VGEAR: Sistema de adquisición de datos mediante OBD-II y bus ISO14230

Ingeniería técnica en telecomunicación, especialidad telemática

Estudiante
Daniel-Javier Zayas García

Consultor
Jordi Bécares Ferrés

15/06/2015

Agradecimientos

Quisiera agradecer a Joan Rosset compañero de trabajo y amigo por su inestimable ayuda durante el desarrollo de VGEAR.

Y en especial a mi pareja por su paciencia y apoyo durante todo el tiempo que ha durado este proyecto.

Resumen

Proyecto final de carrera de sistemas embebidos

Este proyecto consiste en la conexión al bus de diagnóstico ODB de una motocicleta Suzuki Intruder M800 mediante los estándares ISO14230 e ISO15031 obteniendo diferentes parámetros: velocidad, marcha y RPM del motor.

Estos parámetros se obtienen en la fase I mediante el producto VGEAR y son enviados a la placa de desarrollo LPC1769. En la fase II se usa únicamente la placa de desarrollo LPC1769. También se usa el módulo MMA7361 para obtener las aceleraciones directamente conectado a la placa LPC1769.

En ambos casos los datos son enviados por Bluetooth a una aplicación para Android bajo el protocolo VAP a través de un módulo HC05 conectado al LPC1769. Esta aplicación para Android también usa la señal de GPS y del propio acelerómetro del dispositivo móvil para cruzar datos.

Índice de contenido

Índice de figuras

1 Introducción

Este proyecto final de carrera consiste en la conexión al bus de diagnóstico de la ECU (Engine Control Unit) de la motocicleta Suzuki Intruder M800 para extraer información de estado y mostrarla en una aplicación para Android. La comunicación del sistema con la aplicación Android se hace mediante Bluetooth.

La ECU implementa el estándar ISO14230 a nivel físico, enlace y sesión mientras que a nivel de aplicación usa el protocolo estándar ISO15031-5.

Los datos que se obtienen son: velocidad, RPM, apertura de la mariposa de admisión, temperatura del refrigerante del motor y temperatura de entrada del aire del motor.

Debido a que no existe información pública del fabricante de la ECU (Suzuki) resulta complejo recolectar más información del resto de sensores de la motocicleta.

Meses previos a la elaboración de este proyecto se desarrolló el dispositivo VGEAR (ver figura 1). Este dispositivo se conecta al bus de diagnóstico de la ECU de la Suzuki Intruder M800 y muestra en un LCD PCD8544 (usado en los terminales Nokia 3310 y Nokia 5110) las RPM actuales y la marcha. Mediante un botón permite alternar la vista anterior con otra que muestra la temperatura del refrigerante y la temperatura de la entrada de aire del motor.

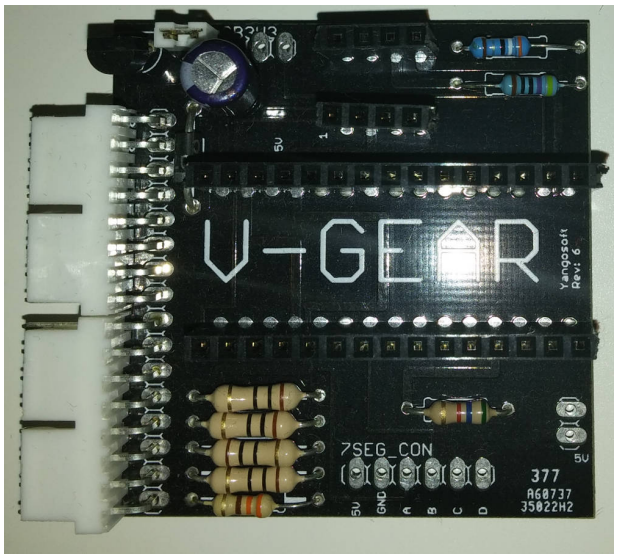

Figura 1: Dispositivo VGEAR 1.0.

1.1 Justificación

El dispositivo VGEAR se desarrolló debido a la necesidad de mostrar las revoluciones por minuto y la marcha actual en la Suzuki Intruder M800 que poseo. Este modelo de motocicleta no incluye esa información en el cuadro de instrumentos a pesar de disponer de los sensores y de que la ECU monitoriza estos datos.

Se decidió abordar el proyecto mediante el conector de diagnóstico de la ECU ya que existían diversos dispositivos en el mercado que ya realizaban esta función.

El siguiente modelo de VGEAR debía incorporar conexión Bluetooth mediante un módulo HC05 para poder visualizar los datos en un terminal móvil inteligente dado que disponen de una pantalla de mayor resolución y gran capacidad de proceso.

1.2 Objetivos

El proyecto se dividió en 2 fases para cumplir con la siguiente lista de objetivos:

1.2.1 Fase I

- Dotar a VGEAR de conexión Bluetooth.
- Monitorizar más datos: velocidad, apertura del gas, aceleraciones. Este último dato se obtiene a partir del acelerómetro MMA7361.
- Interfaz gráfica para terminales Android.
- Incrementar la robustez del sistema y recuperación de situaciones de error.

1.2.2 Fase II

- Usar exclusivamente la placa de desarrollo LPC1769 para eliminar VGEAR debido a sus limitaciones de potencia de cálculo, memoria, pines y UART.
- Mejorar la interfaz gráfica de la aplicación de Android.
- Dotar de más funcionalidades al proyecto: aprovechar el terminal móvil para cruzar datos con los sensores que éste dispone.

1.3 Metodología

El proyecto se ha abordado dividiéndolo en diferentes partes, implementando su funcionalidad, comprobando errores con juegos de pruebas y finalmente integrando.

Primero se analizaron los requisitos y se validaron con el director de proyecto Jordi Bécares Ferrés. Se decidió dividir el proyecto en dos fases.

En la primera fase se crearon los drivers necesarios para interactuar con las UART y el HC05 en el LPC1769. Estos se probaron exhaustivamente con juegos de pruebas, en el capítulo 11.2 se detallan. También se creó el driver para usar el acelerómetro MMA7361.

Se definió el protocolo de comunicación VAP (VGEAR At Protocol) para luego ser implementado tanto en VGEAR como en LPC1769. Más adelante en el capítulo 4.2.1.1 se especifica en detalle el protocolo.

Se realizaron diversos juegos de pruebas para comprobar el correcto “parseo” del protocolo por ambas partes, de nuevo en el capítulo 11.2 se detallan los juegos de pruebas.

Un vez implementados estos protocolos se procedió a comunicar VGEAR con LPC1769 a través de las UART serie. Se implementó el protocolo VAP en ambas partes.

Terminada la comunicación VGEAR con LPC1769 se procedió a implementar la aplicación de Android. Esta aplicación debía implementar el protocolo VAP. Se probó junto a un juego de pruebas.

Finalmente se procedió a integrar todos los subsistemas en la aplicación que ejecuta el LPC1769: BlueGEAR.

En la segunda fase se portó el código de VGEAR a LPC1769 dado que éste es más potente que VGEAR.

La memoria ha sido escrita a lo largo del proyecto, tomando notas de las diferentes tareas y de los problemas encontrados.

Todo el software desarrollado, esquemáticos y documentación están disponibles en un repositorio Git. Durante todo el desarrollo ese repositorio estuvo albergado en un servidor propio externo.

1.4 Planificación

A lo largo del proyecto se ha seguido la planificación pero dedicando más horas de las previstas en algunas tareas. La figura 2 es un diagrama de Gantt con la planificación inicial del proyecto.

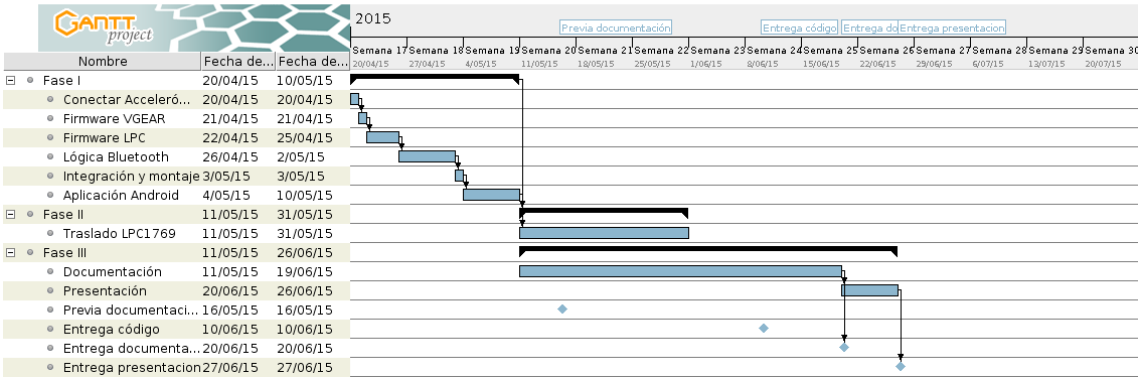

Figura 2: Gantt de la planificación del proyecto.

En concreto la tarea “Aplicación Android” que corresponde a VGEAR para Android 1.0 se retrasó 2 días debido a problemas con el terminal móvil y el módulo HC05 responsable de dotar de conexión Bluetooth al sistema. El módulo HC05 parecía no responder al intento de conexión. La pila Bluetooth no es inicializada automáticamente al salir del modo comandos, se debe forzar su inicialización con el comando AT+INIT.

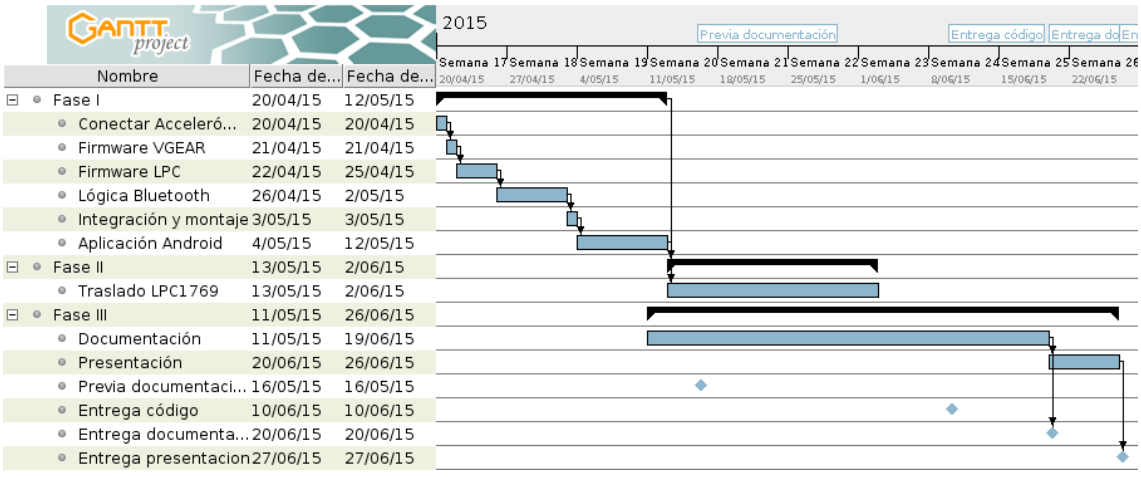

Figura 3: Planificación final.

1.5 Recursos

A continuación se listan los recursos utilizados para desarrollar el proyecto:

1.5.1 Software

LPCExpresso
Netbeans
Git
Dokuwiki
Eagle
GCC
Python
Qt5
Yed
Doxygen

1.5.2 Hardware

VGEAR
Protoboard
Terminal Android LG G3
HC05

1.5.3 Componentes para los prototipos

Electrónica de control para el módulo HC05

- 1x Placa de prototipado perforada
- 1x Transistor NTF5P03
- 1x Resistencia 1/4W 100Ω
- 2x Resistencia 1/4W 1kΩ

1.6 Productos obtenidos

1.6.1 Fase I

- VGEAR con comunicación serie con protocolo VAP.
- BlueGEAR 1.0: firmware del LPC1769 con comunicación Bluetooth y comunicación con VGEAR.
- VGEAR Android 1.0: Aplicación para Android que muestra los datos recogidos por VGEAR y enviados por BlueGEAR en LPC1769 mediante el driver HC05.

1.6.2 Fase II

- BlueGEAR 1.1: LPC1769 que permite conectarse al bus ISO14230 de la Suzuki Intruder M800, enviar los datos por Bluetooth y mostrar por el PCD8544 los mismos datos.
- VGEAR Android 1.1: Aplicación Android mejorada.

2 Antecedentes

2.1 Estado del arte

Actualmente existen en el mercado diferentes productos que implementan parte de la funcionalidad de VGEAR. En concreto los indicadores de marcha funcionan de dos formas: con sensores de efecto Hall[1] o bien conectándose al bus de diagnóstico de la ECU[2].

Los indicadores que se conectan al bus de diagnóstico de la ECU son capaces de implementar diversos protocolos de conexión. VGEAR sólo implementa el ISO14230 y el ISO15031-5 siendo capaz sólo de leer los datos de las ECU de Suzuki Intruder M800. Probablemente todas las Suzuki con bus ISO14230 implementan el mismo protocolo pero no se ha podido constatar.

Una ventaja de VGEAR es que reúne en un solo producto varios productos comerciales: indicador de velocidad, indicadores de marcha, acelerómetro, medidor de inclinación, indicador de RPM e indicador de temperatura y además dispone de una interfaz para terminales Android.

2.2 Estudio de mercado

Un cliente potencial del sistema VGEAR son los pilotos aficionados de motocicleta que quisieran tener sistemas de telemetría similares a los del mundo profesional pero que no pueden adquirirlos. A pesar de que VGEAR sólo soporta motocicletas del fabricante Suzuki es posible adaptar el código para obtener esos mismos datos de otras ECU.

Otro potencial cliente es el propietario de motocicletas *custom* que usualmente suelen tener indicadores más espartanos o sencillos.

Según datos de la Dirección General de Tráfico, en 2013 Suzuki representaba el 8,84% del total de matriculaciones de motocicletas del estado Español[3].

1 http://electronics-lab.com/projects/automotive/006/index.html

2 http://www.tracking.uk.com/digital-gear-indicators-23-c.asp

3 http://www.dgt.es/Galerias/seguridad-vial/estadisticas-e-indicadores/matriculaciones-definitivas/vehiculos/motocicletas/doc/motosProvinciaMarca017-2013.xls

2.3 Tecnologías inalámbricas

Para la conectividad inalámbrica se ha seleccionado Bluetooth debido a que está disponible en el 90% de los terminales móviles del mercado[4] incluyendo terminales que no son de tipo smartphone. Wifi también es una alternativa como medio de conexión inalámbrico con el terminal móvil. Zigbee se descartó debido a su poca presencia en los terminales móviles actuales.

Wifi	Bluetooth	Zigbee
Mayor velocidad de transferencia	Menor velocidad de transferencia	Menor velocidad de transferencia
Mayor consumo energético	Menor consumo energético	Menor consumo energético
Mayor alcance	Menor alcance	Menor alcance

Dado que el dispositivo está ubicado cerca del terminal móvil del usuario, que la cantidad de datos a enviar es baja y que la aplicación se ejecutará en terminales móviles con batería; Bluetooth es la tecnología seleccionada.

2.4 Microcontroladores

La elección de un microcontrolador se puede basar en diversos factores:

- Económicos
- Disponibilidad en el mercado
- Soporte del fabricante
- Características y periféricos integrados
- Capacidad de proceso

En el caso de VGEAR se seleccionó Arduino como plataforma de desarrollo del prototipo por su sencillez, precio y soporte.

4 Datos del IHS Technology Connectivity in Consumer de 2014.
http://www.bluetooth.com/Pages/Press-Releases-Detail.aspx?ItemID=209

Para el prototipo final ha primado el criterio de disponibilidad del fabricante, de cantidad de periféricos y que el microcontrolador cumpla el estándar de automoción.

Para este producto no es necesario una gran capacidad de proceso ya que no se realizan cálculos intensivos.

2.5 Terminales móviles

En el segundo cuadrimestre de 2014 las ventas mundiales de teléfonos inteligentes alcanzaron un total de 295,2 millones de unidades[5].

Figura 4: Mercado de telefónos móviles 2do cuadrimestre 2014.

En lugar de desarrollar una aplicación únicamente para el sistema operativo predominante se puede optar por utilizar Qt5 que permite desarrollar la aplicación en C++ y compilar para diversas plataformas entre ellas: Android, Windows Phone e iOS.

5 Datos extraídos de Strategy Analytics http://bgr.com/2014/07/31/android-vs-ios-vs-windows-phone-vs-blackberry/

3 Descripción funcional

3.1 Diseño funcional Fase I

El diseño funcional de la Fase I consiste en el uso de VGEAR en modo pasivo, LPC1769 como maestro, HC05 en modo esclavo y el acelerómetro.

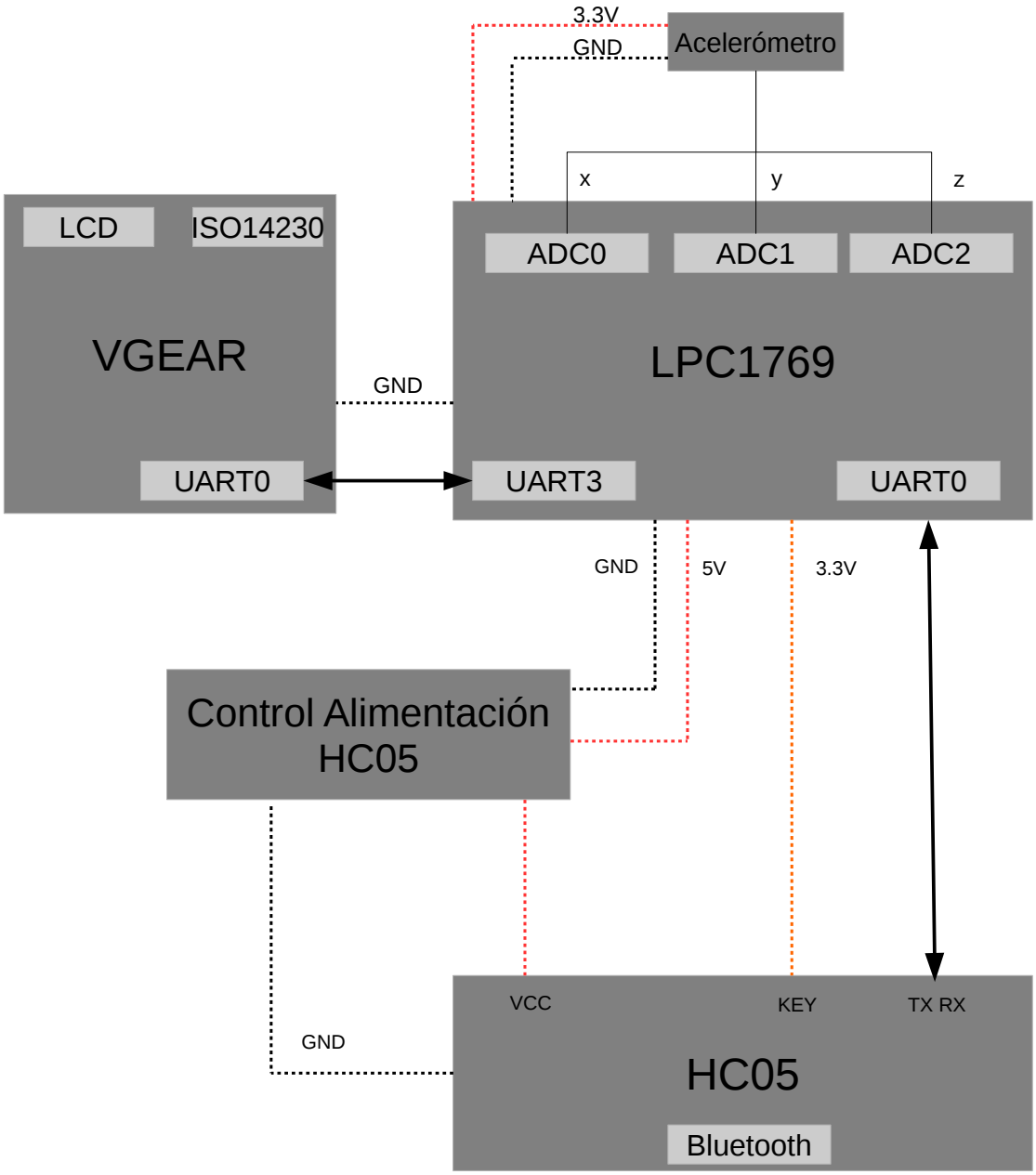

Figura 5: Diseño funcional fase I.

3.1.1 VGEAR

VGEAR se conecta a la ECU de la Suzuki Intruder M800 mediante el bus ISO14230. Por otra parte se comunica con el módulo LPC1769 mediante el protocolo VAP para responder a las solicitudes de datos que éste realice.

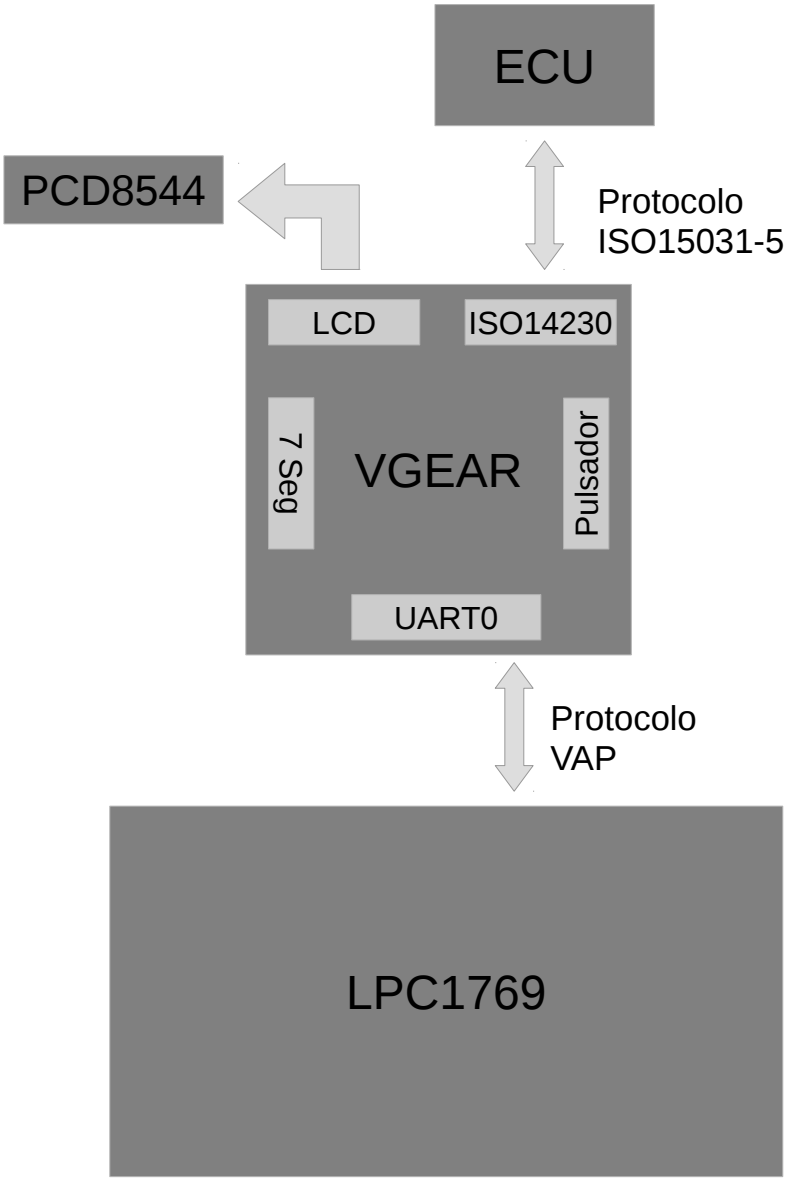

Figura 6: Diseño VGEAR.

Adicionalmente el módulo VGEAR permite conectar una pantalla LCD (ver Figura 7 y siguiente) con controlador PCD8544 o un display de 7 segmentos.

Figura 7: Pantalla LCD de VGEAR.

Figura 8: Vista del conector de la pantalla LCD de VGEAR.

La vista de la pantalla LCD permite ser cambiada mediante un simple pulsador.

3.1.2 BlueGEAR 1.0

El software BlueGEAR que ejecuta el LPC1769 actúa como controlador del sistema. Por una parte se conecta al VGEAR para solicitar datos del estado de la ECU cada 500ms mediante el protocolo VAP.

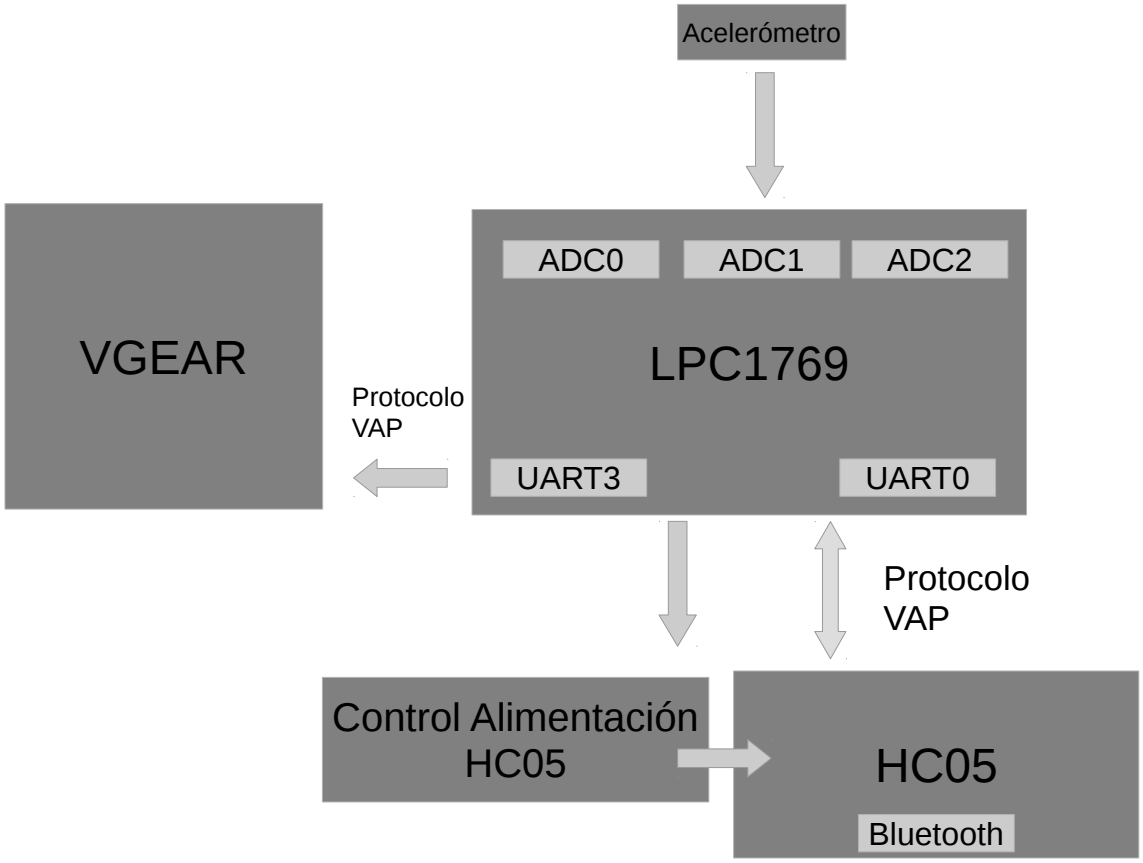

Figura 9: BlueGEAR se ejecuta sobre el LPC1769.

Existe un segundo proceso que muestrea el acelerómetro y almacena las diferentes aceleraciones.

Finalmente un tercer proceso se conecta al módulo HC05 mediante la UART0 para configurarlo como esclavo. Una vez configurado como esclavo espera recibir peticiones de estado mediante el protocolo VAP. Cuando recibe una petición contesta con el último estado leído de VGEAR más la información del acelerómetro.

Figura 10: Diseño por capas del proyecto.

El sistema se basa en diferentes capas:

- FreeRTOS: sistema operativo en tiempo real. Aunque no está representada, FreeRTOS se ayuda de la biblioteca CMSIS2.
- LPC1769_UOC_Library: biblioteca desarrollada en diversas PAC y ampliada con los drivers HC05, ADC y módulos VGEAR.
- BlueGEAR: aplicación controladora del sistema. Se encarga de interactuar con los diferentes dispositivos y subsistemas. Esta aplicación está supervisada por un watchdog.

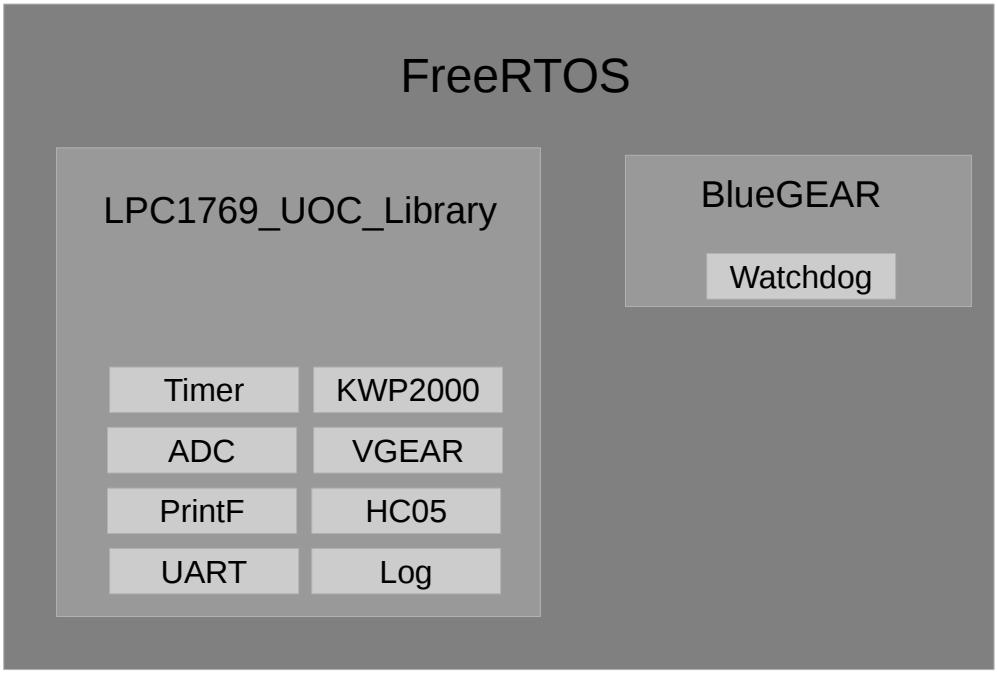

Figura 11: Principales bloques de software.

3.1.3 VGEAR Android 1.0

La aplicación VGEAR para Android se encarga de conectarse por Bluetooth al módulo HC05 y mostrar los datos que la aplicación BlueGEAR proporciona.

Se compone de una interfaz sencilla que muestra la velocidad actual, la marcha actual, las RPM, las temperaturas del refrigerante y de la entrada de aire del motor y el porcentaje de apertura de la mariposa de admisión. También muestra las aceleraciones que el módulo MMA7361.

Adicionalmente usa el GPS del terminal y el acelerómetro para mostrar la velocidad y las aceleraciones que el dispositivo registra.

Se ha diseñado para que muestre la información más interesante en un tamaño mayor y con una elección de colores adecuada para ser vista con la luz solar.

Figura 12: VGEAR 1.0 para Android.

Cuando la diferencia entre la velocidad obtenida del GPS y de VGEAR está entre 3 y 10 km/h la etiqueta que muestra la velocidad cambia su color de fondo para indicar al usuario que esté atento ya que hay una diferencia importante de velocidad entre las dos medidas.

3.2 Diseño funcional Fase II

El diseño funcional de la fase II elimina VGEAR y deja LPC1769 como interfaz con la ECU. Implementa el protocolo ISO14230 para conectarse a la ECU y la UART0 para el protocolo VAP con el módulo Bluetooth HC05.

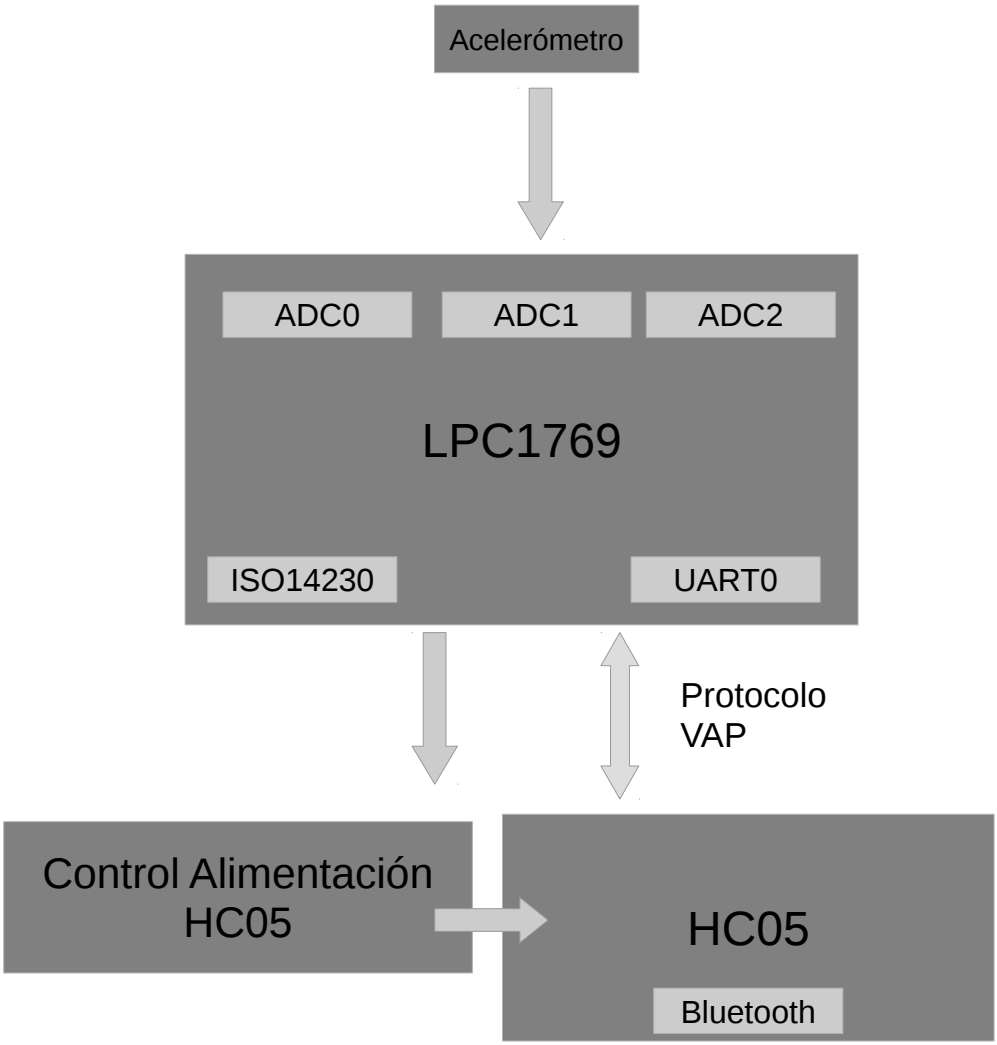

Figura 13: Diseño funcional fase II.

Se elimina la gestión del LCD ya que se usa un terminal móvil como interfaz de usuario. Tampoco se implementa la gestión del display de 7 segmentos.

3.2.1 BlueGEAR 1.1

En esta versión la tarea encarga de comunicar con VGEAR ya no existe. El mismo software realiza la gestión de la conexión al bus ISO14230. La implementación se encuentra en el módulo KWP2000 dentro de la biblioteca LPC1769_UOC_Library.

Con el objetivo de reducir el tiempo entre petición del terminal móvil y respuesta BlueGEAR envía los últimos datos de la ECU por Bluetooth sin esperar a que se haga un comando de petición.

3.2.2 VGEAR Android 1.1

Se modificó la aplicación con el objetivo de mostrar los datos en un menor tiempo. Ya no se realizan comandos de petición de datos, en su lugar sólo se atiende a la conexión Bluetooth y se parsean los datos que BlueGEAR envía.

4 Diseño del sistema

4.1 Hardware

4.1.1 Diseño de VGEAR

VGEAR nació de la necesidad de mostrar en una pantalla o en un display de 7 segmentos la marcha actual en los modelos de motocicleta Suzuki Intruder M800. Funciona de manera similar a una herramienta de diagnóstico o tester.

Está diseñado para incluir un Arduino Nano como unidad de proceso. Se escogió esta plataforma por la velocidad de prototipado y por la gran cantidad de bibliotecas disponibles.

A pesar de estar limitado a una UART hardware su capacidad de proceso permite implementar UART por software. VGEAR requiere de una UART para la comunicación serie (PC o LPC1769) y una segunda UART para la comunicación con la ECU mediante el driver L9637D.

Para mostrar la información de la ECU se puede usar una pantalla LCD con el popular controlador PCD8544 (usado en los modelos Nokia 5110 y Nokia 3310 entre otros). Además dispone de un conector para incrustar un botón (que permite alternar la información en pantalla) y 4 pines libres (conector 7SEG_CON) que permiten, por ejemplo, conectar un display de 7 segmentos o 2 UART software o simplemente ser usados como pines de entrada y salida.

La conexión con la ECU se hace mediante un conector Sumitomo MT090-6 conectado mediante un cable al puerto QACCESS. Tanto el puerto QACCESS como el puerto HEADER_LCD y el puerto 7SEG_CON son conectores Molex.

4.1.1.1 Prototipos

Antes de crear un PCB para VGEAR se usó una placa de prototipado para validar el diseño y comprobar el correcto funcionamiento del software.

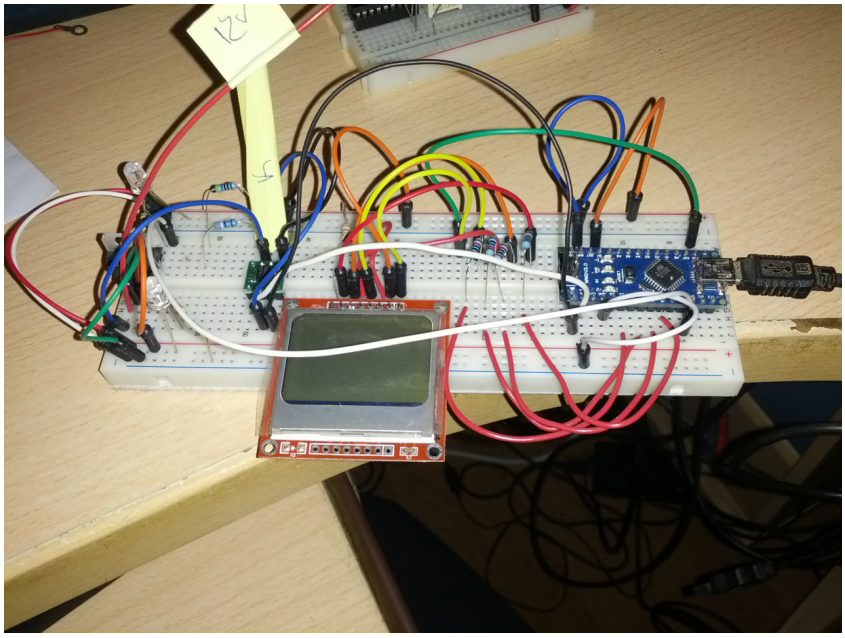

Figura 14: Prototipo de VGEAR en placa de prototipos.

El siguiente paso fue realizar el diseño en una placa perforada con un diseño similar al del PCB.

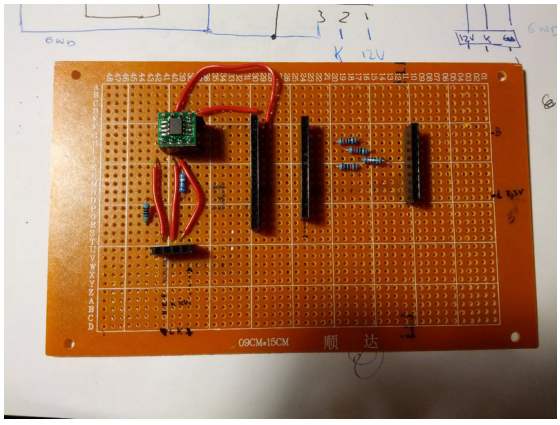

Figura 15: Prototipo de VGEAR en placa perforada.

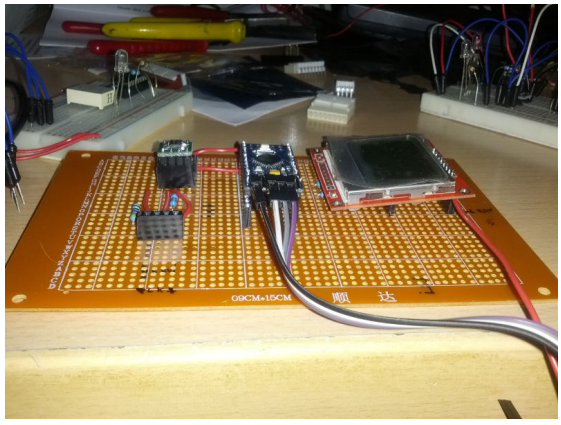

Figura 16: Componentes montados.

Dado que el PCB se estaba diseñando en base al prototipo de la figura 15 y 16 validar y comprobar que funcionaba era indicativo de que el traspaso sería correcto.

4.1.1.2 Alimentación

VGEAR se alimenta directamente de los 12 V de la batería de la motocicleta. Esta tensión de entrada está filtrada por un filtro (Diodo + Condensador) para absorber los diferentes picos de tensión que produce la dinamo que carga la batería.

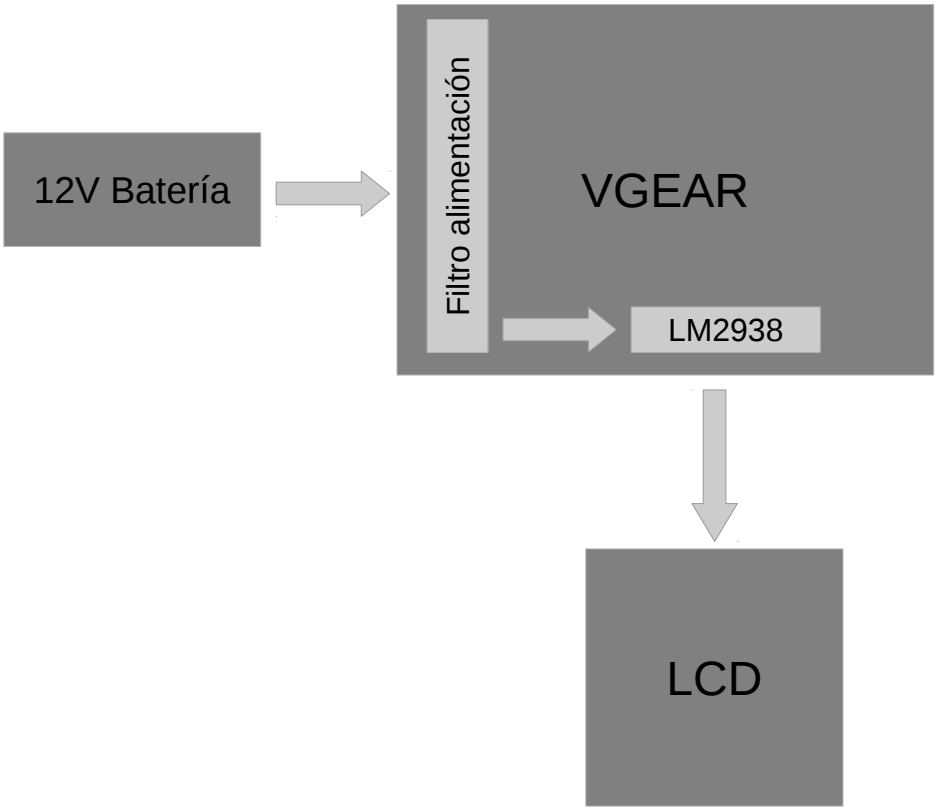

Figura 17: Esquema alimentación VGEAR.

Dependiendo de la longitud del cable del LCD es necesario alimentarlo directamente a 5V y calcular la caída producida por la longitud del cable para añadir una pequeña resistencia (si es necesario) en el PCB del LCD. Se puede deshabilitar el regulador mediante un simple jumper.

Las tensiones del bus de datos SPI del LCD se pueden alterar mediante el conjunto de resistencias del PCB dependiendo – de nuevo – de la longitud del cable del LCD y de la resistencia que éste ofrezca.

El proyecto VGEAR se realizó en tres fases: prototipado, validación e implementación. El prototipado consistió en simular una ECU en VGEAR y comprobar la correcta conexión con un ELM327.

El ELM327 es una herramienta de diagnóstico que dispone de un conector ODB que puede ser insertado en el puerto de diagnóstico de los vehículos modernos. Por el otro extremo dispone de un puerto USB o bien Bluetooth o WiFi y permite ser configurado mediante comandos. Esta herramienta implementa numerosos protocolos de comunicación entre ellos ISO14230.

El procedimiento consistió en simular una ECU mediante VGEAR y permitir que el ELM328 comunicara peticiones.

La validación se realizó mediante un prototipo conectado al bus ISO14230 de una ECU real. Por último se realizó un PCB con el diseño.

4.1.1.3 Problemas durante el desarrollo

Debido a que la alimentación de las motocicleta es altamente inestable (en este caso se pudo constatar fluctuaciones entre 10V y 16V) el PCB alberga un filtro de alimentación.

Este filtro previo a la alimentación de VGEAR consiste en un diodo 1N4002 para evitar retornos de corriente y un condensador de 220µF.

4.1.1.4 Cálculo de tiempos

Durante el desarrollo de VGEAR se constató que la ECU de la Suzuki Intruder M800 se comportaba de manera anómala cuando se solicitaban datos con un periodo inferior a 100 ms. La inyección parecía verse afectada cambiando la respuesta del motor.

Debido a este problema se decidió aumentar el tiempo de solicitud de datos a 200ms + 70ms (10ms por byte). La ECU parece comportarse de forma correcta con un periodo de 270ms.

El estándar ISO14230 especifica que el tiempo entre peticiones (del tester a la ECU) comprende el intervalo de 55ms a 5000ms y que el tiempo entre bytes debe comprender el intervalo 5ms a 20ms.

Para una descripción más detallada de los tiempos definidos en el estándar ISO14230 se puede consultar el capítulo 5.1.6.

4.1.1.5 Conector Sumitomo MT090-6

Este conector es utilizado por Suzuki como puerto para sus ECU. Otros fabricantes usan este mismo conector para sus ECU. Una característica interesante es que es impermeable. La conexión queda aislada del agua, polvo y otros elementos externos.

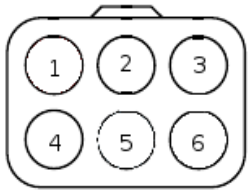

Figura 18: Conector Sumitomo MT090-6 visto desde el lado de los cables.

Pin	Descripción
1	Positivo batería
2	No conectado
3	No conectado
4	No conectado
5	K-Line
6	GND

Figura 19: Ubicación del conector Sumitomo MT090-6 en la Suzuki Intruder M800.

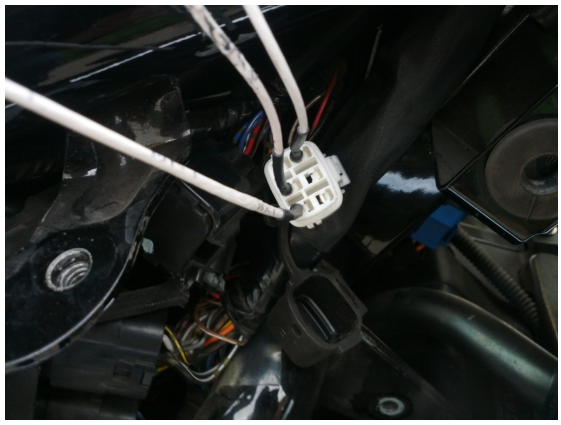

Figura 20: Detalle del conector Sumitomo MT090-6.

4.1.1.6 PCB

El PCB para VGEAR consta de 2 capas y tamaño 5cm x 5cm. Este tamaño se podría reducir bastante diseñando el PCB con componentes SMD.

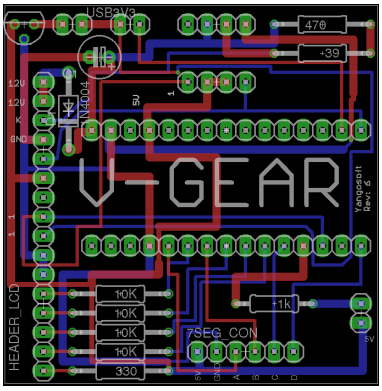

Figura 21: PCB de VGEAR 1.0.

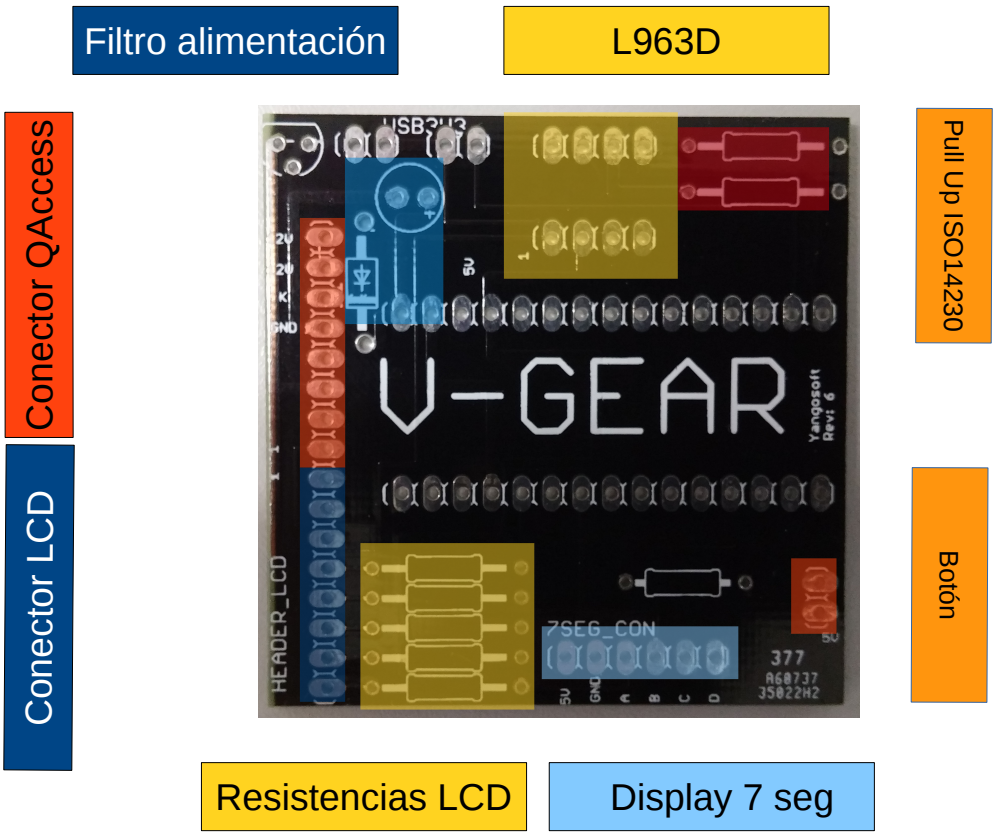

Figura 22: Bloques de VGEAR 1.0.

La figura 22 muestra los diferentes bloques que componen el PCB de VGEAR 1.0.

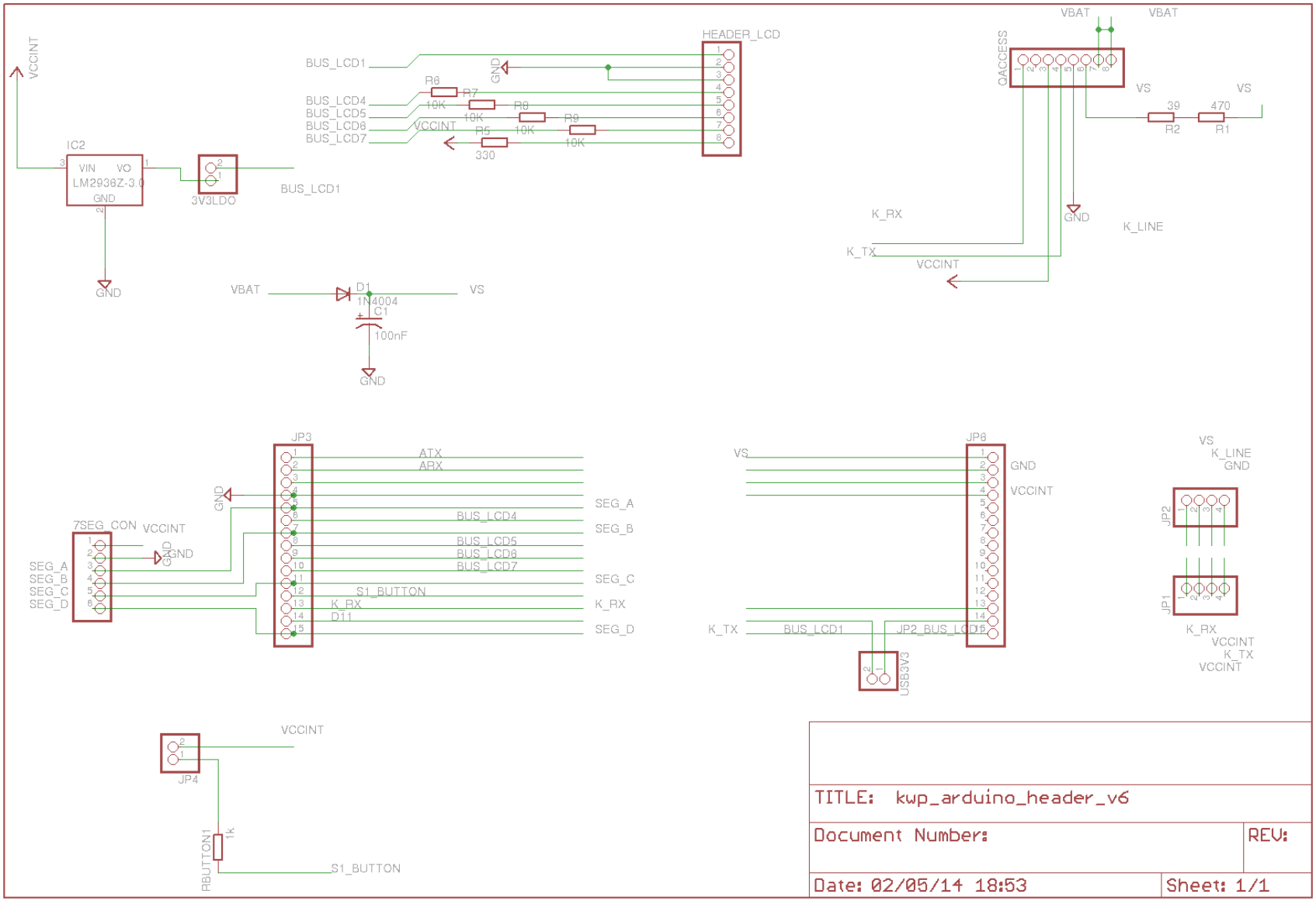

Figura 23: Esquemático de VGEAR 1.0.

4.1.1.7 Máquina de estados VGEAR

El funcionamiento del programa se basa en una máquina de estados sencilla representada en la figura 24.

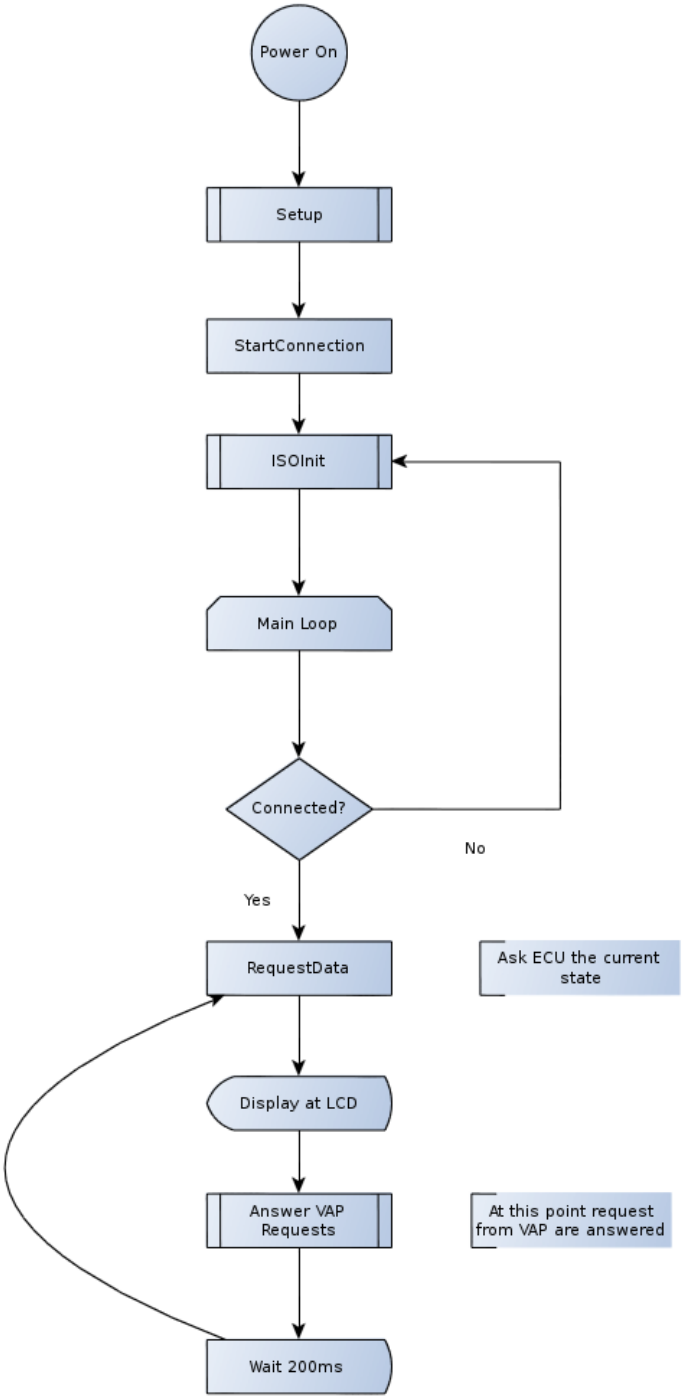

Figura 24: Máquina de estados de VGEAR.

4.1.2 Electrónica de control para el módulo HC05

El módulo HC05 admite un modo comandos para poder ser configurado. Para poder acceder al modo comando es necesario alimentar el módulo una vez el pin “KEY” está a 3.3V.

Sólo se permite salir del modo comando usando el comando AT+INIT, bajando el pin “KEY” a 0V, apagando el módulo HC05 y finalmente alimentándolo de nuevo.

El driver HC05 implementando en el LPC1769 contempla el manejo del modo comandos mediante una electrónica auxiliar diseñada específicamente.

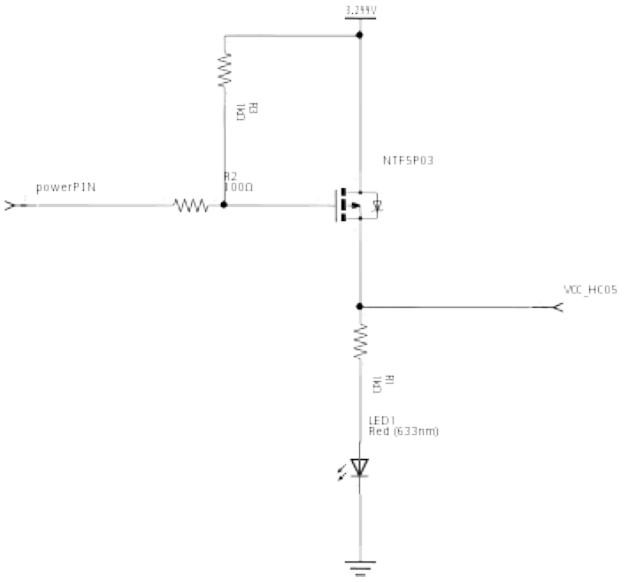

Figura 25: Esquema electrónica auxiliar HC05.

Consiste en un transistor MOSFET de canal P NTF5P03 que controla la alimentación del módulo HC05.

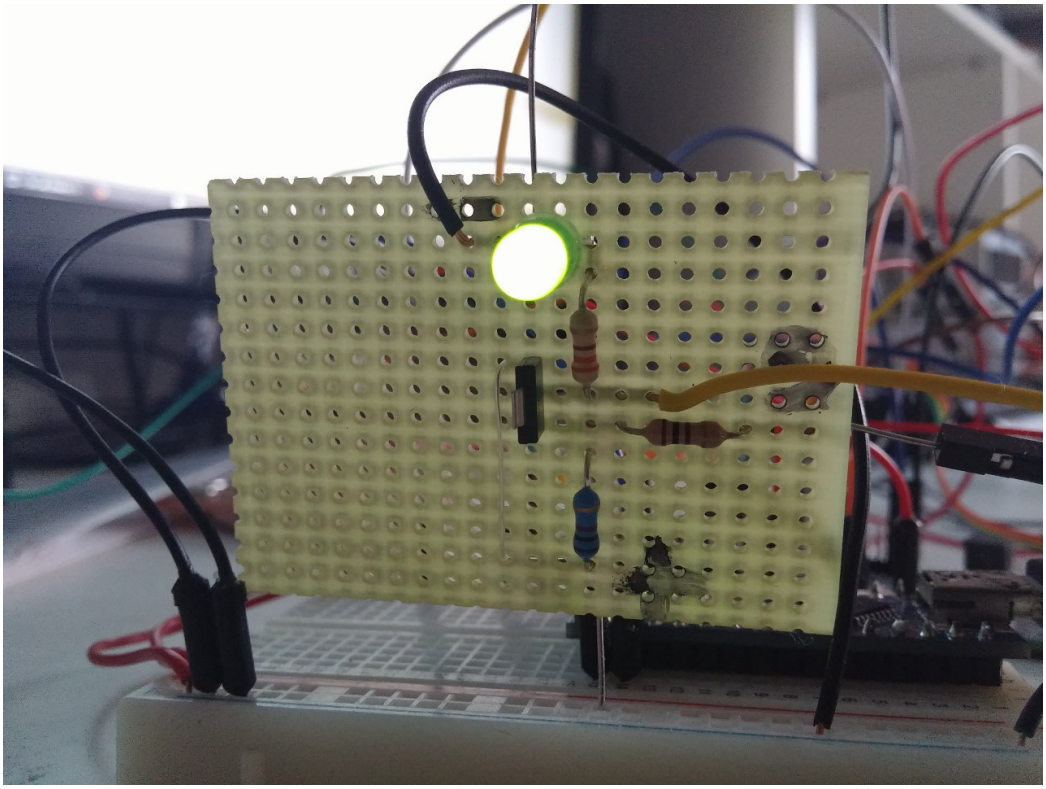

Figura 26: Prototipo de la electrónica de control para el HC05.

4.2 Software

4.2.1 Driver HC05

El driver para el HC05 es un módulo de la biblioteca LPC1769_UOC_Library. Este módulo se inicializa asociando una de las UART, un baudrate y dos pines: powerPin y keyPin.

Todas las llamadas están protegidas con mutex para que el módulo se pueda utilizar desde diversos procesos.

El pin powerPin es usado para encender y apagar el módulo HC05 mediante la electrónica de control descrita en el capítulo 4.1.2 mientras que keyPin es usado para activar el modo comandos.

La comunicación en modo comandos está preparada para un baudrate de 38400bps. Una vez se sale del modo comandos se reinicializa la UART con el baudrate que se haya fijado en la

llamada inicial del módulo HC05.

Para salir del modo comandos es necesario apagar el módulo HC05, conectar a GND el keyPin y finalmente conectar el módulo HC05.

El envío de datos y la lectura de datos del módulo HC05 se debe hacer mediante el acceso a la UART o bien mediante el módulo PrintF desarrollado en las PAC de esta asignatura. De esta forma evitamos repetir código y aprovechamos el trabajo desarrollado y validado.

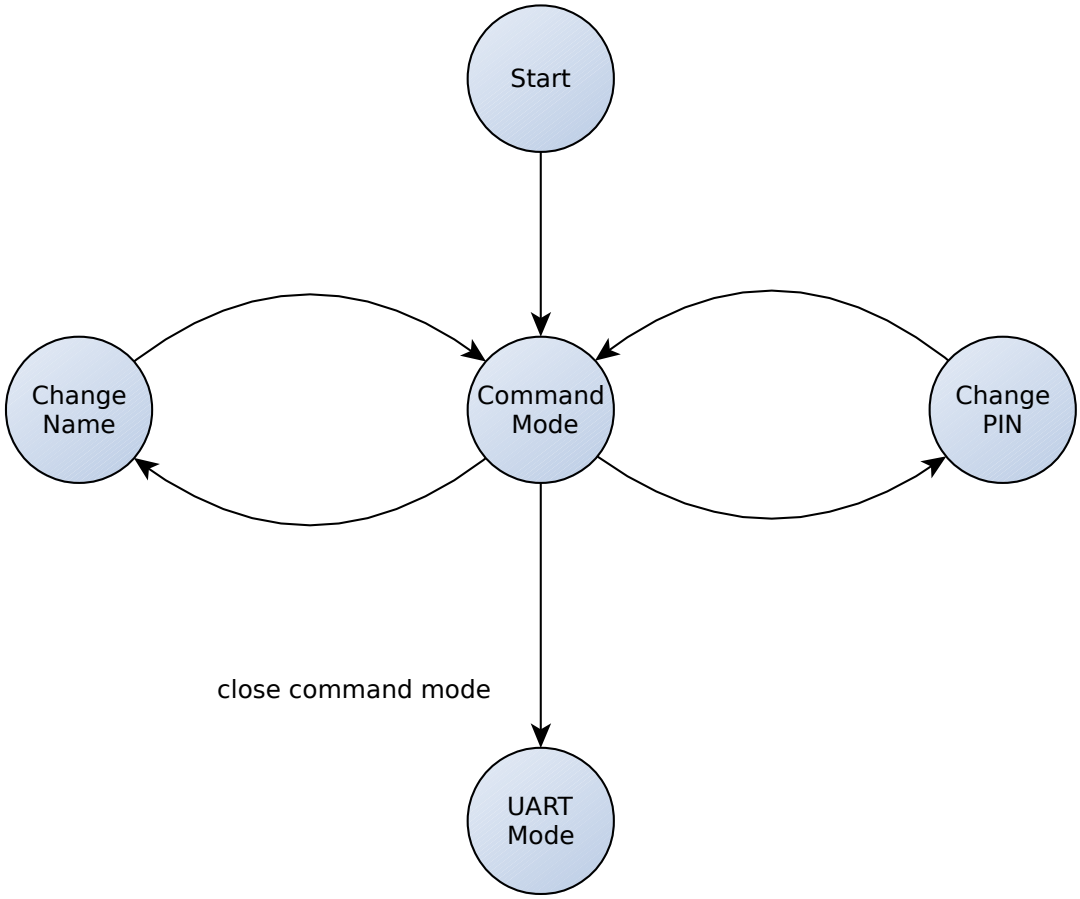

Figura 27: Máquina de estados del driver HC05.

4.2.1.1 Protocolo VAP

El protocolo VGEAR AT Protocol (VAP) es un protocolo de aplicación diseñado para la comunicación entre LPC1769 y VGEAR y entre LPC1769 y la aplicación VGEAR para Android.

Se basa en comandos en texto plano (ASCII) que empiezan con AT+ seguido de un identificador de comando y terminando en un salto de línea ("\n").

La única diferencia entre la implementación VAP de VGEAR y LPC1769 es que VGEAR no retorna datos del acelerómetro mientras que LPC1769 sí por lo que la trama AT+ALL devuelve 9 valores.

En caso de enviar un comando erróneo el protocolo VGEAR define la respuesta AT+ERR 1 que indica que el comando no se ha reconocido.

Todas las respuestas acaban en un salto de línea ("\n").

Este protocolo es fácilmente ampliable y permite ser implementado de forma sencilla y rápida en cualquier microcontrolador.

<table>
<tr><th>Petición</th><th>Respuesta</th></tr>
<tr><td>AT+STATUS</td><td>C
D
E
C: conectado a ECU correctamente
D: desconectado de la ECU
E: error</td></tr>
<tr><td>AT+ALL</td><td>Si no se dispone de datos del acelerómetro:

<table>
<tr><td>Apertura del gas</td><td>RPM</td><td>Temperatura aire</td><td>Temperatura refrigerante</td><td>Marcha</td><td>Velocidad (km/h)</td></tr>
<tr><td>uint32_t</td><td>uint32_t</td><td>float</td><td>float</td><td>uint32_t</td><td>uint32_t</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><td></td><td>El formato float es un entero de menos de 3 dígitos separado por un punto y otro entero que representa la parte decimal con 2 dígitos. Por ejemplo: 120.00.

Si hay datos del acelerómetro:
Se añaden 3 elementos más a la trama:

<table><tr><td>Aceleración X</td><td>Aceleración Y</td><td>Aceleración Z</td></tr><tr><td>uint32_t</td><td>uint32_t</td><td>uint32_t</td></tr></table>
Los valores de aceleración son directamente la lectura del módulo MMA7631 por lo que es necesaria su conversión a m/s^2.</td></tr>
<tr><td>AT+SPEED</td><td>uint32_t con la velocidad en km/h.</td></tr>
<tr><td>AT+GEAR</td><td>uint32_t con la marcha.</td></tr>
<tr><td>AT+ATEMP</td><td>Float que representa la temperatura de la entrada de aire del motor en Celsius.

El formato float es un entero de menos de 3 dígitos separado por un punto y otro entero que representa la parte decimal con 2 dígitos. Por ejemplo: 120.00.</td></tr>
<tr><td>AT+ETEMP</td><td>Float que representa la temperatura del refrigerante del motor. El formato es el mismo que el comando AT+ATEMP.</td></tr>
<tr><td>AT+RPM</td><td>uint32_t RPM del motor.</td></tr>
<tr><td>AT+TRH</td><td>uint32_t representado el porcentaje de apertura del gas de la motocicleta.</td></tr>
<tr><td>AT+ACC</td><td><table><tr><td>Aceleración X</td><td>Aceleración Y</td><td>Aceleración Z</td></tr><tr><td>uint32_t</td><td>uint32_t</td><td>uint32_t</td></tr></table>
Los valores de aceleración son directamente la lectura del módulo MMA7631 por lo que es necesaria su conversión a m/s^2.</td></tr>
</table>

4.2.2 BlueGEAR

BlueGEAR es el software controlador del sistema. Es capaz de conectarse a la placa VGEAR, al dispositivo Bluetooth HC05 y muestrear el acelerómetro.

4.2.2.1 BlueGEAR 1.0

La aplicación BlueGEAR se ejecuta sobre el LPC1769. Ejecuta tres tareas:

- Muestrear el acelerómetro
- Petición de datos a VGEAR
- Atender peticiones por Bluetooth

Además está supervisada por un watchdog hardware que es refrescado cada vez que se muestrea el acelerómetro. De esta forma garantizamos que si el microcontrolador entra en un estado problemático reiniciará en unos segundos.

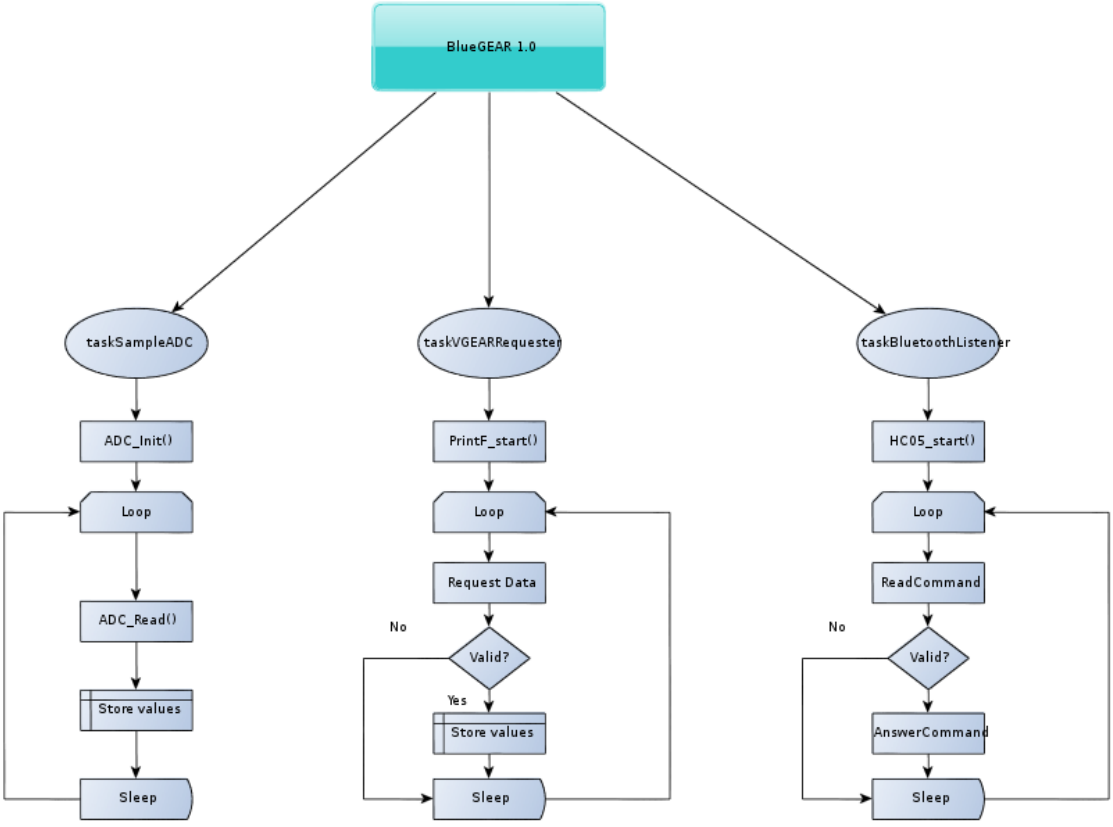

Figura 28: Máquinas de estado de BlueGEAR 1.0.

taskSampleADC

Esta tarea muestra con el ADC el módulo acelerómetro. Guarda los datos en la estructura ecuStatus de tipo ECU_STATUS.

taskVGEARRequester

Esta tarea se encarga de conectar con VGEAR, solicitar datos de la ECU y almacenarlos en la estructura ecuStatus.

taskBluetoothListener

Esta tarea se encarga de leer de la UART conectada al módulo Bluetooth HC05. Si detecta un comando válido le proporciona respuesta.

4.2.2.2 BlueGEAR 1.1

La versión 1.1 modifica la tarea taskVGEARRequester y ahora es la encargada de conectar con la ECU de la motocicleta eliminando así la necesidad de VGEAR. Usa el módulo KWP2000 de la biblioteca LPC1769_UOC_Library.

Esto consigue reducir la latencia de las comunicaciones ya que el LPC1769 es el encargado de conectar con la ECU.

4.2.3 Aplicación VGEAR para Android

Esta aplicación se ejecuta sobre terminales Android 4.1 o superior. Permite mostrar los datos de BlueGEAR obtenidos mediante una conexión Bluetooth.

4.2.3.1 VGEAR Android 1.0

La aplicación VGEAR para terminales Android se diseñó con Qt5 debido a que permite ser compilado para diversas plataformas con un mínimo de cambios.

En particular Qt5 permite compilar para Android (cualquier versión), para IOS 8.1 y para

Windows Phone por lo que la aplicación puede ser instalada en el 99,1% de los terminales móviles (porcentajes extraídos de International Data Corporation[6] para el cuadrimestre 4 de 2014).

El funcionamiento de la aplicación se basa en el protocolo VAP (VGEAR At Protocol ver capítulo 4.2.1.1).

Cada 200ms se solicita el mensaje AT+ALL al LPC1769, este retorna la última información disponible de la ECU y se muestra por pantalla.

Además se muestran datos del acelerómetro junto con la velocidad obtenida del GPS del terminal móvil.

6 http://www.idc.com/prodserv/smartphone-os-market-share.jsp

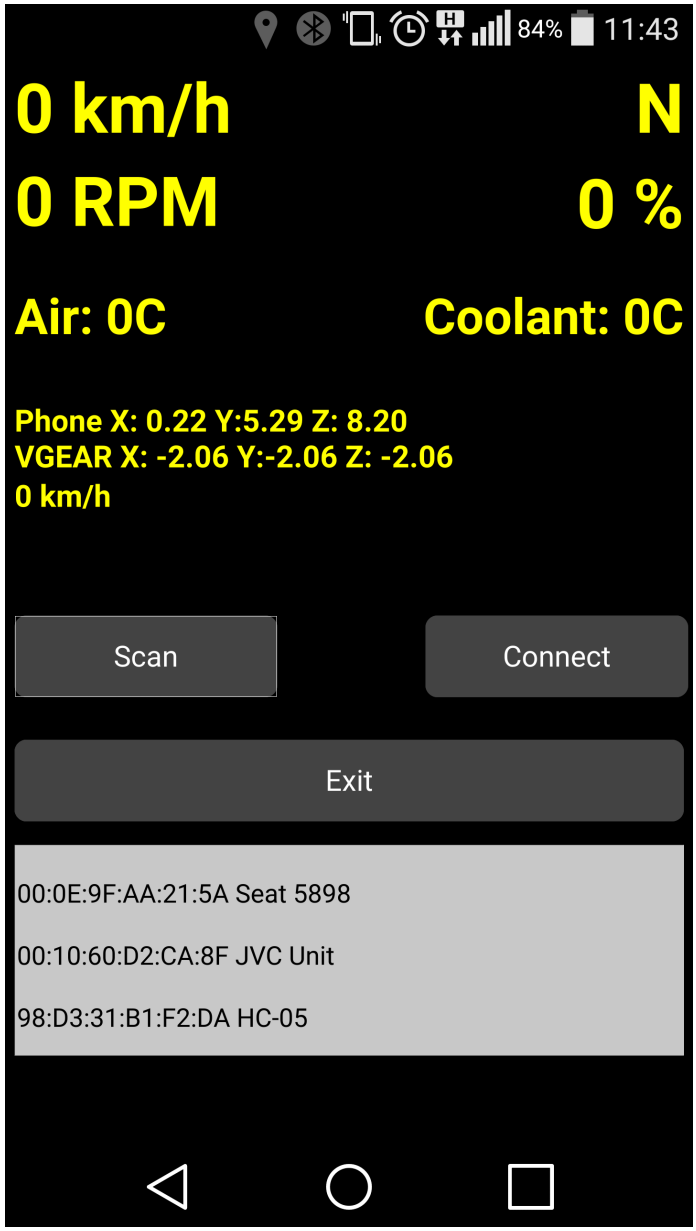

Figura 29: VGEAR Android 1.0.

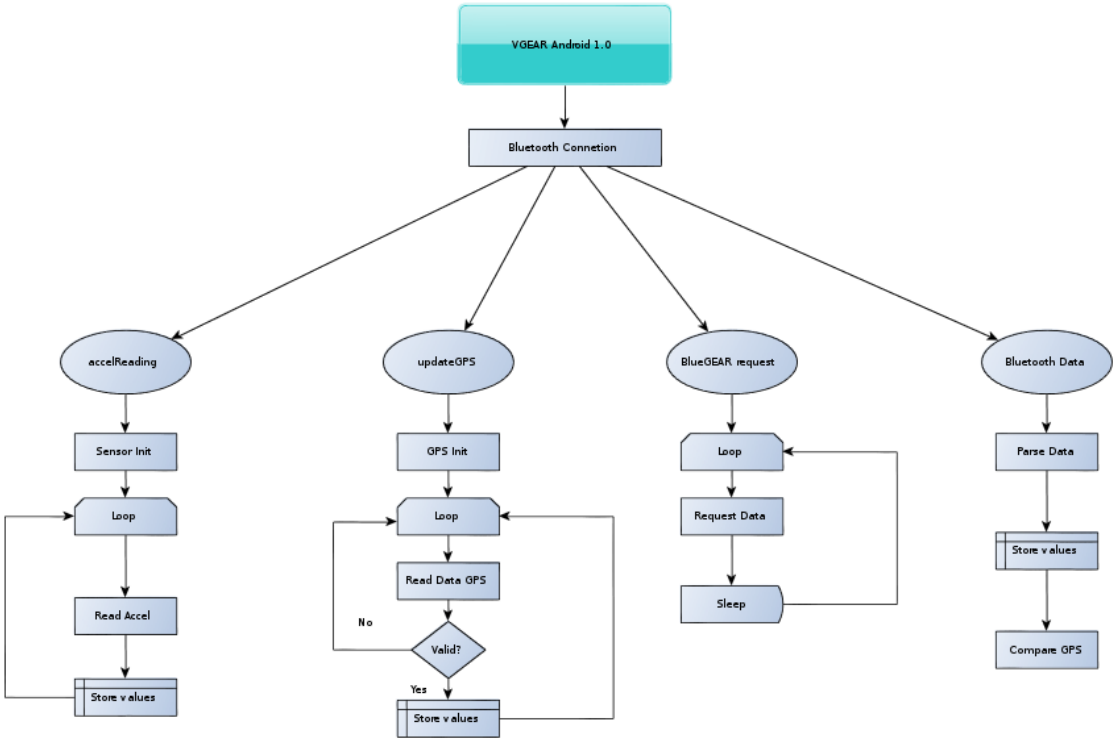

Figura 30: Flujos de ejecución en VGEAR Android 1.0.

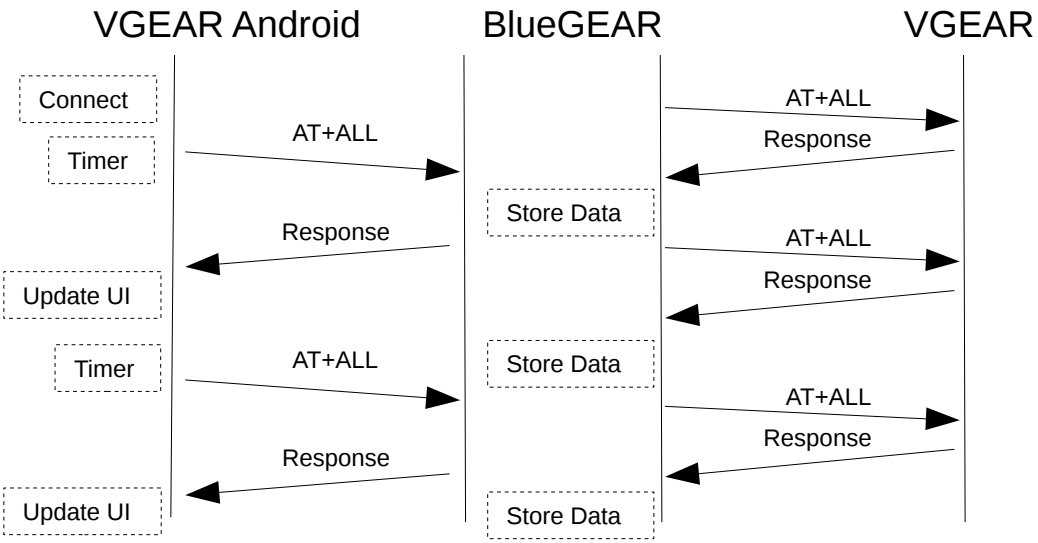

Figura 31: Comunicación entre VGEAR Android 1.0, BlueGEAR y VGEAR.

4.2.3.2 VGEAR Android 1.1

La aplicación VGEAR para Android versión 1.1 elimina las peticiones de datos y directamente espera recibir los datos de BlueGEAR. De esta manera se consigue disminuir la latencia en las comunicaciones.

Además se elimina el cuadro de texto inferior que servía de log y se añade un nuevo gráfico para mostrar el porcentaje de apertura de la válvula de admisión.

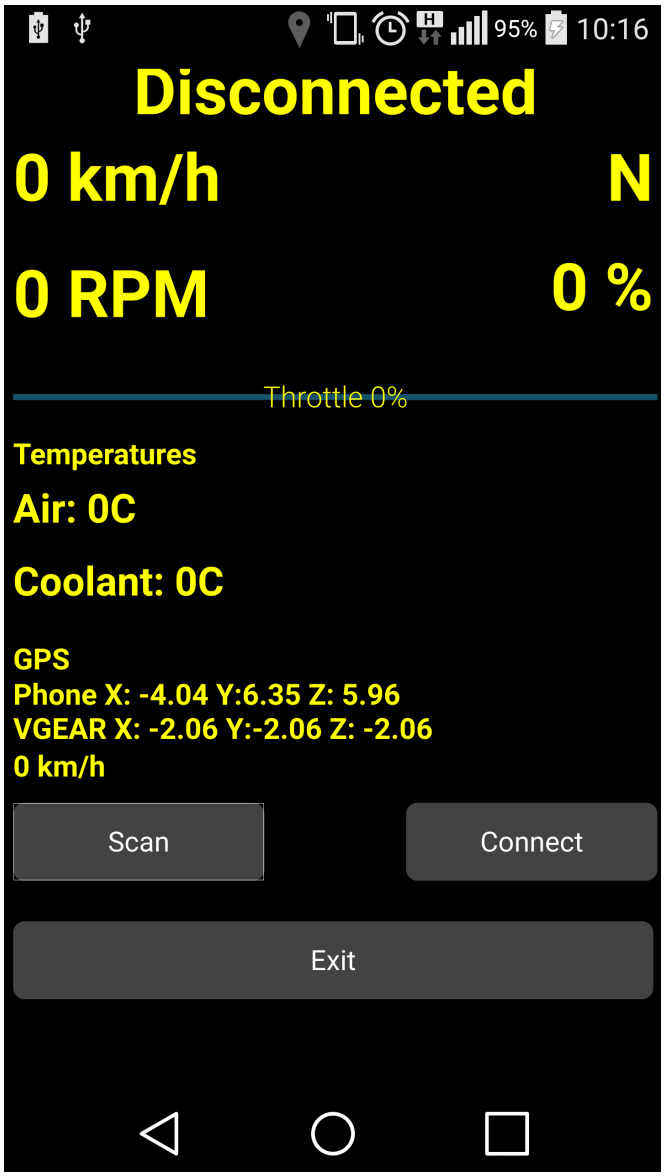

Figura 32: VGEAR para Android 1.1.

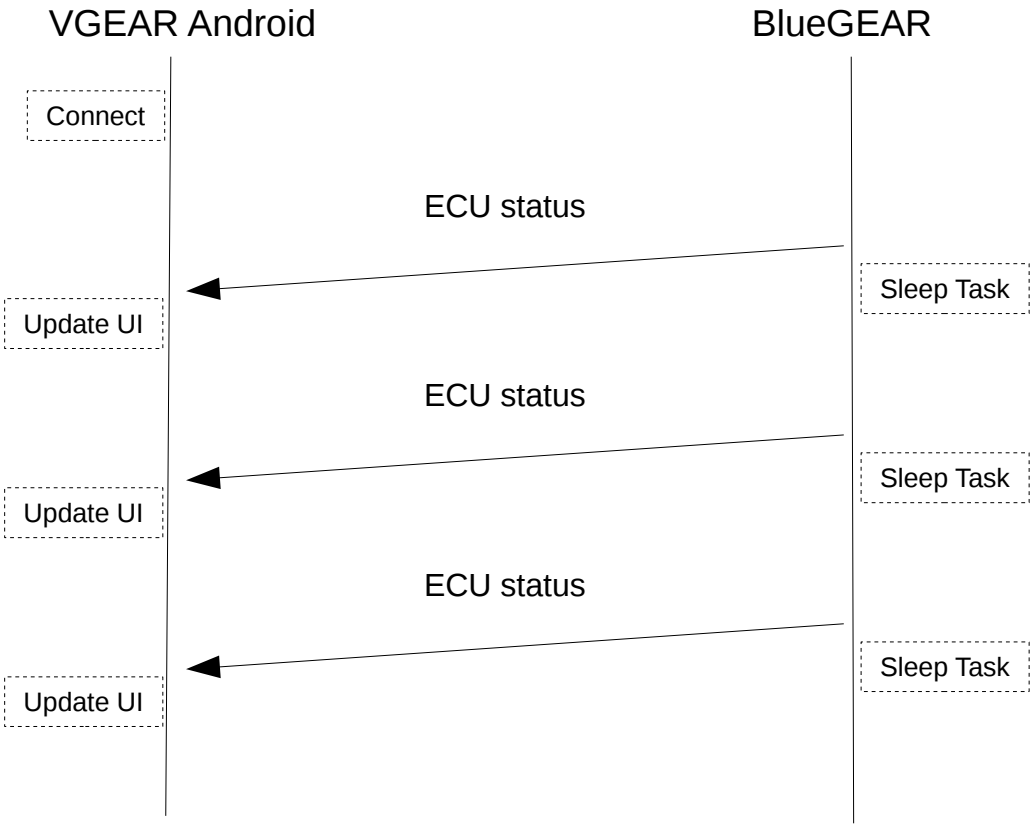

Figura 33: Comunicación entre VGEAR para Android y BlueGEAR.

5 Conexión con la ECU

5.1 Protocolo ISO14230

El protocolo de comunicaciones ISO14230 también es conocido como Keyword Protocol 2000. Era ampliamente utilizado por la industria de la automoción para los sistemas de diagnóstico a bordo (On-board Vehicle Diagnostics). Actualmente está siendo desplazado por el UDS (Unified Diagnostic Services ISO14229-1:2013).

La capa física es la misma que la de ISO9141: una línea serie denominada K-line bidireccional. Opcionalmente se puede implementar la linea L-line para operaciones de “wakeup”. La tasa de transferencia puede variar desde 1,2 kbps a 10,4 kbps.

Cuando no se implementa la línea L-line se debe realizar una de las dos secuencia de inicialización: fast-init o bien 5-baud wakeup. Estas secuencias consisten en el envío de un pulso a la ECU de motor seguido de una trama de 5 bytes concreta.

El estándar es muy amplio y soporta muchas posibilidades. En los apartados siguientes se hace un resumen de las partes más significativas relativas a la comunicación de VGEAR con la ECU de la Suzuki Intruder M800.

5.1.1 Terminología

El documento del estándar ISO14230 nombra al dispositivo que se conecta a la ECU herramienta tester o tester.

5.1.2 Topología

El diagrama siguiente muestra cómo es la topología de una conexión al bus K-Line.

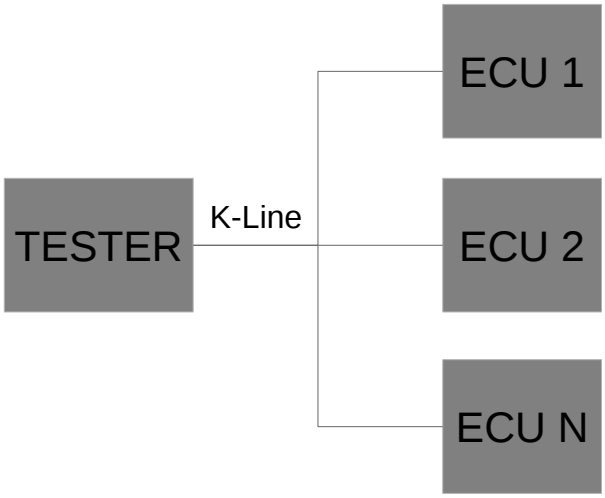

Figura 34: Topología de red ISO14230.

5.1.3 Características eléctricas

A continuación se especifican algunos valores que son necesarios cumplir según el estándar. Se indican los valores para 12V y para 24V.

	Voltaje batería VBat	
Características del tester	12V	24V
Tensión que debe soportar el tester	8V a 16V	16V a 32V
Rango de temperatura	-20C a 50C	-20C a 50C
K-Line no conectada a ECU	Pullup a VBat con 510Ω ± 5%	Pullup a VBat con 1KΩ ± 5%
K-Line con 1 lógico	90% VBat con 510Ω ± 5%	90% VBat con 1kΩ ± 5%
K-Line con 0 lógico	10% Vbat con 100mA drenaje	
Resistencia mínima que se debe considerar entre K-Line y GND del vehículo	5KΩ	10kΩ
Capacidad máxima del tester hasta el conector de diagnóstico	2nF	

5.1.4 Formato de trama

Las tramas que se envían y reciben constan de 3 partes:

- Cabecera
- Payload
- CRC

Cabecera				Mensaje			CRC
FMT	DST	SRC	LEN	Payload	...	...	CRC

Cabecera

La cabecera de la trama se puede formar de dos formas: incluyendo un byte con el tamaño del payload o bien sin incluirlo.

Cabecera de 4 bytes			
FMT	DST	SRC	LEN

Cabecera de 3 bytes		
FMT	DST	SRC

FMT: 1 byte formato de mensaje.

DST: 1 byte destino del mensaje.

SRC: 1 byte origen del mensaje.

LEN: 1 byte longitud del mensaje.

El byte FMT codifica en 2 bits (A1 y A0) el tipo de direccionamiento y en los 6 siguientes (L5 a L0) la longitud del mensaje. Si de L5 a L0 es 0 entonces se debe incluir un byte con el tamaño del payload y la cabecera pasará a constar de 4 bytes permitiendo un payload de hasta 255 bytes.

	MSB							LSB
	A1	A0	L5	L4	L3	L2	L1	L0
Direccionamiento físico	1	0	X	X	X	X	X	X
Direccionamiento lógico	1	1	X	X	X	X	X	X

DST es la dirección física o lógica (en función de los bits A1 y A0 del byte FMT).

SRC es la dirección del tester, usualmente las herramientas de diagnóstico se identifican como 0xF1.

Payload

Aquí se codifica el tipo de petición y los argumentos de esta.

CRC

El CRC simplemente es la suma de todos los bytes de la trama (excepto el mismo byte de CRC) módulo 256.

Si Bi es es el Byte en la posición i de la trama de longitud N Bytes entonces CRC es:

$$CRC = \left(\sum_{i=0}^{N-1} B_i\right) mod\, 256$$

5.1.5 Fast-init

El estándar define dos formas de iniciar la comunicación con la ECU por parte del tester. 5-baud wakeup y fast-init.

VGEAR sólo implementa fast-init por lo que sólo se explicará este método. La principal diferencia respecto a 5-baud init son los bit por segundo de la primera trama y la duración de los pulsos de inicialización.

El proceso consiste en poner la línea K-Line a 0 durante 25ms, a 1 durante 25ms y enviar una

trama con petición de servicio *startCommunication* a 10400bps con un tiempo entre bytes de 5 a 20ms. Más información de los tiempos del protocolo se encuentra en el capítulo 5.1.6.

FMT	DST	SRC	0x81	CRC
Byte FMT.	ECU destino.	Identificador del tester.	Servicio *startCommunication*.	Byte de CRC.

La ECU responde con la siguiente trama:

FMT	DST	SRC	LEN	RES	KB1	KB2	CRC
Byte FMT.	Destino tester.	Origen del mensaje.	Longitud del payload.	Respuesta positiva: 0xC1.	Key byte 1.	Key byte 2.	Byte CRC.

Los KB1 (Key byte 1) y KB2 (Key byte 2) son dos bytes que codifican los servicios que la ECU dispone.

KB1							
MSB							LSB
7	6	5	4	3	2	1	0
Paridad del byte.	1	TP1	TP0	HB1	HB0	AL1	AL0

Bit	Valor 0	Valor 1
TP1	Tiempos extendidos.	Tiempos normales.
TP0	Tiempos normales.	Tiempos extendidos.
HB1	Target y Source en cabecera no soportado.	Target y Source en cabecera sí soportado.
HB0	Cabeceras de 1 byte no soportadas.	Cabeceras de 1 byte sí soportadas.
AL1	No se soporta byte adicional de información de longitud.	Se soporta byte adicional de información de longitud.
AL0	No se soporta información de longitud del mensaje en el byte FMT.	Se soporta información de longitud del mensaje en el byte FMT.

Key byte 2

Este byte sirve como CRC y simplemente se envía el valor 0x0F en los bits de 0 a 6 y 1 para indicar que los bits a 1 son pares.

KB2							
MSB							LSB
7	6	5	4	3	2	1	0
Paridad del mensaje.	0	0	0	1	1	1	1

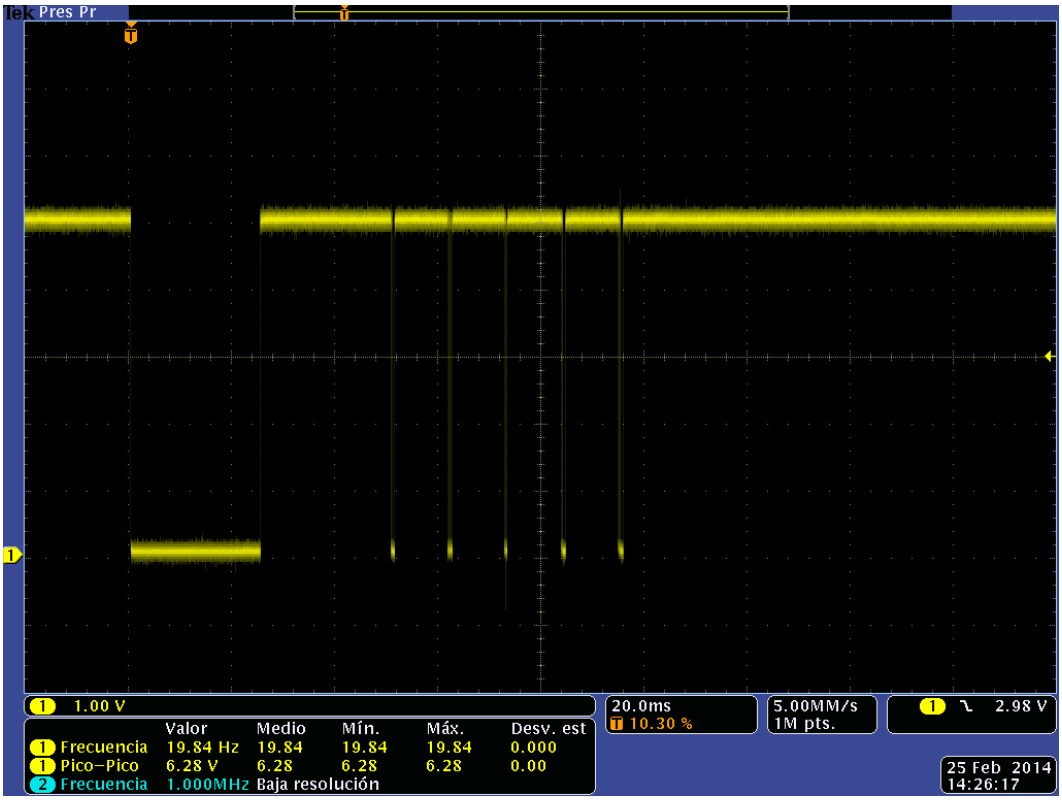

Figura 35: Captura de osciloscopio mostrando la secuencia de inicialización fast-init.

5.1.6 Tiempos

El estándar define los siguientes tiempos relevantes para el desarrollo de herramientas de diagnóstico que se conecten al bus.

Mínimo (ms)	Máximo (ms)	Descripción
0	20	Tiempo entre bytes en la respuesta de la ECU.
5	20	Tiempo entre bytes en el comando del tester.
25	50	Tiempo entre el fin de un comando del tester y la respuesta de la ECU o entre respuestas de la ECU.
25	5000	Tiempo entre el final de una respuesta de la ECU y la siguiente petición del tester o tiempo que debe esperar el tester a realizar una nueva petición si la ECU no response

5.1.7 Ejemplo de comunicación

En este ejemplo se muestra como VGEAR inicia las comunicaciones con la ECU de la Suzuki Intruder M800.

VGEAR se identifica como tester 0xF1 y los mensajes son enviados a la ECU 0x12.

VGEAR				
0x81	0x12	0xF1	0x81	0x05
FMT	Destino	Origen	Payload	CRC

FMT: Se codifica la longitud en el mismo byte FMT (1 byte en total). Direccionamiento físico.
Destino del mensaje: ECU 0x12.
Origen del mensaje: tester 0xF1.
Payload: 0x81 Comando StartCommunication.
CRC: 0x05.

ECU							
0x80	0xF1	0x12	0x03	0xC1	0xEA	0x8F	0xC0
FMT	Destino	Origen	Longitud	Respuesta	Key Byte 1	Key Byte 2	CRC

FMT sin información de longitud.
Destino del mensaje: tester 0xF1.
Origen del mensaje: ECU 0x12.

Longitud del payload: 3 bytes.
Respuesta: 0xC1 comando StartCommunication aceptado.
Key Byte 1: 0xEA.
Key Byte 2: 0x8F.
CRC: 0xC0.

5.2 Protocolo ISO15031-5

El protocolo estándar de automoción SAE J1979[7] o ISO15031-5 define, entre otros, los diversos servicios que una ECU debe proporcionar. Este protocolo es de nivel de aplicación (nivel 7 en la pila OSI) y es implementado por una gran cantidad de fabricantes de automóviles y motocicletas.

Entre los servicios disponible se encuentra el 0x21 que sirve para leer datos de la ECU dado un identificador. La ECU responde con una ristra de bytes con el contenido que el fabricante decida enviar. Mediante este servicio VGEAR extrae la información de la ECU tal y como veremos a continuación.

5.3 Suzuki Diagnosis System

La herramienta de diagnóstico oficial de Suzuki es capaz de extraer muchísima información de la ECU. Consiste en una herramienta industrial de diagnóstico que se conecta al bus ISO14230 de las motocicletas Suzuki y al puerto serie (o USB según modelo) del computador. Como protocolo de comunicaciones con la motocicleta a nivel de aplicación usa ISO15031-5.

Como interfaz de usuario (figura 36) utiliza un software propietario que sólo funciona en Microsoft Windows. Este software permite realizar pruebas de diagnóstico a la ECU y extraer información detallada de los parámetros que la ECU monitoriza.

7 http://standards.sae.org/j1979_201202/

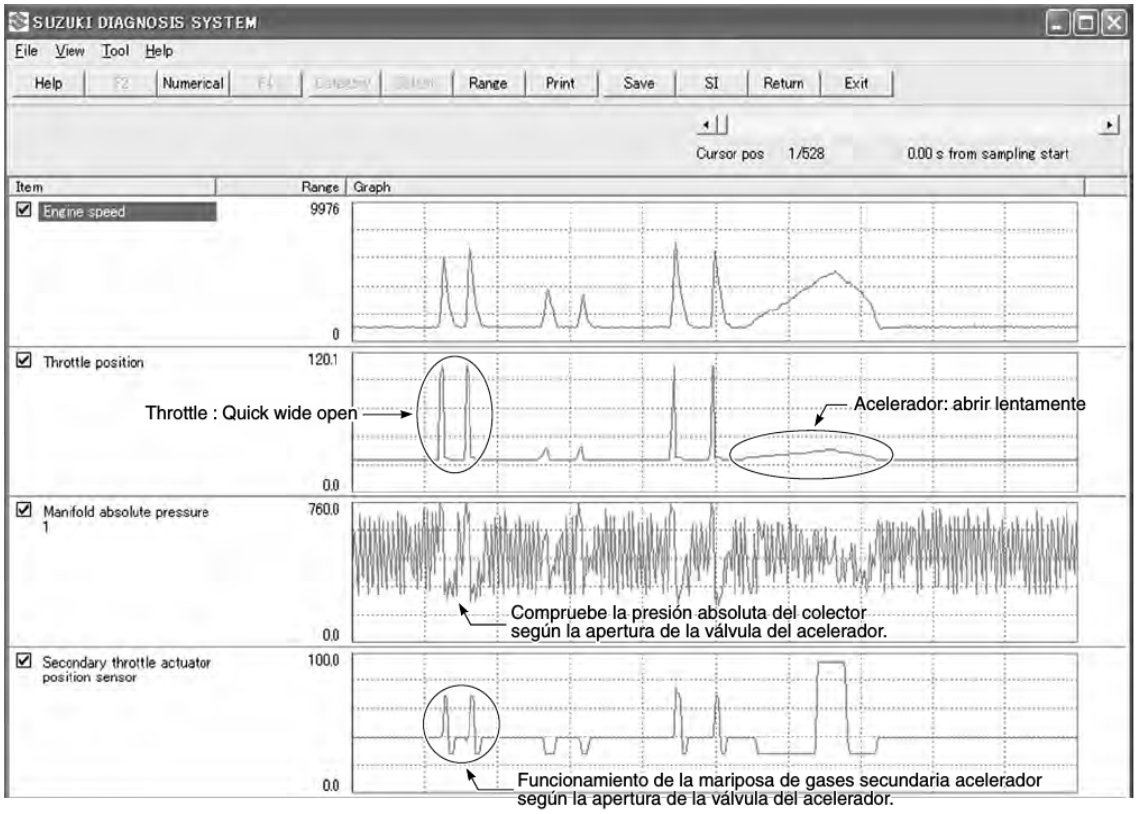

Figura 36: Captura de pantalla de Suzuki Diagnosis System

Gracias al trabajo de ingeniería inversa de una persona que prefiere mantenerse en el anonimato se localizó un punto en el programa donde se solicitaba el servicio 0x21 definido en el estándar ISO15031-5 a la ECU de la Suzuki Intruder M800 y se pudo averiguar alguno de los campos de la respuesta.

Lamentablemente no se dispone del permiso de esa persona para publicar las capturas de datos y sólo se detallan algunos de los campos de los cuales sí se ha obtenido permiso.

Trama enviada por la herramienta SDS:

0x80	0x12	0xF1	0x02	0x21	0x08	0xAE
Cabecera del mensaje	ECU destino	ID del tester	Longitud de los datos de la petición	Servicio de lectura por identificador local	Identificador local	CRC del mensaje

Respuesta recibida:

80 F1 12 34 61 08 02 05 05 A0 17 69 A2 FF FF FF 00 00 00 37 B8 6B 61 B9 00 FF 00 FF 5E 1F FF 00 00 00 00 00 00 00 00 FF FF 40 40 40 40 FF 1A 00 CB 1A 30 00 04 00 FF FF 07

Los 4 primeros bytes corresponden a la cabecera de la respuesta. A continuación se listan algunos de los valores de la respuesta y su interpretación:

Byte	Valor	Descripción
16	00	Velocidad Velocidad = byte16 * 2 km/h
17	00	RPM byte1
18	00	RPM byte2 RPM = 10 * byte17 + byte18 / 10
19	37	Apertura mariposa de admisión. 0x37 es 0%
21	6B	Temperatura del motor. (byte 22 - 48) / 1.6 en Celsius
22	61	Temperatura de la entrada de aire del motor. (byte22 - 48) / 1.6 en Celsius
26	00	Indicar de marcha engranada. 0 indica ninguna marcha.

6 Viabilidad técnica

El proyecto es viable técnicamente y económicamente aunque se deben seleccionar componentes de rango automoción ya que probablemente sería necesaria una homologación. Esto encarecerá el precio del producto.

Puntos débiles	Puntos fuertes
Sólo soporta motocicletas Suzuki	Fácilmente ampliable
No dispone de electrónica de automoción	Diseño sencillo
No se sabe cuándo el módulo HC05 pasará a obsoleto.	Especificaciones abiertas (puede interesar a aficionados)

6.1 Elección del microcontrolador

El microcontrolador LPC1769 no es válido para automoción. No se podría llegar a homologar VGEAR basado en este diseño.

El microcontrolador de gama automoción debe proporcionar al menos dos UART y posiblemente interfaces para bus CAN, LIN y FlexRay para futuras ampliaciones.

6.2 Modelo de negocio

Se propone comercializar la unidad VGEAR homologada para automoción y permitir que aplicaciones de terceros puedan conectarse por Bluetooth. Esto ayudará a popularizar el producto y puede generar comunidades de entusiastas que programen software para él.

Un ejemplo de ello es ELM327, existe gran cantidad de software desarrollado por entusiastas y empresas que implementan el protocolo necesario para comunicar con este dispositivo.

7 Valoración económica

7.1 Alternativas económicas

Dado que el procesador ARM LPC1769 no es de grado automoción se debe seleccionar un microcontrolador adecuado para poder homologar VGEAR.

Un candidato podría ser la gama Hercules TMS470M de Texas Instruments. En concreto el microcontrolador TMS470MF03107 consta de 2 UART, 2 CAN y ADC de 10 bits y 16 canales. Su precio es de 6,17USD/unidad para 1000 unidades.

La gama TMS570L y en concreto el TMS570LS0332 proporciona 2 UART, 2 CAN, 2 FlexRay, 2 LIN y 2 ADC de 12 bits con 41 canales. Su precio es de 11€ para 1 unidad.

	TMS470MF03107	**TMS570LS0332**
UART	2	2
CAN	2	2
FlexRay	-	2
LIN	-	2
ADC	10 bits (16 canales)	2x12 bits (41 canales)
Velocidad	80MHz	300MHz
Precio	6,17USD/unidad (5000)	5,92USD/unidad (5000)

Si descartamos arquitectura ARM y optamos por Microchip disponemos del dsPIC33FJ256GP710A a un precio de 6,20USD/unidad para 5000 unidades. Dispone de 2 UART y ADC de 12 bits además de 2 interfaces de CAN.

El TMS570LS0332 de Texas Instruments nos permitiría, en un futuro, ampliar la conectividad del dispositivo.

7.2 Coste del prototipo

El prototipo de VGEAR Fase II con el microcontrolador TMS570LS0332 a partir de 5000 unidades se desglosa en la siguiente tabla.

Fabricación PCB[8] (5cmx5cm) 2 caras	0,25 €
Montaje PCB	30 x 0,08 €
TMS570LS0332	5,92 USD[9] (5,39 €)
HC05	2 USD (1,82 €)
L9638	0,76 USD (0,69 €)
Componentes discretos varios	2 €
Caja	2 €
Cableado (1m)	1,5€
Conector Sumitomo MT090	1 USD (0,91 €)
Total	16,96€

Se estima que un diseño en SMD de 5cmx5cm y dos capas es suficiente para albergar toda la electrónica a excepción del módulo HC05. También se estiman unos 30 componentes SMD: resistencias, reguladores de tensión, diodos y condensadores.

Estos costes pueden ser decrementados si se realiza una producción a mayor escala.

8 Precios consultados en CIPSA Circuits http://www.cipsacircuits.com/

9 Tipo de cambio dólar / euro: 0,91€/USD.

8 Conclusión

8.1 Conclusión

Para este proyecto final de carrera (PFC) han resultado esenciales las siguientes asignaturas:

- Fonaments tecnològics II: conocimientos de electrónica básica para el diseño de la alimentación del HC05 (ver punto 4.1.2)
- Sistemes electrònics digitals: conocimientos básicos de electrónica digital.
- Projectes: planificar y dividir un proyecto correctamente.

La experiencia en la realización de otro PFC en otros estudios me ha servido para evitar cometer errores importantes:

- División de tareas incorrecta: no saber dividir el proyecto en diferentes unidades.
- Planificación irreal: no saber asignar a las tareas el tiempo real.
- Poca previsión para cambios: no ser capaz de reaccionar, en un espacio de tiempo corto, a los cambios o imprevistos que pueden surgir.
- Falta de tiempo para pruebas y testeo: no planificar correctamente el tiempo para testear los diferentes sistemas.

Lamentablemente el tiempo que esta asignatura dedica al PFC es muy escueto, se sobrecarga al alumno con demasiadas PAC previas al inicio del proyecto. Además se obliga a seguir una nomenclatura particular. Esa falta de libertad puede ser una ventaja para algunos alumnos que afronten por primera vez un PFC pero quizá también limita a otro tipo de alumno que prefiere diseñar su PFC desde 0.

En conclusión, el PFC es una herramienta útil para poner en práctica los conocimientos adquiridos a lo largo de los estudios y permite al alumno desarrollar ideas propias que pueden llegar incluso a comercializarse o pueden servir de base para productos comerciales.

8.2 Mejoras

Como casi cualquier proyecto o producto este PFC se puede mejorar en muchísimos aspectos.

- Hardware de automoción: elegir un microcontrolador de rango automoción y componentes de automoción.
- Diseño de PCB: diseñar un PCB que integre toda la electrónica.
- Implementar gestión de errores del protocolo ISO15031. Muchos aspectos del estándar no se han implementado porque no eran necesarios pero sí que añadirían robustez al sistema.
- Implementar más buses de comunicación de automoción: CAN, FlexRay, Lin son los estándares habituales en automoción y desplazan a ISO14230 desde hace años.
- Implementar más protocolos de diagnóstico: en concreto ODBII y UDS son ampliamente utilizados en la industria de la automoción.
- Refactorizar el código fuente para que cumpla con el estándar MISRA C.
- Diseñar una capa HAL para independizar el software del microcontrolador seleccionado.

8.3 Autoevaluación

Los objetivos iniciales se han cumplido gracias a la dedicación y al seguimiento de la planificación a pesar de que hay muchas partes que pueden ser mejoradas.

Se ha intentado documentar todo el software desarrollado de la forma más adecuada posible y el código escrito ha intentado seguir el estándar MISRA C en la medida de lo posible aunque esto quedaba fuera del alcance del proyecto. Las funciones escritas devuelven códigos de error que puede ser consultados en los ficheros de cabecera.

El driver HC05 se puede aprovechar para otros proyectos junto con el driver del ADC, esto puede ser una ventaja para otros proyectos.

9 Glosario

ECU. Engine Control Unit. Unidad de control de motor. Es una unidad de control electrónico que administra diversos aspectos de la operación de combustión interna del motor.

ELM327. Microcontrolador comercializado por ELM Electronics con firmware capaz de conectar con múltiples protocolos de diagnóstico de automoción.

FlexRay. Protocolo de comunicaciones para buses de datos en el automóvil. Desarrollado por el consorcio FlexRay entre 2000 y 2009.

GPIO. General purpose input/output. Pin genérico en un chip cuyo comportamiento (entrada o salida) se puede programar por el usuario en tiempo de ejecución.

LIN. Bus de datos para automoción. Creado por el consorcio LIN en 1999.

MISRA C. Subconjunto del lenguaje C desarrollado por la Motor Industry Software Reliability Association. El software de automoción debe cumplir este estándar para poder certificarse.

Mutex. En informática un mutex es un mecanismo o algoritmo para evitar que un recurso compartido sea usado por más de un proceso.

OBD. On board diagnostics, estándar del sector de la automoción que define un método de acceso a la ECU de los automóviles y procedimientos de diagnóstico.

UDS. Unified Diagnostic Services, estándar del sector de automoción que representa una mejora respecto a OBD.

Watchdog. En electrónica, mecanismo de seguridad que realiza un reset del sistema si se bloquea.

10 Bibliografía

Application Note AVR181: Automotive Grade0 - PCB and Assembly Recommendations, Atmel. http://www.atmel.com/Images/doc7760.pdf

Calculating & Specifying Electrical Characteristics of PCBs, Neil Chamberlain http://www.polarinstruments.com/support/cits/cits_index.html

Keyword Protocol 2000 - Part 1 - Data Link Layer, Swedish Implementation Standard, L. Magnusson http://www.alfa145.co.uk/obd/14230-2s.pdf

Keyword Protocol 2000 - Part 2 - Data Link Layer, Swedish Implementation Standard, L. Magnusson http://www.alfa145.co.uk/obd/14230-2s.pdf

Microcontroladores: fundamentos y aplicaciones con PIC, Valdés, Fernando / Pallas, Ramón ISBN: 8426714145, 1ª edición 1997.

The art of electronics 2nd edition, Horowith, Paul / Hill, Winfield. ISBN: 0521370957, 2ª edición 1989.

11 Anexos

11.1 Licencias del software

Todo el software de este proyecto se licencia bajo General Public License Version 2 GPLv2. Una copia de la licencia se puede descargar de https://www.gnu.org/licenses/gpl-2.0.html.

11.2 Anexo I: Programas de pruebas

A continuación se listan los programas de prueba junto con una descripción de su objetivo.

11.2.1 Arduino

Todos los programas distribuidos para Arduino se pueden compilar usando Makefile sin necesidad de abrir el editor de Arduino. Sólo es necesario modificar el fichero makefile para indicar la ruta a Arduino Makefile.

Arduino Makefile es un proyecto de software libre gestionado por Sudar Muthu y Simon John. Se puede descargar de https://github.com/sudar/Arduino-Makefile.

eco_test.ino

Programa simple que incializa una UART software en los pines 10 y 11. Todo lo que recibe de esa UART lo reenvía a la UART hardware y a su vez a la software haciendo eco.

El objetivo de este programa es comprobar la comunicación con el LPC1769.

test_command_parser.ino

Este software lee de la UART hardware una cadena de texto acabada en '\n' e intenta parsearla con el parser de protocolo VAP. Si lo consigue devuelve una respuesta acorde con el comando solicitado.

Mediante este software se puede comprobar fácilmente si el protocolo VAP está bien implementado en Arduino.

11.2.2 LPC1769

Los programas de test de LPC1769 se incluyen en el workspace de LPCExpresso. Se deben importar en este software para poder ser compilados.

adctest

Este programa de test usa la biblioteca de LPC1769_UOC_Library para leer del ADC y mostrar los valores por consola. Necesitamos configurar el proyecto como “semihost” para poder visualizar los valores leídos.

Utilizando la siguiente tabla es posible leer los valores que el módulo MMA7631 proporciona de forma sencilla.

PIN LPC1769	PIN MMA7631
28 VOUT (3.3V)	3V3
	Sleep
1 GND	GND
15 ADC0	X
16 ADC1	Y
17 ADC2	Z

11.2.3 X86

Los programas de esta categoría se pueden compilar mediante GCC en plataformas x86, x64 y ARM ya que exclusivamente usan libc.

test_vap_x86

Programa para comprobar el parseo del protocolo VAP en x86. Dado que es más rápido desarrollar en PC y testear se implementó el módulo command.c y luego fue portado a LPC1769 y Arduino.

Primero el programa parsea mediante la función parseCommand una petición de tipo AT+ALL, esta funcionalidad es la misma que hace VGEAR en Arduino. Devuelve una respuesta y es mostrada por el canal estándar de salida.

A continuación simula una respuesta "69 1200 180.32 200.55 4 45\n" y la parsea mediante VGEAR_StringToECUStatus introduciéndola en una estructura de tipo ECU_STATUS mostrándola por el canal estándar de salida.

Esta funcionalidad es la misma que hace BlueGEAR 1.0 cuando recibe una respuesta de tipo AT+ALL de VGEAR. También es usada en VGEAR Android.

11.3 Anexo III: ELM327

El ELM327 es un microcontrolador ya programado comercializado por ELM Electronics[10]. Es un PIC18F2480 que implementa numerosos protocolos OBD (On-Board Diagnostics) y permite conectarse a las ECU de automóviles y motocicletas de una forma sencilla. Dispone de un modo comandos que permite configurar multitud de parámetros.

Se pueden encontrar copias baratas del ELM327 versión 1.0 ya que ELM Electronics no protegió el firmware contra lectura en su primera versión y fue copiado por diversos fabricantes de origen asiático. Además se comercializan con toda la electrónica necesaria para ser conectados al conectar ODB de los automóviles.

Figura 37: Copia de origen Chino del ELM327.

Antes de tener lista la electrónica de VGEAR se utilizó un ELM327 de origen asiático (figura 37) para comprobar el correcto funcionamiento del protocolo.

10 http://elmelectronics.com

11.3.1 Conector OBDII J1962

El conector J1962 está presente en los automóviles europeos de gasolina a partir del año 2000 y en los automóviles diésel desde 2003. La Directiva Europea 98/69EG obligaba a que todo automóvil vendido en Europa a partir de las fechas indicadas implementara este estándar y dispusiera de este conector.

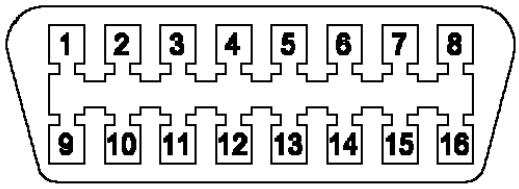

Figura 38: Conector J1962 hembra.

El pinout es el siguiente:

Pin	Descripción	Pin	Descripción
1	Libre uso para el fabricante	9	Libre uso para el fabricante
2	SAE J1850 Positivo	10	SAE J1850 Negativo
3	Libre uso para el fabricante	11	Libre uso para el fabricante
4	GND al chasis	12	Libre uso para el fabricante
5	GND de la señal	13	Libre uso para el fabricante
6	CAN High	14	CAN Low
7	K-Line	15	L-Line
8	Libre uso para el fabricante	16	Positivo batería

11.4 Anexo IV: Software entregado

```
├── android
│   ├── vgear_android: VGEAR para Android 1.0
│   ├── vgear_android_1_1: VGEAR para Android 1.1
├── arduino
│   ├── eco_test: ECO de la UART para Arduino
│   ├── test_command_parser: test para el protocolo VAP
│   ├── test_vap_x86: test para el protocolo VAP
│   └── vgear_arduino: VGEAR para Arduino
├── Arduino-Makefile
├── doc
└── lpc1769
    └── workspace
        ├── adctest: Test del ADC
        ├── AppTest: Test de la UART
        ├── AppTest1: Test módulo Log
        ├── BlueGear_1_0: BlueGEAR 1.0
        ├── BlueGear_1_1: BlueGEAR 1.1
        ├── CMSISv2p00_LPC17xx
        ├── FreeRTOS_Library
        └── LPC1769_UOC_Library: Biblioteca para BlueGEAR.
```

11.5 Anexo V: Documentación del código

Las aplicaciones VGEAR para Android se han documentado con Doxygen. Se puede acceder a su documentación en el directorio doc de cada una de ellas y en el mismo código fuente.

No se ha documentado con Doxygen el software del LPC1769 porque el sistema de documentación no seguía el esquema de Doxygen desde un principio. Todos los ficheros de cabecera están documentados.