrica.

Putty: Programa para realizar conexiones de comunicaciones.

PWM: Modulación por ancho de pulsos.

RTSP: Protocolo multimedia de flujo en tiempo real.

RS-232: Protocolo de comunicaciones por puerto serie.

Streaming: Flujo de video en tiempo real.

Script: Archivo de procesamiento por lotes.

Smartwearable: Ropa o complemento dotado de sensores, microcontroladores, etc.

Socket: Punto final de una comunicación entre los elementos de una red.

TCP: Protocolo central de comunicaciones en red TCP/IP.

UART: Hardware que traduce datos entre serie y paralelo.

Wi-Fi: Tecnología de transmisión de redes de área local sin cables.

10. BILBIOGRAFÍA

Bibliografía

Using the FreeRTOS Real Time Kernel – A Practical Guide (Richard Barry)

Recursos web:

Wiki de la asignatura

Manual y Datasheet de la placa LPC1769

Manual de LPCXpresso

Manual y Datasheet del módulo Wifly

https://es.wikipedia.org/

http://www.freertos.org/

http://www.cplusplus.com/

http://www.spy-c-tank.com/

http://www.dx.com/

http://rovingnetworks.com/products/RN_XV

http://www.xataka.com/robotica/knightscope-k5-el-robot-de-vigilancia-que-se-mueve-como-una-aspiradora-roomba

http://electronilab.co/tutoriales/tutorial-de-uso-driver-dual-l298n-para-motores-dc-y-paso-a-paso-con-arduino/

https://www.fayerwayer.com/2012/08/los-computadores-del-apolo-11-que-llevaron-a-neil-armstrong-a-la-luna/

11. ANEXOS

11.1. Manual de LPCXpresso

En el presente anexo describiremos la forma de importar, compilar y cargar en la mota el código del proyecto.

Instalación

No se entrará en detalles de instalación y configuración del IDE ya que existe un completo manual del fabricante en el siguiente Link:

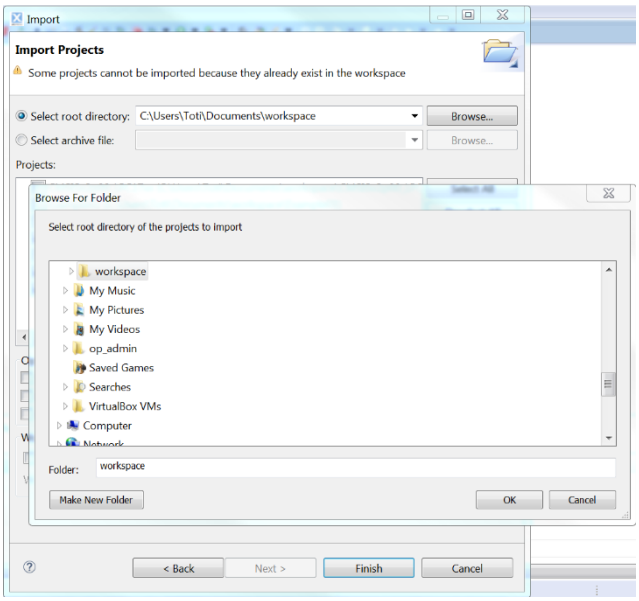
Manual de usuario de LPCXpresso v7

http://www.lpcware.com/system/files/LPCXpresso_User_Guide_3.pdf

Importar código

Una vez instalado el LPCXpresso siguiendo las indicaciones del fabricante procederemos a importar el código.

Para importar el proyecto, usaremos el método habitual de intercambio de proyectos que comentamos en el capítulo 4 cuando describimos este software. Debemos tener preparado el workspace con todo el código necesario, que en nuestro caso es el sistema operativo de tiempo real FreeRTOS (FreeRTOS_Library), la intefaz estándar de software de microcontroladores cortex en la versión 2.00 (CMSISv2p00_LPC17xx) y nuestro proyecto RobotVigilancia.



Copiaremos todo el contenido en la carpeta workspace creada por el software la primera vez que se inicia o seleccionaremos la ruta que deseemos para el workspace al iniciar el IDE por primera vez. Una vez abierto el IDE, iremos a File -> Import y se abrirá el asistente, seleccionamos General -> ”Existing projects into Workspace” y pulsamos Next, marcamos la opción “Select root directory:” y pulsamos Browse, se abrirá otra ventana donde seleccionaremos nuestra carpeta “workspace”

y pulsamos OK, seleccionamos los proyectos a importar (los tres comentados antes) y pulsamos "Finish".

Compilación

Una vez realizados estos pasos nos aparecerá en el Project Explorer los proyectos importados, seleccionaremos RobotVigilancia y compilaremos en Project->Build All.

Una vez compilado y con la mota conectada mediante el USB - JTAG podremos cargar la aplicación contenida en la carpeta Debug del proyecto, pulsando el botón Program Flash y seleccionando el archivo con formato NombreProyecto.axf.

Una vez terminado este proceso la aplicación quedaría correctamente instalada en la mota.

11.2. Uso del Software de control para PC.

Como se ha comentado anteriormente, se ha creado una aplicación para la vigilancia remota a través de la cámara y el control del robot. A continuación explicaremos como ponerlo en marcha y utilizarlo. Se debe seguir el orden marcado en estas instrucciones para asegurar el correcto funcionamiento.

Requisitos:

Python 2.7 con Tkinter.

Esta es la versión del intérprete en la que está hecho el script. Para usar la versión 3 habría que adaptar el código.

No se tratará la instalación en esta guía, debido a que abunda el material en la red. Una posible guía sería la siguiente:

http://www.ehowenespanol.com/instalar-python-windows-como_44790/

Si se utiliza Linux, normalmente ya viene instalado.

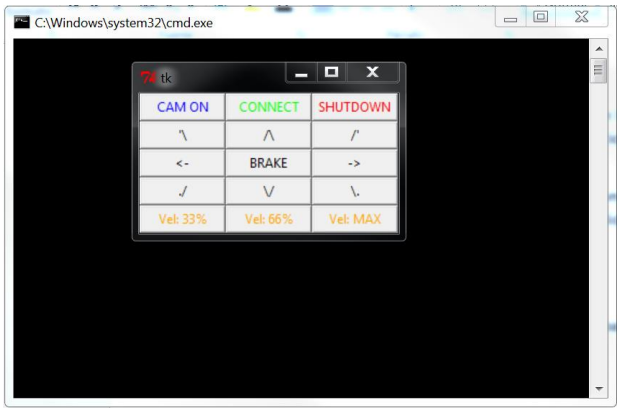
VLC video player.

Necesario para visualizar el Streaming proveniente de la cámara. Puede descargarse de su página web:

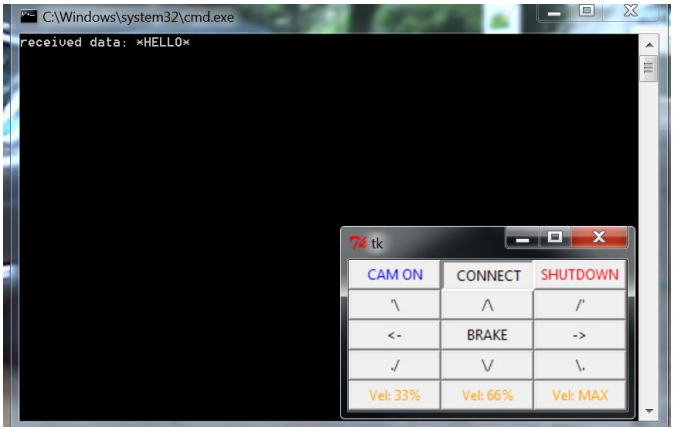
http://www.videolan.org/vlc/

Pasos:

En Windows, con python 2.7 instalado, iniciar RVR.bat, que abre una consola e inicia el script. En Linux ejecutamos el script desde consola, obtendremos un resultado similar al de la imagen. Se trata de la consola que recibirá los mensajes de confirmación de la mota, y de la GUI desde la que se controlará al robot.



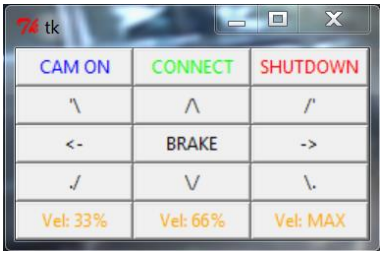
El primer paso es conectar con la mota, la configuración del módulo Wifly para este proyecto es mediante IP estática (192.168.0.200) y el script viene preconfigurado (de cualquier forma se puede cambiar fácilmente configurando el código del proyecto). Para conectar pulsamos el botón CONNECT y esperamos la respuesta del módulo.



El siguiente paso es encender la cámara pinchando CAM ON. La aplicación se bloqueará durante 25 segundos, ya que este es el tiempo de arranque de la cámara, es un comportamiento normal. Una vez finalizado el tiempo de espera se abrirá el VLC automáticamente (NOTA: Puede ser necesario configurar la ruta al VLC dentro del SCRIPT) en la función cámara:

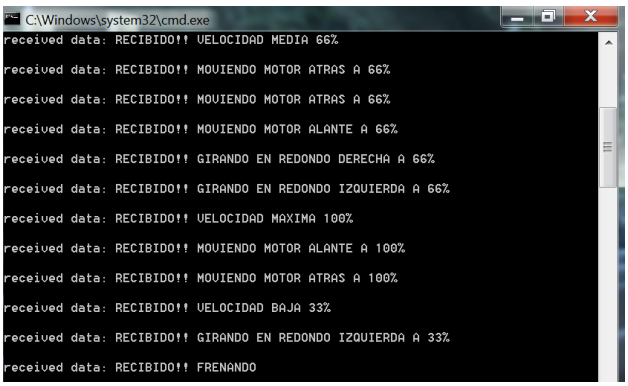
subprocess.Popen(["C:\\Program Files (x86)\\VideoLAN\\VLC\\vlc.exe","rtsp://192.168.0.128:554/12"])

Una vez hecho esto procedemos a mover el robot con normalidad, ya sea mediante un solo clic, o manteniendo pulsado (el tiempo de repetición está configurado a un valor óptimo), pero puede ser cambiado en el script. Los movimientos configurados son adelante, atrás, giro en redondo y diagonales.



Por medio de los botones inferiores se configura la velocidad de movimiento. En la consola podremos ver las confirmaciones, y los mensajes de error si los hubiera (pérdida de paquetes u otro).

Al finalizar se debe cerrar el VLC y el script, el orden es indiferente, pero ambos pertenecen a la misma aplicación y hasta que se cierren la conexión con el Wifly no quedará liberada.



Para cerrar el script se debe pulsar el botón SHUTDOWN que enviará el comando de apagar la cámara, y posteriormente cerrará el script.

11.3. API para el control de motores

Como se ha comentado en el capítulo 4, para controlar las placas de potencia se ha diseñado una serie de librerías que permiten realizar todos los movimientos en motores DC a través de estas enviando señales desde la mota.

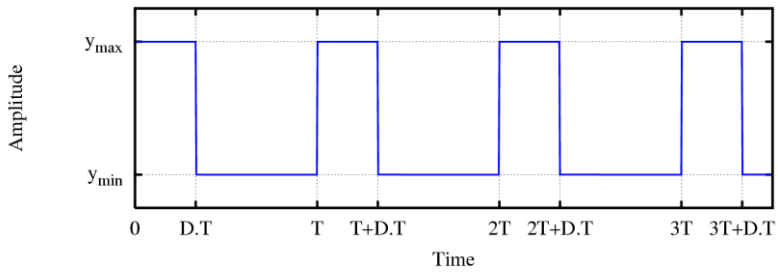
Para conseguir esta funcionalidad se han realizado tres librerías, la primera motor.h, rele.h y pwm.h. La primera es la encargada de combinar adecuadamente las funciones de las otras dos, es decir activar y modificar valores de los puertos PWM y las salidas digitales de la mota para enviar las señales comentadas a las placas de potencia.

Antes de entrar en detalle acerca de las librerías se explicará qué es PWM y para qué se usa en este caso, así comoqué son las entradas/salidas de propósito general de la placa o GPIOs.

PWM:

La modulación por ancho de pulsos (también conocida como PWM, siglas en inglés de pulse-width modulation) de una señal o fuente de energía es una técnica en la que se modifica el ciclo de trabajo de una señal periódica (una senoidal o una cuadrada, por ejemplo), ya sea para transmitir información a través de un canal de comunicaciones o para controlar la cantidad de energía que se envía a una carga.

El ciclo de trabajo de una señal periódica es el ancho relativo de su parte positiva en relación con el período.



La modulación por ancho de pulsos es una técnica utilizada para regular la velocidad de giro de los motores eléctricos de inducción o asíncronos. Mantiene el par motor constante y no supone un desaprovechamiento de la energía eléctrica. Se utiliza tanto en corriente continua como en alterna, como su nombre lo indica, al controlar: un momento alto (encendido o alimentado) y un momento bajo (apagado o desconectado), controlado normalmente por relés (baja frecuencia), MOSFET o tiristores (alta frecuencia).

En nuestro caso la placa de potencia recibirá la señal y actuará igualmente sobre la velocidad de giro de los motores conectados a ella, siendo esta mayor cuanto mayor sea el ciclo de trabajo (o duty cycle).

GPIO:

GPIO (General Purpose Input/Output, Entrada/Salida de Propósito General) es un pin genérico en un chip, cuyo comportamiento (incluyendo si es un pin de entrada o salida) se puede controlar (programar) por el usuario en tiempo de ejecución.

Los pines GPIO no tienen ningún propósito especial definido, y no se utilizan de forma predeterminada. La idea es que a veces, para el diseño de un sistema completo que utiliza el chip, podría ser útil contar con un puñado de líneas digitales de control adicionales, y tenerlas a disposición ahorra el tiempo de tener que organizar circuitos adicionales para proporcionarlos.

Los pines GPIO pueden activarse o desactivarse, y configurarse como entrada o salida. En nuestro caso la combinación de dos GPIO por motor, configurados como salida digital, que pueda tomar valores alto (1, V+) o bajo (0, GND) seleccionarán la una función u otra en la placa de potencia.

Concretamente trabajaremos sobre los pines de PINSEL4, correspondientes a la parte baja del puerto 2 de la mota, a continuación se muestra una tabla con dichos pines:

Table 84. Pin function select register 4 (PINSEL4 - address 0x4002 C010) bit description

PINSEL4	Pin name	Function when 00	Function when 01	Function when 10	Function when 11	Reset value
1:0	P2.0	GPIO Port 2.0	PWM1.1	TXD1	Reserved	00
3:2	P2.1	GPIO Port 2.1	PWM1.2	RXD1	Reserved	00
5:4	P2.2	GPIO Port 2.2	PWM1.3	CTS1	Reserved [2]	00
7:6	P2.3	GPIO Port 2.3	PWM1.4	DCD1	Reserved [2]	00
9:8	P2.4	GPIO Port 2.4	PWM1.5	DSR1	Reserved [2]	00
11:10	P2.5	GPIO Port 2.5	PWM1.6	DTR1	Reserved [2]	00
13:12	P2.6	GPIO Port 2.6	PCAP1.0	RI1	Reserved [2]	00
15:14	P2.7	GPIO Port 2.7	RD2	RTS1	Reserved	00
17:16	P2.8	GPIO Port 2.8	TD2	TXD2	ENET_MDC	00
19:18	P2.9	GPIO Port 2.9	USB_CONNECT	RXD2	ENET_MDIO	00
21:20	P2.10	GPIO Port 2.10	$\overline{\text{EINT0}}$	NMI	Reserved	00
23:22	P2.11[1]	GPIO Port 2.11	$\overline{\text{EINT1}}$	Reserved	I2STX_CLK	00
25:24	P2.12[1]	GPIO Port 2.12	$\overline{\text{EINT2}}$	Reserved	I2STX_WS	00
27:26	P2.13[1]	GPIO Port 2.13	$\overline{\text{EINT3}}$	Reserved	I2STX_SDA	00
31:28	-	Reserved	Reserved	Reserved	Reserved	0

Concretamente hemos utilizado en el proyecto el P2.0 y P2.1 para PWM, y de 2.2 a 2.5 para las salidas digitales HIGH/LOW.

Motor

En motor.h se ha realizado una configuración para controlar dos motores. Esta solución es extrapolable a tantos motores como canales de PWM tengamos.

Para simplificar nos centraremos en uno y explicaremos en detalle qué combinaciones podemos realizar y cuál es la respuesta de la placa en cada caso. A continuación se describen las funciones de esta librería que permitirán realizar lo comentado.

motorSetup()

Inicializa el PWM y los GPIO que se utilizarán mediante las correspondientes funciones de iniciañlizacion de pwm.h y rele.h. Además se crea un semáforo de exclusión mutua.

uint32_t motorI(uint8_t mode, uint32_t percent)

Esta es la función para el motor izquierdo, pero como comentamos anteriormente la solución es igual al derecho, simplemente cambian el canal de PWM y los GPIO.

Configura el ciclo de trabajo del PWM, y las GPIO, para controlar el motor, sentido de giro y velocidad. Esta función recibe como parámetros el modo de funcionamiento del motor, y el valor de offset del PWM. En este caso se trata como un porcentaje ya que el offset toma como valor mínimo 0 y máximo 100.

Los modos de funcionamiento son los siguientes:

0. Desactiva PWM (ENA en placa a 0)
1. Giro sentido agujas de reloj (ENA velocidad, IN1 HIGH, IN2 LOW)
2. Giro contrario agujas de reloj (ENA velocidad, IN2 HIGH, IN1 LOW)
3. Freno (ENA potencia frenado, IN1 LOW, IN2 LOW)

- *ENA: Entrada de la placa de potencia que recibe la señal PWM*
- *IN1: Entrada del primer GPIO*
- *IN2: Entrada del segundo GPIO*

Como se ha mencionado antes, para controlar la señal PWM y las salidas GPIO se utilizan las librerías pwm.h y rele.h respectivamente. A continuación describimos cada una.

Pwm

Por medio de esta librería iniciamos puerto PWM 1 de la mota (PWM1.X). Una vez iniciada podemos configurar el canal que queramos, en el proyecto sólo están predefinidos el 1 y el 2, y asignarle un valor de offset, que marcará el tamaño de la onda cuadrada generada. Las funciones de la librería pwm son las siguientes;

uint32_t PWM_Init()

Inicializa la PWM1 y el semáforo de exclusión mutua, devuelve TRUE si el semáforo fue creado con éxito.

uint32_t PWM_Set(uint32_t ChannelNum, uint32_t offset)

Configura el tamaño de la onda mediante el parámetro offset y el canal de PWM1, en este caso PWM1.1 o PWM1.2.

Si se necesitaran valores más precisos, cambiando el valor de #define ENDPWMVAL en pwm.h podemos cambiar la resolución de offset, por ejemplo definiéndolo como 1000, offset podría tomar valores de 0 a 1000, pudiendo asignar un valor más preciso a la longitud de la onda con respecto al período.

Rele

A través de esta librería configuramos y activamos las GPIO. Las funciones que contiene son las siguientes:

*uint32_t releInit(const char *dispositivo)*

Crea el semáforo e inicia la GPIO correspondiente a cada dispositivo. Los dispositivos están enumerados en rele.h. Concretamente son IN1 e IN2 de los dos motores, y la GPIO de la cámara.

*uint32_t releEstado(const char *dispositivo, uint8_t funcion)*

Cambia el estado de la salida GPIO asociada al dispositivo indicado, a HIGH (1) o LOW (0) dependiendo del valor de “función”.

En rele.h tenemos definidos los pines asígnados a cada dispositivo, así como el número de bit de cada uno, como observamos en la tabla 84 del manual de la mota, que comentamos anteriormente.

11.4. Módulo de cámara OV7670

Para la realización de la parte de del robot correspondiente al video, se estudiaron diferentes opciones, decidiendo finalmente utilizar una cámara IP de seguridad con Wi-Fi integrado, si bien cabe destacar que también existe la opción de utilizar un módulo de cámara para sistemas embebidos como el OV7670.

Las cámaras IP actuales tienen unas características muy superiores a este módulo, pero para ciertas aplicaciones donde no se requiera una gran calidad de imagen, ni un stream de video fluido, podría convertirse en la mejor opción, ya que se trata de un elemento pequeño, muy económico y de bajo consumo, que coincide a la perfección con la filosofía de los sistemas embebidos.

Descripción

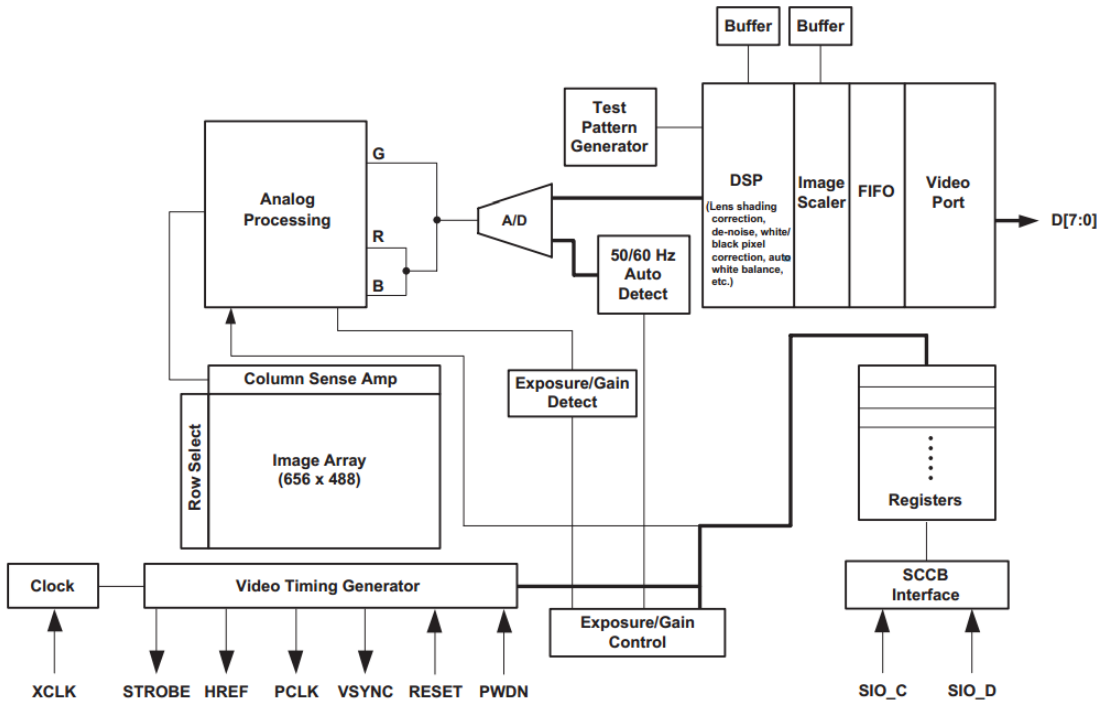
Es un módulo basado en el sensor de imagen CMOS OV760 de Omnivision, de pequeño tamaño, con procesador de imagen VGA y un consumo muy reducido. Existen diversas variantes, todas ellas económicas, aunque la que trataremos en este documento es la que incorpora un buffer FIFO AL422. La diferencia reside en que para capturar video con este módulo es recomendable utilizar una memoria externa, de forma que podamos leer el FIFO a la velocidad adecuada y no necesitemos tanta memoria de la mota para almacenar los datos generados por la cámara.

Características principales:

- Resolución VGA (640x480 pixel)
- Imagen de 8 bits
- Formatos YUV 4:2:2, RGB565, GRB 4:2:2, Raw RGB
- Máxima velocidad de transferencia (30 frames por segundo)
- Diversos filtros y sistemas de corrección de imagen (Exposición, Flicker, UV, etc)
- Consumo de 60mW en uso, y menor a 20uA en espera.
- AL422 3Mbit de DRAM y tiempo de ciclo Read/Write de 20ns

Diagrama de bloques

A continuación podemos observar el diagrama de bloques de las funciones del sensor de imagen de OmniVision:



Compatibilidad

La cámara puede ser usada en sistemas embebidos basados en diversas motas, si bien la calidad y velocidad de la imagen dependerá de la memoria y velocidad de reloj de cada una. Tras estudiar diferentes proyectos concluimos que este módulo es perfectamente compatible con la LPC1769, así como otros sistemas basados microcontroladores como el Arduino, Raspberri Pi o LaunchPad de TI.

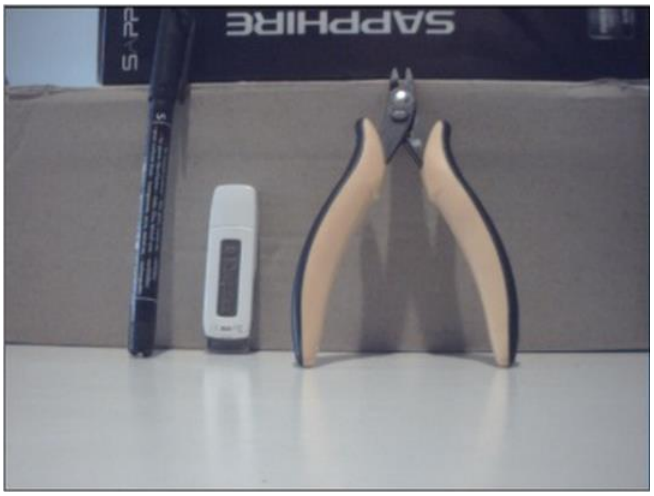
Podemos encontrar código fuente de diferentes proyectos en la red, tanto para la mota usada en este proyecto, como para otros muchos sistemas como los antes mencionados, así como software para PC desde el que podemos visualizar el resultado.

Resultados:

Para concluir mostramos algunos ejemplos de imágenes obtenidas con diversas configuraciones de hardware:

LPC1769, OV7670 sin AL422

PC1769, OV7670 con AL422



Arduino Uno, OV7670 sin AL422

MSP430 LaunchPad, OV7670 con AL422