UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

Grado en INGENIERÍA INFORMÁTICA

TRABAJO FIN DE GRADO

Diseño e implementación de un sistema de alarma para coches.

Miguel Moreno Rodríguez
Tutor: Álvaro Ortigosa Juárez

Junio 2017

Diseño e implementación de un sistema de alarma para coches.

AUTOR: Miguel Moreno Rodríguez
TUTOR: Álvaro Ortigosa Juárez

Dpto. Ingeniería Informática
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Junio de 2017

Resumen (castellano)

En este Trabajo Fin de Grado se cuenta el desarrollo de una sistema de alarma para coche, que servirá para comprender y evaluar el funcionamiento de la misma.

Para conseguir este propósito, se realiza y desarrolla un software, para el tratamiento de señales provenientes del vehículo y la comunicación con el usuario, y un hardware, para la captura y chequeo de distintos sensores pertenecientes al vehículo.

A continuación, partiendo de una idea inicial sobre programación y circuitería integrada, se avanza siguiendo las pautas establecidas por el diseño a través de la programación de una serie de métodos y el uso de módulos hardware para que cumpla con todos y cada uno de los requisitos y objetivos que necesita este trabajo, dentro de las restricciones encontradas.

Por último, una vez logrado lo mencionado anteriormente, se lleva a cabo una serie de pruebas de integración, caja blanca y comunicación donde se interactúa con la aplicación para observar el correcto funcionamiento de la misma. En consecuencia de ello, permite tener en cuenta posibles ideas futuras con la finalidad, durabilidad, seguridad y el deseo de una mejora del diseño del proyecto.

Abstract (English)

This paper describes the design and development processes of a car alarm system, which will serve to understand and evaluate the operation of the same.

To achieve this purpose, software is developed for the treatment of signals coming from the vehicle and communication with the user, and a hardware for the capture and checking of different sensors belonging to the vehicle.

Then, starting from an initial idea about programming and integrated circuitry, progress is made following the guidelines established by the design through the programming of a series of methods and the use of hardware modules to comply with each and every one of the requirements And objectives that this work needs.

Finally, once achieved the aforementioned, a series of tests is carried out where the application is interacted to observe the correct functioning of the same. Consequently, it allows to take into account possible future ideas with the purpose and desire of an improvement of the design of the project.

Palabras clave (castellano)

Hardware, software, GPRS, relé, Raspberry Pi B+, fuente de alimentación, GPIO, UART, conexión asíncrona, Cantata++.

Keywords (inglés)

Hardware, software, relay, RaspberryPi B+, power supply, GPIO, UART, Asynchronous connection, Cantata++.

Agradecimientos

Agradezco a mis padres la ayuda y esperanzas que me dieron en su día para poder empezar este proyecto, ya que me hubiera costado más aun, y sobre todo a mi niña, a la que quiero y admiro por la increíble paciencia que tiene por soportar a tal ingeniero como un servidor.

Y como no, a toda aquella gente que, a lo largo de mi vida, me ha dejado '*cacharrear*' y aprender a su lado, sin ellos esto no habría sido posible.

Gracias.

INDICE DE CONTENIDOS

INDICE DE FIGURAS

INDICE DE TABLAS

1 Introducción

1.1 Motivación

El ser humano, por naturaleza, se ha caracterizado por evolucionar constantemente. Ha conseguido, a través de la indispensable motivación, alcanzar los objetivos marcados que logran mejorar nuestra calidad de vida. El campo de los sistemas de alarma trabaja con la finalidad de ser parte de esta mejora.

Las alarmas son, en gran medida, un gran avance. Este campo expone, por sí mismo, una multitud de características y usos que hacen tenerlas muy en cuenta. Tienen una gran presencia en nuestra sociedad y ofrecen una gran variedad de funcionalidades dependiendo del sistema y/o del medio en donde se instalan (en mi caso será en el interior de un vehículo).

La clave de por qué elegí diseñar una alarma para mi coche surge de la innumerable cantidad de despistes cometidos a lo largo del paso del tiempo, en los que a veces, por llegar cansado a casa no me he fijado en el estado en el que dejaba el vehículo, dudando y pasando malas noches hasta que despertaba, iba a comprobarlo y, a veces tenia suerte y lo había cerrado y otras no. La suerte que tuve fue que nunca tuve un percance, pero siempre tuve ese miedo del qué podría haber pasado.

1.2 Problema actual

Para intentar solventar los problemas planteados en el apartado anterior, tuve que darle varias vueltas de cómo poder solucionar el problema de comunicarme con un coche viejo (del 2001) y poder entenderme con él.

Dado que existen múltiples soluciones, y algunas bastante costosas en cuanto a dinero y esfuerzo, tuve que estudiar los sistemas existentes y cómo adaptarlos al problema que tenía, ya que no cualquiera me valía puesto que debía de poder comunicarme a mucha distancia, por lo que una comunicación Buetooth o WiFi no me valía, y traspasar las barreras y muros de hormigón que hay cuando se deja el coche en un garaje (en mi caso, 2 pisos bajo tierra).

1.3 Mi solución

Como las interfaces que puedan existir para realizar este proyecto serían demasiado costosas, he tenido que estudiar la formar de poder conectar el coche a 'algo' que pueda servirme como interfaz, y en este punto opté por la Raspberry, capaz de recoger señales por cualquier de los GPIO designados para ello.

Pero como la Raspberry no trae unos módulos extra para la comunicación sin cables, había que optar por uno de los siguientes:

- Módulo Buetooth
- Módulo WiFi
- Módulo GSM

De los cuales opté por el GSM, ya que el radio de acción es a nivel mundial, siempre y cuando se disponga de cobertura y saldo. Cosa que el WiFi y Buetooth no pueden hacer.

Para la parte de conexión con el vehículo, opté por un módulo de relés, compatibles con la Raspberry para poder manejar voltajes mayores a los soportados por sus GPIO. De esta forma protegemos la placa frente a subidas de tensión y voltajes superiores a 3.3V.

1.4 Organización de la memoria

Durante la elaboración del presente documento, se ha pretendido que su organización siguiera el orden coherente de la construcción del mismo. A partir de sus ocho distintos capítulos se observa el desarrollo del sistema de alamar para coche:

- Capítulo 1- Introducción

Es el capítulo actual. Se desarrolla una introducción del proyecto explicando de forma breve el tema a tratar, se redactan las motivaciones por las que se ha realizado el presente estudio, se explican los objetivos fundamentales que se pretenden conseguir, se describen los objetivos más particulares que hacen posibles los anteriores y se lleva a cabo la estructura del actual informe.

- Capítulo 2 – Historia y estado del arte.

En esta parte del proyecto se realiza un "viaje" a través de las distintas alarmas disponibles que se encuentran en el mercado, cómo se iniciaron y cómo fueron las primeras alarmas de los vehículos y algunos tipos de alarma usados con Raspberry.

- Capítulo 3 – Análisis

Se describirá un análisis del problema más en profundidad, junto con los requisitos funcionales y no funcionales que debe cumplir, las restricciones con las que cuento para el diseño de este sistema y un detalle del diseño de todo el sistema.

- Capítulo 4 – Diseño

En el presente apartado se explica la composición que forma la estructura del sistema de alarma. Se divide en dos partes, una en hardware y otra en software. En cada una de estas partes se estudian los distintos tipos que derivan de ellas, se describen y analizan los elementos que las componen.

- Capítulo 5 – Desarrollo

En el siguiente capítulo se analizan los ensamblajes e interconexiones del. Se desarrolla el diseño de todo el software de tratamiento, se analizan estructuras de código, las funciones que forman el sistema de alarma y el diseño del hardware.

- Capítulo 6 – Pruebas y resultados

Para esta parte del proyecto se describe como se han realizado los diferentes ensayos experimentales y se explica con todo detalle los resultados obtenidos para los ensayos propuestos. Se puede observar el funcionamiento y casos de prueba del sistema de alarma.

- Capítulo 7 – Conclusiones y trabajos futuros

Se redactan las conclusiones obtenidas a lo largo de la realización de cada una de las partes que dan forma a este proyecto. También, se describen posibles alternativas futuras con el fin y el deseo de una mejora del diseño de este sistema de alarma.

- Referencias

Se registran y se citan todos los elementos de consulta que han sido utilizados para la investigación sobre este trabajo de fin de grado.

- Glosario

Catálogo alfabetizado de las palabras y expresiones utilizadas en la memora de este proyecto.

- Anexos

En esta parte se detallará el manual de instalación y algunas curiosidades respecto al inicio de las alarmas.

2 Estado del arte

2.1 Situación actual de las alarmas para vehículos

Desde el lanzamiento de los primeros vehículos, surgieron de la mano los ladrones de coches. Desde ese momento, el propietario de un vehículo buscaba la manera de proteger su bien de cualquier manera que pudiera impedir o alejar la idea de sustraer y/o robar el vehículo, usando cadenas, bastones de fijación, cortacorrientes y alarmas sonoras.

En la revista Popular Mechanic se publicó un artículo el día 20 de Junio del 1920 que un inventor del estado de Nebraska creo el primer sistema de alarma de vehículos y el diseño era un interruptor de llave (switch) una caja de acero con un bobinado y una sirena. Este sistema estaba puesto en uno de los ejes y al dar vueltas activaba el campo magnético, generando un voltaje que hacía sonar la sirena, claro esto simple y cuando su dueño la activara con el interruptor de llaves.

Pero en los años 70 y 80 las alarmas ya consistían en un dispositivo similar al explicado anteriormente, solo que con varias funciones integradas: cortacorriente, sensor de impacto, sirena, sensor de voltaje y beeper.

Actualmente, se disponen de 2 tipos de alarmas:

1) Pasivas

Las pasivas no utilizan beeper y se activan una vez el auto este apagado y las puertas cerradas. Muchos de estos sistemas se activan cuando la puerta del conductor se cierra.

Si el sistema (alarma pasiva) es *aftermarket* se activa al cerrar la última puerta.

2) Activas

Estas utilizan un beeper o control remoto estos tienen de 2, 3, 4 botones como mínimo y función es activar la alarma cuando el usuario lo desee con el toque de un botón.

Actualmente, las alarmas aportan las siguientes funciones:

- Cortacorriente
- Cierre centralizado
- Accionamiento de luces de intermitencia
- Sirena multitono

[9]

2.2 Alarmas diseñadas con Raspberry Pi

No solo existen alarmas en los coches, sino que en edificios y en ciertos sitios oportunos también las podemos encontrar, ya sean por sensores de presencia, de ultrasonidos, por luz… Hay gran variedad de sistemas.

Pero en este punto, nos centraremos en las alarmas que han sido diseñadas con una Raspberry, puesto que de aquí parte mi idea.

Al tratarse de un miniordenador, la Raspberry es muy versátil, tanto como su vecino *Arduino*. Cuenta con entradas y salidas de señales, entrada de video, sonido, microSD, etc... Y es lo suficientemente potente como para poder usarse como interfaz entre el dispositivo de detección y el usuario.

A continuación se describirán algunas alarmas las cuales he considerado oportunas para centrar mi trabajo.

2.2.1 Alarma con aviso por mail

Dado que la Raspberry puede conectarse a la red, podemos aprovechar los protocolos de comunicación para utilizar nuestro correo electrónico como medio de comunicación con la Raspberry.

Utilizando un sensor de presencia, podemos controlar una habitación, un recinto o incluso una casa entera utilizando varios sensores. De este modo, cualquier intruso podría detectarse y enviarnos un aviso por mail para poder ponernos en contacto con las autoridades o realizar las tareas oportunas que creamos convenientes, como podría ser el cierre de ventanas y puertas (en caso de contar con domótica en el hogar), activar una bocina que provoque sordera temporal o inhiba los sentidos, dejando inconsciente al intruso, etc…

2.2.2 Alarma con video vigilancia

Utilizando una cámara conectada a la entrada HDMI y configurando raspbian para poder operar con ella, se consigue realizar fotos, videos o gifts que nos son enviados a nuestro dispositivo móvil en el momento que se detecta una presencia utilizando algún tipo de sensor sónico, móvil… También se podrá configurar la calidad de las imágenes enviadas para conseguir mayor rapidez y fluidez a la hora de avisa, puesto que no nos interesa que envíe una imagen una gran resolución ya que al pesar tanto sería lento el aviso. Pero en caso contrario, si podríamos almacenar una foto de gran calidad en la Raspberry y enviar una imagen que aporte información sobre el hecho de menos calidad, consiguiendo una mayor eficiencia durante la comunicación.

2.3 Desarrollo de Raspbian

Es una distribución derivada de Debian que está optimizada para el hardware de Raspberry Pi, lanzada durante Julio de 2012.

1. Imagen RaspBian

A la GPU se accede mediante una imagen del firmware de código cerrado (un *blob* binario, en inglés), que se carga dentro de la GPU al arrancar desde la tarjeta SD. El archivo está asociado a los controladores del núcleo Linux que también son de código cerrado. Las aplicaciones hacen llamadas a las bibliotecas de tiempo de ejecución que son de código abierto, y las mismas hacen llamadas a unos controladores de código abierto en el núcleo Linux. La API del controlador del núcleo es específica para estas bibliotecas. Las aplicaciones que usan vídeo hacen uso de OpenMAX, las aplicaciones tridimensionales usan OpenGL ES y las aplicaciones 2D usan OpenVG; OpenGL ES y OpenVG hacen uso de EGL y este último, del controlador de código abierto del núcleo.

El 19 de febrero de 2012, la fundación lanzó un prototipo de imagen de tarjeta SD que almacenaba un sistema operativo y que podía ser cargado en una tarjeta SD. La imagen se basaba en Debian 6.0 (Squezze), con el escritorio LXDE y el navegador Midori, más algunas herramientas de programación. La imagen funcionaba bajo QEMU permitiendo que el Raspberry Pi pudiera ser emulado en otros sistemas.

El 8 de marzo de 2012, la fundación lanzó Raspberry Pi Fedora Remix (actualmente llamada Pidora), que en el momento de era la distribución recomendada por la fundación, y fue desarrollada en la universidad de Séneca, en Canadá. También se propuso crear una tienda de aplicaciones para que la gente intercambiara programas.

El 24 de octubre de 2012, Alex Bradbury, director de desarrollo Linux de la fundación, anunció que todo el código del controlador de la GPU Videocore que se ejecuta en ARM sería de código abierto, mediante licencia BSD modificada de 3 cláusulas. El código fuente está disponible en un repositorio de la fundación en GitHub.

El 5 de noviembre de 2012, Eben Upton anunció el lanzamiento del sistema operativo RISC OS 5 para Raspberry Pi a la comunidad, pudiéndose descargar la imagen de forma gratuita desde la web de la fundación. Su relación con la comunidad RISC OS se remontaba a julio de 2011, cuando habló en ella de una hipotética versión. El sistema operativo incluye una gran cantidad de aplicaciones como *!NetSurf* para la navegación web, *!StrongED* para editar texto, *!Maestro* para editar música, *!Packman* para la gestión de paquetes o una tienda de aplicaciones llamada *!Store* donde se puede encontrar aplicaciones gratuitas o de pago. Además se incluyen manuales para crear aplicaciones en BASIC para el sistema operativo.

El 24 de noviembre de 2012, se anunció en la Minecon de París, el juego *Minecraft: Pi Edition* para Raspberry Pi, basado en la versión *Minecraft: Pocket Edition* para teléfonos inteligentes y tabletas. La descarga se hizo disponible de forma oficial y gratuita por primera vez el 12 de febrero de 2013 desde el blog del juego, como versión 0.1.1 alpha, junto a instrucciones para ejecutarlo en Raspbian Wheezy. Una de las características principales de este lanzamiento es poder interaccionar con el juego mediante programación, con la intención de motivar a los niños a aprender a programar.

El 25 de mayo de 2013, la fundación informó de que se estaba trabajando en una versión del servidor gráfico Wayland para Raspberry Pi, para sustituir al sistema de ventanas X. Con este cambio se lograría suavidad al usar la interfaz gráfica del escritorio, ya que el procesamiento lo realizaría el núcleo de video de la GPU y no la CPU, sin interferir en el renderizado 3D.

El 3 de junio de 2013, fue lanzado en la web de la fundación para su descarga la aplicación NOOBS (New Out of Box Software), utilidad que facilita la instalación de diferentes sistemas operativos para Raspberry Pi. Se distribuye en forma de archivo zip que se copia descomprimido a una tarjeta SD de 4 o más GB, y una vez arrancada la placa con la tarjeta por primera vez, aparece un menú en que se da la opción de instalar una de las diferentes distribuciones en el espacio libre de la tarjeta de memoria, o acceder a internet con el navegador Arora integrado. Más adelante si se desea, es posible acceder a este menú apretando la tecla shift durante el arranque para reinstalar el sistema operativo, elegir otro, o editar el archivo config.txt. NOOBS contiene las distribuciones GNU/Linux de carácter general Raspbian, Arch Linux ARM y Pidora; las distribuciones para mediacenter con Kodi Openelec y RaspBMC; y el sistema operativo Risc OS 5.

El 26 de septiembre de 2013, se añadió a los repositorios de Raspbian una versión oficial de Oracle Java JDK ARM con soporte para coma flotante por hardware, que ofrece bastante más rendimiento que la versión OpenJDK ARM ya existente hasta ese momento y más compatibilidad con aplicaciones. También se anunció que esta versión de Oracle Java JDK se incluiría dentro de la distribución en futuras versiones de Raspbian. [6]

3 Análisis

3.1 Análisis del problema

Una vez se han entendido los conceptos necesarios para poder trabajar con Raspberry y, junto con los conocimientos que he ido desarrollando a lo largo de los años en el ámbito del automóvil (tanto en la parte mecánica como en la eléctrica), comencé a estudiar las dificultades con las que me iba a topar, pues no son pocas dado que en mi carrera (Grado en Ingeniería Informática) no se estudian diversos factores necesarios para la realización del prototipo.

- **COMUNICACIÓN**

Uno de los primero problemas, es la comunicación. Dado que yo necesito una comunicación a larga distancia, la única forma de conseguirla es utilizando la red móvil, no WiFi que apenas funciona a 50m y, mucho menos, si hay hormigón de por medio.

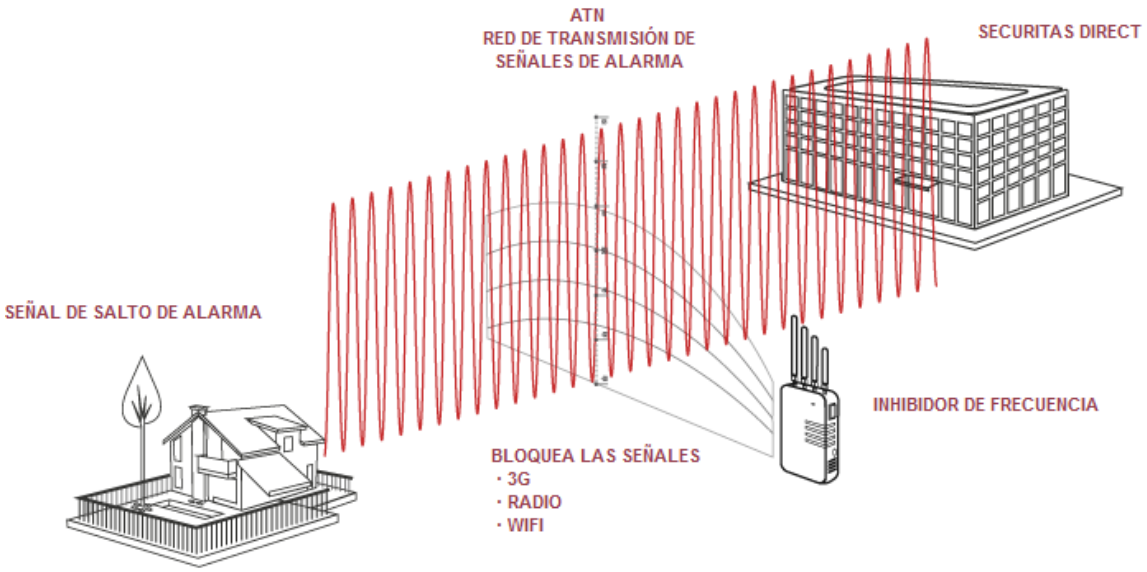
De igual manera sucede con la comunicación vía Buetooth, no es factible pues requiere una distancia menos a la del WiFi.

Así pues decidí, por movilidad y fiabilidad, utilizar una comunicación vía GSM/GPRS. Esta idea fue tomada de las alarmas que se instalan en los hogares, pues ya no se utiliza, o en muy pocos casos y algunos obsoletos, el cable de teléfono, pues puede ser cortado y muchas de las alarmas que lo usan, solo realizan una llamada al día a una determinada hora para realizar el control rutinario de estado.

¿Cuál es el problema de esto? Pues si un ladrón sabe que a las 3 de la mañana, la alarma realiza esta llamada, posteriormente puede cortar el cable con total seguridad pues no va a volver a llamar la alarma, ni la empresa distribuidora va a saber de este incidente, quedando desprotegido el hogar. Por eso se opta por una alarma conectada a la central por GPRS ya que realizan varias llamadas al día, utilizando una tarifa plana que al cliente apenas le supone un coste mayor al que podría tener con una línea de cable telefónico.

Cabe destacar, que estas alarmas tampoco son infalibles pues existen inhibidores de frecuencia que podrían alterar el correcto funcionamiento de la comunicación, dejando desprotegido el hogar hasta que los ladrones salieran, lo que vendría a ser desastroso para el inquilino.

Frente a este inconveniente, poco se puede hacer y la única solución es conseguir un *anti inhibidor de frecuencia*, que consiste en superar las frecuencias de corte que utilizan los inhibidores.



2. Imagen anti inhibidor de frecuencia

Pero para mi propósito, que es la protección de un vehículo, no cuento con este sistema puesto que no es un requisito.

El motivo de la elección del uso de la red GSM está enfocado al mayor alcance del que se dispone, puesto que en muchos parkings subterráneos muchas veces disponemos de una cobertura leve de la cual me voy a surtir para poder tener mi coche controlado. También quiero destacar que es más complicado piratear un SMS que un paquete de red.

- **SEGURIDAD**

Como he mencionado anteriormente, el pirateo de un sms original es posible pero muy costoso. Para este inconveniente he optado por registrar un número de teléfono propio con el cual cotejo la recepción del mensaje.

Es bastante complicado replicar un mensaje de texto utilizando un número de teléfono distinto al que he agregado en la implementación.

3.2 Requisitos

3.2.1 Requisitos funcionales

Los requisitos funcionales de mi sistema deben ser:

- El sistema debe poder comunicarse de manera rápida y eficaz.
- Debe instalarse en un lugar donde no se vea afectada la señal GSM y tenga buena corriente de alimentación.
- El sistema debe cumplir unos requisitos de seguridad con respecto al usuario, es decir, evitar problemas tales como la sobrecarga de circuitos que origine un corto o un fuego por una mala refrigeración del sistema de alarma.
- La solución debe ser capaz de enviar mensajes de texto claros, que indiquen la hora y la incidencia ocurrida o, en caso de utilizar el chequeo, enviar un reporte de estado completo.
- Se deberán de implementar las siguiente funcionalidades:
    - Apertura y cierre de puertas.
        - Debe ser capaz de accionar el sistema de cierre centralizado para abrir las puertas.
        - Debe ser capaz de accionar el sistema de cierre centralizado para cerrar las puertas.
        - Se debe de obtener el estado del sensor de cierre de cada puerta.
    - Apertura y cierre de ventanillas.
        - Debe ser capaz de accionar el motor de elevalunas para bajar las ventanillas.
        - Debe ser capaz de accionar el motor de elevalunas para subir las ventanillas.
        - Se debe establecer un tiempo de funcionamiento del motor elevalunas.
    - Informe de chequeo del estado del vehículo.
        - Debe reportar el estado del cierre centralizado: Abierto/Cerrado.
        - Debe accionar los motores elevalunas.
        - Debe montar un mensaje de texto, conectarse a la red GSM y enviar el mensaje al propietario.
    - Ha de enviar un mensaje de aviso en caso de que la alarma del vehículo se accione por medios ajenos a este sistema, tales como la propia alarma del vehículo.

3.2.2 Requisitos no funcionales

Los requisitos no funcionales de mi sistema son:

- Escalabilidad
    - El diseño debe contemplar el uso óptimo de los recursos del hardware.

- Mantenibilidad
    - La disponibilidad del sistema debe contemplar un periodo de tiempo operativo garantizado, entorno a los 4 días aproximadamente.
    - Ha de ser actualizable de manera sencilla.
    - Deben contemplarse mecanismos frente a fallos de software y hardware, tales como una interrupción de corriente como el aviso en cuanto se obtenga una señal válida de cobertura.
    - El sistema debe ser construido e implantado de tal manera que pueda modificarse de manera sencilla, tanto para la parte hardware como software.

- Seguridad
    - Debe poder ser capaz de parar ataques originados desde dispositivos que no sean de confianza.

3.2.3 Restricciones

Dado que es un sistema de alarma creado por mí y seré quien lo instale, el diseño ha de ser flexible a leves modificaciones puesto que se pueden dar problemas a la hora de su funcionamiento.

Existen restricciones de carácter físico por las cuales posiblemente sea muy complicada la instalación, tales como el hueco en donde se aloja el dispositivo y el conexionado del vehículo.

Se ha de tener en cuenta el alcance de la señal GSM, ya que en algunas partes podemos quedarnos sin cobertura, como en la montaña o un parking subterráneo.

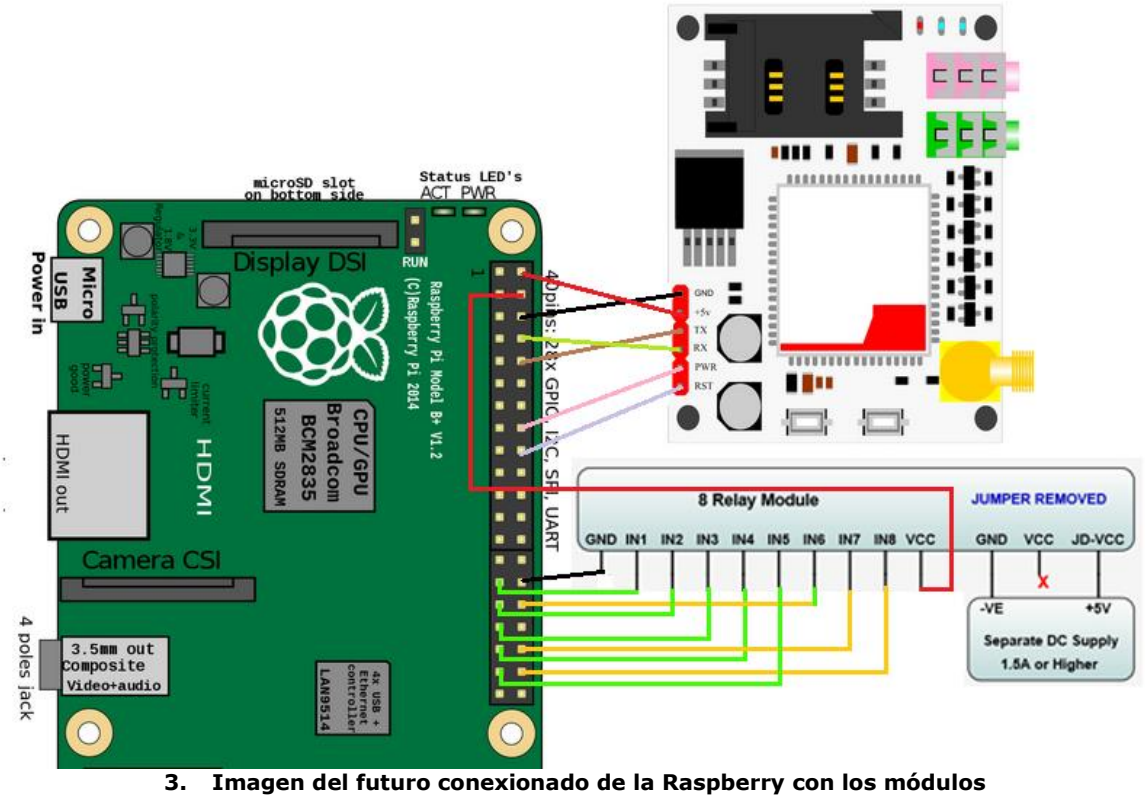
Otra restricción es el tipo de tarjeta SIM que utilicemos. En caso de utilizar una tarjeta prepago, el usuario se verá obligado mantener su saldo cada cierto tiempo. Aun no se ha implementado una funcionalidad que nos avise del saldo existente, pues debiera de ser una consulta directa hacia la compañía y remitir la respuesta al dispositivo del usuario, pudiendo no llegar en caso de superar el límite de saldo. O, en el caso de utilizar una tarjeta de contrato, el usuario se verá más a gusto puesto que el consumo es 0€ (en mi caso).

4 Diseño

4.1 Introducción

Para el diseño de este sistema se ha buscado la facilidad y bajo coste del desarrollo, pero sin dar de lado la eficiencia del producto.

4.2 Hardware del sistema



3. Imagen del futuro conexionado de la Raspberry con los módulos

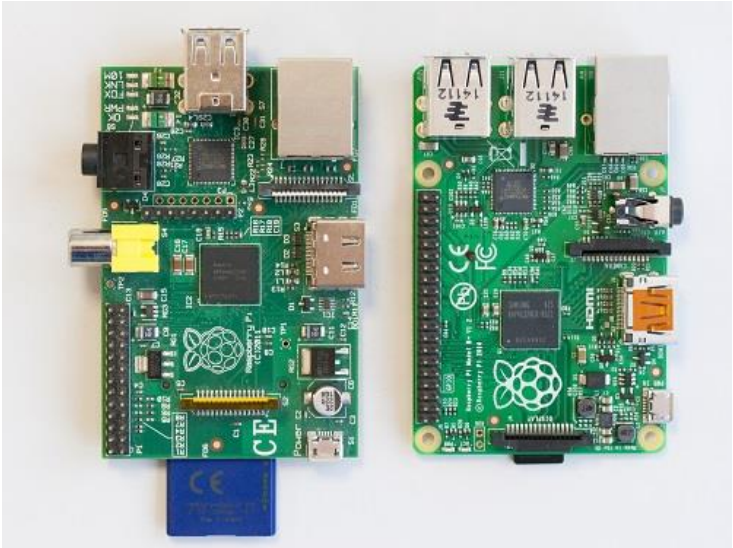
4.2.1 Raspberry Pi B+

Este modelo escogido es la versión final de Raspberry Pi modelo B+ pero se reemplaza por las nuevas Raspberrys: Pi 2 y 3.

No tuvo un gran salto en cuanto al desarrollo del procesador, pero se encuentran grandes cambios en cuanto al formato y diseño, añadiendo entre otras cosas, dos puertos USB adicionales. En cuanto a la visual, se ha reducido el tamaño y modificado las posiciones de los GPIO (entradas/salida).

Algunas de las diferencias significativas:

- **4 puertos USB 2.0** comprado con los 2 en el modelo B.
- Ranura para **Micro SD**. Se sustituye la memoria SD por una micro SD.
- Optimización en **consumo de eléctrico**.
- Mejor audio.
- **Más GPIO**. Pasa de 26 pines a 40 pines.
- Cambio de diseño general en la placa: Ahora **todas las entradas/salidas están distribuidas en sólo dos laterales**.



4. Imagen Raspberry Pi B – RaspberryPi B+

Para poder alimentar a la Pi con estos cambios, se ha rediseñado el circuito de energía y han conseguido una optimización en cuanto consumo eléctrico logrando una reducción del consumo de en torno al 30%, bajando entre 0.5W y 1W.

El cambio de tarjetas SD a micro SD también es destacable. Además, el enganche da un salto de calidad. Esta mejora evitará que sobresalga tanto la tarjeta y los posibles problemas al sufrir golpes en esa zona. El nuevo enganche de la micro SD hace que ésta se quede enganchada y no se extraiga a no ser que presiones hacia dentro. [1]

4.2.2 Módulo EFCom Pro GPRS/GSM

Introducción

Es un módulo inalámbrico muy compacto y fiable.
Compatible con AVR/ARM/FPGA/CPLD.
Su sistema está basado en SIM900 de cuádruple banda GPRS/GSM.

Se configura y controla a través de su UART (Universal Asynchronous Reciver-Transmitter: es el dispositivo que controla los puertos y dispositivos serie) utilizando comandos AT (registrarse a la red, enviar/recibir llamadas/mensajes).

Se utiliza en un amplio abanico de campos, ayudando en los desarrollos de proyectos inalámbricos y de control remoto.

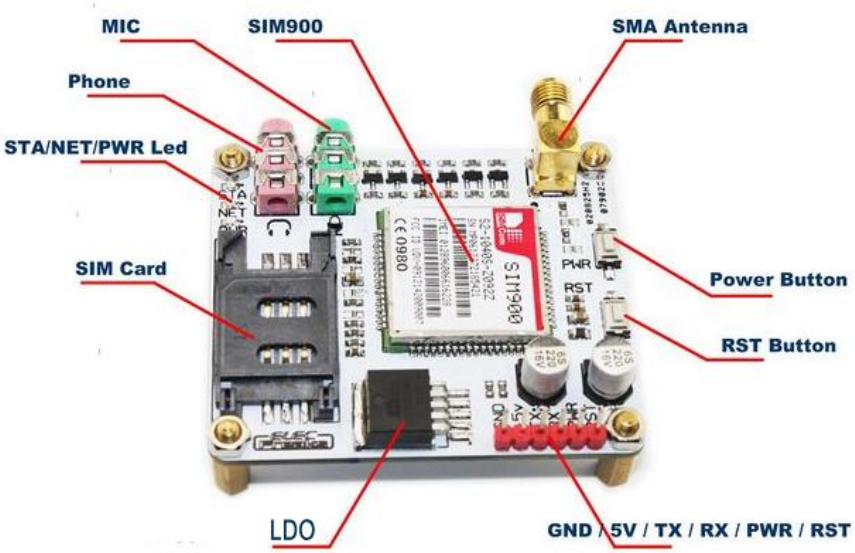
5. Imagen módulo GSM EFcom Pro

[1]

Características principales:

- Alimentación: 5V
- Totalmente compatible con AVR/ARM/FPGA.
- Conexión de puerto serie libre.
- Se puede encender y reiniciar utilizando el PIN de la placa, no solo el botón.
- Quad-band 850/900/1800/1900MHz
- GPRS multi-slot clase 10/8.
- GPRS Mobile station class B.
- Cumple con la fase GSM 2/2 +.
- Control a través de comandos AT.
- Rango de voltaje de alimentación: 3,1 – 4,8V.
- Bajo consumo: 1,5mA
- Temperatura de funcionamiento: -40º a 85ºC
- Dimensiones: 60mm x 53mm

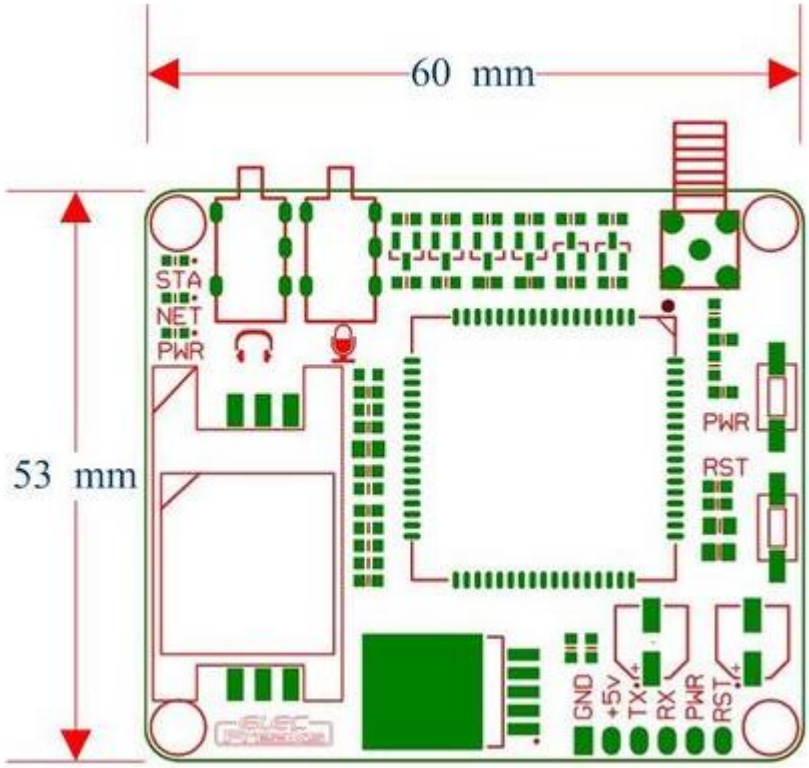
Estudio externo de la placa



6. Imagen módulo GSM EFCom Pro, descripción

- SIM Card: Slot para la colocación de la tarjeta SIM que usaré para registrarme en la red GSM y comunicarme a través de él con la Raspberry Pi.
- MIC & Phone: Slots para un micrófono y un auricular para realizar una llamada telefónica.
- SMA Antenna: Slot para la colocación de la antena para mejorar la señal de red.
- STA/NET/PWR led: Leds indicadores de estado. Se detallan más adelante.
- Power/Resset Button: Botones para el encendido/apagado y reinicio de la placa.
- GND/5V/RX/PRW/RST: Pines de control de la placa.
- LDO: Regulador de tensión.
- SIM900: Chip de comunicación GSM/GPRS.

Dimensiones



7. Imagen dimensiones Módulo GSM EFCom Pro

Indicadores LED

LED	***ESTADO***	***DESCRIPCIÓN***
Status	Off	Encendido
	On	Apagado
Net light	Off	Sim900 no está funcionando.
	64ms ON / 800ms Off	Sim900 no encuentra una red.
	64ms ON / 3000ms Off	Sim900 ha encontrado una red.
	64ms ON / 300ms Off	Comunicación GPRS.

1. Tabla módulo GSM

4.2.3 Módulo de relés

Introducción

Se trata de un módulo de 8 relés opto-acoplados que permiten trabajar a 3,3V (cada relé) utilizando un micro-controlador (en este caso una Raspberry Pi) pudiendo trabajar con voltajes y tensiones muy grandes.

Características principales:

- Módulo con 8 relés opto-acoplados (chip negro de 4 patas).
- Diodos de seguridad.
- 5V entrada.
- Se controla directamente con un micro-controlador.
- Trabaja con AC250V 10A; DV30V 10A.
- Leds indicadores de estado (abierto/cerrado).

Estudio externo de la placa

8. Imagen módulo relés

- GND: toma de tierra.
- VCC: Entrada de tensión, 5v.
- n1 a n8: Entrada de señal n-ésima que activa cada relé.
- Cada relé tiene su entrada/salida.
- Dimensiones: 13,8 x 5,7 x 1,8 cm.

Indicadores LED

LED	***DESCRIPCION***
ON	Paso abierto
OFF	Paso cerrado

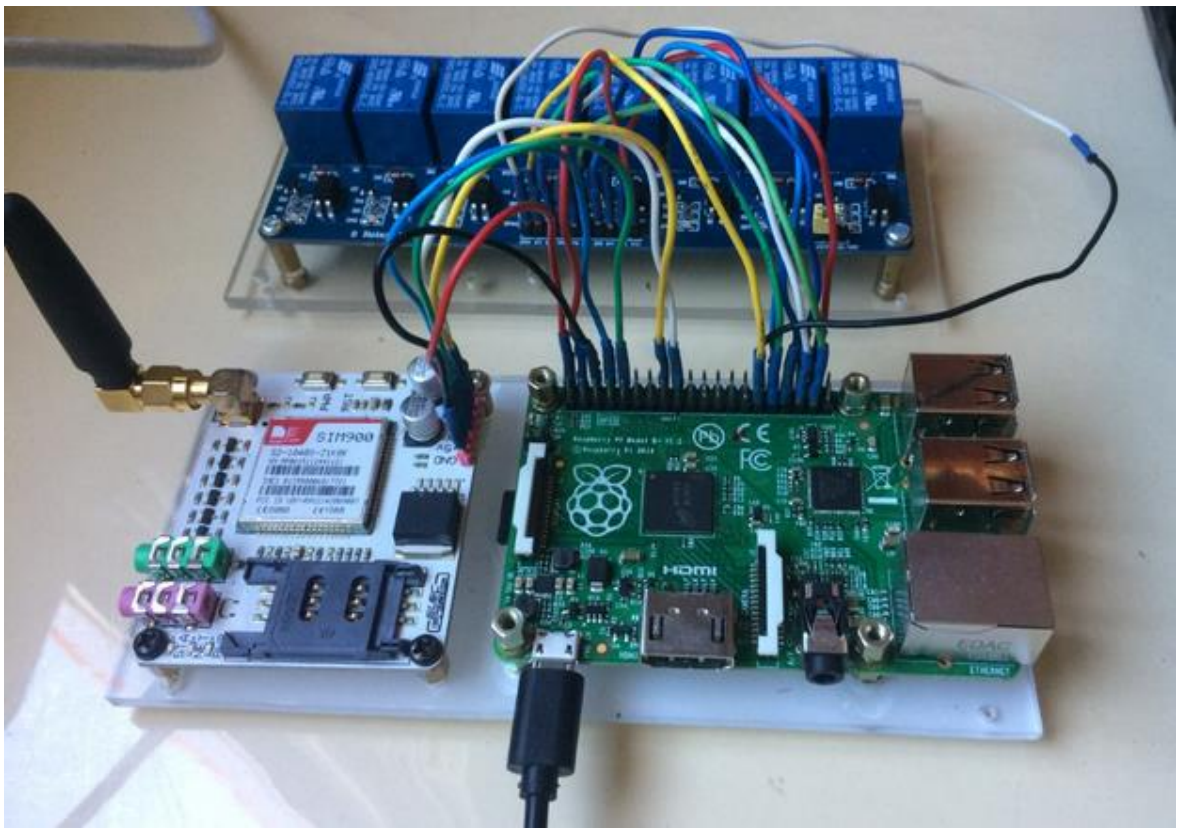
2. Tabla relés

5 Desarrollo

5.1 Introducción

Como el trabajo desarrollado se trata de un conjunto software y hardware, se realizará la instalación sobre una base móvil para poder mostrarla, en lugar de instalarla en su lugar para el que cual fue diseñada, el coche.

A continuación, vemos las primeras conexiones realizadas:



9. Imagen RaspberryPi conectada al módulo GSM y de relés

5.1 Conexionado

El conexionado de las placas es relativamente simple. Solo hay que seguir las conectar los pines oportunos de entre las tres placas: RaspberryPi B+, módulo GSM y módulo de relés.

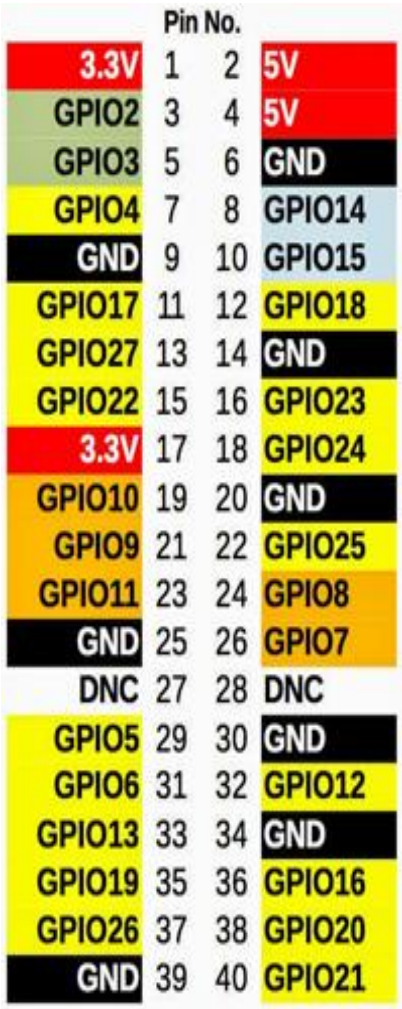
Se ha tratado de implementar de forma simple para evitar cortocircuitos o confusiones a la hora de pinchar cada cable en su sitio correspondiente, de tal forma consigo que queden las conexiones separadas unas de otras y divididas por módulo.

Cabe destacar, que en este conexionado se han empleado cables y conectores reciclados provenientes de fuentes de alimentación de PC.

5.1.1 Esquema de conexiones

A continuación se detallan las conexiones realizadas:

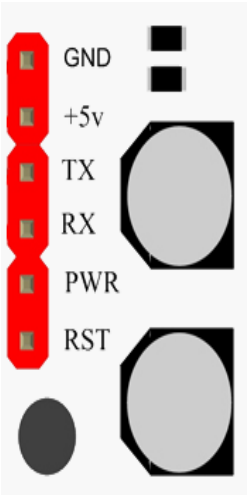
10. Imagen Pines RBPi

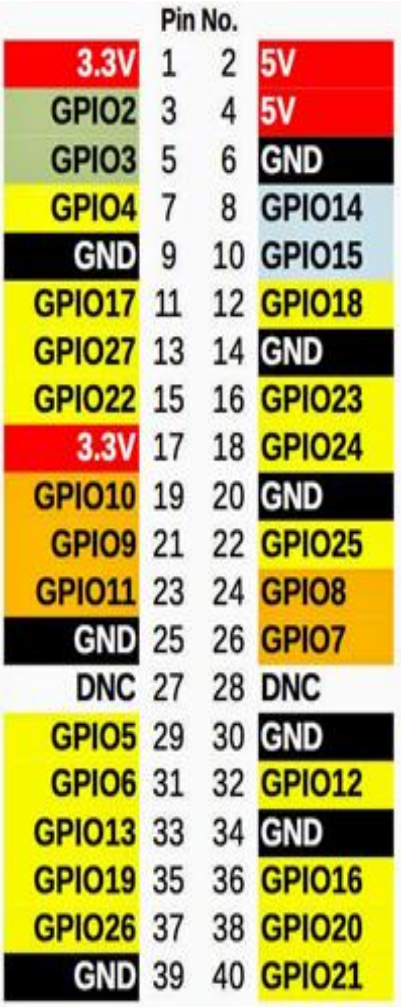


EFCOM PRO GSM		RaspBerri Pi B+	
PIN	***Descripción***	***Descripción***	***PIN***
1	GND	GND	6
2	5V	5V	2
3	Tx	GPIO15	10
4	Rx	GPIO14	8
5	PWR	GPIO23	16
6	RST	GPIO24	18

3. Tabla conexiones RBPi – Módulo GMS

11. Imagen conexiones módulo GSM EFCom Pro





12. Imagen coenxiones RBPi

Módulo relés		**RaspBerri Pi B+**	
PIN	***Descripción***	***Descripción***	***PIN***
1	GND	GND	30
2	5V	5V	4
3	IN1	GPIO5	29
4	IN2	GPIO6	31
5	IN3	GPIO13	33
6	IN4	GPIO19	37
7	IN5	GPIO26	37
8	IN6	GPIO12	32
9	IN7	GPIO16	36
10	IN8	GPIO20	38

4. Tabla conexiones RBPi – Módulo relés

13. Imagen conexiones módulo relés

[6]

5.2 Diseño del software

La parte software ha sido relativamente simple una vez alcancé el conocimiento necesario para trabajar con la Raspberry Pi B+, el módulo GSM y los relés.

Una vez resuelto el problema, realicé el código en varios ficheros:

1) *main.c*
2) *logicData.h*

En *main.c* se ha desarrollado la lógica del programa. ([3], [5], [6])
Se ha estructurado en varias funciones para poder localizar mejor los errores y las partes críticas del software:

- int main();
- void setup();
- void powerOn();
- int8_t enviarComandoAT(const char* comAT, const char* res, unsigned int timeout);
- void recibirSMS();
- void enviarSMS(char* telefono, char* sms);
- void procesarTarea(char* tarea);

El último pasó, ha sido crear un script capaz de arrancar el programa como un *demonio* al efectuar el arranque del sistema.

5.2.1 main()

Parte inicial del programa, en esta parte se crea un log con el nombre ***tfg.log*** para ir escribiendo las llamadas a procedimientos, mensajes de entrada y salida y mensajes de error, si lo hubiera, se comprueba que la librería *wirinPi.h* haya arrancado correctamente y comienza la ejecución del programa con el metodo *setup*.

5.2.2 Setup

Esta función es la encargada del registro del módulo GSM en la red con la que operamos (Movitar, Vodafone, Orange…), pero antes es necesario comprobar que el módulo está operativo y ha arrancado correctamente, de esta parte se encarga la funcion *porweON.*

Se registran las caracteristicas necesarias para que el registro en la red sea lo más eficiente posible, como la velocidad, el pin de la tarjeta, el modo de los mensajes a enviar/recibir, etc...

5.2.3 powerON

Como se mencionó anteriormente, comprobamos que el módulo ha arrancado correctamente enviando el código '*AT*, en caso afirmativo, enviaremos un pulso de luz que nos mostrará la placa y quedamos a la espera de recibir mensajes nuevos mediante el envio del comando '*AT*' cada 2 segundos.

5.2.4 enviarComandoAT

Esta función es la encarga de comunicar con la red para el envio y recepcion de información. Mientras la funcion *Serial.read()* no devuelva un 1, esperaremos la respuesta.

5.2.5 recibirSMS

Esta función es la encargada de obtener el mensaje en bruto recibido por parte del terminal,es decir, nuestro teléfono móvil.

Una vez obtenido el mensaje en *sms*, lo parseamos para poder recoger la información que nos interesa.

Cabe destacar que en esta primera implementación, me hubiera gustado haber utilizado algún tipo de seguridad y/o cifrado, a la hora de enviar los mensajes para conseguir mayor seguridad en el intercambio de información, entre usuario y vehículo. He optado por una comprobación simple, pero difícil de falsear: utilizando el número de teléfono del usuario.

Este número es fijo y se trata de una constante dentro de la implementación. Este modelo de seguridad esta sacado de las nuevas alarmas del hogar, que no van por cable, sino por tarjeta SIM.

Cabe destacar, que se ha pensado en la cobertura movil, para conseguir mayor alcance. Al utilizar un módulo GSM/GPRS consigo mayor cobertura dentro de un parking a dos plantas bajo tierra. Gracias a esta cobertura de distancia, se puede comprobar el estado del vehículo desde cualquier parte del planeta, mientras tengamos cobertura y batería en el vehículo, aunque el consumo del proyecto es muy bajo.

Cuando el mensaje recibido ha pasado el control de identidad, éste es enviado a *procesarTarea*, función encargada de realizar la tarea pertinente que el usuario desea.

5.2.6 enviarSMS

Esta función es la encargada de enviar el mensaje al usuario, respondiendo a su necesidad.

Cabe destacar que, antes de enviar cualquier comando y/o mensaje, debemos registrarnos de nuevo en la red, ya que es un tipo de conexión no persistente.

5.2.7 procesarTarea

Esta función será la encargada de procesar la tarea solicitada por el usuario, tales como:

1) Abrir puerta.
2) Cerrar puerta.
3) Subir ventana.
4) Bajar ventana.
5) Transmitir el reporte destado del vehículo.
6) Avisar mediante un sms cuando la propia alarma del vehículo se accione.

En caso de no haber entendido el comando, se avisará al usuario de que vuelva a repetir la tarea a realizar.

5.3 Protocolo de comunicación

Como se ha mencionado anteriormente, la comunicación utilizada es **UART** *Universal Asynchronous Receiver-Transmitter* (**Transmisor-Receptor Asíncrono Universal**).

Las funciones principales del chip UART son:

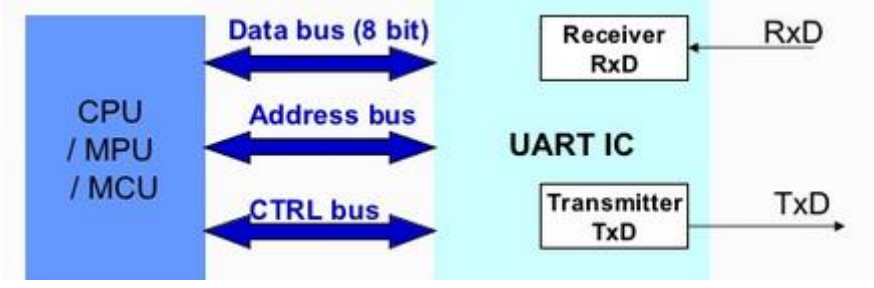
- Manejar las interrupciones de los dispositivos conectados al puerto serie.
- Convertir los datos en formato paralelo, transmitidos al bus de sistema, a datos en formato serie, para que puedan ser transmitidos a través de los puertos y viceversa.

El controlador del UART es el componente clave del subsistema de comunicaciones series de una computadora. Toma bytes de datos y transmite los bits individuales de forma secuencial. En el destino, un segundo UART recompone los bits en bytes completos. La transmisión serie de la información digital (bits) a través de un cable único u otros medios es mucho más efectiva en cuanto a costo que la transmisión en paralelo a través de múltiples cables.

Normalmente no genera directamente o recibe las señales externas entre los diferentes módulos del equipo. Usualmente se usan dispositivos de interfaz separados para convertir las señales de nivel lógico del UART hacia y desde los niveles de señalización externos.

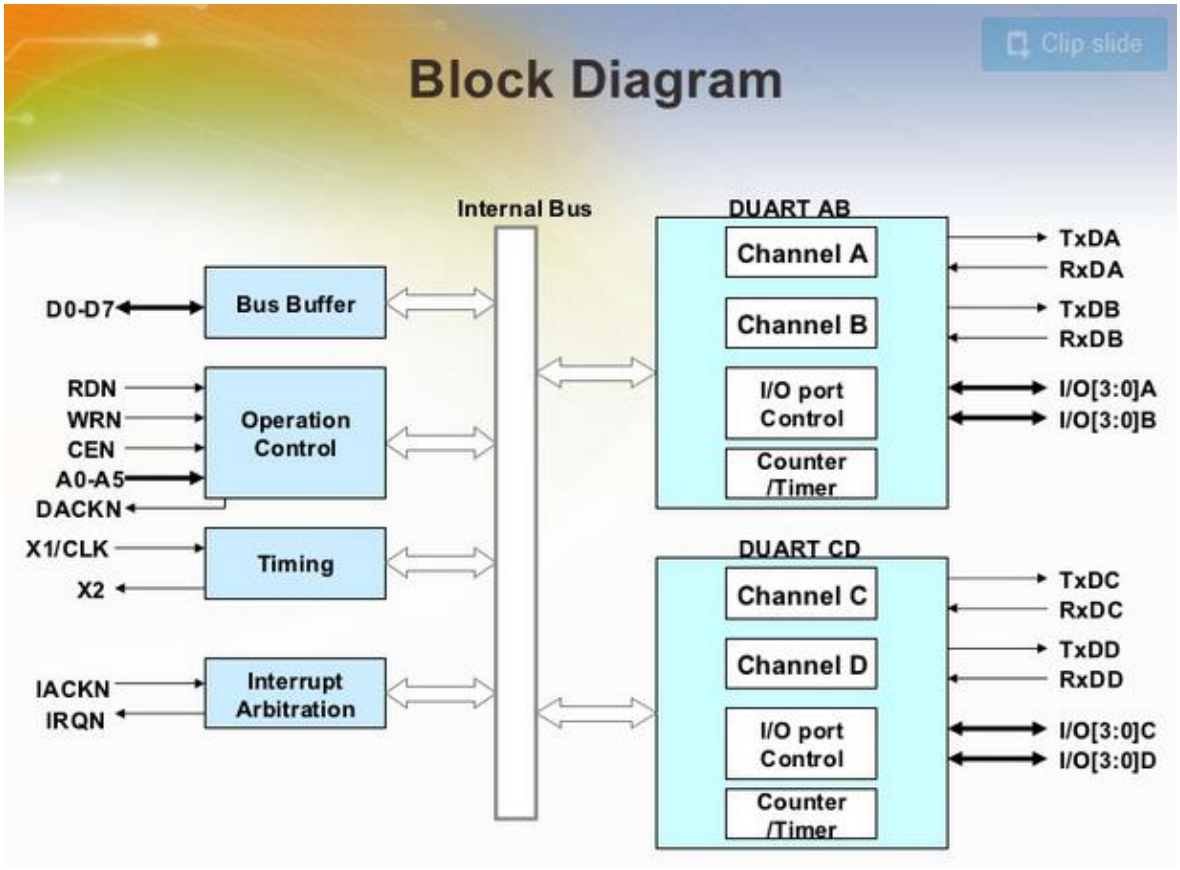
14. Imagen Chip UART 8250

Como vemos en la imagen, este es el chip UART encargado de la transmisión y recepción. El 8250 incluyó en el chip un generador de bit rate programable, permitiendo el uso, tanto de los bits rates comunes, como los de propósitos especiales, que podían ser precisamente derivados desde una arbitraria frecuencia de referencia de oscilador de cristal. Particularmente, el 8250 original podía repetir la transmisión de un carácter si la línea Clear To Send (CTS) fuera activada asincrónicamente durante el primer intento de transmisión.



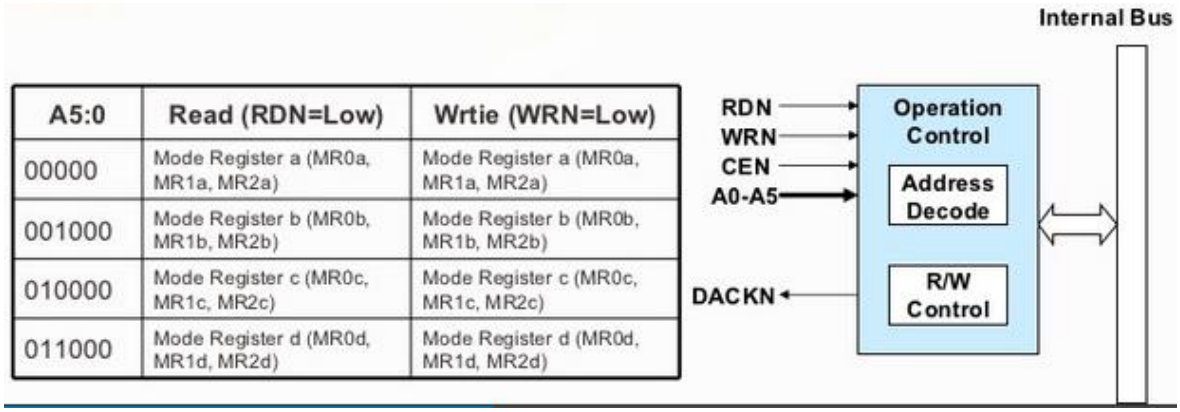
15. Imagen arquitectura UART 8250

Se utiliza para comunicaciones en serie, usualmente a través de un cable, y proporciona una conversión de datos paralelo a serie y serie a paralelo tanto en las secciones de transmisión como de recepción.



16. Imagen arquitectura QUART 8250

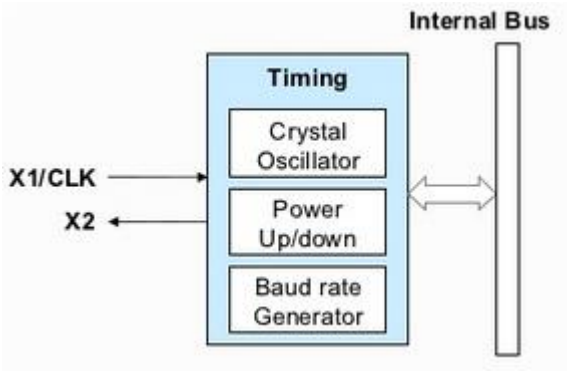
El bloque de control (**Operation Control**) recibe comandos de operación desde la Raspberry Pi y genera señales apropiadas a secciones internas para controlar el funcionamiento del dispositivo.



A5:0	Read (RDN=Low)	Wrtie (WRN=Low)
00000	Mode Register a (MR0a, MR1a, MR2a)	Mode Register a (MR0a, MR1a, MR2a)
001000	Mode Register b (MR0b, MR1b, MR2b)	Mode Register b (MR0b, MR1b, MR2b)
010000	Mode Register c (MR0c, MR1c, MR2c)	Mode Register c (MR0c, MR1c, MR2c)
011000	Mode Register d (MR0d, MR1d, MR2d)	Mode Register d (MR0d, MR1d, MR2d)

17. Imagen arquitectura QUART 8250

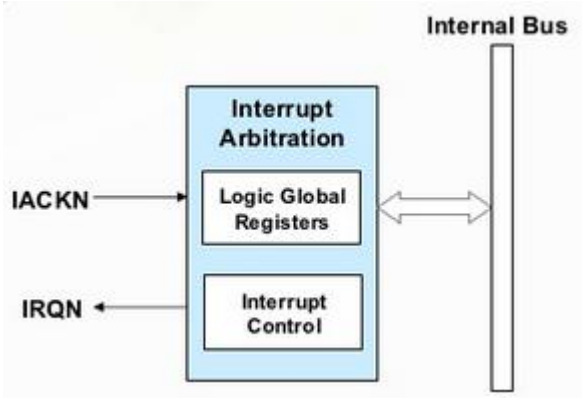
El bloque de sincronización (**Timing block**) opera directamente con un cristal de 3.6864MHz conectado a través de las entradas X1/CLK y X2. Contiene un generador de velocidad en baudios (opera desde el oscilador o entrada de reloj externo, capaz de generar 18 velocidades de transmisión de comunicaciones de datos comúnmente utilizadas, que van de 50 a 38.4K baudios) y contiene la lógica la lógica arrancar o suspender.



18. Imagen arquitectura QUART 8250

El bloque de interrupción (**Interrup arbitration**) está diseñado con dieciocho fuentes que pueden causar una interrupción:

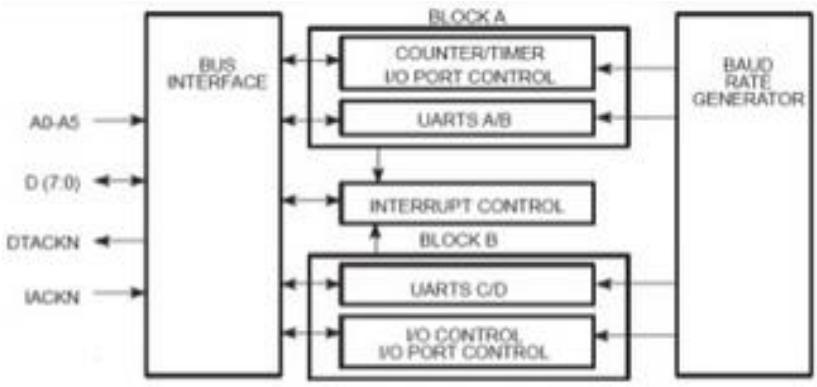
- Cuatro datos de receptor FIFO funciones llenas:
- Cuatro funciones de detección BREAK del receptor.
- Cuatro funciones FIFO de transmisión disponible.
- Cuatro detectores de cambio de estado.
- Dos contadores y/o temporizadores.



19. Imagen arquitecturaQ UART 8250

El bloque de canales se divide en los siguientes puntos descritos:

1) Transmisor.
    - Acepta datos paralelos de la CPU y, convierte y envia en un flujo de bits en serie al pin de salida *TxD*.
2) Transmisor FIFO de 8 Bytes.
    - Los datos se obtienen del registro de desplazamiento de transmisión.
3) Receptor:
    - Acepta datos en serie en el pin *RxD* y, convierte y envia los datos al CPU.
4) Receptor FIFO de 8 Bytes.
    - Los datos se obtienen del registro de desplazamiento de recepción.
5) Control de entrada/salida.
    - Hay 16 pines de uso, 4 por cada UART
6) Contador/reloj.
    - Contador: tiempo de espera.
    - Temporizador: onda cuadrada.
    - Time-out: controla el tiempo entre los datos recibidos.



20. Imagen bloque QUART 8250

REFERENCIAS [7] Y [8]

5.4 Función principal

La función principal de este proyecto es mantener una conexión segura con el vehículo y poder alertarnos en todo momento sobre el estado en el que se encuentra.

Para realizar este propósito, es necesario conectar este proyecto al vehículo. Hubiera sido más eficiente si se pudiera tocar el ordenador del coche: la centralita, ya que es esta la encarga de hacer funcionar el vehículo (desde abrir una puerta con la pulsación de un botón a distancia hasta saber en qué momento inyectar una gota de carburante en el cilindro).

Pero al ser demasiado costoso y muy complicado (de momento), se ha optado por un conexionado ajeno a ella y directo a sensores que permiten darnos dicha información buscada si sabemos cómo obtenerla. Así bien, comenzaré a explicar los conexionados que se han de realizar:

- Control de las puertas: Abierto y cerrado.

Para saber si el coche está abierto o cerrado, basta con saber cuál es el sensor de cierre de la puerta. Estos suelen cerrar un circuito mientras la puerta está abierta, y abrirlo en caso contrario.

De esta manera y pinchando sensor a la Raspberry, podemos saber con un **1** cuando el coche está abierto y, con un **0**, cuando está cerrado. También utilizaremos parte de este circuito para abrir o cerrar el cierre centralizado del vehículo, solo que deberemos de pinchar justo a la salida del interruptor de cierre interno.

- Control de ventanas: Subidas y bajadas.

Para la funcionalidad de la bajada y subida de ventanas, se ha usado la misma filosofía anterior, salvo que en este caso tendremos que mantener una pulsación de 5-10 segundos al cable que acciona el motor del elevalunas. Como dichos motores no están polarizados, dependiendo de cómo enviemos la corriente, subirá o bajará la ventana. Uno de los posibles problemas a la hora de realizar esta implementación, ha sido cuándo saber en qué momento parar, puesto que no podemos saber a qué altura se encuentra el cristal de la ventana. Pero como los vehículos nuevos llevan un sensor de presión que corta la corriente, me he decantado por enviar un pulso del mayor tiempo de trabajo (tiempo total de subir y bajar ventana).

- Estado del vehículo.

Para realizar este paso, me he decantado por el uso de varios sensores, unidos todos en una misma entrada, puesto que cuando el vehículo este cerrado, ningún sensor enviará corriente a la Raspberry. Pero, en el momento que se habrá una puerta o salte la alarma, lo detectará y nos avisará.

Esta implementación se ha conseguido utilizando el mismo sensor de cierre anterior, dado que nos dará el estado de los sensores de las puertas, de la luz del plafón o luz interior, ya que es otro indicativo de vehículo abierto, y de la señal que envía la alarma del vehículo a la bocina o claxon. Obviamente, si salta la alarma nos enteraremos por el ruido, pero esta conexión está pensada para cuando la alarma salte muy lejos de nosotros y no escuchemos nada ni tengamos a un vecino cerca que nos pueda alertar.

6 Pruebas y resultados

Las siguientes pruebas se han realizado en un entorno controlado y con buena señal de cobertura.

No se ha probado con el vehículo, pero si se han simulado todas las conexiones descritas anteriormente.

6.1 Desarrollo de las pruebas

A continuación se detallan las pruebas que se han realizado para comprobar la comunicación entre el sistema y el usuario, de tal manera que en todos los casos correctos hemos obtenido, tanto una respuesta vía sms como la comprobación visual de que realiza lo pedido en la placa de entrenamiento. En estas pruebas se utilizará un led como indicador de la realización de la tarea pedida.

En los casos *apertura y cierre de puertas* se hace lucir el led simulando la señal que acciona el motor del cierre centralizado. Para los casos de *apertura y cierre de ventanas* se hace lucir un led durante un intervalo de tiempo específico y similar al que tardaría una ventana en subir y bajar. Para el caso de intermitencia y claxon, se simulará mediante 2 leds la intermitencia y un pequeño zumbador de ordenador el pitido.

Para el caso del disparador de la alarma del vehículo se simula desconectando un cable de la placa de ensayo, provocando una señal de entrada que estimula el sistema y envíe el mensaje de alerta.

Dado que mi trabajo consiste en la realización de *Unit Tests* para una empresa importante, he tenido la oportunidad de realizar las pruebas de caja blanca para comprobar toda la funcionalidad de los métodos está controlada y funciona según lo esperado utilizando una distribución privada de Eclipse: **Cantata++**. Con ello he podido realizar pruebas parar obtener un 100% en todos los casos tanto de cobertura como de decisión y retorno.

7 Conclusiones y trabajo futuro

7.1 Conclusiones

Con este proyecto he aprendido mucho sobre cómo manejar una Raspberry y cómo utilizar la red GSM sin necesidad de un teléfono.

He superado muchos problemas de diseño e implementación, pues muchas cosas no se han enseñado en la carrera o simplemente se han visto por encima. Por suerte, el mundo hardware siempre me fascinado y nunca me he detenido en mis intentos, por muy complicados que fueran.

He aprendido mucho sobre cómo se transfieren los mensajes de otra manera que no sea por una red WiFi o Ethernet como se estudia en la carrera, y es muy interesante, dado que es muy similar a escribir instrucciones en un terminal, solo que en este caso las escribimos en un mensaje que es enviado por ondas.

He realizado un gran esfuerzo para conseguir la mayor integración posible con el coche, pero muchas funciones no descritas en este documento son inviables pues se requieren otros tipos de métodos de comunicación. Al hablar de estas funcionalidades, me refiero a la nueva función que aportan algunos BMW de clase alta. Consiste en poder arrancar, mover hacia adelante y atrás y apagar el vehículo simplemente con el mando del coche.

21. Imagen llave BMW i8

No es que sea una funcionalidad necesaria, pero nunca viene mal para ajustar nuestro coche en un sitio estrecho, pero desde fuera.

Por último, decir que me he sentido muy a gusto con el trabajo, pues es una idea que he conseguido desarrollar de principio a fin, pudiendo incluso hacer uso de ella. También, que los conocimientos aprendidos son muy valioso para futuros trabajos orientados a la domótica industrial y/o del hogar.

7.2 Trabajo futuro

El trabajo que se espera realizar sobre la herramienta en un futuro se centrará en el poder ampliar sus funcionalidades.

Se intentará aumentar el porcentaje de cobertura, pudiendo emplear en lugar de la red GSM se utilice la red por satélite, con mucho mayor radio de cobertura hasta en los lugares más inhóspitos.

También se intentará garantizar la seguridad en la transmisión de datos desde el terminal de usuario, realizando una aplicación que encapsule el paquete, enviando un mensaje cifrado en SHA256.

Se pretenderá añadir funcionalidades nuevas a medida que estén disponibles en nuevos vehículos, pero sin salir de la línea inicial, la cual pretende mantener y perpetuar la seguridad del interior del vehículo. Tales como incluir cortacorrientes en diversos componentes eléctricos del propio vehículo para evitar que puedan ponerlo en marcha, tales como cortacorrientes o bloqueos hidráulicos accionados por el sistema.

Y, como punto fijo de desarrollo, se desarrollará una configuración muy bien medida para conseguir optimizar el consumo de energía, haciendo que sea obtenga lo necesario de la batería del vehículo sin llegar a agotarla en un periodo de tiempo que sea superior a una semana, pues si consumismo toda la batería no podremos arrancarlo.

A raíz del problema de la batería, se ha pensado en incluir un arrancador de coche eléctrico. Estos arrancadores ya no son como eran antes, del tamaño de 3 cajas de zapatos, sino que ahora son del tamaño de un teléfono móvil. Con ellos, podríamos 'pincharlos' en el módulo de relés, activando este sistema en el momento en el que veamos que no podemos arrancar el vehículo por falta de energía.

22. Imagen arrancador portatil de coche

Referencias

[1] http://www.elecfreaks.com ‹ Elecfreaks › 2015 [En linea]
[2] http://www.ebay.es ‹ eBay › 2015 [En linea]
[3] http://www.hackster.io ‹ hackers › 2016 [En linea]
[4] https://www.ibaivalencia.com ‹ ibaivalencia › 2016 [En linea]
[5] https://hipertextual.com ‹ hipertextual › 2016 [En linea]
[6] https://www.Raspberrypi.org ‹ Raspberrypi › 2016 [En linea]
[7] https://es.wikipedia.org/wiki/Universal_Asynchronous_Receiver-Transmitter ‹ Wikipedia › 2017 [En linea]
[8] http://www.ti.com/lit/ug/sprugp1/sprugp1.pdf ‹ ti groups › 2017 [En linea]
[9] https://ortizayala.wordpress.com/2011/02/26/hello-world/ ‹ ortizayala wordpress › 2017 [En linea]
[10] https://github.com/micahwalter/arduino-pi/blob/master/arduPi ‹ Micah Walte › 2017 [En linea]
[11] https://geekytheory.com/tutorial-Raspberry-pi-gpio-parte-1-control-de-un-led ‹ geekytheory › 2017 [En linea]

Glosario

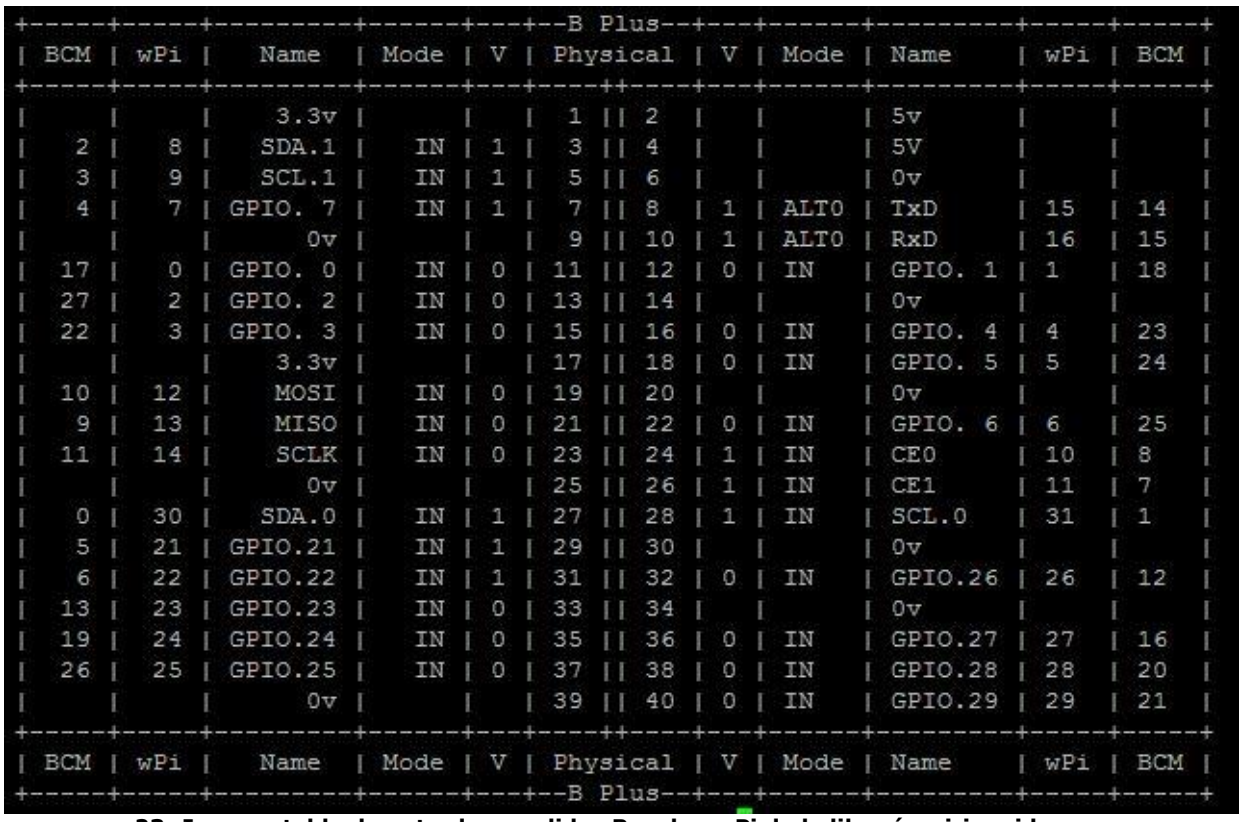
Aftermarket	Segunda mano.
Beeper	Dispositivo pequeño que permite la recepción de mensajes y emite un pitido (*beep*) cuando lo recive.
Buetooth	Es una especificación tecnológica para redes inalámbricas que permite la transmisión de voz y datos entre distintos dispositivos mediante una radiofrecuencia segura.
Cantata++	Distribución comercial para la realización de tests unitarios en c++.
Gift	Imagen animada
GPIO	Conexión de entrada o salida, según se configure en la Raspberry.
GPRS	Servicio general de paquetes vía radio. Extensión de la transmisión de datos GSM.
GSM	Sistema de radiotelefonía celular digital europeo.
Hardware	Conjunto de elementos físicos o materiales que constituyen una computadora o un sistema informático.
HDMI	Es una norma de audio y vídeo digital cifrado sin compresión apoyada por la industria para que sea el sustituto del euroconector.
MicroSD	Formato de tarjeta de memoria flash más pequeña que la miniSD.
SIM	Módulo de identificación de abonado, es una tarjeta inteligente desmontable usada en teléfonos móviles y módems
Software	Conjunto de programas y rutinas que permiten a la computadora realizar determinadas tareas.
Switch	Interruptor analógico.
UNIX	Sistema operativo portable, multitarea y multiusuario
WIFI	Mecanismo de conexión de dispositivos electrónicos de forma inalámbrica

Anexos

A Manual de instalación. Conexionado

Para no hacer una memoria demasiado extensa, la parte de explicación de cómo instalar este sistema de alarma en un vehículo se realizará durante la presentación de manera oral.

B Conexiones wiringPi.h



BCM	wPi	Name	Mode	V	Physical	V	Mode	Name	wPi	BCM
		3.3v			1 || 2			5v		
2	8	SDA.1	IN	1	3 || 4			5V		
3	9	SCL.1	IN	1	5 || 6			0v		
4	7	GPIO. 7	IN	1	7 || 8	1	ALT0	TxD	15	14
		0v			9 || 10	1	ALT0	RxD	16	15
17	0	GPIO. 0	IN	0	11 || 12	0	IN	GPIO. 1	1	18
27	2	GPIO. 2	IN	0	13 || 14			0v		
22	3	GPIO. 3	IN	0	15 || 16	0	IN	GPIO. 4	4	23
		3.3v			17 || 18	0	IN	GPIO. 5	5	24
10	12	MOSI	IN	0	19 || 20			0v		
9	13	MISO	IN	0	21 || 22	0	IN	GPIO. 6	6	25
11	14	SCLK	IN	0	23 || 24	1	IN	CE0	10	8
		0v			25 || 26	1	IN	CE1	11	7
0	30	SDA.0	IN	1	27 || 28	1	IN	SCL.0	31	1
5	21	GPIO.21	IN	1	29 || 30			0v		
6	22	GPIO.22	IN	1	31 || 32	0	IN	GPIO.26	26	12
13	23	GPIO.23	IN	0	33 || 34			0v		
19	24	GPIO.24	IN	0	35 || 36	0	IN	GPIO.27	27	16
26	25	GPIO.25	IN	0	37 || 38	0	IN	GPIO.28	28	20
		0v			39 || 40	0	IN	GPIO.29	29	21

23. Imagen tabla de entradas y salidas RaspberryPi de la librería wiringpi.h

C Código fuente del programa

```
/************
* LIBRERIAS *
*************/
#include "arduPi.h"
#include <wiringPi.h>
#include <iostream>
#include <fstream>
#include <string.h>

/************
*  DEFINES  *
*************/
/* Configuracion del dispositivo */
#define VELOCIDAD 115200
#define PIN "2011"
#define NUM_TEL "608806226"
#define ON_MODULE_PIN 2

/* Comandos de entrada al dispositivo */
#define CMD_IN_N1 = "CERRAR"             // Activar cierre centralizado
y subir ventanillas
#define CMD_IN_N2 = "ABRIR"                    //   Desbloqueo   de   las
puertas y bajar ventanillas
#define CMD_IN_N3 = "BAJAR VENTANAS"     // Bajar ventanillas
#define CMD_IN_N4 = "SUBIR VENTANAS"     // Subir ventanillas
#define CMD_IN_N5 = "ESTADO"             //   Chequea   y   transmite   el
estado  del  vehiculo:  estado  ventanas,  estado  cierre,  estado  alarma
vehiculo.
#define CMD_IN_N6 = ""                         // SIN DEFINIR
#define CMD_IN_N7 = ""                         // SIN DEFINIR
#define CMD_IN_N8 = ""                         // SIN DEFINIR
```

```
/**************************
* DECLARACION DE FUNCIONES *
**************************/
int8_t enviarComandoAT(const char* comAT, const char* res, unsigned int
timeout);
void powerON();
void enviarSMS(char* telefono, char* sms);
void recibirSMS();


/************************
*   VARIABLES GLOBALES  *
************************/
using namespace std;
ofstream registro;
int8_t respuesta;
char aux[30];

/***************************************************************************
* Nombre: setup
* Descripcion: funcion de "espera y respuesta". Se mantiene a la espera
de la recepcion de un SMS. En el momento de la recepcion, realizará las
instruccines pertinentes.
* Retorno: (-)
***************************************************************************/
void setup()
{
    registro << "Seleccionando puerto y velocidad de salida\n";
      pinMode(ON_MODULE_PIN, OUTPUT);
      Serial.begin(VELOCIDAD);

      // COMPROBANDO ESTADO DEL MODULO GPRS
      powerON();
      delay(3000);

      // Asignamos el PIN de la tarjeta SIM
    registro << "Inserccion PIN de la tarjeta\n";
      sprintf(aux, "AT+CPIN=%s", PIN);
      enviarComandoAT(aux, "OK", 2000);
      delay(3000);

      // Nos registramos en la red del operador (Movistar)
    registro << "Registrando tarjeta en la red\n";
    while( (enviarComandoAT("AT+CREG?", "+CREG: 0,1", 500) ||
            enviarComandoAT("AT+CREG?", "+CREG: 0,5", 500)) == 0);

      // Configuracion del SMS en modo TEXTO
    registro << "Configurando modo de mensaje a texto\n";
      enviarComandoAT("AT+CMGF=1", "OK", 1000);

    recibirSMS();
}
```

```
/**************************************************************************
* Nombre: enviarSMS
* Descripcion: Se encarga de realizar las tareas necesarias para mandar
* un sms:
*           - Insertar el PIN de la SIM
*           - Registrar la tarjeta SIM en la red del operador
*           - Enviar un sms a un destinatario
* Retorno: (-)
**************************************************************************/
void enviarSMS(char* telefono, char* sms)
{
      // Envio del SMS
    registro << "Enviando SMS\n";
      sprintf(aux, "AT+CMGS=\"%s\"", telefono);
      respuesta = enviarComandoAT(aux, ">", 2000);

      if(respuesta == 1)
      {
            Serial.println(sms);
            Serial.write(0x1A);
            respuesta = enviarComandoAT("", "OK", 20000);
            if(respuesta == 1)
                  registro << "Mensaje enviado\n";
            else
                  registro << "Error al enviar el mensaje\n";
      }
      else
      {
            registro << "Error al enviar el mensaje\n";
      }
}
```

```
/*************************************************************************
* Nombre: procesarTarea
* Descripcion: Se encarga de las tareas necesarias para leer un sms y
realizar las tareas pertinentes
*           - Insertar el PIN de la SIM
*           - Registrar la tarjeta SIM en la red del operador
*           - Procesar el SMS recibido realizando las operaciones
* correspondientes
* Retorno: (-)
*
*************************************************************************/
void procesarTarea(char* tarea)
{
      if(strcmp(tarea, CMD_IN_N1))
    {
        pinMode(5, OUTPUT);
            digitalWrite(5, HIGH);
            delay(1000);
            digitalWrite(5, LOW);
            enviarSMS(numeroDestino, "VEHICULO.CERRADO: OK\n");
    }
      else if(strcmp(tarea, CMD_IN_N2))
    {
            pinMode(6, OUTPUT);
            digitalWrite(6, HIGH);
            delay(1000);
            digitalWrite(6, LOW);
            enviarSMS(numeroDestino, "VEHICULO.ABIERTO: OK\n");
    }
    else if(strcmp(tarea, CMD_IN_N3))
    {
            pinMode(13, OUTPUT);
            digitalWrite(13, HIGH);
            delay(15000);
            digitalWrite(13, LOW);
            enviarSMS(numeroDestino, "VEHICULO.VENT_B: OK\n");
      }
    else if(strcmp(tarea, CMD_IN_N4))
    {
            pinMode(19, OUTPUT);
            digitalWrite(19, HIGH);
            delay(15000);
            digitalWrite(19, LOW);
            enviarSMS(numeroDestino, "VEHICULO.VENT_S: OK\n");
      }
      else if(strcmp(tarea, CMD_IN_N5))
    {
            registro << "Comprobando estado del vehiculo:\n"
            registro << "Checkeo estado VENTANILLAS\n";

            //Se suben las ventanillas
            pinMode(19, OUTPUT);
            digitalWrite(19, HIGH);
```

```
            delay(15000);
            digitalWrite(19, LOW);

            enviarSMS(numeroDestino, "VEHICULO.VENT_S: OK\n");

            //Chequear Cierre centralizado:
            if(digitalRead(6))
                  enviarSMS(numeroDestino, "VEHICULO.ESTADO: ERROR.
Detectada puerta ABIERTA\n");
            else
                  enviarSMS(numeroDestino, "VEHICULO.ESTADO: OK\n");
    }
    else
        registro << "Comando ["; tarea; "] NO permitido\n";

      registro << "Tarea ["; tarea; "] REALIZADA con exito\n";
}


/*********************************************************************
* Nombre: recibirSMS
* Descripcion: Se encarga de las tareas necesarias para leer un sms y
realizar las tareas pertinentes
*           - Insertar el PIN de la SIM
*           - Registrar la tarjeta SIM en la red del operador
*           - Procesar el SMS recibido realizando las operaciones
*                 correspondientes
* Retorno: (-)
*********************************************************************/
void recibirSMS()
{
      int8_t x=0;
    int i=0, flag=0;
    char sms[100]="\"REC
READ\",\"+34608806226\",\"\",\"fechaaaaaa\"\nPrueba1\"0";

    char* numeroDestino=NULL;
    char* fecha=NULL;
    char* mensaje=NULL;
    char* ptr;
    char deli[3] = "\"\n";

    registro << "Seleccionando partes de guardado del sms...\n";
      enviarComandoAT("AT+CPMS=\"SM\",\"SM\",\"SM\",\"SM\"", "OK", 1000);
//Selecciona la memoria

    registro << "Leemos sms\n";
      respuesta = enviarComandoAT("AT+CMGR=1", "+CMGR:", 2000); //
Lectura del primer mensaje
      if(respuesta == 1)
      {
            respuesta = 0;
            while(Serial.available() == 0);
```

```
            do
            {
                  if(Serial.available() > 0)
                  {
                        sms[x] = Serial.read();
                        x++;
                  registro << sms[x];
                  // printf("%c", sms[x]);
                        if(strstr(sms, "OK") != NULL)
                  {
                      registro << "\n";
                      //printf("\n");
                                respuesta = 1;
                  }
                    }
            } while(respuesta == 0);

            // sms contiene el mensaje completo
            //sms[x] = '\0';
        registro << '\0';
      }
      else
      {
            registro << "Error " << respuesta << "\n";
      }

    ptr = strtok(sms, deli);
    ptr = strtok(NULL, deli);
    ptr = strtok(NULL, deli); strcpy(numeroDestino, ptr);
    ptr = strtok(NULL, deli);
    ptr = strtok(NULL, deli);
    ptr = strtok(NULL, deli); strcpy(fecha, ptr);
    ptr = strtok(NULL, deli); strcpy(mensaje, ptr);

    registro << "Mensaje leido correctamente\n";
      if(numeroDestino != NUM_TEL)
      {
            registro << "Detectado remitente no permitido [";
numeroDestino; "]\n";
            registro << "Mensaje deshechado\n"
      }
      else
      {
                procesarTarea(mensaje);
      }
}
```

```
/**********************************************************************
* Nombre: powerON
* Descripcion: Funcion que  verifica si el modulo SIM esta funcionando
* Retorno: (-)
**********************************************************************/
void powerON()
{
      uint8_t dev = 0;

      // Comprobamos que el modulo esté  encendido
    registro << "Comprobando modulo...\n";
    registro << "AT";
      dev = enviarComandoAT("AT", "OK", 2000);
      if(dev == 0)
      {
            // Power on pulse
            digitalWrite(ON_MODULE_PIN, HIGH);
            delay(3000);
            digitalWrite(ON_MODULE_PIN, LOW);

            while(dev == 0)
                  dev = enviarComandoAT("AT", "OK", 2000);
      }
    registro << "Modulo OK\n";
    registro << "\n";
}
```

```
/**************************************************************************
* Nombre: enviarComandoAT
* Descripcion: Funcion encargada de introducir el mensaje en el buffer
UART para después enviarlo. El envio se hace dentro de un tiempo timeout.
* Retorno: respuesta (int8_t)
**************************************************************************/
int8_t enviarComandoAT(const char* comAT, const char* res, unsigned int
timeout)
{
      uint8_t x=0, dev=0;
      char respuesta[100];
      unsigned long previo;

      // Inicializacion del String
      memset(respuesta, '\0', 100);
      delay(100);
      while(Serial.available() > 0)
            Serial.read();    // Limpia el Buffer

      // Enviamos el comando AT
    registro << "Enviando comando AT: " << comAT;
      Serial.println(comAT);
      x=0;
      previo = millis();

      do
      {
            if(Serial.available() != 0)
            {
                  // Recogemos en dev el retorno del comando y lo
mostramos por pantalla
                  respuesta[x] = Serial.read();
                  registro << respuesta[x];
            //printf("%c", respuesta[x]);
                  x++;
                  if(strstr(respuesta, res) != NULL)
                        dev=1;
            }
    // Esperamos por una respuesta antes del timeout
    } while((dev == 0) && ((millis() - previo) < timeout));

    registro << "Respuesta de: " << comAT << "=" << respuesta;
      return dev;
}
```

```
/**************************************
* Nombre: main                        *
* Descripcion: programa principal     *
* Retorno: 0 -> OK                    *
**************************************/
int main()
{
    registro.open("tfg.log");
    registro << "Iniciando sesion...";

    if(wiringPiSetup() == -1)
    {
        registr << "Error al iniciar la libreria wiringPi."
        return -1;
    }

      setup();
    registro << "Fin de la sesion.";
    registro.close();
      return 0;
}
```

D Historia de las alarmas del hogar

D.1 Introducción

Un sistema de alarma es un elemento de seguridad pasiva. Esto significa que no evitan una situación anormal, pero si son capaces de advertir de ella, cumpliendo así su función disuasoria frente a posibles problemas.

Los sistemas de alarma están constituidos por instalaciones destinadas a avisar al personal en caso de siniestro. Son la combinación de dispositivos de entrada (contactos en puertas y ventanas, detectores de movimiento, detectores duales, detectores de humos, etc…), una unidad centrar de proceso (PLC, panel de control) y los dispositivos de salida (sirenas, luces estroboscópicas, interfaces telefónicas, etc…).

La mayoría de los fenómenos físicos pueden ser detectados por sensores, monitoreados a través de amplificadores y circuitos de disparos, que pueden ser presentados en medidores, campanas, sirenas u ordenadores personales. Estos sistemas de protección utilizan sensores para detectar la luz, temperatura, presión, velocidad…

D.2 Clasificación

Podemos agrupar los distintos tipos de alarmas en 2 categorías: *Manuales* y *Automáticas.*

Una alarma manual consta de estaciones de aviso distribuidas por todo el recinto. Consisten en llaves o timbres cuyo accionamiento manual hace sonar la alarma.

Con el objetivo de impedir que alguien la oprima inadvertidamente, están protegidas por vidrios. Deben estar colocadas al alcance de los usuarios, de manera que no sea necesario recorrer más de 30 metros para encontrar una.

Una alarma automática pueden accionarse por dos mecanismos. Uno es un detector que indica un aumento de temperatura ambiente sobre cierto límite, y el otro es un detector sensible a una variedad brusca de la temperatura ambiental.

D.3 El prototipo de Pope

Augustus Russell Pope de Sommerville, Botón, fue un hombre muy habilidoso, que tras años de estudio, patentó el 21 de Junio de 1853 la primera alarma del mundo. Hasta

entonces, la gente confiaba en los ruidos omitidos por gansos y la fidelidad de sus perros guardianes para detectar posibles intrusos.

Pope ideó un dispositivo que funcionaba con pilas, que reaccionaba al cerrar un circuito eléctrico, en el cual las puertas y ventanas estaban conectadas como unidad independiente a una conexión en paralelo. De este modo, en caso de abrirse la puerta o una de las ventanas, y con ello el circuito eléctrico conectado a ellas, la corriente eléctrica creada dentro de los imanes del sistema producía una vibración. Estas oscilaciones eléctricas se transmitían a un martillo, en cual golpeaba una campana de latón. Lo realmente importante de este primer sistema de seguridad, es que la alarma n se podría desconectar simplemente cerrando la puerta o la ventana, ya que por encima de la puerta, en la pared, iba montado un muelle que mantenía el circuito en continuo funcionando y hacía que la campana siguiera sonando.

Sin embargo es a otra celebridad a la que se le suele considerar el padre de las modernas instalaciones de alarmas: Edwin Holmes.

D.4 Edwin Holmes: un estratega inteligente

Aunque Holmes no tuviera el espíritu inventor de Pope, sí demostró ser un sagaz estratega. En cuanto a sus medios de publicidad, hay que decir que se adelantó a su época. Con el fin de soslayar el miedo a la electricidad tan extendido durante el siglo XIX, publicó en periódicos de Nueva York una lista con nombres de clientes importantes que estaban dispuestos a reafirmar públicamente su confianza en las instalaciones de alarma. Así, encargó imprimir un anuncio en el cual aparecía siempre una foto de su "telégrafo alarma antirrobo", siempre junto a la mención del nombre de su empresa. De forma totalmente instintiva, Holmes siguió las leyes del marketing moderno, de manera que el invento de Pope se terminó convirtiendo poco a poco en su marca.

Holmes aprovechó también la confianza y fascinación de la población en una época en la que la telegrafía era algo aun realmente nuevo y lo hizo con un doble fin comercial. Por un lado, como nombre de producto para sus sistemas de alarma y por otro, con una finalidad técnica: aprovechar sus numerosos derechos de patente para el aislamiento de los cables telegráficos.

No le llevó mucho tiempo a Holmes crear una central donde convergían las líneas telegráficas que instalaba. Para lograr que los cables llegaran directo a cada uno de sus clientes, mudó la empresa al último piso de un edificio en el centro de Nueva York.

Sin embargo, la novedad fue introducida por Edwin Holmes hijo, quien -ante la cantidad de pedidos de nuevos clientes- pensó en usar los cables de teléfono de la ciudad, que se usarían para la alarma durante la noche, cuando los negocios cerraban. Así, la expansión de la red de alarmas creció considerablemente.

D.5 Un nuevo servicio

Otro hito en la historia de los sistemas de seguridad modernos fue la idea de enviar a los servicios de asistencia frente a emergencias, como los bomberos y la policía, a las casas en las que suenen las alarmas.

En 1867, un joven llamado Edward Calahan inventó el primer tablero de cotizaciones, que permitió que las fluctuaciones en la bolsa sean transmitidas mucho más rápido a los inversores.

¿Qué tiene ésto que ver con la **central de monitoreo de alarmas**? El inversor que creó una empresa a partir del invento de Calahan, y que ahora era su jefe, Elisha Andrews, fue sorprendido por ladrones que entraron a su casa de noche. Impactado por el incidente de su jefe; frente a esto, a Calahan se le ocurrió adaptar su invento para conectar botones de pánico entre distintas casas, con sonidos de alarma diferentes para cada una. Entonces, si en la casa A se presionaba el botón, las casas B y C sabrían que era la casa A la que había sufrido una intrusión.

Mientras trabajaba en su sistema de seguridad, Calahan tuvo una idea revolucionaria: cuando sonara una alarma, se podría enviar a las fuerzas policiales y bomberos, según la situación lo requiriese. Decidió entonces que su invento (al que llamó *call-box*) se conectaría a una central desde donde se recibirían las señales, para luego alertar a los servicios de emergencias y enviarlos al domicilio.

D.6 Sistemas de alarma de alta tecnología

El siglo veinte también trajo importantes desarrollos al mundo de la tecnología de alarmas.

Después de la Segunda Guerra Mundial, las cajas de alarma tipo Calahan eran más asequibles, lo cual permitió colocarlas en más y más puntos de alarma de emergencias médicas, comisarías y bomberos, con lo que la seguridad de la población también se vio mejorada. En la década de 1970, los técnicos integraron los primeros detectores de movimiento en los sistemas de alarmas. Los años 1980 y 1990 estuvieron caracterizados por una creciente estandarización, lo cual a su vez redundó en un uso cada vez más extendido para la protección de edificios. Finalmente llegaron los primeros sistemas de alarma inalámbricos al mercado, que revolucionaron la tecnología de alarmas también a nivel práctico, ya que por fin acabaron con el desbarajuste de cables.

Hoy en día incluso los terrenos más escondidos pueden quedar protegidos prácticamente sin zonas muertas gracias a la combinación de modernos detectores de movimiento, la tecnología de vigilancia de vídeo en alta definición y detectores electrónicos. Las innovaciones tecnológicas más relevantes siguen sorprendiendo a la gente. Así es como los desarrolladores de productos del centro de seguridad de ABUS consiguieron integrar en 2008 dentro de la moderna tecnología de alarmas inalámbricas la protección mecánica y electrónica en un único sistema de alarmas. Su alto nivel de protección mecánica permite repeler los intentos de robo al tiempo que se detectan electrónicamente.

Por ejemplo, si el ladrón intenta entrar por una ventana, las correas de acero encajadas una dentro de otra hacen que penetrar en la vivienda sea prácticamente imposible. Además, el intento de apertura se transmite a una central que activa la alarma y asusta al ladrón provocando su huida.

Seguro que la historia de los modernos sistemas de alarmas aún nos tiene preparados capítulos interesantes, así que dejémonos sorprender con lo que nos depararán las innovaciones técnicas en los próximos años. ABUS desea que todos los técnicos profesionales y los aficionados a los aparatos sofisticados sigan teniendo espacio de sobra para la inspiración y nuevas ideas. Así la historia de la seguridad continuará progresando con pleno éxito.