Universidad Zaragoza
1542

Trabajo Fin de Grado

Desarrollo de ECU en EV mediante trabajo cooperativo de Microcontrolador y Microprocesador

Autor

Javier Martínez Lahoz

Director

David Asiaín Ansorena

Escuela Universitaria Politécnica La Almunia

2016

ESCUELA UNIVERSITARIA POLITÉCNICA

DE LA ALMUNIA DE DOÑA GODINA (ZARAGOZA)

MEMORIA

Desarrollo de ECU en EV mediante trabajo cooperativo de Microcontrolador y Microprocesador

424.16.93

Autor: Javier Martínez Lahoz

Director: David Asiaín Ansorena

Fecha: Diciembre 2016

INDICE DE CONTENIDO

1. RESUMEN 1

2. ABSTRACT 2

3. INTRODUCCIÓN 3

3.1. MOTIVACIÓN 3

3.2. OBJETIVOS 3

4. MARCO TEÓRICO 5

4.1. MÉTODO DE DISEÑO EN V 5

4.2. AISLAMIENTO ELECTRÓNICO 6

4.2.1. Alimentación aislada 7

4.2.2. Aislamiento de salidas 9

4.3. SUPERCONDENSADORES 10

4.4. PLACAS MICROCONTROLADORAS 10

4.4.1. Arduino Yún 11

4.5. COMUNICACIONES 12

4.5.1. Modulación por ancho de pulso - PWM 12

4.5.2. Bus SPI 13

4.5.3. CANBus 13

4.5.4. NMEA 0183 15

4.5.5. Transmission Control Protocol - TCP 16

5. DESARROLLO 17

5.1. ESTUDIO DE NECESIDADES 17

5.1.1. Necesidades de la ECU 19

5.2. DISEÑO 20

5.2.1. Establecimiento de subsistemas 20

5.2.2. Diseño de subsistemas 22

5.2.2.1. ECU 23

5.2.2.2. CANBus Aislado 27

5.2.2.3. Key-Switch 33

5.2.2.4. Security Maneuver 38

5.2.2.5. GLVS Measures 41

5.2.2.6. Power Manager 44

5.2.2.6.1. Fuente de alimentación 44

5.2.2.6.2. Acumulador de energía 45

5.2.2.6.3. Interruptor de potencia 50

5.2.2.6.4. Acondicionamiento del circuito 51

5.2.2.7. Subsistemas auxiliares 53

5.2.2.8. Interface 55

5.2.3. Diseño de la PCB 59

5.2.3.1. Restricciones de diseño 59

5.2.3.2. Decisiones de diseño 60

5.2.3.3. Disposición de componentes 61

5.2.3.4. Diseño de las pistas 64

5.3. PROTOTIPADO 66

5.3.1. Fabricación 66

5.3.2. Montaje y comprobación 67

5.3.2.1. Power Manager 68

5.3.2.2. ECU 69

5.3.2.3. Security Maneuver 70

5.3.2.4. Key-Switch 71

5.3.2.5. GLVS Measures 73

5.3.2.6. CANBus 74

5.3.3. Fin del prototipado 75

5.4. DISEÑO SOFTWARE 76

5.4.1. Comunicaciones 76

5.4.1.1. Comunicaciones Motocicleta - CANBus 76

5.4.1.2. Comunicaciones internas - NMEA0183 78

5.4.1.3. Comunicaciones hacia el exterior - TCP 79

5.4.2. Software del microcontrolador 79

5.4.2.1. Cometidos del programa 80
5.4.2.2. Paradigma de programación 81
5.4.2.3. Estructura base del programa 82
5.4.2.4. Código del Setup 84
5.4.2.4.1. Serial1.begin 84
5.4.2.4.2. IMD_Config 84
5.4.2.4.3. CAN_Config 85
5.4.2.4.4. InputOutputConfig 86
5.4.2.5. Código del Loop 86
5.4.2.5.1. CAN_Watch 87
5.4.2.5.2. Rd_Serial1 88
5.4.2.5.3. IMD_Conv 88
5.4.2.5.4. Comms_Heartbeat 89
5.4.2.5.5. WatchSecMan 90
5.4.2.5.6. UpdateStatus 91
5.4.2.5.7. Watch_Reload 92
5.4.2.5.8. UpdateDisplay 93
5.4.2.5.9. Status_LED 93
5.4.2.5.10. ECU_Stat 94
5.4.3. Software del microprocesador 95
5.4.3.1. Cometidos del microprocesador 95
5.4.3.2. Lenguaje y Paradigma de programación 96
5.4.3.3. Estructura base del programa 97
5.4.3.4. Código de Python 98
5.4.3.4.1. Librerías, Variables y Configuración 98
5.4.3.4.2. Hilo principal de ejecución 99
5.4.3.4.3. Hilo de transmisión TCP 102
5.4.3.4.4. Hilo de recepción TCP 103
5.4.3.4.5. Hilo de lectura de teclado 103
6. RESULTADOS 104
6.1. RESULTADO DE LAS COMUNICACIONES 104
6.2. RESULTADO DE LA PLACA 107
6.3. RESULTADOS FINALES DE COMPORTAMIENTO 110
7. CONCLUSIONES 112
8. BIBLIOGRAFÍA 115

8.1. REFERENCIAS BIBLIOGRÁFICAS _____ 115

8.2. OTRAS FUENTES CONSULTADAS _____ 115

8.3. DATASHEETS CONSULTADOS _____ 116

INDICE DE ILUSTRACIONES

Ilustración 1. Representación gráfica del modelo en V _____ 5

Ilustración 2. Estructura interna de una fuente conmutada _____ 9

Ilustración 3. Partes destacables del Arduino Yún _____ 11

Ilustración 4. Onda PWM (abajo) y energía entregada (arriba) _____ 12

Ilustración 5. Distribución de conexiones en un bus SPI _____ 13

Ilustración 6. Distribución del Bus CAN _____ 14

Ilustración 7. Trama de bits de un mensaje CANBus con el mensaje natural (arriba) y el mensaje con los bits de sincronismo en morado (abajo) _____ 15

Ilustración 8. Ejemplo de mensaje en protocolo NMEA 0183 _____ 16

Ilustración 9. Esquema de conexionado de buses entre los sistemas de la motocicleta _____ 19

Ilustración 10. Esquema de bloques de la ECU _____ 22

Ilustración 11. Representación electrónica del Arduino y su conexionado _____ 26

Ilustración 12. Esquema de conexionado de dos canales CANBus vía SPI _____ 28

Ilustración 13 Conexionado del canal de comunicaciones CAN GLVS _____ 32

Ilustración 14. Esquema electrónico del Key-Switch _____ 38

Ilustración 15. Esquema de conexionado de la maniobra de seguridad _____ 40

Ilustración 16. Esquema interno del HCPL-2231 _____ 42

Ilustración 17. Conexionado de los tres diferentes optoacopladores del subsistema _____ 43

Ilustración 18. Esquema electrónico del subsistema Power Manager _____ 52

Ilustración 19. Distribución de las conexiones en el clavijero selector de circuitos auxiliares _____ 55

Ilustración 20. Distribución de pines en el conector de baja tensión _____ 58

Ilustración 21. Distribución de pines en el conector de alta tensión _____ 59

Ilustración 22 Disposición de componentes sobre la PCB 62

Ilustración 23. Diseño final de la PCB prototipo 65

Ilustración 24 Diseño definitivo de la PCB 75

Ilustración 25. Código principal del programa del microcontrolador 83

Ilustración 26. Definición de los pines de Arduino de acuerdo a sus funciones 84

Ilustración 27. Límites de memoria según el Arduino IDE 106

Ilustración 28. Datos recibidos de la base de datos 111

Ilustración 29. Gráficas obtenidas con MatLab con los datos de telemetría 111

Ilustración 30. Motocicleta de EUPLA Racing Team donde se instaló la ECU 112

INDICE DE TABLAS

Tabla 1. Tecnologías de aislamiento 6

Tabla 2. Topologías de fuentes de alimentación 8

Tabla 3. Comparativa entre los protocolos I2C y SPI 27

Tabla 4. Mensajes de CANBus 78

Tabla 5. Tiempos de trabajo de las funciones 104

Tabla 6. Ocupación de canales de comunicación 106

Tabla 7. Consumos 109

1. RESUMEN

El presente trabajo desarrollará todo el proceso de diseño, fabricación y programación de una centralita electrónica para vehículo eléctrico. Se comenzará realizando un estudio completo del sistema y de las posibles soluciones que se pueden adoptar para resolver cada cuestión.

El asunto principal en el que se centrará este proyecto consiste en el control inteligente del vehículo y cómo llevarlo a cabo. La opción planteada es el trabajo cooperativo entre microcontrolador y microprocesador. Comenzaremos logrando una comunicación entre ambos subsistemas. Posteriormente, y tras definir las funciones generales que deberá llevar a cabo la ECU en su conjunto, se repartirán las funciones entre el microcontrolador y el microprocesador. Cuando se conozcan todas las funciones a desarrollar, y quién debe llevarlas a cabo, se procederá a un análisis de las mismas y al planteamiento de diferentes soluciones para cada una.

Se escogerán las opciones que se consideren adecuadas en cada caso, y se planteará un prototipo inicial de ECU. Se montará este primer prototipo de ECU, y se realizará un programa prototipo que compruebe las funcionalidades individualmente de cada subsistema.

Tras realimentar el diseño con los resultados de las pruebas del primer prototipo, se procederá a realizar otro prototipo. Este segundo prototipo se considerará ya definitivo, por lo que se realizará de una forma profesional y se montarán los componentes definitivos. Sobre esta versión definitiva se desarrollará el software final. Este software se irá desarrollando poco a poco, añadiendo la conectividad de sistemas externos de uno en uno, hasta que finalmente funcionen todos de forma adecuada.

Finalmente se realizarán unas pruebas funcionales definitivas y se ajustará el software hasta lograr un comportamiento óptimo. El objetivo final es disponer de una centralita inteligente con registro de datos completamente adaptada al sistema en cuestión.

2. ABSTRACT

This essay is going to show the whole process of the design, manufacturing and programming of an ECU made for an EV. The fist step is to make a full study of the complete system and the posible solutions for every situation.

The principal concept that we are interested in this essay is the inteligent control of an electric vehicle and how to develop it. The proposed option is the cooperative work between a microcontroller and a microprocessor. We will begin achieving a simple communication between both subsystems. Later on, after defining the general functionalities that will involve the ECU, the tasks will be divided up between MCU and CPU.

When the functionalities to develop are defined and who must carry out them, the next step will be analyzing and set out solutions for each one. In each case we will choose the solution we consider will work better and develop a first ECU prototype. The second prototype will be considered the definitive one, therefore it will be realized in a professional way and assemble the final components. The needed software will be developed according to this version. This software will be developed step by step, adding the connectivity of external systems one by one, until the fully integration between all the systems.

Finally we will realize full functionality tests and adjust the software until we get an optimal behaviour. The final objetive is getting an inteligent ECU and data logger completelly adapted to the motorbike.

Keywords: Electronic Control, electric vehicle, Motorbike, ECU, CANBus,

3. INTRODUCCIÓN

3.1. MOTIVACIÓN

La reciente aparición de las placas microcontroladoras de bajo coste ha propiciado su empleo en multitud de aplicaciones docentes y de entretenimiento. En muchas ocasiones las capacidades de la electrónica empleada se encuentran muy por encima de la aplicación desarrollada. En este caso se pretende estudiar la viabilidad de embarcar una placa microcontroladora como ECU, así como la posibilidad de desarrollar esa tarea de forma conjunta con un microcontrolador y un microprocesador.

El satisfactorio empleo de estas dos tecnologías de forma cooperativa podría servir como precedente para emplearlas de forma similar en otras aplicaciones de diversas índoles.

El desarrollo propio de este tipo de sistemas nos dará a su vez la posibilidad de ajustar la electrónica y la programación a nuestras necesidades concretas, cosa que no sería posible empleando electrónica comercial, mucho más generalista.

3.2. OBJETIVOS

El primer objetivo del trabajo será realizar un análisis de sistemas y necesidades, a partir del cual se desarrollará todo el trabajo posterior. En este análisis deben quedar definidas las funciones de cada subsistema y las dependencias entre ellos.

El siguiente objetivo del trabajo será diseñar una centralita electrónica (ECU) que sea capaz de satisfacer todas las necesidades encargadas a dichos sistemas. Una vez completado el diseño general, se realizará un diseño por subsistemas de acuerdo a las diferentes tareas a desarrollar. Resulta imprescindible comprobar el correcto funcionamiento de todos los subsistemas antes del diseño definitivo, por lo que lo fijaremos como siguiente objetivo; comprobación general del sistema, y modificación de aquellos subsistemas que no lleven a cabo su función de la forma esperada.

Cuando todos los subsistemas de la ECU funcionen de la forma deseada, se procederá a integrar y comprobar la programación. Finalmente se procederá a la

interconexión de sistemas. Una vez todos los sistemas estén conectados y sean funcionales, se realizarán las modificaciones pertinentes hasta que la ECU, y todos los demás sistemas, funcionen de forma óptima.

4. MARCO TEÓRICO

A continuación se pretende dar una visión general a los fundamentos teóricos y las tecnologías que están directamente relacionadas con nuestra aplicación.

Debido a la gran cantidad de tecnologías y soluciones que se pueden aplicar en estos casos, se emplearán tablas comparativas en las que se enunciarán las principales características de cada opción.

4.1. MÉTODO DE DISEÑO EN V

A la hora de diseñar un producto resulta conveniente seguir un modelo de diseño. Uno de los modelos más empleados en ingeniería es el modelo V. Este surgió originalmente para el desarrollo de software; sin embargo, es aplicable a cualquier campo de diseño.

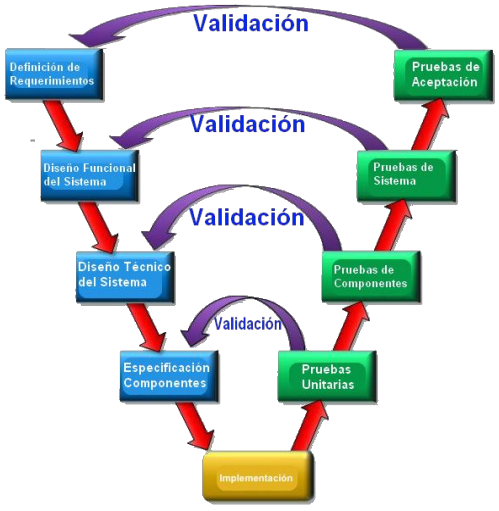

Ilustración 1. Representación gráfica del modelo en V

El modelo en V adquiere su nombre de su representación gráfica. Como se puede observar en la ilustración 1 se divide en dos partes. En la primera parte, mitad descendente de la V, se estudian las necesidades y propuestas de diseño. Se comienza por un estudio global del sistema, en la parte superior, y se realizan diseños cada vez de menor nivel, apartados inferiores, hasta obtener un diseño completo con componentes concretos. El diseño completo se representa en la parte inferior de la V.

La segunda mitad de la V representa las diversas comprobaciones que se van realizando, de menor a mayor nivel. Si alguna prueba detecta un fallo de diseño, se volverá al diseño de ese nivel. Será necesario volver a realizar todas las fases de diseño posteriores.

4.2. AISLAMIENTO ELECTRÓNICO

Uno de los factores más importantes de un producto electrónico es la fiabilidad. Esta viene determinada por la rigurosidad con la que el producto ha sido diseñado. Para obtener una buena fiabilidad en un producto, debemos asegurar que siempre vaya a trabajar en un ambiente admisible, al tiempo que debemos establecer las protecciones adecuadas para evitar o minimizar situaciones que puedan perjudicar al sistema. Una de las principales técnicas empleadas para mejorar la fiabilidad consiste en el aislamiento galvánico del sistema.

El aislamiento galvánico consiste en la separación eléctrica de circuitos electrónicos de forma que no pueda circular corriente directamente entre ellos. De esta manera se logra que una anomalía, como una sobretensión o un cortocircuito, producidos en un lado del circuito no afecte a otras partes del circuito. El nivel de aislamiento galvánico se mide en función de la tensión de rotura. La tensión de rotura es aquella tensión que supera el aislamiento existente y permite el paso de corriente directamente de un circuito al otro. La tensión de rotura se mide en KV, miles de voltios, y cuanto mayor sea, mejor será el aislamiento y la fiabilidad de la electrónica.

Tipo de aislamiento	Ventajas	Desventajas	Componente
Inductivo	• Inmunidad a campos eléctricos • Alta velocidad de transmisión	• Sensible a campos magnéticos	• Transformador de pulsos
Capacitivo	• Inmunidad a campos magnéticos • Alta velocidad de transmisión	• Sensible a campos eléctricos	• Condensador
Óptico	• Inmunidad a campos magnéticos y eléctricos	• Baja velocidad de transmisión mayor consumo energético	• Optoacoplador

Tabla 1. Tecnologías de aislamiento

Autor: **Javier Martínez Lahoz**
424.16.93

Lograr un buen aislamiento implica una separación total entre los conductores de un circuito y otro; sin embargo, es necesario que la información pueda circular correctamente entre ambos circuitos. Actualmente se emplean tres tecnologías que permiten traspasar información entre circuitos a la vez que generan un aislamiento robusto. Estas tecnologías son: el aislamiento capacitivo, el aislamiento inductivo y el aislamiento óptico. La tabla 1 muestra las características principales de cada tecnología de aislamiento. Como puede observarse en la tabla 1, no existe una tecnología claramente superior a las demás. Normalmente se emplea aislamiento óptico cuando no se requiere una velocidad muy elevada ni un consumo muy reducido. Esto se debe a que, al tratarse de una señal digital, es más difícil que se de una lectura fallida. Cuando se requiere de una gran velocidad de transmisión se suele emplear aislamiento inductivo o aislamiento capacitivo en función del circuito.

4.2.1. Alimentación aislada

Para obtener una alimentación eficiente y aislada actualmente se recurre principalmente al uso de fuentes conmutadas. Las fuentes de alimentación conmutadas proporcionan un aislamiento galvánico elevado a la vez que logran una mayor eficiencia que los reguladores lineales de tensión. El principal problema que puede generar este tipo de fuentes se debe a la conmutación a altas frecuencias que realizan para controlar la energía entregada. Esta conmutación provoca ruidos electromagnéticos de alta frecuencia, si no se emplean los filtros de salida de alta frecuencia adecuados pueden provocar problemas en los circuitos electrónicos.

En la tabla 2 se realiza una comparación entre cuatro tecnologías de alimentación. Como podemos observar en la tabla, el principal escollo de las fuentes no reguladas es su falta de estabilidad en la salida, mientras que el de las fuentes reguladas es su baja eficiencia debido a que disipan la energía restante en forma de calor. Además los transformadores de núcleo de hierro resultan mucho más voluminosos y pesados que los transformadores de ferrita. Resulta evidente que las fuentes conmutadas se comportan de una forma mucho más satisfactoria que el resto, con un tamaño menor, si bien requieren de un mayor esfuerzo de diseño.

El funcionamiento paso a paso de las fuentes conmutadas puede apreciarse claramente en la ilustración 2. Las fuentes conmutadas siguen siempre el mismo ciclo de trabajo: filtrado, rectificado (solo AC), conmutación de alta frecuencia, transformación, rectificación de salida, filtrado de salida y control realimentado. El

control toma medidas de la salida en tensión y, tras compararlo con una referencia, actúa sobre la conmutación de alta frecuencia.

Tecnología	Reguladores lineales	Fuente de alimentación no regulada	Fuente de alimentación regulada	Fuente de alimentación conmutada
Componentes principales	• Regulador lineal de tensión	•Transformador con núcleo de hierro	•Transformador con núcleo de hierro y regulador lineal de tensión	•Transformador de pulsos con núcleo de ferrita y transistor de conmutación (MHz)
Tamaño y peso	• Reducido	• Elevado debido al transformador	• Elevado debido al transformador	• Reducido
Voltajes de salida	• Siempre inferior a la entrada	• Cualquiera (dependiendo de la topología)	• Cualquiera (dependiendo de la topología)	• Cualquiera (dependiendo de la topología)
Estabilidad de la salida	• Muy estable frente a cambios de consumos	• Poco estable frente a cambios de consumos	• Muy estable frente a cambios de consumos	• Muy estable frente a cambios de consumos
Eficiencia / Disipación de calor	• Poca eficiencia • Disipa el exceso de tensión restante en forma de calor	• Eficiencia media. • Algunas pérdidas en el transformador	• Poca eficiencia • Disipa la energía restante en forma de calor	• Elevada eficiencia (hasta 95%). • Pérdidas solo por componentes no ideales
Complejidad de diseño	• Reducida	• Reducida	• Reducida	• Elevada. Un diseño más cuidado implica mayor eficiencia
Ruido eléctrico o electromagnético	• Poco ruido y de baja frecuencia	• Poco ruido y de baja frecuencia	• Poco ruido y de baja frecuencia	• Ruido de alta frecuencia que debe ser filtrado

Tabla 2. Topologías de fuentes de alimentación

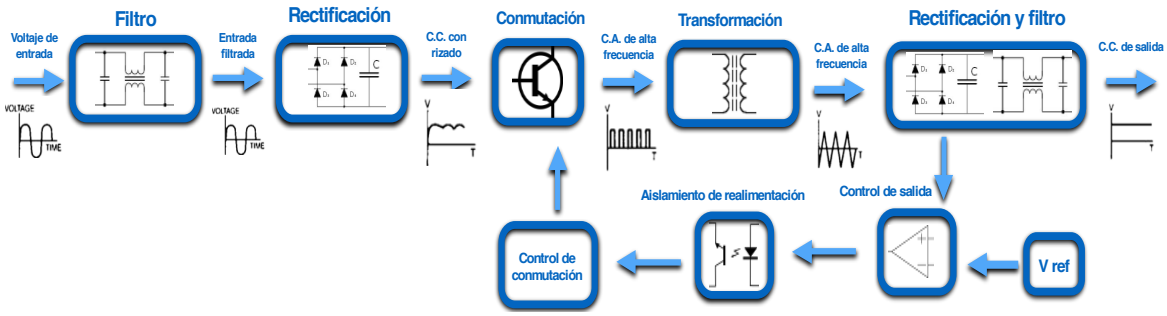

Ilustración 2. Estructura interna de una fuente conmutada

4.2.2. Aislamiento de salidas

El aislamiento y control de salidas requiere del empleo de otras tecnologías. Tradicionalmente se han usado relés electromecánicos, sin embargo, actualmente han aparecido como alternativa los relés de estado sólido (SSR), que se basan en el uso de semiconductores. Estas dos tecnologías difieren considerablemente en comportamiento y características.

Los relés de estado sólido cuentan con numerosas ventajas frente a los relés convencionales: eficiencia, velocidad de conmutación, vida útil, arco eléctrico… Sin embargo, cuentan con un inconveniente muy importante: al estropearse suelen quedar en cortocircuito. Debido a esto no es recomendable emplearlos en aplicaciones de seguridad en las que un fallo debe desconectar el equipo completo.

Los relés electromagnéticos se basan en el movimiento de una pletina metálica, contacto, provocado por un campo magnético. Este campo magnético es generado por una bobina, que es el control del relé. La pletina metálica suele contar con un muelle, este muelle desactiva el relé cuando no se le esta entregando corriente. En la mayoría de los casos este muelle provoca que en caso de darse un fallo de funcionamiento en el relé, el contacto del mismo quede e circuito abierto.

En ambos casos, relés electromecánicos y SSR, es importante tener presentes las tensiones y corrientes de control y de trabajo para un funcionamiento óptimo. Un exceso de corriente o tensión provocará el deterioro o destrucción del componente; mientras que una tensión o corriente inferior a la nominal de control no llegará a cerrar el contacto. Resulta importante destacar que los relés, en la mayoría de los casos e independientemente de su tecnología, requieren una corriente constante para

funcionar. Por lo tanto una menor corriente de disparo implicará un sistema más eficiente.

4.3. SUPERCONDENSADORES

Los condensadores tradicionales generan un campo eléctrico entre dos terminales para acumular energía temporalmente. Los condensadores electrolíticos de doble capa, también llamados supercondensadores, son una familia de condensadores que destacan por las enormes capacidades que poseen, hasta decenas y cientos de faradios. El gran obstáculo para el uso de los supercondensadores es su bajo voltaje nominal, de apenas unos pocos voltios. Para compensar esto se suelen emplear bancos de supercondensadores dispuestos en serie y en paralelo, para aumentar su voltaje de trabajo sin reducir su capacidad global.

Los supercondensadores están sustituyendo a las pilas y baterías químicas en determinados campos debido a la gran energía que son capaces de acumular y el bajo mantenimiento requerido. La energía acumulada por un condensador viene determinada por su capacidad (C) y la diferencia de tensión entre sus bornes (V2-V1).

$$\mathcal{E} = \int_{q_1}^{q_2} V \mathrm{d}q = \int_{q_1}^{q_2} \frac{Q}{C} \mathrm{d}q = \frac{Q^2}{2C} = \frac{1}{2C}(C(V_2 - V_1))^2 = \frac{1}{2}C(V_2 - V_1)^2$$

4.4. PLACAS MICROCONTROLADORAS

Los microcontroladores son sistemas SoC (System on a Chip) que integran todo lo necesario para su funcionamiento: ALU, memoria RAM, memoria EEPROM, reloj, módulos E/S... Las placas microcontroladoras que surgen a día de hoy son una evolución de los microcontroladores. Se trata de microcontroladores con determinados componentes conectados para facilitar su uso, especialmente en tareas de prototipado y educación.

Una de las primeras placa microcontroladoras de bajo coste, y la que sentó las bases de las actuales, fue desarrollada por Arduino. La placa microcontroladora Arduino Uno está basada en el microcontrolador Atmega328P, con una frecuencia de trabajo de 16MHz. Cuenta con 19 pines de E/S, 6 de ellos analógicos, otros 6 con posibilidad de salida PWM, regulador de tensión integrado, memoria flash y memoria

EEPROM entre otras características; además permite comunicarse mediante puerto serie y mediante protocolo SPI. Se trata de un componente muy versátil que puede ser empleado en una gran variedad de aplicaciones.

4.4.1. Arduino Yún

El Arduino Yún es una evolución del Arduino Uno. El microcontrolador integrado es el ATmega32u8, empleando en paralelo el microprocesador Atheros AR9331, que incorpora el sistema operativo OpenWRT basado en Linux. En la ilustración 3 se encuentran señalados todos los subsistemas importantes del Arduino Yún, incluidos los dos núcleos de procesamiento.

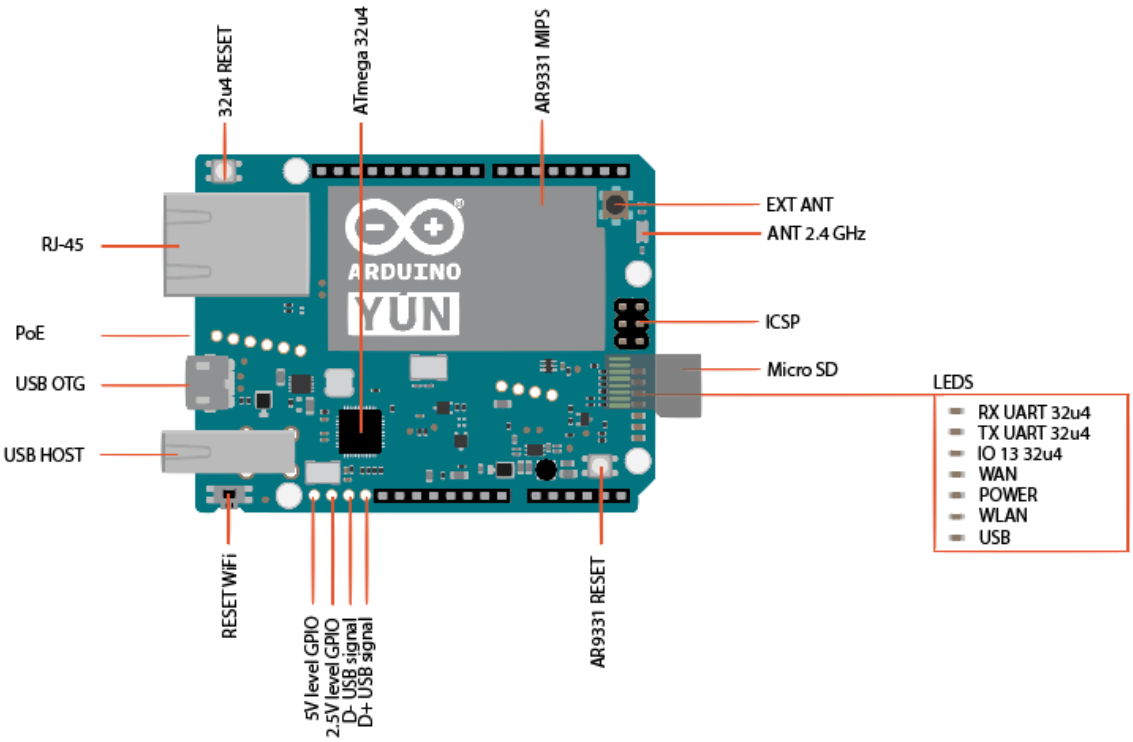

Ilustración 3. Partes destacables del Arduino Yún

El ATmega32U8 es un microcontrolador SoC con una arquitectura RISC de 8 bit basada en AVR, cuenta con 32 KB de memoria flash de programa, 2’5KB de memoria RAM estática, 1KB de memoria EEPROM, USB 2.0 integrado, 12 canales con conversor analógico-digital de 10 bits.... Su velocidad máxima de trabajo es de 16MHz, permitiendo ejecutar 16MIPS (Millones de instrucciones por segundo).

El Atheros AR9331 es un chip SoC diseñado para llevar a cabo tareas relacionadas con internet. Este microprocesador cuenta con interface WiFi *IEEE 802.11b/g/n*, e interface Ethernet *IEEE 802.3 10/100Mbit/s*. El Atheros AR9331 se emplea como complemento al Atmega32U4, por lo que la gran mayoría GPIOs no están accesibles desde el exterior de la placa. Sin embargo, cuenta con un puerto USB2.0 y un lector de tarjetas microSD. El microprocesador incorpora el sistema operativo OpenWrt, que es un firmware basado en Linux y diseñado originalmente para ser empleado en sistemas como los routers. A pesar de ello, debido a su reducido tamaño y la gran cantidad de paquetes existentes este sistema operativo ha ganado una gran expansión. Empleando los paquetes adecuados, OpenWrt puede emplearse para cualquier casi aplicación.

4.5. COMUNICACIONES

4.5.1. Modulación por ancho de pulso - PWM

La modulación por ancho de pulsos (pulse width modulation, PWM) es una técnica para transmitir información o controlar la energía entregada a una carga. Se basa en la modificación del ciclo de trabajo de una onda cuadrada para codificar información o controlar la energía entregada. Cuenta con dos parámetros básicos: la frecuencia y el ciclo de trabajo. Se pueden usar ambos para la codificación de información. En la ilustración 4 se muestra una onda PWM, en la parte inferior, y sus equivalente analógicas, en la parte superior.

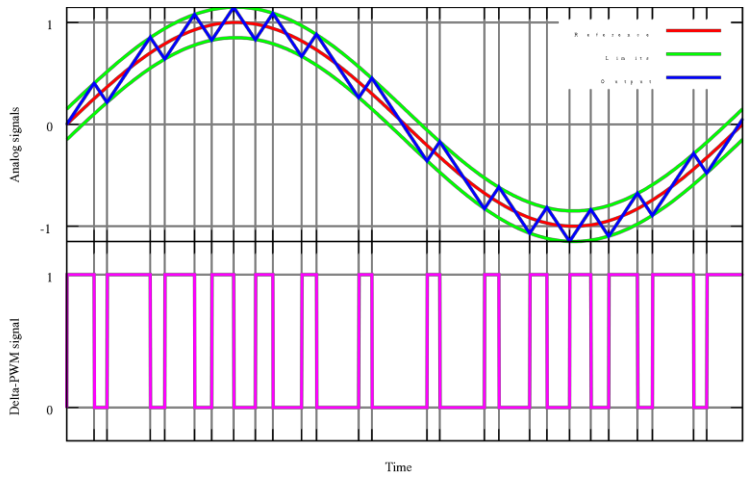

Ilustración 4. Onda PWM (abajo) y energía entregada (arriba)

Autor: **Javier Martínez Lahoz**
424.16.93

4.5.2. Bus SPI

El bus SPI es un protocolo de comunicaciones muy empleado en la transferencia de información entre sistemas electrónicos. Se trata de una comunicación tipo serie síncrona. El bus está compuesto por un componente maestro y diversos componentes esclavos. Su topología se muestra en la ilustración 5. El bus esta formado por tres líneas compartidas: una línea de reloj (SCLK), una línea del maestro a los esclavos (MOSI) y una línea de los esclavos al maestro (MISO). Además es necesario que exista una línea adicional a cada esclavo (SS), que selecciona al esclavo deseado para cada comunicación que vaya a realizarse. Los paquetes de información no están predefinidos ni en longitud ni en formato por el protocolo, lo que otorga una gran flexibilidad frente a otros estándares de comunicación más rígidos, como I2C o CANBus. La velocidad de transmisión está marcada por el maestro, por lo que tan solo está limitada por las capacidades de cada componente que intervenga en la transmisión.

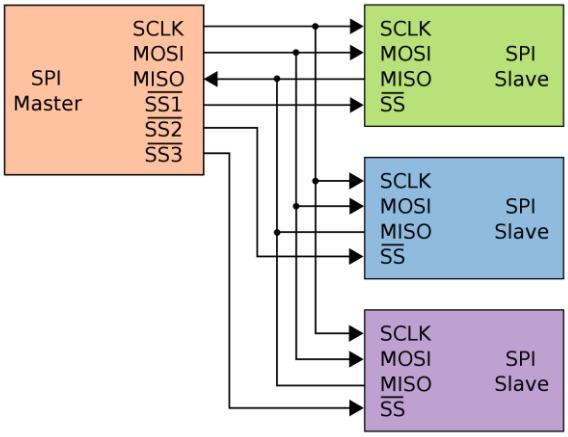

Ilustración 5. Distribución de conexiones en un bus SPI

4.5.3. CANBus

El protocolo CAN es un estándar de comunicaciones a nivel mundial, especialmente utilizado en industria y automoción. Una de sus principales características es el empleo de controladores de bus. Estos controladores se encargan de gestionar toda la comunicación de cara al bus, por lo que minimizan el tiempo que el procesador principal debe consumir gestionando comunicaciones. Algunos microprocesadores y microcontroladores incluyen un controlador de CANBus para facilitar la integración de este protocolo.

En la ilustración 6 se puede observar la topología de un bus CAN. El bus esta formado por un par diferencial con impedancias terminadoras de 120Ω en ambos extremos; todos los dispositivos conectados son derivaciones individuales de dicho bus. Mediante esta conexión se pueden lograr velocidades de transmisión de información de hasta 1Mbit/s. Cuanto mayor sea la longitud del bus, menor será la tasa de transferencia de datos, pues se podrían generar problemas de sincronización de bits entre ambos extremos del bus. Cada nodo del bus CAN debe incorporar un transceptor antes del controlador.

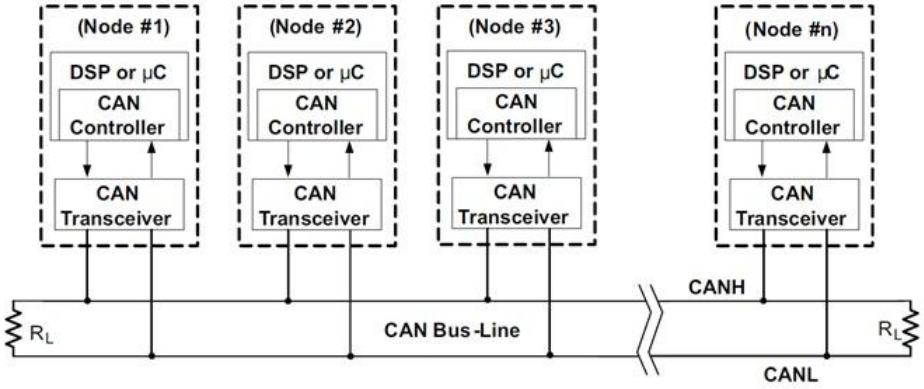

Ilustración 6. Distribución del Bus CAN

Las tensiones existentes en el bus dependen del estado en el que se encuentre, existen dos posibilidades: dominante y recesivo. En estado dominante, bit 0 siendo transmitido por el bus, existe una diferencia de tensión entre CAN-H y CAN-L de entre 1'5V y 3V; no se especifica la tensión respecto al plano de masa pues puede ser distinto. En estado recesivo, bit 1 siendo transmitido por el bus, las tensiones de CAN-H y CAN-L se encuentran en modo común, 0V entre ellos.

La trama inicial de todos los mensajes es similar. Se comienza con un bit dominante, bit 0, que marca el inicio de la transmisión; los siguientes 11 bits son el identificador de mensaje. Estos bits indican cuál es el objetivo del mensaje, de esta forma cada nodo puede discriminar qué mensajes le transmitirá a su procesador, y qué mensajes no. Además sirve para definir la prioridad en que los mensajes serán publicados en el bus. La capa de enlace de datos, definida en el estándar de CANBus, incorpora el arbitraje de mensajes, los bits de sincronismo, el acuse de recibo y otras características que hacen de CANBus un protocolo de comunicaciones robusto y fiable.

En la ilustración 7 se observa una trama de bits de un mensaje CANBus. En verde se resalta el identificador de mensaje. Los siguientes bits identifican algunos

Autor: **Javier Martínez Lahoz**
424.16.93

aspectos relevantes del mensaje: si es extendido o si es una petición remota. El campo resaltado en amarillo sirve para conocer la longitud de los datos incluidos en el mensajes; dichos datos se encuentran a continuación resaltados en rojo. Finalmente se localizan el CRC (Verificación por redundancia cíclica) y el ACK (acuse de recibo) para asegurar que el mensaje fue transmitido correctamente. Los bits resaltados en morado son bits de sincronismo, que se intercalan tras 5 bits del mismo valor, para asegurar que el sincronismo entre los diversos sistemas es el adecuado y no se producen problemas de desincronización.

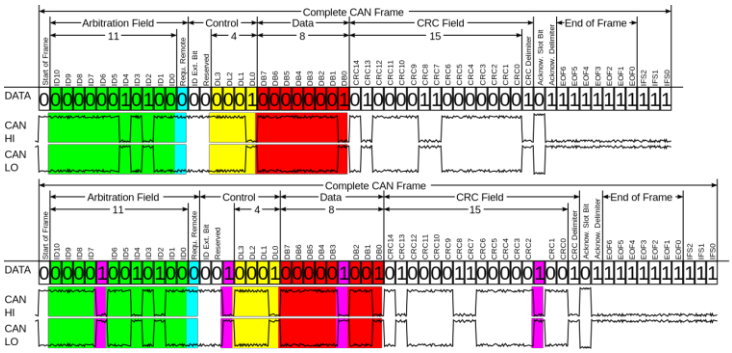

Ilustración 7. Trama de bits de un mensaje CANBus con el mensaje natural (arriba) y el mensaje con los bits de sincronismo en morado (abajo)

4.5.4. NMEA 0183

NMEA 0183 es un protocolo de comunicaciones diseñado específicamente para electrónica e instrumentación marítima. Las siglas NMEA provienen de *National Marine Electronics Association*. La comunicación se realiza mediante caracteres ASCII transmitidos sin codificar, de forma similar a la comunicación serie. El protocolo establece la sintaxis y la velocidad de los mensajes (300bits/s), para que puedan ser descodificados de forma sencilla por los sistemas receptores.

Como puede observarse en la ilustración 8, cada mensaje comienza con el símbolo del dólar como identificador de mensaje. Los siguientes cinco caracteres identifican al emisor del mensaje, empleando dos caracteres, y el tipo de mensaje, con tres caracteres. El nombramiento de emisor y el nombramiento del tipo de mensajes se encuentran estandarizados, para que cualquier sistema pueda conectarse a cualquier red NMEA0183. A continuación se encuentran los campos de datos del mensaje, separados entre sí por comas. Tras el último campo de datos se dispone un asterisco si se emplea CheckSum. El CheckSum es un número de dos dígitos hexadecimales que se emplea para detectar errores de transmisión. Dicho número se

logrará al realizar la operación lógica XOR entre todos los caracteres del mensaje. Como identificador de final de mensaje se empleará el carácter ASCII de retorno de carro seguido del carácter de cambio de línea.

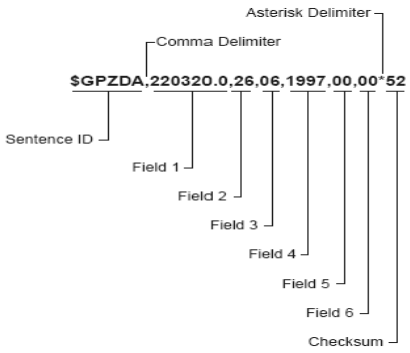

Ilustración 8. Ejemplo de mensaje en protocolo NMEA 0183

4.5.5. Transmission Control Protocol - TCP

Uno de los protocolos más comunes para comunicaciones en red es el TCP (Transmission Control Protocol). Se trata de un protocolo de comunicaciones que da soporte a multitud de aplicaciones de más alto nivel, si bien también puede usarse independientemente como método de transmisión de datos en bruto. El protocolo TCP garantiza que todos los paquetes de datos serán entregados sin errores y en el orden original. Este protocolo además permite diferenciar entre aplicaciones dentro del mismo sistema, empleando un identificador llamado puerto.

Una aplicación basada en TCP es SSH (Secure Shell). Se trata de un intérprete de órdenes cifrado, es decir, permite tomar el control de un sistema a través de la red. Gracias a esto podemos efectuar ordenes y redirigir información en una máquina remota, estando toda la comunicación y la información cifrada. La conexión SSH establece un canal seguro de tipo túnel entre la máquina emisora y la máquina receptora de la comunicación mediante un cifrado de la conexión TCP. El acceso a la máquina remota mediante este protocolo está limitada según el usuario con el que nos conectemos, en el caso de conectarnos con el usuario raíz (root) tendremos pleno acceso a la máquina. También podemos emplear aplicaciones TCP orientadas a la transmisión de ficheros, el FTP (File Transfer Protocol) y el SFTP (Secure File Transfer Protocol).

5. DESARROLLO

El primer paso para realizar el diseño de cualquier sistema consiste en un estudio de necesidades. Al comprender los requisitos que dicho sistema deberá satisfacer, así como otras necesidades no imprescindibles que puedes resultar útiles, podremos realizar un diseño adecuado. Ambos tipos de condicionantes se deducirán a partir de un estudio del sistema que se va a llevar a cabo; en este caso una motocicleta. También se podrá realizar un estudio de los sistemas empleados comercialmente o otros prototipos similares.

5.1. ESTUDIO DE NECESIDADES

Para poder definir las necesidades que deberemos satisfacer es necesario comprender el funcionamiento completo del prototipo sobre el que irá montado. El prototipo será una motocicleta eléctrica de competición. Eléctricamente consta de tres partes fundamentales: un acumulador, un regulador y un motor. Estas se encargan de almacenar, controlar y consumir la potencia eléctrica de la motocicleta. Estos dispositivos trabajan con tensiones y corrientes elevadas, por lo tanto nos referiremos a este conjunto como sistema de alta tensión (HVS). El acumulador será una batería de 130 celdas de litio-polímero, logrando una tensión de bus de 110V y una energía total de 5KWh. El regulador empleado será comercial, de la marca *Sevcon*, diseñado para esta clase de aplicaciones y recomendado para ser empleado junto con nuestro motor. El motor utiliza tecnología *Brushless* para su funcionamiento: cuenta con dos conjuntos síncronos de bobinas, cada uno de ellos de tres bobinas independientes y cuatro pares de polos por bobina. Para que el regulador y el motor funcionen correctamente sincronizados se requiere de una realimentación de posición angular. Esta realimentación es proporcionada por un sensor de posición anclado al eje del motor, el *encoder*. En nuestro caso este sensor incluye un sensor de temperatura interna del motor. Lógicamente para que el piloto pueda controlar la potencia entregada al motor, se cuenta con un puño acelerador electrónico. Con el objetivo de lograr una estandarización de planos y evitar diversas nomenclaturas, nos referiremos a todos ellos en inglés: *Battery*, *Controller*, *Motor*, *Encoder* y *Throttle* respectivamente; o por sus nombres comerciales.

Desarrollo

De forma paralela existen diversidad de elementos eléctricos y electrónicos encargados de proporcionar una mayor inteligencia, interactividad, control o seguridad al sistema de alta tensión. En nuestro caso, y tras un estudio detallado, los elementos que incluimos en el diseño son: el vigilante de aislamiento (*IMD*), el supervisor de la batería (*BMS-Master*), los circuitos de balanceo de celdas (*BMS-Slaves*), una fuente de alimentación que proporciona la tensión de alimentación de la electrónica, el *Display*, la maniobra de seguridad junto con los contactores y la unidad de control electrónico (*ECU*). Además en nuestro prototipo se incluirá un sistema innovador de detección de vehículos cercanos en pista para evitar accidentes, denominado *Driver Detection System*. Todos estos sistemas se encuentran representados en los planos 424.16.93.01 y 424.16.93.02.

Debemos tener en cuenta dos principios fundamentales desde el momento que comencemos a realizar los diseños. En primer lugar, el circuito de alta tensión del prototipo deben estar eléctricamente aislados del circuito de baja tensión, así como de cualquier parte accesible por un usuario. En segundo lugar debemos permitir que la información producida por cualquier subsistema (tanto interno como externo) pueda llegar a cualquier otro subsistema que pueda modificar su comportamiento debido a esa información. Estos principios hemos de cumplirlos para que nuestro prototipo sea seguro, eficiente e inteligente, tal y como pretendemos, y como se espera de una escuela de ingeniera. Se decide además que se empleará el protocolo CANBus para la transmisión de información por la motocicleta. Esto se debe a que es un protocolo que permite una rápida transmisión de grandes cantidades de información, es fácil de implementar, y está muy extendido tanto en el mundo del automovilismo como en la industria. El regulador escogido ya cuenta con este medio de comunicación, por lo que también ha influido en la toma de esta decisión.

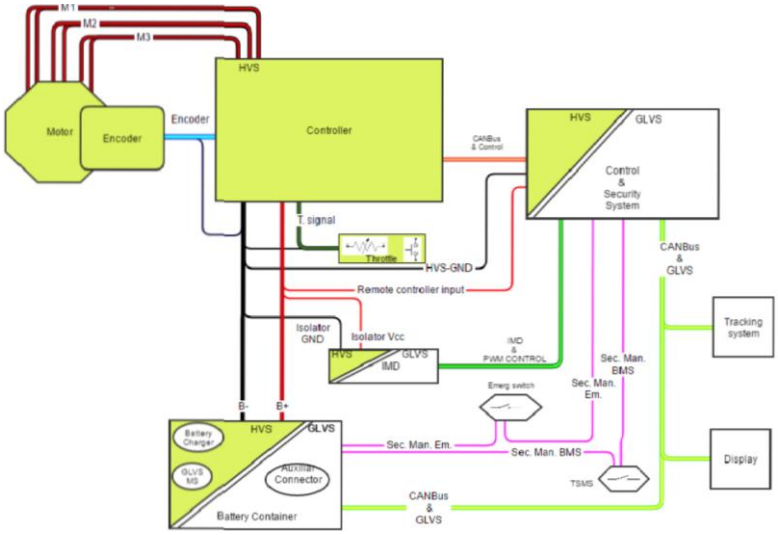

Ilustración 9. Esquema de conexionado de buses entre los sistemas de la motocicleta

La interconexión entre los diferentes sistemas se recoge en la ilustración 9. Hay que señalar los buses de potencia partiendo de la batería al regulador, y del regulador al motor. En color amarillo se resaltan los sistemas de la red de alta tensión. Podemos observar que la batería, el IMD y la ECU están conectados a ambas redes, por lo que será importante tener precaución con el aislamiento de estos sistemas.

5.1.1. Necesidades de la ECU

El diseño de la ECU debe tener en cuenta casi todos los sistemas de la motocicleta, pues de una forma u otra todos ellos afectarán o serán afectados por otros; la ECU deberá ser capaz de leer la mayor cantidad posible de información para poder procesarla y actuar de acuerdo a ella. Nuestro primer paso será determinar que subsistemas van a componer la ECU, para ello debemos determinar cómo se relacionan los sistemas y qué necesidades debe cubrir la ECU. Comenzamos analizando el esquema de alto nivel de conexionado entre sistemas de la motocicleta, en la ilustración 9.

De este diagrama podemos concluir que una de las tareas fundamentales de la ECU será lograr el aislamiento general del regulador. Esto se debe a que el regulador debe transmitir información, actuar sobre la maniobra de seguridad y ser alimentado. Todo ello está en estrecha relación con la red de baja tensión, sin embargo, debe permanecer aislado. Se decide encargar estas funciones a la ECU, con el fin de optimizar recursos y minimizar las placas electrónicas a incluir.

A continuación realizamos un estudio en profundidad del conexionado y del funcionamiento de los distintos subsistemas. De dicho estudio obtenemos abundante información relacionada con las funciones que deberá cumplir la ECU. Por lo tanto podemos comenzar a definir ya las restricciones de diseño que deberemos cumplir. Las funciones mínimas atribuidas a la ECU son las siguientes:

- Controlar el Key-Switch que activa regulador.
- Mantener comunicación CANBus con el regulador.
- Recibir la señal del regulador relacionada con la maniobra de seguridad.
- Leer el estado de aislamiento general.
- Recibir la señal del IMD relacionada con la maniobra de seguridad.
- Mantener comunicación con todos los subsistemas de baja tensión.

Existen además una serie de funciones adicionales que se deciden incorporar a la ECU para mejorar sus características, si bien no son necesarias para el correcto funcionamiento de la motocicleta. Estas características son:

- Monitorización del estado de la maniobra de seguridad
- Salvado de datos en tiempo real para su posterior análisis.
- Lectura de datos por el usuario en tiempo real.

5.2. DISEÑO

Una vez fijadas las funciones que deberá llevar a cabo la ECU, se puede comenzar a diseñar el sistema. En primer lugar se decidirán los subsistemas que coexistirán en la ECU, y posteriormente se analizará y concretará cada uno de ellos individualmente.

5.2.1. Establecimiento de subsistemas

Se deberán fijar los subsistemas y sus funciones, para posteriormente valorar las diferentes formas de implementación de cada uno de ellos de acuerdo a diversos aspectos. Estos aspectos pueden ser las dependencias, la seguridad, la complejidad, el espacio… Los subsistemas, junto con su función, que diferenciaremos dentro de la ECU son los siguientes:

- Key-Switch: Control del encendido del regulador.

- HV CANBus: Comunicación con el regulador.

- GLVS CANBus: Comunicación con la red de baja tensión.

- Security Maneuver: Control sobre una parte de la maniobra de seguridad.

- Power Manager: Alimentación todos los demás los subsistemas.

- ECU: Procesamiento de datos y toma de decisiones.

- Interface: Conectividad con el exterior.

Durante esta fase de análisis se propone una característica adicional de la ECU, el aislamiento eléctrico completo, tanto del circuito de alta tensión como del de baja. De esta forma, la robustez de la electrónica será mucho mayor y se minimizará la posibilidad de un fallo interno debido a una perturbación electromagnética en el exterior de la misma. Asimismo también se propone incluir un sistema de entradas y salidas auxiliares. El origen de esta decisión se debe a la naturaleza del proyecto: puesto que el prototipo completo se encuentra en fase de diseño, cabe la posibilidad de requerir alguna entrada de información o salida de control más; en consecuencia, y para prevenir futuros problemas, se preparan módulos de entradas y salidas auxiliares, tanto de alta como de baja tensión. Por lo tanto contamos con tres subsistemas nuevos:

- GLVS Measures: encargado de aislar las medidas tomadas del GLVS

- Auxiliar GLVS: E/S Auxiliares de la red de baja tensión

- Auxiliar HVS: E/S Auxiliares de la red de alta tensión

A continuación establecemos las dependencias entre subsistemas. Para ello realizamos un esquema de bloques, y marcamos con líneas las dependencias entre subsistemas. El esquema final se muestra en la ilustración 10. Podemos observar que la mayoría de dependencias son direccionales; un claro ejemplo es el subsistema de alimentación, Power Manager, que depende del medio externo, GLVS, y del que dependen todos los demás subsistemas, incluso por duplicado, pues el subsistema GLVS Measures debe ser alimentado, a la vez que recibe las señales.

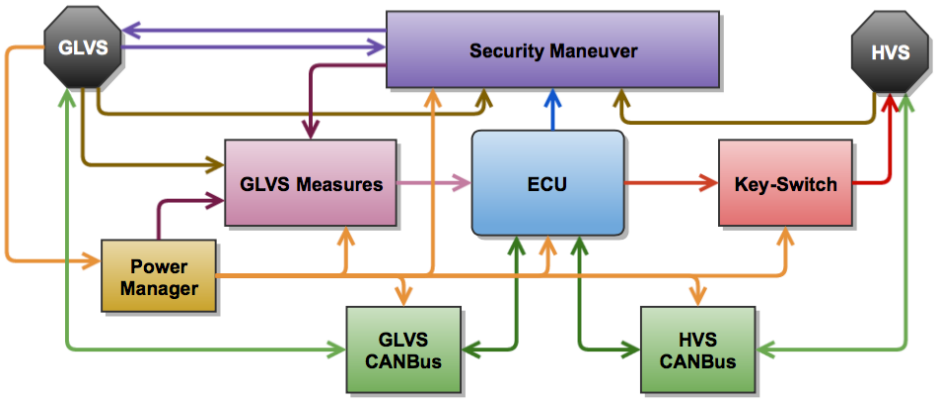

Ilustración 10. Esquema de bloques de la ECU

Podemos observar en la ilustración 10 que las únicas dependencias bidireccionales son las de los subsistemas de comunicaciones, el resto de dependencias son todas direccionales. La ECU recibe información exclusivamente del subsistema GLVS Measures y de las comunicaciones CANBus y actúa sobre la maniobra de seguridad, el Key-Switch y los canales de comunicaciones.

También debemos centrar la atención en la maniobra de seguridad, Security Maneuver, en la parte superior de la ilustración 10. Este subsistema recibe señales de la ECU y de tres orígenes distintas del exterior. Dos de estas señales provienen del sistema de baja tensión, una de ellas es la señal entrante de la maniobra de seguridad, que viene de la batería pasando por la seta de emergencia, mientras que la otra es la señal del relé del IMD. La tercera señal recibida proviene del regulador, y se emplea para controlar su relé en la maniobra. Este subsistema a su vez genera dependencias sobre dos elementos. La dependencia más importante es con el exterior, pues la maniobra de seguridad debe alcanzar la batería, una vez pasado el TSMS, para que la alta tensión pueda ser activada. También depende el subsistema de mediciones, GLVS Measures, de la maniobra de seguridad, pues se realizarán medidas en distintos puntos del subsistema para conocer su estado exacto en todo momento.

5.2.2. Diseño de subsistemas

Para poder realizar un diseño satisfactorio y eficiente nuestro siguiente objetivo será realizar un análisis individual de cada subsistema: función exacta, diferentes posibilidades de implementación, dependencias… Cada subsistema deberá ser definido completamente en este apartado.

Autor: **Javier Martínez Lahoz**
424.16.93

5.2.2.1. ECU

Es la unidad de control encargada de administrar toda la información de la moto y efectuar la toma de decisiones consecuente. Deberá tener una potencia de procesamiento lo suficientemente alta para poder asimilar el volumen de información recibida, así como su procesamiento. A la vez interesa tener una buena conectividad de cara al exterior, puesto que al tratarse de un prototipo debemos ser capaces realizar modificaciones de funcionamiento rápidamente.

Es el elemento principal de todo el sistema, y de su elección depende el resto de subsistemas del sistema. Se valorará por lo tanto a la hora de su elección la potencia de procesamiento, la conectividad, la facilidad de reprogramación y la adaptabilidad De cara al resto de subsistemas, la ECU debe contar con una importante cantidad de pines de entrada/salida, pines de interrupción y un canal de comunicación, como por ejemplo SPI. Existen dos principales tecnologías para nuestro núcleo principal: un microcontrolador y un microprocesador.

Un microcontrolador es más fácil de programar, se encuentra completamente integrado, y facilita la comunicación con el exterior. Las órdenes de programa en un microcontrolador se ejecutan directamente contra el núcleo de procesamiento, lo que aumenta la estabilidad del sistema, siempre y cuando la programación sea correcta.

Como alternativa a los microcontroladores se encuentran los microprocesadores. Ofrecen una mayor capacidad de procesamiento, así como funciones más complejas, como pueden ser el multiprocesado o la conexión a internet. Sin embargo, el coste en desarrollo de la aplicación es más mayor, su precio más elevado y pueden sufrir mayores problemas que los microcontroladores, ya que funcionan con un sistema operativo.

Finalmente, se llega a la conclusión de que la opción que más se adapta a nuestras necesidades pasa por la combinación de ambas tecnologías. Esta decisión se basa fundamentalmente en una característica: la adaptabilidad. Si nos decantamos por microcontrolador o microprocesador, perdemos las posibilidades que el otro nos podría dar. Se decide emplear una placa de desarrollo comercial híbrida, que cuente con un microprocesador y un microcontrolador. De esta forma los componentes de acondicionamiento necesarios ya se encuentran integrados. Concretamente se empleará como elemento central de nuestro sistema el Arduino Yún. El Arduino Yún integra un microcontrolador ATmega32u4 con un microprocesador Atheros AR9331.

Los esquemas internos del Arduino Yún se muestran en el anexo 3: Esquemas Arduino Yún.

Al integrar el microcontrolador y el microprocesador obtenemos una gran cantidad de recursos. De esta forma podemos, por ejemplo, emplear el microcontrolador para tareas de control básico de bajo nivel, mientras el microprocesador se encarga de las tareas de interconectividad de alto nivel.

Se emplearán clavijeros rectos para conectar el Arduino Yún a la PCB. Para asegurar la conexión entre el Arduino y la PCB se realizarán orificios en la placa para realizar una unión mediante tornillos. Se emplearán separadores de nylon o fibra de vidrio de forma que protejan la placa de cortocircuitos producidos por la cabeza del tornillo o la tuerca.

Adicionalmente se ha decidido colocar un indicador luminoso controlador por el Arduino. De esta manera se podrá indicar el estado general del sistema directamente al usuario. Hay que proteger dicho LED mediante una resistencia colocada en serie con el diodo, esta resistencia limitará la corriente a la nominal del LED para su correcto funcionamiento.

La elección del diodo LED que emplearemos apenas cuenta con restricciones. En primer lugar se decide el encapsulado, para minimizar el espacio necesario se ha decidido emplear un encapsulado SMD. De entre todas las posibilidades que encontramos, escogemos de acuerdo al color deseado y a dos parámetros eléctricos: la caída de tensión y la corriente necesaria. La combinación de estos valores determinará la resistencia de limitación que debemos emplear y el consumo del conjunto.

Se acuerda que el LED empleado será de color azul, para destacar frente al resto de posibles indicadores que empleemos. Tras una pequeña búsqueda se decide emplear el LED LNJ926W8CRA de Panasonic. Principalmente se ha buscado un componente cuyo consumo sea reducido, así como su tamaño. Según el Datasheet proporcionado por el fabricante (Ref: *Panasonic. (2008). Light Emitting Diode LNJ926W8CRA*), este LED genera una caída de tensión de 2'9V y su corriente nominal es de 5mA, aunque su corriente máxima es de hasta 20mA. El encapsulado que emplea este componente es el SMD 0603. Su tamaño es bastante reducido, pero no debería causar grandes problemas cuando nos dispongamos a soldarlo.

Autor: **Javier Martínez Lahoz**
424.16.93

Empleamos la ecuación 1, derivada de las leyes de Kirchhoff y Ohm, para calcular el valor de la resistencia de limitación en función de la caída de tensión del LED (2'9V), la tensión de trabajo (5V) y la corriente recomendada por el fabricante (5mA). Todos los datos se introducirán en unidades del sistema internacional (Voltios, Amperios y Ohmnios).

Ecuación 1

$$R = \frac{V \quad V_{LED}}{I_{LED}} = \frac{5V \quad 2'9V}{5mA} = 420$$

Una vez calculado el valor de la resistencia, 420Ohm, debemos escoger un modelo concreto de entre todos los que existen en el mercado. La primera restricción que debemos realizar es, lógicamente, la de valor óhmico; también se realiza una restricción respecto al tamaño, y así se opta por emplear el encapsulado 0805 [Métrica 2012]. Se trata de un encapsulado SMD de tamaño reducido, pero que permite la soldadura manual sin dificultades. Suele ser necesario tener en cuenta la potencia que disipará la resistencia. En este caso al darse corrientes tan reducidas no debería suponer un problema, aun así se calcula dicha potencia. Para ello emplearemos la fórmula de potencia eléctrica. Al introducir en esta fórmula la resistencia en ohmnios y la corriente en amperios obtenemos la potencia disipada en watios. La ecuación 2 desarrolla esta fórmula para calcular la potencia disipada en la resistencia.

Ecuación 2

$$P = R \cdot I^2 = 420 \quad \cdot (5mA)^2 = 10'5mW$$

Si se trabaja con con tensiones elevadas, también deberá tenerse en cuenta la tensión máxima que permite la resistencia. Como en este caso la tensión es de 5V no existirá ningún problema, pues todas las resistencias que valoramos aguantan más de esta tensión.

Como criterio final se empleará el precio del componente, ya que se va a montar el prototipo y no se dispone de un presupuesto muy elevado, por lo que se optará por componentes más económicos, siempre que las características eléctricas sean similares. Se combinan todos estos criterios en la búsqueda, y finalmente se decide emplear la resistencia MC01W08051422R fabricada por la empresa Multicomp. Este modelo cumple con todas nuestras necesidades, y además, es capaz de disipar hasta

100mW según sus datos técnicos (Ref: Multicomp. (2013). Thick Film Chip Resistor 0805), más de lo necesario.

Finalmente se incorporará un botón de reinicio forzado de la ECU, para casos de emergencia o pruebas. Este botón actuará directamente contra el pin de reinicio del Arduino. El funcionamiento de este pin es el siguiente: cuando el pin se conecta al plano de referencia del Arduino, este se bloquea; cuando el pin se libera a 5V o a un estado una alta impedancia, pin al aire, el Arduino comienza su funcionamiento. El circuito empleado consta tan solo un botón con una pata conectada al pin de Arduino y otra pata conectada a masa. Lógicamente hemos de asegurarnos que esas patas estén desconectadas cuando el botón este liberado, y se conecten al pulsarlo. Este circuito apenas maneja corriente, y la tensión máxima será de 5V, por lo que no resultan restricciones reales a la hora de escoger el componente. El único requisito que debe cumplir el botón, aparte que se enclave, está relacionado con el tamaño. Se opta por emplear un botón de pequeñas dimensiones, ya que no es esperable que tenga que ser pulsado en muchas ocasiones, de manera que ocupe poco espacio y no moleste al resto de componentes. El botón que se escoge está fabricado por la empresa Sparkfun, y está pensado para trabajar con Arduino, concretamente se escoge el modelo COM-00097, ya que cuenta con unas dimensiones muy reducidas.

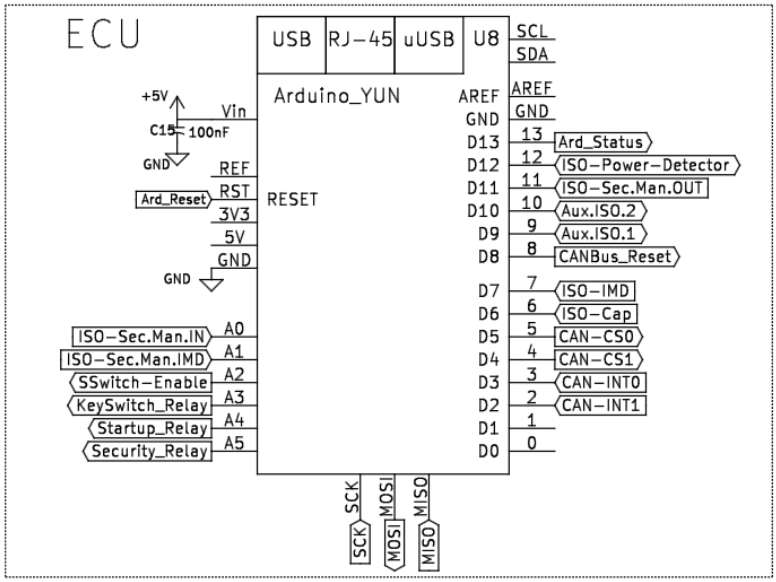

Ilustración 11. Representación electrónica del Arduino y su conexionado

En la ilustración 11 se muestra la representación del Arduino dentro del esquema electrónico general. Podemos observar que todos los pines se encuentran conectados a otros subsistemas, que veremos más adelante. Las excepciones son los pines D0 y

D1, que están conectados al bus serie interno y no deben ser conectados externamente; y los pines SDA y SCL, ya que no emplearemos ninguna conexión I2C.

5.2.2.2. CANBus Aislado

Ambos subsistemas de comunicaciones será similares, pues servirán para mantener una comunicación vía CANBus aislada con el exterior, si bien cada uno con su respectivo bus. El objetivo es implementar dos redes de CANBus de alta velocidad, el regulador emplea inicialmente una velocidad de transmisión de 500kbits/s, de manera que estandarizamos dicha velocidad para ambos buses.

Algunos microcontroladores y microprocesadores integran un controlador de CANBus. El Arduino Yún no integra dicho controlador, así que es necesario emplear un controlador externo. Dicho controlador debe poder comunicarse con la ECU, por lo que debe admitir uno de los protocolos de comunicación que integra el Arduino Yún. Los protocolos de información que podemos emplear son I2C y SPI.

	I2C	SPI
Nº Cables	2	3 + n (nº slaves)
Multimaestro	Soportado	No soportado
Selección de receptor	Mediante Bus	Mediante CS
Transmisión y recepción	Serializados (Half Duplex)	Paralelizados (Full Duplex)
Velocidad de transmisión	Reducida (frente a SPI)	Elevada (frente a I2C)

Tabla 3. Comparativa entre los protocolos I2C y SPI

En la tabla 3 podemos observar que el protocolo I2C emplea un menor número de cables, principalmente porque la selección del receptor se realiza mediante la cabecera de los mensajes. Además, al solo emplear un cable para toda la transmisión de datos, se reduce a la mitad el volumen máximo de información para la misma velocidad de reloj; Half Duplex frente a Full Duplex. Todo esto hace el bus I2C mucho más adaptable, pero considerablemente más lento.

Nuestro sistema incorporará un doble canal de transmisión a 500kbits/s y además no se prevé la inclusión de más subsistemas al bus interno, en consecuencia se decide primar la velocidad de transmisión frente a la adaptabilidad, por lo que emplearemos el bus SPI. El esquema de conexionado de los dos controladores CANBus, con sus respectivos transceptores, se muestra en la ilustración 12. En dicho esquema se han incluido los pines de los vectores de interrupción de ambos canales.

Dichos pines no son necesarios para la comunicación SPI, pero mejoran la velocidad de transmisión, pues el controlador puede avisar a la CPU de que ha recibido un mensaje. De esta forma se logra una lectura inmediata de los mensajes entrantes.

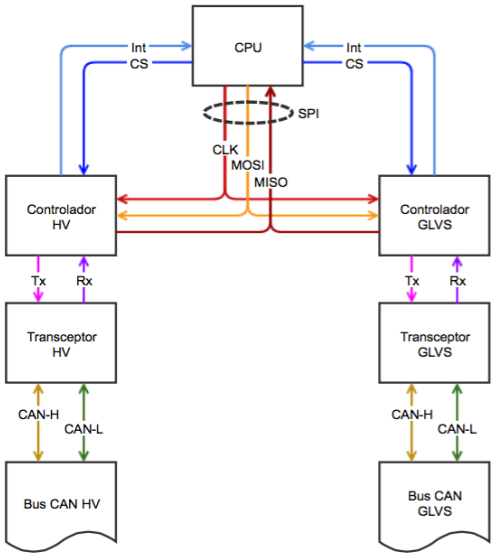

Ilustración 12. Esquema de conexionado de dos canales CANBus vía SPI

El controlador a emplear deberá por lo tanto admitir comunicación SPI y CANBus de hasta 500kbits/s. Se realiza una búsqueda de los controladores existentes en el mercado que cumplan con estas expectativas, además se valorará la simplicidad del circuito de acondicionamiento, la cantidad de buffers y la cantidad de máscaras y filtros. Esto se debe a que se pretende simplificar el esquema electrónico, al tiempo que se prima buscar una gran adaptabilidad y robustez software.

Finalmente el controlador escogido es el MCP2515 de Microchip. Dicho controlador ha sido empleado en *shields* de Arduino para comunicaciones CANBus, por lo que se supone una buena compatibilidad con Arduino. Cumple con todas las necesidades del sistema, como podemos observar en el Datasheet del componente (Ref: Microchip. (2012). MCP2515 Stand-Alone CAN Controller with SPI Interface), este controlador permite comunicación SPI de hasta 10MHz y comunicación CANBus V2.0B hasta 1Mb/s. También destacan la implementación de seis filtros junto con dos máscaras, de 26 bits todos ellos, y dos buffers internos, con prioridad de mensajes. Adicionalmente sobresalen su bajo consumo, de 5mA típicos, y su amplio rango de temperaturas, de -40ºC a +85ºC (ampliable a 125ºC en rango extendido).

El transceptor, además de las funciones normales, será el que proporcione el aislamiento galvánico al sistema. Las características del transceptor más relevantes a

la hora de escogerlo son la velocidad de transmisión y el nivel de aislamiento que proporciona.

Se opta por emplear como transceptor el ISO1050DUB de Texas Instruments, que proporciona 2'5KV de aislamiento capacitivo y permite comunicaciones CANBus de hasta 1Mbit/s según su hoja técnica (Ref: Texas Instruments. (2015). ISO1050 Isolated CAN Transceiver). Además proporciona robustez al estar protegido frente a tensiones de bus de entre -27V y 40V. De este transceptor destaca su implementación, pues no requiere de ningún circuito de acondicionamiento externo, más allá de a alimentación en ambos lados del integrado.

Una vez escogidos el controlador y el transceptor es necesario acondicionar y proteger el sistema. Para comenzar se realizar la búsqueda de una fuente de alimentación aislada al transceptor. Usaremos una fuente conmutada con entrada y salida a 5V, de esta forma mantendremos en todo momento el aislamiento entre ambas mitades del transceptor. Consideramos que esta opción resulta más eficiente y más segura que obtener los 5V de alimentación a partir de la alimentación de alta tensión, de 110V. Emplearemos la fuente conmutada TMA0505S de Tracopower que proporciona hasta 1W de potencia, con una eficiencia del 71% según el Datasheet (Ref: Traco Power. (2013). DC/DC Converters TMA Series, 1 Watt). Además proporciona un aislamiento de hasta 1KV con un rango de temperaturas de trabajo de entre -40ºC y 85ºC. Típicamente genera un ruido de 100mV de amplitud a una frecuencia de 20MHz debido a la conmutación, para compensar este ruido emplearemos un condensador de desacoplo de mayor capacidad a la entrada de alimentación aislada del ISO1050. Colocaremos a la entrada de la fuente de alimentación aislada un condensador para mejorar el comportamiento del sistema. El condensador recomendado por el fabricante en el Datasheet es de una capacidad de 4’7uF.

Existen una gran cantidad de condensadores en el mercado. Para escoger un modelo en concreto fijamos determinados criterios de búsqueda. En primer lugar es necesario que su tensión nominal de trabajo sea superior a la tensión a la que va a trabajar, 5 voltios. También realizamos restricciones de tamaño y encapsulado, escogiendo el encapsulado SMD 0805, para estandarizar con las resistencias. Finalmente realizamos un filtrado de acuerdo a tecnología de fabricación. Escogemos condensadores cerámicos frente a condensadores de tantalio, esto se debe a que optamos por no usar condensadores polarizados, eliminando un posible fallo de

montaje por colocarlos al revés. La decisión del componente concreto es arbitraria, optando por un modelo económico. Se ha optado por emplear el modelo 0805X475K6R3CT de la marca Walsin (Ref: Walsin Technology Corporation. (2015). MULTILAYER CERAMIC CAPACITORS General Purpose Series).

También es recomendable colocar un condensador a la salida de la fuente de alimentación. Este condensador reducirá el ruido entrante al ISO1050 debido a la conmutación de la fuente. El fabricante recomienda emplear un condensador de 2'2uF en el apartado *Input Source Impedance* de la página 73 del *Application Note: TMA Series* (Ref: Traco Power. (s.f.). Application Note TMA Series). Para mantener la estandarización con los condensadores empleados anteriormente elegimos un modelo de la misma familia que el anterior, el modelo 0805F225Z160CT de Walsin, cuya capacidad es 2'2uF, sus datos específicos se encuentran en el Datahseet de la familia (Ref: Walsin Technology Corporation. (2015). MULTILAYER CERAMIC CAPACITORS General Purpose Series).

El controlador requiere de una señal constante de reloj para poder funcionar correctamente. Esta señal se puede obtener a partir de otro subsistema que ya cuente con reloj interno, que implica una dependencia, o a partir de un circuito oscilador. Por lo tanto decidimos incluir un circuito oscilador en cada subsistema para que cumpla con esa función. A fin de que el circuito funcione correctamente será necesario emplear un cristal de cuarzo y dos condensadores. La principal característica que en la que nos fijamos a la hora de escoger el oscilador es su velocidad. La velocidad máxima de trabajo del MC2515 es de 25MHz según su Datasheet, por lo tanto buscaremos un cristal de cuarzo de una frecuencia igual o inferior. El Arduino Yún trabaja a 16MHz, así que para mantener una estandarización optamos por emplear un circuito oscilador de esta frecuencia también en el MCP2515. El cristal de cuarzo que decidimos emplear es el modelo 9C-16.000MAAJ-T de la marca TXC. La desviación máxima es de 30ppm mientras se trabaje entre -20ºC y 70ºC, (Ref: TXC. (s.f.). Quartz Crystals SMD HC-49S 9C SERIES). Para que el oscilador cumpla correctamente con su cometido es necesario incluir a su vez dos condensadores, entre sus terminales y la masa de referencia. El Datasheet no especifica concretamente el valor de dichos condensadores para un cristal de cuarzo de 16MHz; para un cristal de cuarzo de 8MHz se emplean condensadores de 22pF, para un cristal de cuarzo de 20MHz se emplean condensadores de 15pF y para un resonador cerámico de 16MHz se emplean condensadores de 22pF. Decidimos emplear condensadores de 22pF, y en caso de que

no funcionen de forma adecuada se probaran condensadores de menor capacidad. Se decide mantener la estandarización de tamaño y fabricante en todos los condensadores, por lo que emplearemos el modelo 0805N220J101CT de Walsin (Ref: Walsin Technology Corporation. (2015). MULTILAYER CERAMIC CAPACITORS General Purpose Series).

Para dotar al sistema de mayor robustez decidimos proteger los canales de comunicación frente a posibles señales transitorias de alta tensión. Para ello emplearemos diodos supresores de transitorios en el lado exterior del transceptor. Esta medida de seguridad viene recomendada en el Datasheet del ISO1050, *Figure 27 Application Circuit*. Estos diodos actuarían como toma de tierra en el hipotético caso de que el voltaje de los canales de comunicación se disparase. Colocaremos uno de estos diodos en CAN-H y otro en CAN-L. Entre los diversos diodos de transitorios del mercado encontramos uno que destaca. Se trata de un componente diseñado para actuar como diodo supresor de transitorios en canales de comunicación diferencial, como CANBus. Integra dos diodos bidireccionales supresores de transitorios en un único encapsulado. Este componente es el NUP2105 de ON Semiconductor, en concreto emplearemos el modelo SMD de encapsulado SOT-23, cuya referencia completa es NUP2105LT1G. Esta diseñado para trabajar con comunicaciones de automoción tipo CANBus (On Semiconductor. (2015). NUP2105L, SZNUP2105L, Dual Line CAN Bus Protector).

Además se considera oportuno que cada canal cuente con la opción de reinicio hardware, para ello la ECU contará con un pin para reiniciar ambos canales simultáneamente. Para limitar la corriente en esta línea, pues se trata tan solo de una señal de control, se empleará una resistencia de limitación. Debido a la baja corriente que circulará por el circuito, así como a la baja potencia, no resulta necesario realizar la búsqueda de la resistencia en función de su capacidad de disipación. Según el Datasheet del fabricante esta resistencia debe oscilar entre 1K y 10K. Decidimos emplear una resistencia de 10K, ya que minimiza el gasto del circuito. También se pretende mantener una estandarización similar a la que se mantiene en los condensadores, por lo que se empleará la resistencia MCWR08X1002FTL de la marca Multicomp, cuyos datos concretos se encuentran el Datasheet general de la serie (Ref: Multicomp. (2013). Thick Film Chip Resistor 0805).

Finalmente se implementa la resistencia de fin de línea entre CAN-H y CAN-L de 120Ohm, como indica el estándar CANBus de alta velocidad. Sin embargo no

deseamos que esta resistencia se encuentre en todo momento conectada, queremos que pueda ser activada o desactivada a voluntad. Esto se realiza porque el estándar CANBus expresa que deben existir dos resistencias terminadoras en el bus, una en cada extremo, pero no en los nodos intermedios. Por lo tanto activaremos dicha resistencia si la ECU es un nodo final del bus CAN, y no lo activaremos si es un nodo intermedio. La resistencia empleada es la MC01W08055120R de la misma familia que las resistencias anteriormente empleadas, (Ref: Multicomp. (2013). Thick Film Chip Resistor 0805).

La activación o desactivación de estas resistencias se llevara a cabo mediante un jumper de dos posiciones y un clavijero de dos posiciones. Este clavijero estará conectado en un extremo a la línea CAN-L de CANBus y, por el otro a la resistencia terminadora. El otro extremo de la resistencia terminadora estará conectada a la línea CAN-H de CANBus. Todo ello se muestra en la ilustración 13. Si colocamos el jumper, ambas líneas de CANBus quedan unidas a través de la resistencia de fin de línea. En caso de no conectar el jumper, la resistencia queda en circuito abierto. Para la implementación se empleará un clavijero de dos posiciones, en concreto el modelo 2211S-02G de Multicomp (Ref: Multicomp. (2012). 2211S - MC34 Series).

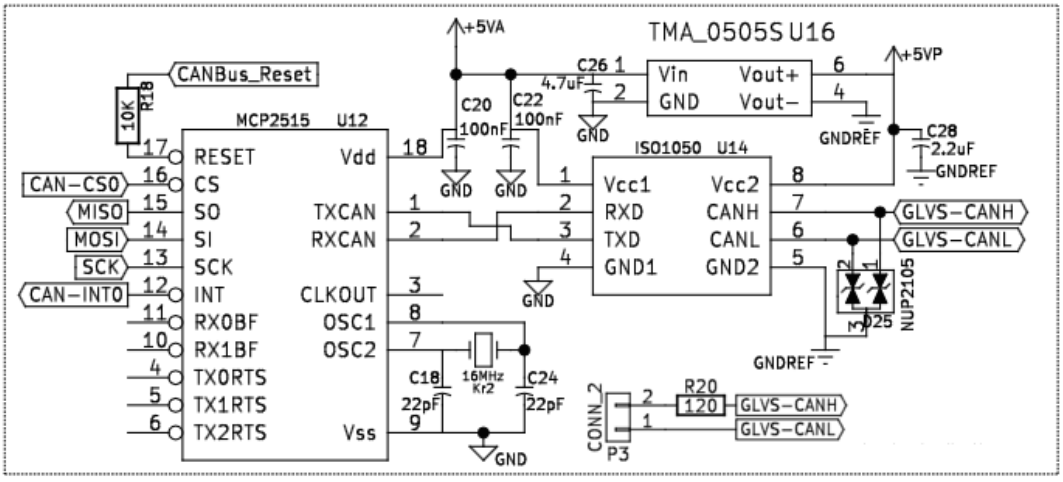

Ilustración 13 Conexionado del canal de comunicaciones CAN GLVS

En la ilustración 13 que representa el conexionado electrónico del subsistema de comunicaciones CANBus podemos destacar los tres componentes principales: el controlador a la izquierda, el transceptor a la derecha y la fuente de alimentación encima del transceptor. También quedan claramente representados el circuito oscilador, abajo a la derecha del controlador, y la resistencia de fin de línea con su clavijero, abajo a la derecha.

5.2.2.3. Key-Switch

Este subsistema es el encargado de controlar el encendido y apagado del regulador. El elemento central será un relé, por motivos de seguridad se ha decidido emplear un relé electromecánico frente a los relés de estado sólido para cumplir este cometido. Sin embargo la ECU por sí sola no es capaz de activar un relé de las características adecuadas, por lo tanto se decide incluir como etapa de potencia un transistor. Se plantean diversas alternativas de etapa de potencia, algunas con una mayor simplicidad, mientras que otras dotan de una mayor seguridad. La solución empleada será un circuito simple de activación de relé mediante etapa de potencia por transistor. Se ha optado por una solución simple para minimizar la posibilidad de errores de soldadura y conexionado en el montaje final. Por lo tanto los dos componentes base de este subsistema son el relé y el transistor.

El relé empleado en el Key-Switch debe tener una tensión de control de 5V, independientemente de la corriente, gracias al transistor empleado como etapa de potencia. Aun así interesa que el consumo sea el mínimo, para aumentar la eficiencia energética del subsistema. También hay que tener en cuenta que la corriente de contacto en ningún caso deberá estar por debajo de tres amperios, pues es lo que podría llegar a consumir el regulador, según especifica el fabricante. A su vez habrá que controlar la tensión de contacto admisible, de 110V al menos, para que no se produzcan arcos eléctricos ni desgastes indebidos del componente.

El relé que hemos encontrado que cumple con todas las necesidades es el ST1-DC5V-F de Panasonic. Se trata de un relé que permite conmutaciones de hasta 250Vcc y 8A. El consumo de la bobina es de 240mW; lo que implica 47mA a 5V. Además proporciona una tensión de aislamiento entre el contacto y el control de 3750V. Esta información y más detalles se puede encontrar en su Datasheet (Ref: Panasonic. (2012). ST RELAYS, 1a1b/2a 8A polarized power relays)

El transistor que emplearemos debe cumplir con las especificaciones del relé: 5V y 50mA. Por seguridad realizamos una búsqueda de transistores de hasta 100mA a más de 10V. La ganancia de dicho transistor deberá ser tal que permita la activación del relé con la señal de control del Arduino. Si queremos optar a generar corrientes de hasta 100mA con una señal pequeña, inferior a 1mA para minimizar el trabajo del Arduino, necesitamos un transistor con una ganancia de 100 o mayor. El transistor que se empleará es un transistor de uso general de la marca NXP, el BC847. Cuyos datos, según el fabricante (Ref: NXP Semiconductors N.V. (2014). BC847 series),

cubren perfectamente nuestras necesidades: corriente de colector de hasta 100mA, tensión de trabajo hasta 45V y ganancia 200. Además funciona correctamente hasta los 150ºC, otorgando una gran fiabilidad.

Para completar el circuito de activación del relé necesitaremos una resistencia y un diodo. La resistencia se colocará entre el pin de control y la base del transistor, para que actúe de resistencia de polarización en el transistor y límite la cantidad de corriente entrante. De esta forma controlamos la corriente de salida del transistor, evitando que supere los límites admisibles del mismo. Si conocemos la corriente base-emisor del transistor y su ganancia, podemos conocer la corriente colector-emisor necesaria mediante la siguiente fórmula:

Ecuación 3

$$I_{CE} = I_{BE} \times$$

Paralelamente podemos relacionar mediante la Ley de Ohm la corriente base-emisor y la resistencia de base conociendo la tensión base-emisor.

Ecuación 4

$$R_B = \frac{V_{BE}}{I_{BE}}$$

Combinando las ecuaciones 3 y 4, y empleando los valores adecuados: corriente de disparo del relé (Ice - 50mA), ganancia del transistor (β - 200), y tensión de trabajo (Vbe - 5V), podemos calcular la resistencia que debemos emplear. Estas fórmulas determinan que la resistencia que deberemos implementar será de 10kOhm. Usaremos por lo tanto la misma resistencia que se empleó para el reset de los controladores de CANBus, apartado 5.2.2.2, el modelo MCWR08X1002FTL de Multicomp, (Ref: Multicomp. (2013). Thick Film Chip Resistor 0805).

El diodo lo utilizaremos como protección del relé. Los relés electromecánicos (convencionales) generan una importante fuerza contraelectromotriz en su desactivación; si no los protegemos, esta fuerza puede generar picos de tensión que deterioren la electrónica y el relé hasta hacerlos fallar. Para evitar este fenómeno, colocamos un diodo en contraparalelo de la bobina del relé. De esta forma la contraelectromotriz de la bobina circula por dicho diodo y nuevamente por la bobina del relé hasta que desaparece, evitando el deterioro del componente. Para decidir el

Autor: **Javier Martínez Lahoz**
424.16.93

diodo se realiza una búsqueda de acuerdo a tres de las principales características que definen un diodo: voltaje, corriente nominal, corriente instantánea y caída de tensión. Se pretende además que dicho diodo pueda ser empleado en otros subsistemas, por lo que se busca un diodo con unas características más permisivas.

En primer lugar debemos escoger un diodo que permita trabajar en bloqueo con voltajes superiores a 24V, pues será la tensión máxima de trabajo, exceptuando los subsistemas de alta tensión. Por motivos de seguridad fijamos que el voltaje mínimo admisible será de 30V.

El segundo paso es definir la corriente que el diodo será capaz de conducir trabajando a favor de la tensión. Hemos estimado que la corriente admisible estará en torno a 1A. Se trata de un valor estándar que permitirá su uso en cualquier subsistema de la placa. La conmutación de los relés puede provocar picos instantáneos de corriente superiores a 1A, por lo que debemos tener en cuenta también la corriente instantánea soportada por el diodo; consideramos que dicha corriente debe ser superior a 10A por seguridad.

Finalmente debemos tener en cuenta la caída de tensión producida en el diodo cuando trabaja a favor de la tensión. Esta característica resulta especialmente relevante en el subsistema Power Manager, como explicaremos más adelante en el apartado 5.2.2.6. Cuanto menor sea la caída de tensión mejor se comportará el sistema y menos perdidas por calor aparecerán. Establecemos que dicha caída de tensión debe ser inferior a 1V para no afectar al comportamiento de los subsistemas.

El diodo que hemos escogido, de entre aquellos que cumplen con todas nuestras necesidades, es el SBR1A40S1 de Diodes INC. Este diodo podrá emplearse en la mayoría de subsistemas sin problemas debido a las características según fabricante (Ref: Diodes Incorporated. (2015). SBR1A40S1, 1A SBR® SUPER BARRIER RECTIFIER, DS33306). Sus características principales son su tensión inversa de rotura, de al menos 40V, y su corriente nominal de 1A. La caída de tensión producida es de aproximadamente 0'5V. Concluimos con el empaquetado del mismo, SOD123, que mantiene un tamaño similar al empleado en condensadores y resistencias, y que nos permitirá soldarlo manualmente sin problemas.

Para dotar de cierta seguridad de cara al usuario se opta por incorporar un indicador luminoso LED de alta tensión. De esta forma, si la alta tensión circula por la placa, tendremos un indicador para evitar accidentes. La caída de tensión del LED en

este caso resulta poco relevante, pues se alimentará a 110V. Se mantendrá la estandarización en los diodos LED al emplear modelos de la misma familia, en este caso el componente escogido es el LNJ437W84RA de Panasonic: con una caída de tensión de 2V y un consumo de entre 1mA y 20mA, en función de la luminosidad (Ref: Panasonic. (2010). LNJ437W84RA, Hight Bright Surface Mounting Chip LED).

A continuación procedemos a calcular la resistencia de limitación de dicho LED. En esta ocasión resulta más complejo porque la tensión de alimentación no permanecerá constante, puede variar entre los 110 y los 60V. Por lo tanto calcularemos el valor máximo y mínimo que podrá tomar dicha resistencia. El valor mínimo de la resistencia será aquel que con la máxima tensión (110V) implique la máxima corriente (20mA); pues una resistencia menor implicaría más corriente y podría deteriorar el LED. El valor máximo será aquel que con la tensión mínima (60V) produzca la mínima corriente necesaria para que el LED luzca (1mA). Ambos valores se hayan calculados de acuerdo a la ecuación 5.

Ecuación 5

$$R = \frac{V \quad V_D}{I}$$

$$R_{\min} = \frac{110V \quad 2V}{20mA} = 5.4k$$

$$R_{\max} = \frac{60V \quad 2V}{1mA} = 58k$$

Se decide emplear una resistencia de un valor intermedio, 22Kohm. Las corriente máxima y la corriente mínima que circularán por el LED serán 4'9mA y 2'6mA respectivamente, dicho cálculo se realiza empleando la ecuación 5.

Finalmente, debemos tener en cuenta la potencia que se disipará en la resistencia, pues las resistencias SMD solo suelen permitir disipar mínimas cantidades de energía. El cálculo de potencia puede realizarse de dos formas distintas, si bien son similares. La definición clásica indica la potencia consumida como el producto de la caída de tensión por la corriente circulante. Sin embargo, para el cálculo de potencia disipada en resistencias se suele preferir emplear una versión derivada de la anterior al añadirle la ley de Ohm para despejar el voltaje, ecuación 2. Según esta versión la

Autor: **Javier Martínez Lahoz**
424.16.93

potencia disipada es el valor de la resistencia multiplicada por el cuadrado de la corriente que la atraviesa.

Ecuación 8

$$P = 22k \cdot (4'9mA)^2 = 496mW$$

El cálculo de la potencia de la ecuación 8, empleando los datos 22Kohm y 4'9mA, indica que será necesario que la resistencia sea capaz de disipar como mínimo 496mW de potencia. Por seguridad decidimos emplear una resistencia que permite una disipación mayor, por ejemplo 0'75W. Las resistencias SMD de pequeño encapsulado, como el 0805, suelen permitir disipaciones de calor del orden de 125mW, por lo que no empleamos restricciones de tamaño en la búsqueda. Tras emplear una búsqueda centrada en potencia y valor óhmico, acabamos escogiendo la resistencia ASC2010-22KFT4, de Welwyn Components (TT Electronics). Esta resistencia es de 21’5KOhm, valor que consideramos suficientemente cercano al calculado, y nos permite una disipación de 750mW con un tamaño empaquetado relativamente pequeño, 2010; todo ello según los datos de la hoja técnica del fabricante (Ref: TT Electronics (s.f.). Resistors, Anti Sulphur Chip Resistors, ASC Series).

Se decide la incorporación de un fusible de protección, esta elección se basa en la corriente que circulará por estas pistas y la posibilidad de que exista un cortocircuito externo al sistema. Concretamente se empleará un portafusiles de cuchilla para poder cambiar rápidamente de fusible en caso de que se queme. Esto se debe a que el regulador obtiene parte de su energía de este punto, por lo tanto debemos proteger el cable, al igual que cualquier otra salida de potencia que exista en la placa. Con esta protección, si se produce un cortocircuito en el exterior, se evita que las pistas de la placa se quemen aumentando la robustez del sistema. El portafusibles seleccionado es el 178.6165.0001 de Littelfuse. Se trata de un componente pensado para automoción, por lo que se adapta a nuestras necesidades Ref(Littelfuse. (2008). Blade Fuse Holders - FLR PCB Fuse Holder for ATO® Style Blade Fuse Rated 80V). Emplearemos fusibles de cuchilla en el mismo, en concreto se usarán fusibles de 3A, que protegen las pistas a la vez que permiten alimentar el regulador correctamente.

El esquema electrónico final de este subsistema se muestra en la ilustración 14, donde se puede apreciar perfectamente el conexionado entre los distintos componentes.

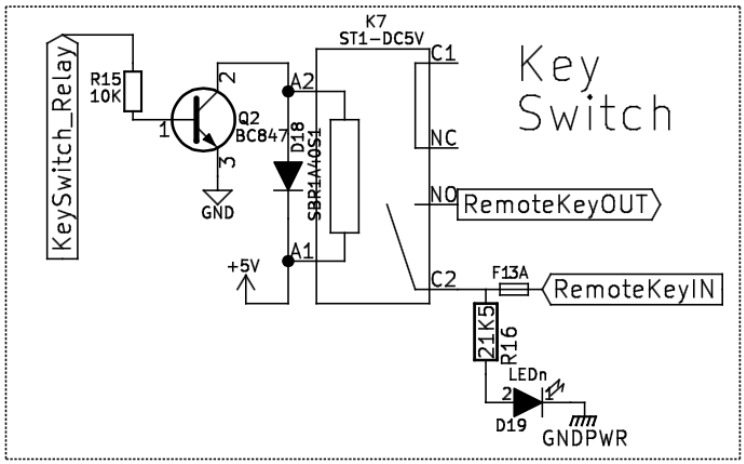

Ilustración 14. Esquema electrónico del Key-Switch

5.2.2.4. Security Maneuver

Este subsistema contendrá todos los mecanismos que actúen directamente sobre la maniobra de seguridad. Estará formado principalmente por tres relés controlados por distintos sistemas: el primero de la propia ECU, el segundo del IMD, y el último del regulador. También cuenta con un relé adicional, esto se debe a que con el regulador apagado la maniobra de seguridad no podría completarse, por lo que no se puede dar la alta tensión para encender el regulador y debido a esto nunca se podría iniciar la marcha. Para solucionar este problema se incorpora un cuarto relé, colocado en paralelo con el del regulador. Este nuevo actuador, al que denominaremos Start-Up Relay, estará controlado por la ECU, y se activará tan solo cuando el regulador este apagado. Una vez el regulador se encienda, este relé se desactivará, permitiendo que este pueda actuar sobre la maniobra de seguridad si detecta algún problema.

Por temas de seguridad, al igual que en el subsistema del Key-Switch (apartado 5.2.2.3), se decide emplear relés convencionales para realizar la maniobra de seguridad. Los cuatro relés que formarán la maniobra de seguridad serán controlados por tres sistemas distintos, por lo tanto será necesario contar con tres modelos distintos de relé: uno para el IMD con control a 24V, otro para el regulador con control a 48V o 96V, y dos más de la ECU con control a 5V. Todos ellos se conectarán en serie, excepto el relé del Start-Up que se colocará en paralelo con el del regulador.

La tensión de contacto de todos los relés será de 24V, con una corriente de paso de menos de 1A; pues se trata de una maniobra indirecta que actúa contra la bobina de un contactor. Este relé pertenece a otro sistema de la moto, la batería, y no se

encuentra definido todavía; por lo que optamos por 1A de corriente máxima, es asumible que la corriente real será mucho menor.

Para la implementación de los relés de la ECU se procura que la corriente de control sea inferior a los 20mA que el Arduino Yún puede suministrar directamente, evitando el uso de transistores para una etapa de potencia intermedia. Las características que influyen a la hora de escoger el componente son: tensión y corriente de contacto, tensión y corriente de control, precio y tamaño. Finalmente el modelo escogido es el TXS2-4.5V de Panasonic. Según su Datasheet (Ref: Panasonic Electric Works Co. (s.f.). TX-S RELAYS) conmuta con una corriente de contacto de apenas 11mA (50mW), lo que permite un control directo desde la mayoría de microcontroladores, incluyendo el ATmega32U8. La tensión de aislamiento entre la bobina y el control es de 1800V.

El segundo relé del subsistema que definimos es el controlado por el IMD, este deberá conmutar con 24V de corriente continua. Al igual que en caso anterior, se tendrán en cuenta la tensión y la corriente de contacto. Además se procurará que la corriente de disparo sea la menor posible, aunque el IMD cuenta con una etapa de salida mediante un transistor. Un relé que cumple con las necesidad de una forma eficiente es el PCJ-124D3M de TE Connectivity. Este se activa con una potencia en la bobina de 200mW a 24V, una corriente de menos de 10mA. La tensión de aislamiento proporcionada entre control y contacto es de 4000V. Estos datos se obtienen de su hoja técnica (Ref: Tyco Electronics. (2005). Miniature PCB Relay PCJ.).

Finalmente el relé del regulador debe estar preparado para una tensión de control de 48V de corriente continua, se escoge emplear esta tensión frente a 96V por motivos de seguridad eléctrica. Este relé será probado antes de su validación para asegurar su correcto funcionamiento. Esto se debe a que la señal del regulador no es una señal continua de 48V, sino una señal modulada por ancho de pulso (PWM) de 96V cuya tensión media resultante es de 48V. Realizamos una búsqueda de relés con bobina a 48V, tensión de contacto de 24V y corriente de contacto de 1A. El relé que emplearemos para esta función es PCH-148D2,000 de TE Connectivity. Según los datos proporcionados por el fabricante (Ref: TE Connectivity Ltd.. (2012). Miniature PCB Relay PCH.) este relé permite conmutar hasta los 30V con cargas más de 5A. Entre sus características destacamos la potencia de conmutación, 200mW, y la tensión de aislamiento entre el control y el contacto, 4000V.

Desarrollo

Todos los relés requieren de la protección de la bobina mediante un diodo en contraparalelo en las bobinas como ya vimos en el Key-Switch. Por lo tanto necesitaremos cuatro diodos, uno para cada relé. El diodo empleado anteriormente en el Key-Switch (Apartado 5.2.2.3), el SBR1A40S1, permite trabajar con tensiones inversas de hasta 40V, por lo que podemos emplearlo para proteger los dos relés de la ECU y el del IMD. Sin embargo, el relé del regulador conmuta a 48V, por lo que es necesario emplear otro diodo.

Decidimos que el nuevo diodo permita trabajar con tensiones de más de 110V, para que pueda ser empleado en cualquier otra parte del sistema. En este relé admitiremos una caída de tensión superior a 1V, puesto que trabajará con tensiones muy elevadas. Consideramos que 1A de corriente directa nominal servirá para todas las aplicaciones. Además decidimos emplear el mismo fabricante que el otro modelo de diodos de la placa, Diodes Inc. El diodo encontrado que cumple, según el fabricante, con estas características es el S1G-13-F (Ref: Diodes Incorporated. (2014). S1A/B - S1M/B 1.0A SURFACE MOUNT GLASS PASSIVATED RECTIFIER). Soporta una tensión inversa de 400V antes de su rotura, por lo que se asegura poder emplearlo en cualquier parte del sistema. Admite una corriente directa de 1A constante, hasta 30A instantáneos, asegurándonos que puede funcionar correctamente como protección del relé del regulador, así como en casi cualquier otra situación. La caída de tensión del diodo es de 1'1V. Se trata de una caída de tensión considerablemente mayor a la del diodo empleado anteriormente, de apenas 0'5V. Este diodo emplea empaquetado SMA, por lo que resulta más voluminoso que el otro diodo empleado; sin embargo, tampoco resulta excesivamente grande y no supondrá ningún problema.

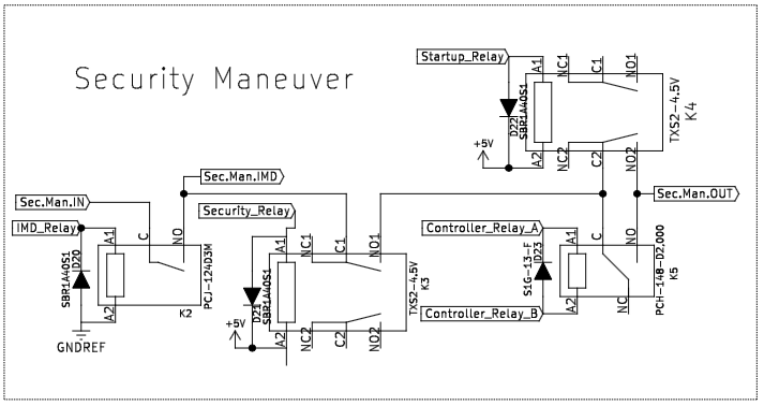

Ilustración 15. Esquema de conexionado de la maniobra de seguridad

Autor: **Javier Martínez Lahoz**
424.16.93

En la ilustración 15 se muestra la serie de relés que forman la maniobra de seguridad, así como el relé en paralelo con el del regulador a la derecha del todo. En esta ilustración también podemos destacar los diodos volantes en todas las bobinas de control y las derivaciones realizadas en diversos puntos de la maniobra para las medidas que se explicarán en el siguiente apartado, 5.2.2.5.

5.2.2.5. GLVS Measures

Este subsistema se encargará de aislar señales procedentes de distintas partes del circuito de baja tensión y adaptarlas a niveles adecuados que el procesador pueda leer directamente. Existen seis medidas que resultan de interés, por lo que se aislarán mediante este subsistema. Dos de estas medidas tendrán relación con el sistema de alimentación, indicando si hay alimentación de servicio externa y si el SAI se encuentra cargado; una estará relacionada con el IMD, y nos servirá para realizar la lectura del nivel de aislamiento eléctrico de la motocicleta; y las tres mediciones restantes nos indicarán el estado de la maniobra de seguridad, detectando en que punto concreto se corta la energía de la maniobra. De esta forma podemos conocer en todo momento el estado del IMD, de la maniobra de seguridad y de la alimentación del sistema.

El uso de optoacopladores resulta ser la mejor solución para cumplir con las necesidades de aislamiento. Si bien la cantidad de optoacopladores comerciales en el mercado es muy elevada, reducimos el abanico al tomar otra decisión relevante del diseño; decidimos emplear optoacopladores de salida digital con el objetivo de minimizar los posibles errores de lectura que pudieran cometerse. Al emplear esta tecnología logramos una compatibilidad máxima con el microcontrolador y reducimos la posibilidad de cometer un error de lectura.

Durante la búsqueda de optoacopladores de salida digital encontramos la familia HCPL-22XX de Broadcom Limited (Ref: Avago Technologies. (2010). Very High CMR, Wide VCC Logic Gate Optocouplers). En concreto resulta interesante el estudio del modelo HCPL-2231. Este modelo cuenta con dos optoacopladores independientes, como se puede observar en el esquema de la ilustración 16 extraída del Datasheet del componente. Se decide emplear este modelo para reducir el número de componentes y conexiones. La corriente de activación de los LEDs internos es superior a la de otros modelos, con un mínimo de 1'8mA para la activación y una corriente recomendada de 2'5mA para evitar errores de lectura.

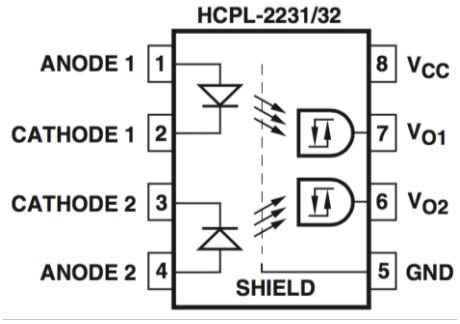

Ilustración 16. Esquema interno del HCPL-2231

En el exterior, lado izquierdo de la ilustración 16, es necesario tomar algunas medidas de protección para evitar el deterioro o rotura de los optoacopladores. Se decide proteger los LEDs frente a tensiones inversas con un diodo rectificador. Ya que todas las tensiones con las que se trabajará están por debajo de los 40V, y las corrientes están por debajo de 1A, mantendremos la estandarización al emplear el diodo SBR1A40S1, previamente mencionado en el apartado 5.2.2.3.

Como con cualquier diodo LED, es necesario emplear una resistencia de limitación de corriente. Dicha resistencia se puede calcular conociendo la tensión de trabajo (24V), la caída de tensión en el optoacoplador, 1'5V típicamente según fabricante; la caída de tensión en el diodo de protección, 0'5V según fabricante; y la corriente que queremos que circule, corriente recomendada por el fabricante del optoacoplador, 2'5mA. Con estos datos se puede calcular el valor de la resistencia en la ecuación 6, similar a la ecuación 1 del apartado 5.2.2.1.

Ecuación 6

$$R = \frac{24V \quad 1'5V \quad 0'5V}{2'5mA} = 8'8k$$

Buscamos en el mercado resistencias con un valor próximo a 8K8; para mantener la estandarización se emplearán resistencias de la marca Multicomp con encapsulado 0805. Aplicando la ecuación 2 con los datos adecuados, obtenemos que la disipación de calor será de 55mW, por lo que emplear un encapsulado 0805 no supondrá ningún problema. La resistencia del valor más próximo al calculado, y que cumple con el resto de restricciones es la MC01W080519K1, con un valor óhmico de 9K1. Al existir una diferencia inferior al 10% del valor calculado se considera un componente válido, pues es inferior a la tolerancia de la misma según el datasheet (Ref: Multicomp. (2013). Thick Film Chip Resistor 0805). Este modelo de resistencia

Autor: **Javier Martínez Lahoz**
424.16.93

se empleará para las medidas de la maniobra de seguridad, el IMD y la alimentación externa.

La medida del nivel de carga de la SAI requiere de una resistencia distinta, pues queremos que se active tan solo por encima de un determinado nivel de tensión del SAI. Decidimos, de forma arbitraria, que con la máxima tensión posible circule 1mA por el circuito. De esta forma cuando la tensión comience a bajar, el optoacoplador se apagará rápidamente. Normalmente existe un margen de seguridad entre los valores que el fabricante indica en el Datasheet y los valores reales, por lo tanto decidimos comenzar con este planteamiento, y en caso de que no funcionase de la forma esperable se cambiaría la resistencia más adelante. El cálculo de la resistencia es idéntico al realizado anteriormente en la ecuación 6, en este caso el valor de corriente que empleamos es 1mA, y por lo tanto el valor de resistencia obtenido es de 22kOhm.

La resistencia empleada para el LED ámbar en el subsistema del Key-Switch, 5.2.2.3 es del mismo valor óhmico; por lo que empleamos el mismo modelo, aunque la capacidad de disipación sea superior a la requerida (Ref: TT Electronics (s.f.). Resistors, Anti Sulphur Chip Resistors, ASC Series).

Finalizamos este subsistema con una protección adicional entre los dos canales del optoacoplador que recibe la señal PWM del IMD, que indica la medida de aislamiento. Se decide emplear la misma protección de supresión de transitorios que la empleada en el apartado 5.2.2.2 en los canales de CANBus, para ello emplearemos un NUP2105 conectado entre ambos canales.

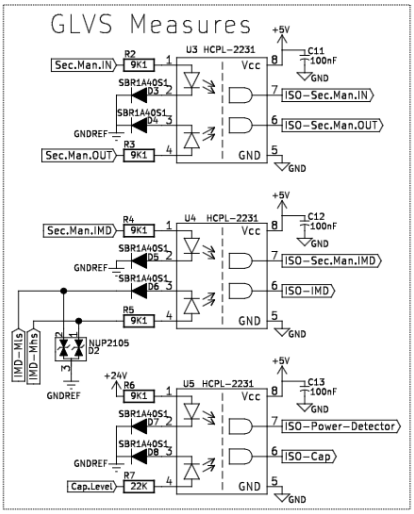

Ilustración 17. Conexionado de los tres diferentes optoacopladores del subsistema

En la ilustración 17 queda patente que los tres optoacopladores empleados se conectan de una forma similar, pero no idéntica. El optoacoplador U4, situado en el centro de la imagen, se encarga de aislar las medidas del IMD, por ello realiza una medida diferencial entre las dos medidas PWM del IMD; además se observa la protección de estos canales mediante el diodo supresor de transitorios. El optoacoplador de la parte inferior se encarga de las medidas relacionadas con la entrada de energía.

5.2.2.6. Power Manager

Este modulo nace de la necesidad primaria de alimentar a los subsistemas internos a partir de una tensión externa mayor, a la vez que se mantiene el aislamiento completo del sistema. Se ha optado por dotar a este subsistema de unas mejores características, para ello se ha decidido incorporar un pequeño sistema de alimentación ininterrumpida (SAI). De esta forma la ECU podrá seguir trabajando durante un periodo de tiempo determinado, incluso después de que la alimentación principal de la motocicleta deje de estar disponible. Esto será útil para poder obtener la información que haya causado un fallo general del sistema sin tener que recurrir a encender nuevamente la motocicleta. También se empleará para dar un margen de tiempo a la ECU a prepararse en caso de corte brusco de energía del sistema. Al mismo tiempo se planea incorporar un interruptor para cortar la alimentación a determinados subsistemas no vitales cuando la alimentación principal desaparezca, de esta forma se optimiza el tiempo útil del acumulador de emergencia (SAI).

Este subsistema lo analizaremos de acuerdo a las tres partes principales que lo componen: el convertidor, el acumulador y el interruptor.

5.2.2.6.1. Fuente de alimentación

En primer lugar estudiaremos el convertidor, se trata del elemento principal que nos proporcionará una alimentación estable para la electrónica, 5V, a partir de la tensión de servicio, de 24V. Resulta importante la elección de este componente, pues si la alimentación presenta problemas, estos repercutirán en el funcionamiento de la electrónica. Resulta imprescindible que el convertidor sea aislado y eficiente, por lo que se empleará una fuente de alimentación conmutada. Por lo tanto generará un ruido de salida debido a la conmutación interna empleada mantener estable la salida.

Decidimos las características que dicho convertidor debe cumplir para funcionar de forma óptima en el sistema. En primer lugar debe minimizar el ruido saliente para

Autor: **Javier Martínez Lahoz**
424.16.93

evitar problemas derivados. Un bajo ruido de salida permitirá que minimizar el número de condensadores de filtro de salida del subsistema y evitará posibles problemas de funcionamiento debido a glitches. También resulta interesante que cuente con un amplio rango de tensiones de entrada. Cuanto mayor sea el rango de voltajes de entrada, más energía podremos obtener del acumulador. Por lo tanto, este valor determinará, junto con la carga del acumulador, cuánta energía se puede aprovechar en el sistema tras la pérdida de la alimentación principal. Además nos interesará que el convertidor no requiera de componentes de acondicionamiento externo, pues cuanto mayor es el número de componentes de acondicionamiento externo, mayores son las posibilidades de que alguno falle debido a una mala conexión o una mala elección de un componente. Empleando un componente con todo integrado minimizamos la posibilidad de que pueda fallar nuestra fuente de alimentación, parte fundamental de la placa.

Tras un estudio de consumo de los diferentes subsistemas se considera que la potencia máxima que consumirá la placa es 3'75W; suponiendo que todos los relés están activados, los optoacopladores funcionando y ambos canales de CANBus transmitiendo; esto se desarrolla en el anexo 5, consumos de los componentes. Por lo tanto debemos escoger una fuente de más de 4W. También se filtra la búsqueda del convertidor en función de su eficiencia. Finalmente la fuente de alimentación escogida es el modelo TEN 5-2411WI de Tracopower. Esta fuente proporciona hasta 5W de potencia, con un ruido de salida de 80mV de amplitud en el peor de los casos a una frecuencia de 20MHz debido a la conmutación. Todos estos datos se obtienen de las hojas técnicas proporcionadas por el fabricante (Ref: Tracopower. (2015). DC/DC Converters TEN 5WI Series, 6 Watt.). Emplearemos condensadores de desacoplo en cada subsistema para suavizar este ruido y que no pueda afectar al funcionamiento de la electrónica.

Una de las mejores características de este componentes es su amplio rango de tensiones de entrada. Permitiendo tensiones de entrada desde 9V hasta 36V; esto nos permitirá obtener una gran cantidad de energía del acumulador.

5.2.2.6.2. Acumulador de energía

La primera decisión que debemos tomar del acumulador es la tecnología que empleará para almacenar la energía. Podríamos emplear celdas químicas (LiPo), sin embargo estas requerirían de una supervisión constante, destinando una gran cantidad de recursos del sistema de los que no resulta conveniente prescindir. La otra

opción es el uso de tecnología capacitiva, en concreto empleando condensadores electrolitos de doble capa, también llamados supercondensadores. Estos condensadores logran una capacidad de carga específica muy superior a los tradicionales, de esta forma podemos lograr acumular carga suficiente para nuestras necesidades, sin requerir supervisión constante.

Una vez se ha decidido que se emplearán supercondensadores como acumulador, es necesario calcular cuánta capacidad necesitaremos. Para realizar estos cálculos emplearemos la fórmula de la energía acumulada en un condensador y la energía consumida a lo largo del tiempo por un sistema. Tendremos en cuenta un consumo constante y una pérdida por eficiencia debido a la fuente de alimentación. Para calcular el tiempo que nuestro sistema se podrá mantener alimentado es necesario calcular la energía acumulada en un condensador en función de la tensión de sus terminales, V1 y Vref, que se muestra en la ecuación 7.

Ecuación 7

$$= \frac{1}{2} C(V_1 \quad V_{ref})^2$$

Además si un terminal permanece a una tensión constante (Vref), y el otro modifica su tensión, de V1 a V2, la energía liberada o absorbida por un condensador se puede calcular empleando la diferencia de cuadrados de las tensiones, en lugar de el cuadrado de la diferencia, esto se muestra en la ecuación 8, derivada de la ecuación 7.

Ecuación 8

$$= \frac{1}{2} C(V_1^2 \quad V_2^2)$$

La formulación matemática de la energía consumida de un sistema puede expresarse de diversas formas, sin embargo la forma más simple es multiplicar la potencia empleada del sistema por el tiempo que permanece encendido. Además, se puede incluir en la ecuación un factor de rendimiento, como se ha realizado en la ecuación 9, para tener en cuenta la eficiencia de transformación de la fuente.

Autor: **Javier Martínez Lahoz**

424.16.93

Ecuación 9

$$= \frac{P \times t}{}$$

Para conocer el tiempo que se mantendrá alimentado el sistema por medio de los condensadores igualamos la potencia disponible y la consumida de las ecuaciones 8 y 9 respectivamente; sustituimos lo valores conocidos: tensión de máxima carga, aproximadamente 22V debido al regulador de corriente y el diodo de protección (explicados ambos más adelante en este mismo apartado), tensión mínima de trabajo de 9V según el Datasheet de la F.A., la potencia consumida la fijamos en función del consumo máximo de 4W y la eficiencia de la fuente de alimentación según el fabricante, 78%. Al igualar y sustituir ambas ecuaciones obtenemos la ecuación 10.

Ecuación 10

$$\frac{4W \times t[s]}{71\%} = \frac{1}{2} C[F] \times ((22V)^2 \quad (9V)^2)$$

El resultado final obtenido tras despejar la ecuación 10, mostrado en la ecuación 11, demuestra que por cada faradio de capacidad de los supercondensadores se podrá alimentar al circuito durante 35'7 segundos.

Ecuación 11

$$t[s] = 35'7 \times C[F]$$

Una vez conocida esta relación debemos establecer el tiempo que queremos que el sistema sea independiente, de donde deduciremos la capacidad de los supercondensadores. Se decide que el SAI deberá ser capaz de mantener la ECU alimentada durante un tiempo mínimo de 20 segundos, valorándose de forma positiva el aumento de dicho tiempo. De acuerdo a la relación entre tiempo y capacidad previamente calculada, será necesario emplear condensadores con una capacidad mínima de 0'5F.

Uno de los principales inconvenientes de los supercondensadores es la baja tensión de trabajo que soportan, por lo tanto será necesario serializar supercondensadores para alcanzar los 24V de trabajo nominal del sistema. Al serializar condensadores se reduce su capacidad equivalente, por lo que es necesario emplear condensadores de más capacidad, o varias ramas en paralelo. El precio de los

supercondensadores es muy superior al de los condensadores tradicionales, por lo que si vamos a emplear una cantidad considerable de ellos, debemos tener en cuenta su precio a la hora de seleccionar un modelo frente a otro.

Tras un estudio de los supercondensadores disponibles en el mercado, teniendo en cuenta su capacidad, tensión de trabajo y precio, se ha decidido emplear supercondensadores de la marca Vishay, en concreto el modelo MAL222091003E3. Este componente soporta tensiones de hasta 2'7V, proporcionando una capacidad de 20F según el Datasheet de esta familia de supercondensadores (Ref: Vishay BCcomponents. (2016). 220 EDLC ENYCAP). Debido a su baja tensión se emplearán diez de estos supercondensadores conectados en serie. La capacidad equivalente del conjunto será de 2F, pudiendo trabajar hasta a 27V; todo esto según las leyes básicas de asociación de condensadores. Esta otorgará una alimentación independiente durante más de un minuto y medio, siempre y cuando los supercondensadores se encuentren completamente cargados.

El principal problema que debemos tener en cuenta al emplear acumulación de energía mediante supercondensadores es el tiempo de carga. Los supercondensadores tiene una resistencia interna muy reducida, por lo que una tensión pequeña entre sus bornes puede suponer una corriente muy elevada atravesando al mismo y las pistas correspondientes de la placa. Esta corriente elevada puede llevar al calentamiento y deterioro del componente o de las pistas. Por lo tanto es necesario emplear algún elemento que limite la corriente entrante. La opción más fácil consiste en emplear una resistencia de limitación de carga, sin embargo esta conlleva determinados problemas: pérdidas continuas por calor, corriente variable, tiempo elevado… Una solución alternativa consiste en emplear un regulador de corriente, de esta forma cargamos los supercondensadores a una corriente constante y conocida, logrando una carga más rápida.

Para implementar el regulador de corriente recurrimos a uno de los reguladores de tensión más empleados proyectos de electrónica a nivel mundial, el LM317, en su versión SMD: el LM317EMP/NOPB de Texas Instruments. El Datasheet de este componente (Ref: Texas Instruments. (2015). LM117, LM317-N Wide Temperature Three-Pin Adjustable Regulator) incluye el circuito típico de aplicación para emplearlo como regulador de corriente, apartado *9.2.8 1-A Current Regulator, Figure 26*. Este componente nos permite cargas de hasta 1'5A.

Autor: **Javier Martínez Lahoz**
424.16.93

Es necesario añadir una resistencia de polarización para que el regulador de corriente entregue la corriente deseada. Dicha resistencia se colocará entre el pin de ajuste y la salida, como se puede observar en el apartado *9.2.8 1-A Current Regulator, Figure 26* del Datasheet. El valor es calculado mediante la Ley de Ohm, empleando la tensión mínima de salida del componente, 1'25V. Al introducir la corriente de carga deseada en la ecuación 11 obtendremos el valor de resistencia que debemos emplear.

Ecuación 12

$$R = \frac{1'25V}{I}$$

Se ha decidido que la potencia consumida por esta parte del sistema no genere un consumo superior a 16W, pues cuanto mayor es el consumo, mayor tendrá que ser la fuente de baja tensión de la moto, o se podrían producir problemas de alimentación en todos los sistemas. En la ecuación 12 se relaciona este valor con la corriente que implica.

Ecuación 13

$$16W = V \times I$$

Posteriormente se calcula la resistencia necesaria en el regulador de tensión según las ecuaciones 12 y 13, teniendo en cuenta que la tensión de trabajo es de 24V. La resistencia que debemos emplear de acuerdo a estas fórmulas será de 1'8Ohm, pero antes de escoger un modelo concreto debemos tener en cuenta cuanta potencia deberá ser capaz de disipar dicha resistencia durante la carga. Emplearemos la fórmula de la ecuación 2 del apartado 5.2.2.1 para calcular la potencia disipada en la resistencia, para ello emplearemos el valor de corriente obtenida en la ecuación 13. Al combinar estos datos obtenemos que la potencia que se disipará en la resistencia será de 0'8W. Por lo tanto debemos emplear una resistencia que sea capaz de soportar una cantidad mayor de energía, por ejemplo 1W. Al tratarse de una resistencia para disipar 1W, no emplearemos ningún criterio de búsqueda por tamaño. Las restricciones para la búsqueda serán el valor de 1'8Ohm, la potencia de 1W y un empaquetado SMD, independientemente de cual.

La resistencia que hemos escogido es el modelo 35201R8JT de TE Connectivity, que permite disipar hasta 1W de potencia, manteniendo un encapsulado SMD 2512

[6432 Metrico] (Ref: TE Connectivity. (2011). SMD Power Resistors Type 3520 Series).

5.2.2.6.3. Interruptor de potencia

Finalmente debemos plantear el método para resolver el interruptor, o switch, que cortará la alimentación a sistemas no vitales tras una perdida de energía. Se valora la opción emplear un transistor para esta función, sin embargo se ha optado por una solución más inteligente y segura. Esta solución pasa por emplear por un Smart-Switch. Este tipo de componentes se activan mediante una señal digital, frente a la polarización analógica de un transistor. Además la mayoría de modelos cuentan con protecciones internas frente a cortocircuitos y sobretemperatura, resultando en un sistema mucho más fiable y robusto. Las características más importantes a la hora de escoger este componente son la tensión de trabajo, la potencia que puede entregar, el encapsulado y las protecciones que integra.

La tensión de trabajo del componente será de 5V y deberá ser capaz de manejar potencias de hasta 1'5W. Esta potencia se fija según el consumo máximo de los subsistemas de comunicaciones CANBus. Finalmente se encuentra el modelo el STMPS2151STR de STMicroelectronics, que cumple con todas la necesidades (Ref: STMicroelectronics. (2013). Enhanced single channel power switches TMPS2141, STMPS2151, STMPS2161, STMPS2171). Este componente actúa como interruptor de potencia a voltajes de hasta 5'5V y corrientes de hasta 500mA. Permite por lo tanto controlar potencias de hasta 2'75W, cumpliendo holgadamente con nuestras necesidades. Emplea un transistor MOS para realizar la conmutación, dicho transistor genera una resistencia equivalente de 130 miliohmnios, que apenas afectará al resto del circuito. Además cuenta con una serie de protecciones que proporcionan mucha seguridad al sistema. Estas protecciones, explicadas en el apartado 3.3 del Datasheet, protegen al componente y a su carga frente a cortocircuitos, sobrecorriente y exceso de temperatura. Este Smart-Switch genera una alta impedancia en su pin de salida cuando se encuentra desconectado para evitar problemas de corriente inversa. Como última característica destacable citaremos su bajo consumo cuando se encuentra en reposo, apenas 12uA según el fabricante.

El Smart-Switch además genera una salida de errores, se decide que sea identificable directamente por el usuario. Para realizar esta comunicación se decide emplear un indicador luminoso de color rojo. Ya hemos empleado LEDs en otros subsistemas, así que decidimos estandarizar, para ello emplearemos modelos de la

misma familia. El diodo LED de color rojo de la misma familia que los empleados anteriormente, es el LNJ237W82RA de Panasonic. En este caso la corriente máxima de trabajo es de 20mA, la corriente nominal es de 5mA y la caída de tensión es de 1'9V. Todo ello ha sido obtenido de los datos técnicos proporcionados por el fabricante (Ref: Panasonic. (2011). LNJ237W82RA Hight Bright Surface Mounting Chip LED).

El cálculo de la resistencia de limitación se realiza de forma similar al desarrollado anteriormente. En este caso se obtiene un valor resistivo de 620Ohm, la obtención de dicho valor se realiza igual que anteriormente, mediante la ecuación 1 y empleando los valores de caída de tensión del LED y la corriente recomendada por el fabricante. Se busca el componente concreto igual que en anteriores ocasiones, tratando de mantener la estandarización. En este caso la resistencia empleada será de 611Ohm frente a los 620Ohm calculados. Esta diferencia no se considera relevante, ya que se trata de una variación del 1'45%. El modelo comercial, de la marca Multicomp, recibe la denominación MC01W08051619R. Emplea la misma hoja técnica que el resto de resistencias de la familia (Ref: Multicomp. (2013). Thick Film Chip Resistor 0805).

5.2.2.6.4. Acondicionamiento del circuito

Para que el subsistema de alimentación funcione correctamente y sea robusto emplearemos dos diodos. El primer diodo se colocará a favor de la corriente a la entrada de la alimentación, de esta forma se logran dos comportamientos favorables. En primer lugar se protege el sistema frente a una posible alimentación externa que pudiera provocar problemas; mientras que a la vez se evita que pueda salir energía de la placa al exterior, concretamente de la SAI, de esta forma toda la energía acumulada se emplea exclusivamente para la alimentación de la placa.

El segundo diodo a emplear se colocará con el ánodo en el terminal positivo de los supercondensadores y el cátodo en la entrada de la fuente de alimentación, como se puede observar en la ilustración 18, donde se referencia como D9. Este diodo evitará la entrada directa de corriente de la alimentación externa a los supercondensadores, pero permitirá que la energía acumulada en los mismos fluya hacia la fuente de alimentación cuando la tensión exterior caiga por debajo de la de los supercondensadores. En ambos casos se empleará el mismo diodo que anteriormente: SBR1A40S1. En este circuito influye en gran medida la caída de tensión del diodo, pues cuanto menor sea, más energía se podrá aprovechar de los supercondensadores. Este modelo en concreto tiene una caída de tensión típica de

0'5V según el fabricante (Ref: Diodes Incorporated. (2015). SBR1A40S1, 1A SBR® SUPER BARRIER RECTIFIER, DS33306).

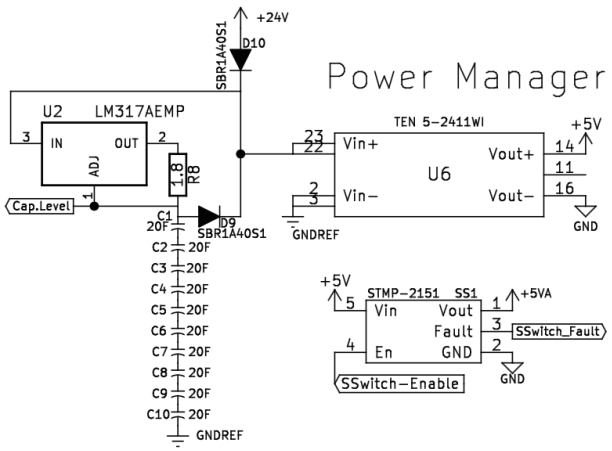

Ilustración 18. Esquema electrónico del subsistema Power Manager

Además será necesario colocar condensadores de filtrado en la alimentación de todos los componentes electrónicos, este condensador recibe el nombre de condensador de desacoplo. Estos deben colocarse lo más cerca posible de la entrada de alimentación de los componentes electrónicos para que cumplan con su cometido. Los condensadores de desacoplo se encargan de contrarrestar el ruido de la fuente y la inductancia generada por las pistas de las placas de circuito impreso.

El valor de estos condensador se suele elegir arbitrariamente, su valor óptimo depende de diversos factores: amplitud del ruido producido por la fuente de alimentación, frecuencia de dicho ruido, consumo del componente, inductancia de las pistas que los unen, etc. Como norma general estos condensadores se suelen colocar de entre 10nF y 100nF; en nuestro caso optamos por los condensadores de 100nF ya que permiten una mayor absorción de ruido.

Una vez aplicados todos los filtros siguen existiendo una gran cantidad de componentes de distintos fabricantes disponible. Para mantener la estandarización se decide emplear condensadores de Walsin con encapsulado 0808, en concreto emplearemos el modelo 0805B104K500CT, de la misma familia que el resto de condensadores. Los datos de este se pueden encontrar en la hoja técnica del fabricante (Ref: Walsin Technology Corporation. (2015). MULTILAYER CERAMIC CAPACITORS General Purpose Series).

Autor: **Javier Martínez Lahoz**
424.16.93

Estos condensadores se emplearán para filtrar la entrada de alimentación de todos los componentes electrónicos dependientes del 5-TEN2411WI: los MCP2515, la mitad aislada de los ISO1050, los HCPL2231 y el Arduino Yún.

Finalmente emplearemos un diodo LED de color verde para indicar al usuario si el sistema se encuentra alimentado. El LED de la misma familia que los empleados anteriormente es el LNJ237W82RA de Panasonic (Ref: Panasonic. (2008). LNJ326W83RA Ultra High Bright Surface Mounting Chip LED.). Las características de este son: Corriente máxima de 15mA, corriente recomendada de 10mA y caída de tensión de 2'05V. Tras el cálculo correspondiente de la resistencia de limitación, similar a la de la ecuación 1 del apartado 5.2.2.1, se obtiene que deberá tener un valor resistivo de 295Ohm.

5.2.2.7. Subsistemas auxiliares

Con el objetivo de permitir una expansión de entradas y salidas, se plantea estos subsistemas. La principal característica de este subsistema debe ser la adaptabilidad, pues esta pensado para suplir la carencia de alguna conexión no prevista previamente. Se implementarán entradas y salidas tanto en el circuito de alta tensión como del de baja tensión. Los componentes y circuitos empleados serán los mismos empleados en otros subsistemas, esto se realiza para mantener la estandarización y reducir el esfuerzo de diseño. Se decide implementar un circuito auxiliar de cada tipo, es decir, un circuito de salida de baja tensión, un circuito de salida de alta tensión, un circuito de entrada de baja tensión y un circuito de entrada de alta tensión.

El circuito de salida de baja tensión estará formado por un relé TXS2-4.5V, se trata del modelo empleado por la ECU para el control de la maniobra de seguridad en el apartado 5.2.2.4. Al igual que en aquel caso será necesario incluir el diodo de protección SBR1A40S1. Todas las conexiones del contacto del relé serán redirigidas hacia el exterior para maximizar la adaptabilidad del circuito.

El circuito de salida de alta tensión será similar al circuito principal del Key-Switch, apartado 5.2.2.3. Sin embargo, no se incluirá en este circuito el LED indicador ni el fusible de protección al no conocerse su posible aplicación. Los componentes empleados serán: el relé, el transistor, la resistencia de polarización y el diodo de protección, todos ellos similares a los empleados anteriormente.

El circuito de entrada de baja tensión será idéntico al empleado para medir la maniobra de seguridad en el apartado 5.2.2.5. Se empleará un optoacoplador doble

con resistencias 9K1 pensadas para conmutar a 24V. Todos los componentes serán similares a los empleados anteriormente, y todas las conexiones del optoacoplador se llevarán al exterior. Se emplearán ambos optoacopladores del HCPL-2231.

El circuito de entrada de alta tensión también empleará un optoacoplador HCPL2231, se debe rediseñar el circuito para que pueda trabajar a 110V en vez de a los 24V que se ha empleado previamente. En este caso se decide emplear tan solo uno de los dos optoacopladores del HCPL-2231.

El primer paso será sustituir el diodo de protección SBR1A40S1 por el S1G-13-F, pues el primero no soporta más de 40V. Tras esta sustitución deberemos recalcular la resistencia de limitación, y en este caso si será necesario tener en cuenta la potencia disipada por la misma. Para realizar los cálculos debemos fijar la corriente máxima, decidimos que dicha corriente sea de 1'8mA, corriente mínima que activa el optoacoplador. Empleamos esta corriente para minimizar la potencia disipada en la resistencia de limitación. Se empleará la fórmula de la ecuación 6 del apartado 5.2.2.5 para calcular el valor de la resistencia que debemos emplear; tras los cálculos pertinentes obtenemos que dicha resistencia deberá ser de 60KOhm y disipar una potencia de 194mW según la ecuación 2.

La resistencia empleada deberá tener un valor cercano a 60kOhm, poder disipar más de 200mW más un margen de seguridad, y soportar tensiones superiores a 110V. Tras una búsqueda empleando estas restricciones se encuentra una resistencia que cumple con todas estas necesidades en un empaquetado SMD0805, si bien no mantiene la estandarización de fabricante del resto de resistencias. Se trata de la resistencia RP73PF2A59KBTDF de TE Connectivity, con un valor de 59Kohm y una disipación de potencia de 250mW, datos técnicos del fabricante (Ref: TE Connectivity. (2012). SMD High Power Precision Resistors Type RP73 Series).

Se han implementado una gran cantidad de E/S auxiliares para dotar a la placa de más adaptabilidad al sistema; sin embargo surge un problema, el Arduino Yún no dispone de pines de E/S suficientes para todos los auxiliares. Se decide que la solución pasará por poder seleccionar manualmente que E/S auxiliares se conectan al Arduino y cuales no. Para lograr esto se debe diseñar un sistema que permita realizar cualquier combinación de conexiones entre las dos pistas que surgen del Arduino destinadas a las E/S auxiliares, y las cinco pistas que surgen de los circuitos auxiliares.

Autor: **Javier Martínez Lahoz**
424.16.93

La solución planteada se muestra en la ilustración 19 y se basa en emplear un clavijero lineal al que se llevan alternativamente las conexiones del Arduino y las conexiones de los circuitos auxiliares. De esta forma cada circuito auxiliar podrá conectarse independientemente a cada uno de los pines de Arduino destinados a entradas auxiliares. Esta solución permite cualquier combinación de las posibles, y no provoca ningún problema, exceptuando que es necesario realizar los cambios de las conexiones de los circuitos auxiliares de forma manual.

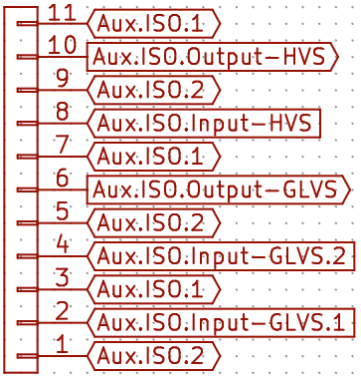

Ilustración 19. Distribución de las conexiones en el clavijero selector de circuitos auxiliares

El clavijero empleado será similar al empleado para las resistencias de final de línea de CANBus, apartado 5.2.2.2. El componente concreto será el MC34735 de Multicomp, datos en su hoja técnica (Ref: Multicomp. (2012). 2211S - MC34 Series).

Cabe destacar que durante la competición se emplearon los dos circuitos auxiliares de alta tensión. El relé se empleó para controlar el interruptor de marcha del regulador, Forward-Switch; mientras que el optoacoplador se conecto a las líneas de alta tensión para monitorizar si existía dicha alta tensión fuera de la caja de la batería, independientemente de las señales del regulador.

5.2.2.8. Interface

Este subsistema se encarga de la conexión física entre el interior y el exterior de la placa. Esta formado principalmente por los conectores, que conectarán los subsistemas internos con los sistemas externos, repartidos por el resto de la motocicleta. Estos conectores se elegirán en función de las necesidades del conexionado, así como el numero de conexiones necesarias.

Con el objetivo de dotar a la placa de adaptabilidad de cara a una expansión de componentes, se decide colocar derivaciones de diferentes voltajes como salida, de

esta forma algún pequeño sistema externo, incluido en un momento tardío del desarrollo, podrá ser alimentado desde la ECU sin necesidad de recurrir a modificar todo el cableado de la motocicleta. También se deberán llevar hasta los conectores derivaciones a masa, para poder disponer de la alimentación completa para cualquier subsistema.

Las derivaciones que se realizan en este subsistema deberán estar protegidas frente a cortocircuito externo, por lo que se decide emplear fusibles en las derivaciones. Sin embargo emplear una gran cantidad de fusibles implicaría una gran cantidad de espacio, así como una gran cantidad de repuestos necesarios, por si fuera necesaria alguna sustitución. La solución final se basa en el uso de fusibles autorearmables SMD. De esta forma evitamos tener que disponer de un stock variado de fusibles para hacer frente a las incidencias, a la vez que reducimos el espacio ocupado por los mismos.

Dividimos en tres grupos las acometidas de potencia: en primer lugar la toma de corriente del IMD, en segundo lugar disponemos cuatro acometidas de potencia sin asignar de 24V y dos de 5V (aislados respecto a los 5V que se emplean para alimentar la electrónica). Cada uno de los grupos se protegerá con fusibles de distintas características.

La alimentación del IMD es algo de vital importancia, debemos emplear un fusible rearmable que nos asegure que la potencia de corte siempre este por encima de la potencia que puede consumir el IMD. Según el Datasheet del medidor de aislamiento (Bender. (2016). ISOMETER® IR155-3203/IR155-3204), el consumo nominal es de 150mA, sin embargo decidimos emplear un fusible muy superior que tan solo se dispare en caso de cortocircuito. La búsqueda de fusibles se realiza de acuerdo a tres parámetros. El más importante es la corriente de corte, sin embargo también cobra importancia la tensión nominal de trabajo, pues si trabajas muy por encima de este pueden generarse fenómenos de arco eléctrico que estropearían el componente. También tendremos que fijarnos en el tamaño del fusible, porque tendremos que disponer siete de ellos cerca del conector, y un empaquetado grande podría dificultar el ruteo en la zona.

Para el IMD escogemos un fusible rearmable de 1’5A, este se encuentra muy sobredimensionado frente al consumo del IMD, sin embargo su tarea es proteger las pistas del sistema y el cable, no el medidor de aislamiento, por lo que se considera un valor admisible. Realizamos una búsqueda de fusibles autorearmables SMD con una

corriente nominal de 1'5A y con una tensión nominal de al menos 24V; llama nuestra atención la familia de fusibles 1812L de Littelfuse, el mismo fabricante que se empleo en el portafusibles del Key-Switch. Todos los fusibles de esta familia son fusibles SMD con empaquetado 1812, como su nombre indica; este empaquetado resulta muy satisfactorio dadas nuestras necesidades. La familia 1812L contiene fusibles con una corriente nominal desde 0'1A hasta 3A y tensiones de entre 6V y 60V, todos los modelos y características se encuentran en su hoja técnica (Littelfuse. (2013). POLY-FUSE Resettable PTCs 1812L Series). Para el fusible de la alimentación del IMD emplearemos el modelo 1812L150/24MR, con una corriente nominal de 1'5A y una tensión nominal de 24V. Según el fabricante, el fusible se dispara con 3A a los 1'5s o con 8A. Además asegura que no se producirán desperfectos en el mismo con corrientes inferiores a 20A.

Para las acometidas de 24V emplearemos un fusible de la misma familia, 1812L de Littelfuse. En este caso reduciremos la corriente nominal a 0’5A, lo que proporcionaría hasta 12W antes de cortar la energía de la acometida. El fusible que cumple con estas restricciones es el 1812L050/30PR. Destacan su rápida actuación, apenas 0'15s con una corriente de 8A, y su corriente máxima sin deterioro, 100A.

Finalmente para las acometidas de 5V emplearemos fusibles de mayor corriente, puesto que un voltaje menor implica menor potencia. Empleando fusibles de 1’1A se podrán suministrar 5’5W por salida. El modelo empleado en este caso es el 1812L110/24DR, muy similar en características al modelo de 0'5A. Las derivaciones de 5 se obtendrán de la DC/DC del canal de comunicaciones CANBus GLVS; de esta forma la alimentación permanecerá aislada del núcleo de procesamiento de la placa.

Se emplearán dos conectores independientes: uno para las conexiones del circuito de alta tensión, y otro para las conexiones del circuito de baja tensión. Para mejorar la robustez del sistema se emplearán conectores de automovilismo con sellos antihumedad. Estos conectores, al igual que en los vehículos comerciales, se dispondrán hacia abajo en la motocicleta. De esta forma se evitará la acumulación de agua en los contactos en caso de que se produzcan precipitaciones.

El regulador cuenta con un conector Ampseal de 35 pines para su conexionado de control. Esta gama de conectores cumple con nuestras necesidades, así que con el objetivo de estandarizar se decide emplear la serie de conectores Ampseal de TE Connectivity para las conexiones de la placa.

Desarrollo

A continuación se realiza un estudio de las salidas del conector de baja tensión. Se emplearán 4 pines para el bus de alimentación y CANBus, 5 para el bus de alimentación e información del IMD, 2 para la maniobra de seguridad, 4 derivaciones de 24V, 2 de 5V, 5 de masa, 6 del relé auxiliar y 4 del optoacoplador auxiliar. En total se contabilizan 34 conexiones independientes, se decide por lo tanto emplear un conector de 35 pines de Ampseal. El sistema completo irá colocado en un lateral de la motocicleta, así que para lograr que los conectores se encuentren orientados hacia el suelo es necesario emplear conectores angulados. El modelo 1-776163-4 de TE Connectivity cumple todas estas necesidades, por lo que es el modelo escogido para su implementación en el sistema. Todas sus conexiones, así como la disposición por bloques de las mismas, se pueden ver claramente en la ilustración 20.

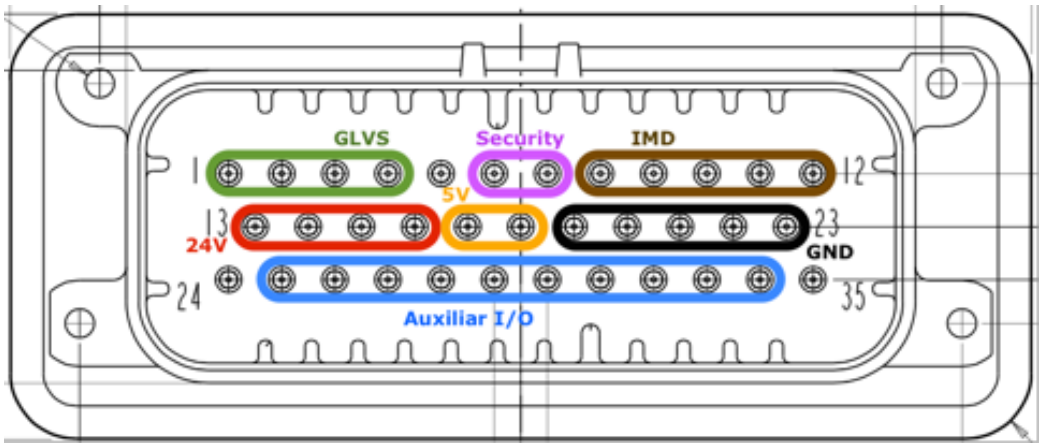

Ilustración 20. Distribución de pines en el conector de baja tensión

El conector de alta tensión tendrá un menor número de conexiones: 2 pines para el CANBus de alta tensión, 2 para el Key-Switch, 2 para el relé de la maniobra de seguridad, 4 para el relé auxiliar, 2 de entrada al optoacoplador auxiliar y 2 derivaciones a masa (podrían resultar útiles en algún momento). Al igual que en el conector de baja tensión, se empleará un conector angulado de Ampseal, concretamente se empleará el modelo 1-776267-1. La distribución de pines de este conector se muestra en la ilustración 21, donde se puede observar que se encuentran menos agrupados por bloques que en el conector de baja tensión.

Autor: **Javier Martínez Lahoz**
424.16.93

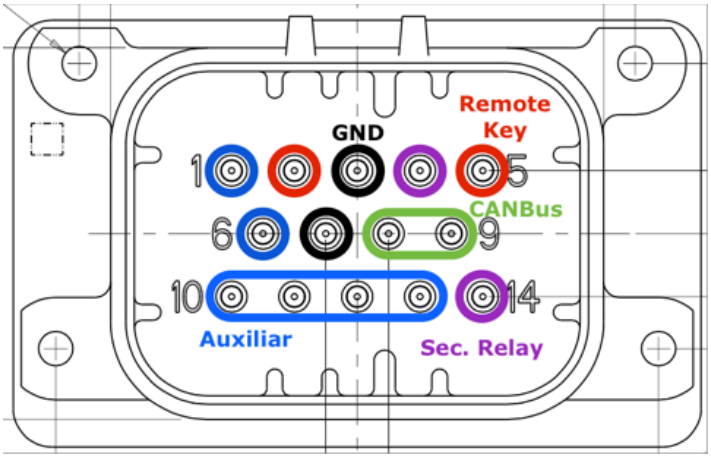

Ilustración 21. Distribución de pines en el conector de alta tensión

Para ambos conectores, la función individual de cada pin, el color original del cable correspondiente y su función se detalla en el Anexo 1, Manual de Conexionado.

Con esta decisión se completa la etapa de diseño del sistema, a continuación procederemos a fabricar un prototipo del sistema. Con este prototipo se comprobará el correcto funcionamiento de todos los subsistemas. En caso de que algún subsistema de problemas se procederá a su rediseño, en caso contrario se procederá a realizar una fabricación definitiva del sistema.

5.2.3. Diseño de la PCB

La placa electrónica que se ha diseñado y que se pretende fabricar cuenta con una importante cantidad de componentes interconectados entre sí. Resulta de vital importancia en estos casos tanto la disposición de los componentes como la estrategia de ruteo empleada. Además, existen una serie de restricciones que debemos tener en cuenta para el diseño del sistema.

5.2.3.1. Restricciones de diseño

En primer lugar debemos tener en cuenta el aislamiento entre los diferentes circuitos del sistema: ECU, GLVS y HVS. También resulta especialmente relevante cuidar el diseño de los circuitos de alta tensión del sistema.

Se emplearán dos conectores principales, más tres existentes en el Arduino. Todas las conexiones deben estar dispuestas en el mismo sentido, e irán apuntando hacia abajo, una vez el sistema se implemente en la motocicleta. Además todos los subsistemas tienen dependencias con el Arduino Yún y la fuente de alimentación.

Finalmente cabe destacar que los dos canales de comunicaciones CANBus tienen tres líneas comunes, el bus SPI, que se conectan con el Arduino Yún.

5.2.3.2. Decisiones de diseño

De acuerdo a las restricciones del diseño del apartado anterior, se toman una serie de decisiones para facilitar el diseño en general de la PCB y su fabricación. Estas decisiones afectarán al diseño de la placa en cuanto a la disposición final de los componentes y su ruteo.

En primer lugar se decide que todos los componentes se dispondrán por la cara superior de la placa. La utilización de ambas capas para disponer componentes solo es empleada en placas con una gran densidad de componentes y que pretenden minimizar el espacio empleado. Además el uso de ambas capas puede dificultar la soldadura de componentes y la fabricación de una envoltura protectora. En nuestro caso el espacio no debería ser un problema, por lo que solo se empleara una capa.

Los dos conectores principales, así como los conectores del Arduino, se colocarán en un mismo sentido, se dispondrán los puertos del Arduino de forma que sean de fácil acceso, orientados hacia el exterior. Así mismo se decide ubicar el Arduino entre ambos conectores. Como se trata del componente principal de la ECU se coloca en el centro para facilitar su conexión con cualquier subsistema de la placa.

Se procurará mantener todos los componentes de cada subsistema próximos entre sí. Se respetarán unas separaciones entre componentes que permitan la soldadura de forma manual; sin embargo, se intentará que cada subsistema se localice en un espacio determinado, para su fácil identificación y reparación. Los dos subsistemas gemelos, los módulos de comunicaciones de CANBus, se dispondrán próximos entre sí para minimizar la distancia del bus SPI.

Resulta bastante relevante que el interface visual de LEDs, así como el botón de reset, sean visibles y accesibles; en consecuencia, se decide disponerlos en un borde de la placa.

Las líneas de alimentación a 5V suministran energía a una gran cantidad de componentes y se decide llevar ambas líneas de forma próxima entre sí. Además se determina que estas líneas no se dirigirán a los pines de alimentación de los componentes, en su lugar se realizaran derivaciones para cada componente. Estas derivaciones serán de menor tamaño que las líneas principales y servirán para

facilitar el ruteo general, frente a llevar un bus que alimente todos los sistemas electrónicos.

Con el objetivo de reducir el coste de fabricación, se decide que el tamaño de pistas mínimo utilizado será de 0'4mm de ancho, pues un tamaño inferior a este aumentaría en gran medida el coste de fabricación de la placa. Además, emplear pistas de anchura muy reducida complica en gran medida el prototipado por ácido de la placa. Cuando sea posible se usará un tamaño de pista de 0'6mm, para reducir la impedancia de las mismas y mejorar el comportamiento del sistema. En el caso de las pistas de alimentación, o pistas por las que vaya a circular una gran corriente, se calculará el tamaño de pista necesario. La regla general que se empleará determina que la pista tendrá 0'7mm de anchura por cada amperio que vaya a circular por ella.

Finalmente, para facilitar el conexionado del Arduino se decide disponerlo en la cara superior de la placa. Irá colocado dispuesto "boca abajo" y conectado mediante clavijeros (Multicomp MC34735) a los pines de Arduino. Por lo tanto existirá un espacio útil debajo del mismo para disponer otros componentes, siempre que no sobrepasen la altura a la que se encontrará el Arduino.

5.2.3.3. Disposición de componentes

Con las restricciones del apartado anterior en mente, se procede a distribuir la disposición de componentes, y posteriormente pistas, sobre la placa. Este proceso de diseño se basa en la modificación constante. Cada vez que se disponga un componente o se diseñe una pista se revisarán los componentes y pistas circundantes y se realizarán modificaciones que faciliten el ruteo general. Por lo tanto la obtención del resultado final no es inmediato, es necesario seguir un proceso lento de diseño y rectificación, cumpliendo las decisiones previamente estipuladas.

Para comenzar con el diseño es necesario definir las posiciones de los componentes más restrictivos: los conectores y el Arduino. Como se ha justificado previamente, estos componentes irán en un mismo lateral de la placa, con el Arduino en medio. Pueden observarse resaltados en color azul claro en la ilustración 22, en la parte superior. El conector de GLVS se dispone a la izquierda, el conector de HVS a la derecha y el Arduino entre ambos. Los fusibles para las acometidas de potencia se colocará lo más próximos posibles al conector de baja tensión.

El subsistema de alimentación, resaltado en naranja en la ilustración 22, se opta por situarlo debajo del conector de GLVS, pues es de donde obtiene la tensión de

alimentación externa. Se deja un espacio entre este subsistema y el conector para permitir la distribución de pistas que surgirán del conector. Además, dentro de este subsistema se decide situar la fuente de alimentación lo más próxima posible al Arduino. Para facilitar esto, los supercondensadores se dispondrán cercanos a la parte exterior de la placa (a la izquierda y abajo).

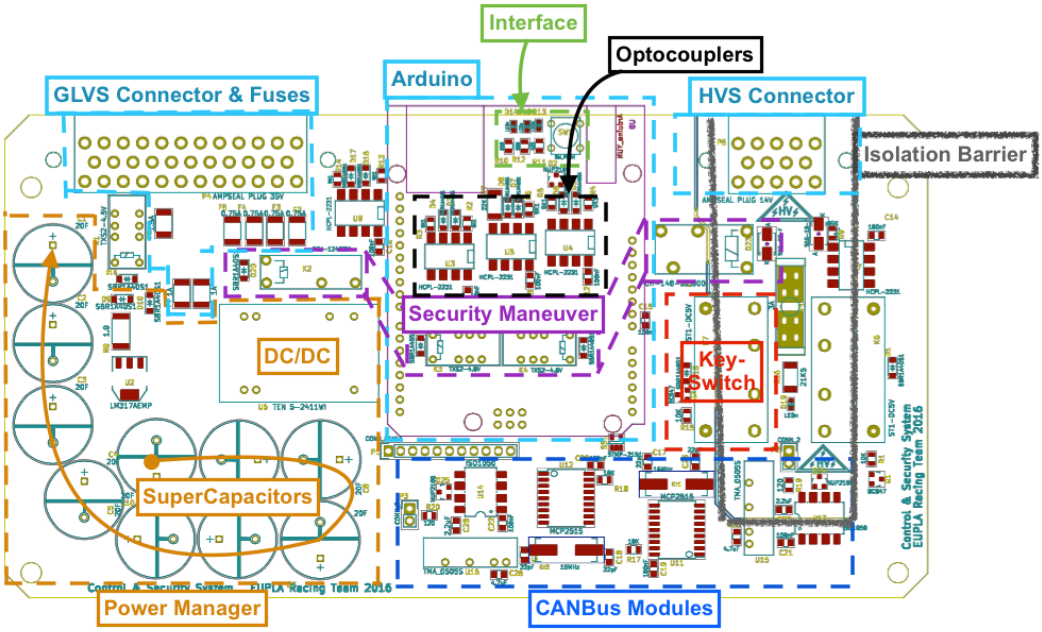

Ilustración 22 Disposición de componentes sobre la PCB

El espacio útil que queda debajo del Arduino lo dividiremos en cuatro partes. La parte superior, situada entre los puertos del propio Arduino, acogerá el interface manual compuesto por el botón de reset y los tres LEDs, resaltado en color verde.

Tras este espacio dispondremos los optoacopladores, así como todos sus componentes periféricos, resaltado en negro en la ilustración 22. Colocaremos los tres optoacopladores en paralelo, con las alineaciones y salidas hacia la parte inferior, y las entradas en la parte superior.

A continuación se deja un espacio tras los optoacopladores que servirá para llevar las pistas de alimentación. Debajo de este pequeño espacio se dispondrá una parte de la maniobra de seguridad, resaltado en morado en la ilustración. En concreto se colocarán los dos relés TXS2-4.5V controlados por el Arduino. Los otros dos relés de la maniobra de seguridad se dispondrán fuera de la zona cubierta por el Arduino. El relé del IMD se colocará a la izquierda, justo encima de la DC/DC. El relé del regulador

Autor: **Javier Martínez Lahoz**
424.16.93

se dispondrá a la derecha, junto con todos los demás elementos relacionados con el circuito HVS.

El último tramo disponible bajo el Arduino se deja libre de componentes. Esto se realiza para tener más espacio posteriormente por donde llevar pistas. En esta zona se encuentra además el puerto ICSP del Arduino, que emplearemos para las comunicaciones SPI. La existencia de este puerto, junto con las pistas adicionales dirigidas hacia los subsistemas de comunicación, que explicaremos a continuación, podrían provocar aquí una gran acumulación de pistas, por lo que decidimos no disponer componentes.

En la parte inferior de la placa, limitando con el subsistema de alimentación a la izquierda y con el Arduino y los circuitos de alta tensión por arriba, se encuentran ambos subsistemas de comunicaciones CANBus, resaltados en azul. Cada uno de estos subsistemas se puede dividir en dos submódulos: el controlador con sus periféricos y el transceptor con sus periféricos. Los módulos de los controladores se situarán en el centro del subsistema, dispuestos de forma que las conexiones del bus SPI queden hacia el centro del subsistema, mientras que las salidas queden hacia los laterales. Al disponerse de forma inversa, los circuitos osciladores se encuentran en posiciones diferentes. El oscilador derecho se encuentra debajo del controlador, mientras que el izquierdo se encuentra encima, esto se realiza así para minimizar la distancia de las pistas entre el oscilador y el controlador.

En los laterales se encuentran los submódulos de los transceptores. El transceptor del bus GLVS está dispuesto a continuación del controlador, con la fuente de alimentación debajo del transceptor, justo al lado del oscilador. El módulo del transceptor del circuito de alta tensión se encuentra rotado 90º respecto al controlador, con las salidas hacia arriba y las entradas debajo. De esta forma se logra que las mitades de salida del transceptor y de la fuente de alimentación se encuentren en la zona de alta tensión, mientras que las entradas se encuentran en la zona aislada.

Entre el modulo CANBus y el espacio libre de debajo del Arduino Yún colocamos el Smart-Switch. De esta forma queda a medio camino entre el Arduino, que le dará la señal de activación, y los componentes CANBus, a los que alimentará.

La parte derecha del circuito es la zona de alta tensión. Se ha dispuesto una separación física entre la zona aislada y la zona de alta tensión, representada en la

ilustración 22 con líneas grises. Los componentes que interactúen con ambas zonas, se encuentran dispuestos sobre dicha línea, sin embargo ningún elemento de la zona aislada se encuentra al otro lado de la barrera de aislamiento, y viceversa. Esta misma política se empleara a la hora de diseñar las pistas, ninguna pista atravesará la mencionada barrera.

En el límite entre ambas zonas podemos encontrar dos elementos ya explicados, el relé de seguridad del regulador, y el módulo del transceptor de CANBus de HV. Además colocamos en esta zona el relé del Key-Switch. Este relé se encuentra ocupando el espacio intermedio entre el relé de seguridad y el módulo de comunicaciones. El transistor de disparo, la resistencia de polarización y el diodo de protección se dispondrán próximos entre sí y al relé.

En el margen derecho de la zona de aislamiento se encuentran los módulos auxiliares de alta tensión. Los módulos auxiliares de baja tensión se encuentran repartidos en las zonas que han quedado libres a la izquierda. Concretamente podemos encontrar al relé auxiliar y al optoacoplador auxiliar debajo y a la derecha del conector de baja tensión respectivamente.

5.2.3.4. Diseño de las pistas

Tras la disposición de todos los componentes se procederá a realizar las conexiones entre ellos. Como la primera prueba que se realizará será un prototipo fabricado mediante atacado por ácidos de forma casera y no se metalizarán las vías, resulta necesario tener en cuenta algunas consideraciones.

Las conexiones de componentes que atraviesen a placa, *Trough-Hole*, solo se podrán soldar por la capa inferior ya que los agujeros no estarán metalizados. Por lo tanto las pistas de dichos componentes deberán partir de la capa inferior. Además tendremos en cuenta que el método de realización de vías que emplearemos se basa en el uso de un cable unifilar para conectar la capa superior con la inferior a través de un agujero. Este cable ocupa un determinado espacio en ambas capas, y por lo tanto no podremos colocar vías debajo de componentes. Para facilitar el ruteo de las pistas seguiremos dos estrategias, además de las mencionadas anteriormente.

En primer lugar procuraremos realizar las pistas de la capa superior verticales, y las pistas de la capa inferior horizontales. De esta forma evitaremos en gran medida cruces de dos pistas en una misma capa, pues todas las pistas de una misma capa serán paralelas. En segundo lugar decidimos que comenzaremos realizando las

conexiones de menor recorrido, y finalizaremos con las de mayor recorrido. De esta forma se asegura que las conexiones de corto recorrido sean lo más simples posibles, puesto que las de gran recorrido pueden valorar diversos caminos por los que llegar a su destino.

Como el conector Ethernet del Arduino es metálico, se evita rutear por la parte superior de la placa en la zona que el conector estará. También se evita rutear debajo del conector USB auxiliar del Arduino por el mismo motivo.

Tras estas consideraciones se procede de rutear de la forma previamente mencionada la placa y, tras diversos intentos desechados, se obtiene una solución que se considera válida. Esta solución cumple con todas las restricciones previamente comentadas, si bien algunos componentes han sido ligeramente desplazados para facilitar el ruteo y la posterior soldadura.

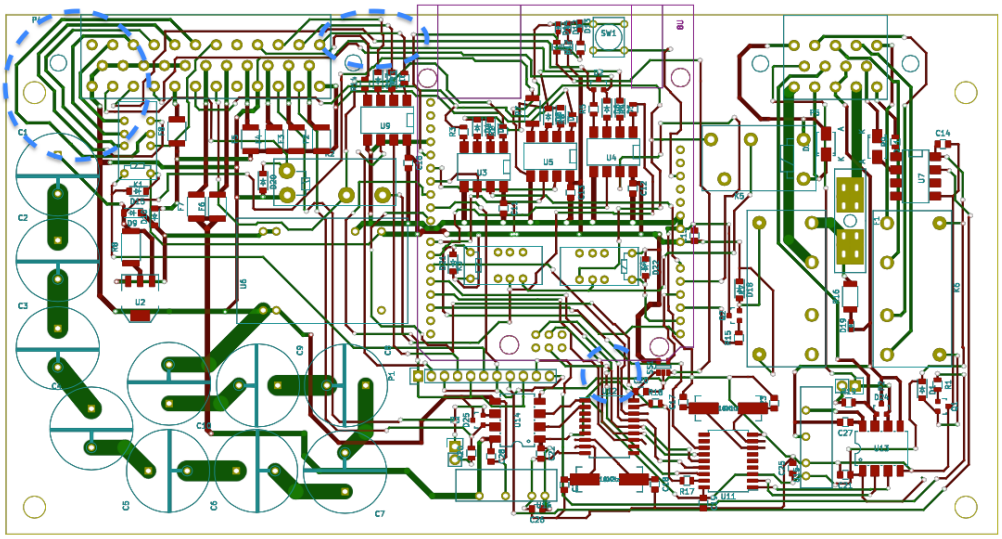

Ilustración 23. Diseño final de la PCB prototipo

Resulta importante destacar que las restricciones que más han afectado al diseño de la placa son las debidas a la fabricación prototipo mediante ácidos. Existen algunos punto críticos en esta placa debido a estas restricciones que se pueden observar en la ilustración 23. En esta ilustración se puede observar una aglomeración de pistas y vías en los alrededores del conector de baja tensión, especialmente a la derecha del mismo. También se observa la gran cantidad de pistas que surgen del conector del baja tensión por la capa inferior, especialmente destacable en el lateral izquierdo. Finalmente podemos observar una gran cantidad de vías en el punto medio

entre el Arduino y el subsistema de CANBus para lograr todas las conexiones necesarias.

También debemos destacar que se ha optado por alimentar los módulos auxiliares de alta tensión a través del Smart-Switch. Esta modificación simplificaba en gran medida el diseño de pistas, por lo que se decidió llevarlo a cabo.

5.3. PROTOTIPADO

5.3.1. Fabricación

La fabricación de placas de circuito impreso por ácidos se basa en cuatro pasos. Se parte de una placa de prototipado, con cobre y material fotosensible, y de una plantilla del esquema deseado plasmada en acetatos. Se introducirá la placa de prototipado, entre las plantillas, en una insoladora durante unos siete u ocho minutos (en función de la placa). La luz ultravioleta de la insoladora eliminará el material fotosensible de la placa en las zonas deseadas. Se procurará que durante este proceso la placa no sea expuesta a ninguna fuente de luz fuerte. También resulta importante que las plantillas de ambas caras queden adecuadamente alineadas, o de lo contrario no coincidirán las vías entre ambas caras.

A continuación se procederá al revelado de la placa, este se realiza mediante un baño de sosa cáustica. Este proceso también se lleva a cabo con una luz de baja intensidad para evitar que se distorsione el diseño. Se mantendrá el baño revelador hasta que el diseño sea completamente visible en la placa. El diseño debe estar plasmado bien definida en la placa o de lo contrario el siguiente paso no funcionará correctamente. En ese momento la placa se sacará con cuidado del baño revelador y se aclarará con abundante agua. Tras este paso ya podemos encender la luz.

El tercer paso de la fabricación es la eliminación del cobre sobrante, esto se consigue al sumergir la placa en una solución ácida, mezcla de ácido clorhídrico y peróxido de hidrógeno. Dicha solución atacará al cobre descubierto y lo eliminará de la placa. La duración de este baño depende de la placa y las proporciones de ácidos; retirar la placa demasiado pronto puede resultar en pistas cortocircuitadas, al no haberse retirado todo el ácido correspondiente, mientras que un atacado demasiado duradero deteriora las pistas, llegando a cortarlas completamente y siendo necesaria su reparación. La velocidad de la reacción puede ajustarse mediante la adición de un

componente neutro a la solución (agua destilada) para reducir la velocidad, en caso deseado también se puede acelerar la reacción al añadir más peróxido de hidrógeno. En cuanto todas las pistas estén claramente definidas, retiraremos la placa del baño de atacado y la lavaremos con abundante agua, pues el ácido es muy corrosivo en la piel.

Finalmente debemos eliminar de forma controlada el material fotosensible restante y barnizar la placa para evitar su oxidación. La eliminación de fotosensible se realizará frotando la placa con un papel impregnado en alcohol de 96º. A continuación se barnizara la placa con laca fundible, de esta forma se protegerá el cobre a la vez que se permitirá la soldadura de componentes sobre el mismo.

5.3.2. Montaje y comprobación

Una vez la placa se encuentra fabricada y con todas las pistas en un estado adecuado podemos proceder a realizar la implementación de los componentes deseados en la placa para probarlos. Este proceso comienza por taladrar todos los agujeros de la placa, entre los que se incluyen los agujeros de las vías, los agujeros de los componentes, y otros agujeros auxiliares. Se realizan todos los agujeros antes de comenzar a soldar, pues los componentes molestarían a la broca de taladrado y podrían resultar dañados durante el proceso.

Al tratarse de un prototipo, y tener que metalizarse las vías manualmente, se decide no montar todos los componentes de la placa. El prototipo se empleará para comprobar el correcto funcionamiento de los subsistemas individualmente y sus conexiones con el Arduino. Por ejemplo tan solo se montará uno de los cinco optoacopladores de la placa, de esta forma aseguraremos su correcto funcionamiento, a la vez que se reduce en gran medida el tiempo empleado entre la soldadura y las comprobaciones.

El montaje final del prototipado incluirá las conexiones para el Arduino, la maniobra de seguridad completa, el módulo completo de alimentación, un módulo de comunicaciones, el módulo del Key-Switch y un módulo de entradas aisladas (un optoacoplador).

Se ha decidido no incluir en el prototipo el montaje de los conectores, ya que su gran cantidad de pines dificultarían en gran medida su desmontaje, y no se consideran necesarios para realizar las pruebas deseadas. Durante todo el proceso de

comprobación se van realizando las metalizaciones de vías necesarias, si bien dicha acción se encuentra omitida durante todo el proceso. También se omite la soldadura de los condensadores de desacoplo, pues no suponen ningún problema ni se realizarán comprobaciones de los mismo.

Al comenzar los taladrados comprobamos que la placa se encuentra bien alineada. En caso de que las guías no estuvieran en su sitio en el apartado anterior, se notará en este punto.

5.3.2.1. Power Manager

El proceso de comprobación comienza con la soldadura del Power Manager, en primer lugar se implementa la fuente de alimentación TEN 5-2411WI junto con su correspondiente diodo de protección frente a corriente inversa. Comprobamos con el osciloscopio que con una entrada de tensión de 24Vcc obtenemos una salida de 5Vcc bastante estables. Realizamos una pequeña prueba de rango de entrada, subiendo la tensión de entrada hasta 28Vcc y posteriormente bajándola hasta los 10Vcc. Comprobamos que la salida en tensión permanece siempre estable a 5Vcc. El ruido existente a la salida concuerda con los datos proporcionados por el fabricante: frecuencia de 20MHz y amplitud siempre inferior a 80mV. Este ruido será filtrado por los condensadores de desacoplo y no provocará problemas en la alimentación de la electrónica. Finalmente comprobamos que con una tensión de entrada inferior a 9Vcc la fuente de alimentación comienza a comportarse de forma intermitente hasta finalmente apagarse.

Una vez comprobado que la fuente funciona correctamente de acuerdo a las características del Datasheet, procedemos a la implementación en paralelo del SAI de supercondensadores. Comenzamos soldando el regulador de tensión LM317, la resistencia de limitación y el diodo correspondiente. Los supercondensadores se soldarán en último lugar debido a su gran tamaño, ya que de soldarlos en primer lugar molestarían a la soldadura del resto de componentes.

Para la primera prueba de los supercondensadores, la prueba de carga, se mide la tensión entre el extremo positivo del primer condensador y la masa de referencia mientras se alimenta la placa. Se observa que la tensión entre los extremos de los supercondensadores aumenta de forma lineal gracias al regulador de tensión. Además observamos que la corriente entregada por la fuente de alimentación, la empleada para alimentar el sistema, es de aproximadamente 0'7A constantes. Damos por válido

el circuito de carga constante formado por el LM317 y la resistencia de limitación. La fuente de alimentación deja de entregar corriente cuando los condensadores alcanzan su carga máxima. En este caso la carga máxima de los condensadores es de 22Vcc, esto se debe a las caídas de tensión existentes en el diodo de protección y el regulador de tensión.

Una vez los condensadores se encuentran a cargados a la carga nominal se procede a la prueba de descarga. Al cortar la alimentación externa se observa como la fuente de alimentación sigue en funcionamiento estable, su tensión de entrada es ahora proporcionada por los supercondensadores. Para lograr la descarga y comprobar su funcionamiento empleamos una resistencia de 50Ohm conectada entre los terminales de salida de la fuente de alimentación, de esta forma generamos un consumo de 100mA. Una resistencia demasiado grande implicaría una corriente muy pequeña y los condensadores apenas se descargarían, mientras que una resistencia demasiado pequeña generaría demasiado calor y podría quemarse. La descarga de los supercondensadores se prolonga durante varios minutos, lo que evidencia la gran capacidad de almacenamiento de energía de los mismos. Cuando la tensión de los supercondensadores alcanza los 9Vcc la fuente de alimentación comienza a actuar de forma pulsante, hasta que finalmente se apaga transcurridos unos segundos. Los supercondensadores quedan por lo tanto cargados a unos 9Vcc. Esta carga se mantendrá durante un prolongado periodo de tiempo, ya que no existe ningún consumo una vez la fuente de alimentación se desconecta más alla de la resistencia en paralelo equivalente de los condensadores que tiende a descargarlos, si bien en el caso de los supercondensadores esta resistencia es de un valor muy elevado y tarda mucho en descargar la energía acumulada.

Con el subsistema de alimentación completamente probado, podemos comenzar con las pruebas de otros subsistemas.

5.3.2.2. ECU

Para probar el Arduino es necesario soldar los clavijeros que lo conectarán con la placa. La soldadura de estos clavijeros requiere mantener una buena perpendicularidad respecto a la placa y una buena alineación entre ellos; de lo contrario el Arduino podría no encajar de forma idónea. Para asegurar una alineación perfecta se introducen previamente los clavijeros en los pines del Arduino. Al soldar los clavijeros a la placa mientras se encuentran conectados al Arduino se asegura la alineación vertical y la posición exacta de cada pin.

Comprobamos que los clavijero encajan perfectamente en los agujeros de la placa. Una vez soldados los clavijeros, y con el Arduino Yún en su sitio, comprobamos las separaciones entre los conectores Ethernet y USB del Arduino Yún y la placa. Ambos conectores cuentan con una ligera separación respecto a la placa, dos o tres milímetro aproximadamente. Esta separación evita que pueda producir cortocircuitos en la placa debido a estos conectores, sin embargo no es una separación lo suficiente mente grande como para albergar componentes debajo; como si ocurre en otras partes de debajo de la placa del Arduino.

La primera prueba realizada con el Arduino Yún se basa simplemente en su encendido mediante la fuente de alimentación. Comprobamos que el encendido es correcto, por lo que se observa su funcionamiento, de esta forma nos aseguraremos de que la alimentación es completamente estable y no generará glitches en la electrónica. Para ello cargaremos un programa ejemplo en el Arduino que provoca un parpadeo en el LED integrado en el pin 13, llamado *Blink*. Comprobamos que el funcionamiento es completamente satisfactorio, por lo que damos por finalizada la comprobación del Arduino.

5.3.2.3. Security Maneuver

A continuación procedemos a soldar y comprobar la maniobra de seguridad, para ello soldamos el relé del IMD, los dos relés del Arduino, y el relé del regulador; todos ellos se sueldan con su correspondiente diodo de protección. Todas las soldaduras se realizarán con el Arduino Yún desconectado de la placa, para evitar posibles deterioros que se pudieran dar al tocarlo con el soldador accidentalmente. Una vez se han realizado las soldaduras de los relés se procede a comprobar la activación de estos. Comenzaremos con la comprobación de los relés de la ECU, para esto se carga un programa en el Arduino que los active y desactive cada dos segundos, similar al ejemplo Blink, pero en los pines que controlan los relés (A4 y A5). Observamos sin embargo que los relés TXS2-4.5V de la ECU no se activan, previamente se habían realizado pruebas en el exterior en las que el Arduino era capaz de activar dichos relés sin incidentes.

Se llevan a cabo diversas pruebas de activación y desactivación sobre los relés, tanto en la placa como en el exterior, con el objetivo de encontrar la fuente del problema. También se realizan mediciones con instrumental de laboratorio sobre la bobina de los relés. Finalmente se encuentra la fuente de dicho problema tras una prueba fuera de la placa: los relés solo se activan en uno de los sentidos. Esto puede

deberse a que los relés cuentan internamente con el diodo de protección. Como en la placa se encuentran conectados de forma que el diodo interno se encuentra en contrasentido del diodo externo la corriente nunca circula por la bobina, y por lo tanto nunca se activa el relé.

Para solucionar este problema se cortan con cuidado las pistas que conducen hasta la bobina del relé, y se reconectan de la forma adecuada mediante un par de cables y estaño. Se realiza esta operación en ambos relés con cuidado de no deteriorar ningún otro componente ni afectar al resto de pistas de la placa. Tras cruzar las conexiones de ambos relés observamos que se comportan de la forma esperada al emplear el programa de Arduino previamente mencionado.

Para comprobar el relé del IMD debemos alimentar ese sistema y controlar si el relé correspondiente se activa o se desactiva. El control del relé se logra uniendo o separando los dos cables que sirven para detectar fallos en el plano de masa. Estos cables se conectaron a un interruptor manual para poder llevara a cabo pruebas de conexión y desconexión rápidamente. Observamos que el relé funciona correctamente, abriéndose al separar los cables (fallo en el plano de masa) y cerrándose al unirlos. También observamos que el tiempo de respuesta del IMD es muy alto, transcurren aproximadamente 12 segundos entre que se produce el cortocircuito y el relé abre el circuito. Tras repasar el manual técnico del medidor de aislamiento observamos que este comportamiento es normal en esta versión del componente. Esto se debe a que el medidor solo pasa cambia al modo de error, o aceptación, si ha recibido diez tomas erróneas, o validas, lo que reduce el tiempo de respuesta sustancialmente.

Para comprobar el relé del regulador debemos emplear dicho componente. Programamos una de sus salidas digitales para trabajar a 48Vcc y empleamos un interruptor externo para controlar dicha salida. Tras conectar el regulador a la ECU y al interruptor, procedemos a activar y desactivar el interruptor y observamos el comportamiento del relé de la maniobra de seguridad. Se observa que funciona perfectamente y sin ningún tipo de retraso en la señal. Damos por concluido por lo tanto la comprobación de la maniobra de seguridad, siendo necesario modificar las conexiones de todos los relés TXS2-4.5V en los esquemas electrónicos.

5.3.2.4. Key-Switch

Una vez todos los relés de la maniobra de seguridad funcionan correctamente, procedemos a comprobar el otro relé de la placa controlado por el Arduino, el ST1-

DC5V-F que actúa como Key-Switch. La experiencia reciente con los relés de la maniobra de seguridad TXS2-4.5V nos empuja a realizar una comprobación previa a la soldadura de los componentes: si el relé funciona en ambos sentido o solo en uno. Empleamos una fuente de tensión fijada a 5Vcc para la comprobación y observamos que el relé solo se activa en uno de los dos sentidos posibles, por lo que también esta protegido internamente. Comprobamos que el modo en el que está conectado dicho relé en la placa es el incorrecto y no activaría el relé, por lo que nuevamente adaptaremos las pistas involucradas cortándolas y rehaciéndolas con cables al igual que en el caso anterior.

Para la comprobación en la placa comenzaremos soldando los componentes auxiliares de menor tamaño: el transistor BC847B, la resistencia de puerta del transistor y el diodo de protección. Por último se suelda el relé a la placa y se carga un programa de Arduino para probar este relé similar al empleado anteriormente, pero en este caso empleando el pin A3. Comprobamos que el relé no se activa con dicho programa, por lo que nuevamente debemos averiguar a que se debe.

Lo primero que realizamos es medir la tensión existente entre los bornes del relé. Observamos que los valores de tensión son los adecuados, por lo que debemos realizar más medidas. Para la siguiente medida desoldaremos el relé y empleamos un medidor de corriente de laboratorio para comprobar la corriente que circula por el circuito. El resultado obtenido al cargar el programa resulta sorprendente. Se obtiene que la corriente máxima que atraviesa el circuito es de 3mA, frente a los 50mA calculados o los 45mA necesarios para activar el relé. La solución a este problema pasa por modificar el transistor o la resistencia de puerta, en nuestro caso se ha optado por reducir la resistencia de puerta. Esto se debe a que, gracias a la estandarización, contamos con diversos modelos de resistencias con el mismo encapsulado y podemos realizar modificaciones fácilmente. Emplearemos una resistencia de menor valor para aumentar la corriente de puerta del transistor, y por consiguiente aumentar la corriente que atraviese la bobina del relé.

La nueva resistencia de puerta se calculará de acuerdo con la corriente actual y la corriente deseada. La corriente obtenida es aproximadamente 20 veces menor a la corriente deseada. De acuerdo a las fórmula de funcionamiento de un transistor, mostrada en las ecuación 3 y 4, la corriente colector-emisor de un transistor varía de forma proporcional a la corriente base-emisor. Por lo tanto si queremos que la corriente colector-emisor aumente un 200%, debemos reducir la resistencia

proporcionalmente. De esta forma obtenemos que la resistencia ideal es de 500Ohm, sin embargo la resistencia existente en nuestro inventario más próxima al valor deseado es de 422Ohm, se trata de la resistencia empleada para limitar la corriente del LED azul, apartado 5.2.2.1. Procedemos a retirar de la placa la resistencia de 10K calculada inicialmente y se coloca en su lugar la nueva resistencia de 422Ohm.

Tras la sustitución de dicha resistencia se vuelve a cargar el programa de comprobación de relés, y en este caso el relé funciona perfectamente, se detecta un paso de corriente de 55mA por lo que nos apuntamos las modificaciones que deberemos efectuar para la versión final de la placa: el cambio de las pistas del relé y el cambio de la resistencia de polarización.

5.3.2.5. GLVS Measures

La siguiente comprobación que realizaremos estará enfocada en los optoacopladores y su correcto funcionamiento. Para ello soldaremos a la placa un optoacoplador, junto con sus correspondientes resistencias y diodos. En concreto se implementará el optoacoplador U5 junto con sus dos resistencias, sus dos diodos y su condensador de desacoplo. Escogemos este optoacoplador por las señales de las que se hace cargo, las señales del sistema de alimentación, ya implementado. Sus dos funciones distintas implican resistencias distintas y por lo tanto comportamientos distintos, por ello se decide que este optoacoplador es el ideal para llevar a cabo las pruebas.

Tras soldar todos los componentes realizamos una prueba para comprobar el funcionamiento de ambos canales. Para ello cargaremos un programa a Arduino que nos indique que si lee alguna señal en los pines D12 o D6, que son aquellos asociados al optoacoplador que se ha implementado. Conectamos la tensión externa y observamos que el pin D12 se activa, pues es el que indica la existencia de alimentación externa. El pin D6 permanece desactivado durante aproximadamente un minuto, mientras se cargan los supercondensadores, tras este tiempo dicho pin se activa indicando que los condensadores se encuentran a media carga. Comprobamos con un medidor de tensión que el canal se activa con una tensión de 14Vcc, una tensión ligeramente menor a la calculada de 16Vcc. Esta diferencia entre las tensiones se debe al comportamiento del optoacoplador, el fabricante no índica corriente mínima de activación del optoacoplador, solo típica, por lo que es lógico que al principio el diodo se pueda activar con una corriente inferior a la nominal, debido a la degradación lógica de estos componentes. A continuación se apaga la fuente de alimentación

externa y se comprueba el comportamiento del optoacoplador. El pin D12 se apaga inmediatamente, y el pin D6 se apaga cierto tiempo después, cuando vez los supercondensadores se encuentran a una tensión inferior a 14Vcc.

El optoacoplador y la toma de medidas del subsistema de alimentación funcionan perfectamente, por lo que podemos proceder a realizar el resto de comprobaciones de la placa.

5.3.2.6. CANBus

Una vez comprobados todos los módulos de entrada y salida, se procede a comprobar los módulos de comunicaciones CANBus. Comprobaremos uno de los módulos y asumiremos que el otro funcionará de forma similar, ya que su conexionado es el mismo. También implementaremos y probaremos el Smart-Switch que alimenta los módulos de comunicaciones y los módulos auxiliares de alta tensión.

En primer lugar se sueldan todos los componentes, comenzando nuevamente de más pequeños a más grandes, y una vez finalizada la soldadura se procede a las comprobaciones. En primer lugar se comprueba que el Smart-Switch funciona correctamente empleando nuevamente un programa de activaciones y desactivaciones cíclicas del pin de control, en este caso el pin A2. Se comprueba mediante un medidor de tensión que el voltaje de alimentación de los integrados de CANBUs es correcto. Se procede a realizar las comprobaciones necesarias de las comunicaciones propiamente dichas. Comenzaremos cargando en el Arduino Yún un programa de ejemplo de envío de mensajes de la librería CANBus. Se empleará otro Arduino, con un *shield* comercial de CANBus, para recibir los mensajes, para ello se le cargará con un programa de ejemplo de lectura de CANBus para comprobar que la comunicación funciona correctamente.

Se comprueba que el Arduino receptor no recibe los mensajes, y que el controlador del Arduino Yún, el emisor, esta devolviendo un mensaje de error de envío. Por lo tanto que las comunicaciones no son correctas, el siguiente paso es buscar el fallo que lo esta produciendo.

Las primeras comprobaciones que se realizan son comprobaciones de continuidad y tensiones con aparatos de medición de laboratorio. También se realizan mediciones en los canales de comunicaciones con el osciloscopio. El fallo sin embargo se encuentra tras revisar los manuales técnicos del controlador y del transceptor. Se llega a la conclusión de que se encuentran mal conectados entre sí. La conexión

realizada entre ambos era Tx-Rx y Rx-Tx, similar a una conexión serie; mientras que la conexión que debe realizarse es Tx-Tx y Rx-Rx. Estas pistas se modifican de la misma forma que se corrigieron previamente las conexiones de los relés. Una vez se encuentran modificadas se vuelve a realizar la prueba de comunicaciones. En esta ocasión la comunicación funciona correctamente, todos los mensajes se reciben correctamente, y se da por válido el subsistema de comunicaciones, pendiente de modificar en los esquemas el conexionado.

5.3.3. Fin del prototipado

Tras todas las comprobaciones realizadas damos por concluidas las pruebas sobre el primer diseño. El siguiente paso relacionado consistirá en adquirir la nueva PCB mediante una fabricación profesional. En primer lugar se corregirán en los esquemas todos los errores encontrados en las verificaciones, y posteriormente se realizará una nueva versión no restringida a los inconvenientes del prototipado con ácido. En este nuevo diseño las vías y la capa por la que soldar los componentes ya no supondrán ninguna problema en el diseño, debido al método de metalización de vías profesional. El nuevo diseño, mostrado en la ilustración 24 y en el plano 424.16.93.06, resulta similar al anterior, si bien el trazado de las pistas es más directo. Se puede observar que las aglomeraciones de pistas y vías de la versión anterior han desaparecido en su mayor parte.

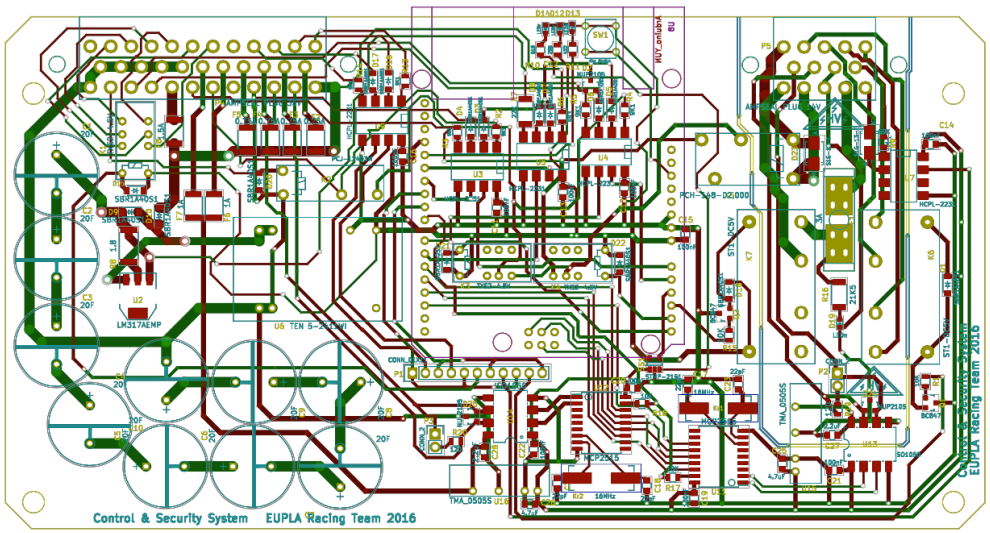

Ilustración 24 Diseño definitivo de la PCB

Tras las comprobaciones y modificaciones realizadas se espera que el diseño corregido no contenga errores de conexionado ni funcionamiento. En caso de encontrar algún error o ser necesaria alguna modificación se afrontará de la forma que se consideré oportuno en cada momento.

Se realiza un pedido de fabricación profesional de placa de circuito impreso con el diseño definitivo. A la espera de recibir la PCB, se comienza con el desarrollo del software de control.

5.4. DISEÑO SOFTWARE

Como ya se ha visto previamente la ECU cuenta con varios sistemas de comunicaciones. La implementación hardware ya se ha resuelto previamente, por lo tanto es necesario determinar los protocolos y los mensajes concretos que se emplearán en cada caso. La programación posterior se realizará de acuerdo a las comunicaciones que se establezcan. Además al disponer de un microprocesador y un microcontrolador independientes con funciones y características diferentes, será necesario desarrollar dos programas completamente diferentes.

5.4.1. Comunicaciones

5.4.1.1. Comunicaciones Motocicleta - CANBus

CANOpen es uno de los protocolos basados en CANBus más extendidos, sin embargo su implementación completa es muy costosa, debido al diccionario de objetos que se debe realizar. Dada la cantidad de tiempo necesaria para generar y gestionar correctamente dicho diccionario, y valorando los beneficios que aportará en el prototipo, se decide no implementar el diccionario de objetos. Sin embargo se empleará el mismo sistema de mensajes de transmisión de datos (PDOs) que CANOpen.

Una vez decidido que no se emplearán diccionarios de objetos es necesario determinar qué información circulará por la red; así como la composición de los mensajes, que se ha ido modificando y ajustando durante todo el desarrollo del proyecto. A continuación se mostrarán y analizarán los mensajes definitivos. Para definir la información y los mensajes que circularán por la red primero es necesario conocer que información generará cada sistema, y cual de esa información resulta

relevante para su análisis. Existen cinco sistemas inteligentes en la moto, y todos ellos generarán información relevante que deberán publicar en el bus.

La ECU obtiene información vital de la maniobra de seguridad. Además es el único sistema conectado tanto al bus CAN de baja tensión como al de alta, por sirve de puente entre los dos canales de comunicaciones. La ECU además enviará la información que deberá ser mostrada en el display.

El regulador obtiene una gran cantidad de información del motor, ya que es el elemento encargado de su control, también obtiene alguna información de la batería, al encontrarse directamente conectados. Destacan principalmente el voltaje y la corriente de batería, la temperatura del motor y del regulador, la velocidad del motor y de la rueda, y el par generado en la rueda. Con estas informaciones se puede conocer el estado del bus de potencia y su rendimiento.

El BMS obtiene una enorme cantidad de información de la batería, tanto de las celdas individualmente como de la batería en su conjunto. Respecto a celdas individuales y sistemas de balanceo los datos más relevantes son el voltaje máximo, mínimo y medio de las celdas, así como la temperatura máxima y media de celda; y temperatura máxima y media de circuitos de balanceo. Entre los valores de la batería que resultan importantes a grandes rasgos podemos distinguir el voltaje total, la corriente suministrada, el nivel de carga, el estado final de la maniobra de seguridad y los posibles fallos detectados en el BMS.

El detector de pilotos situado en la parte posterior cuenta con dos tipos de sensores que consiguen información relevante. En primer lugar cuenta con tres sensores de proximidad para detectar otros pilotos en las cercanías de la moto, y además una IMU de tres ejes, que puede servir para saber las aceleraciones máximas que sufre la motocicleta.

El display implementado en la motocicleta cuenta con dos botones que pueden ser accionados por el piloto, y que se derivarán al bus para poder realizar modificaciones.

Los mensajes estándar de CANBus están limitados a 8 bytes de datos, por lo que debemos agrupar la información de forma que nunca se supere dicho límite. Como se puede observar en la tabla 4 emplearemos ocho mensajes. Como los mensajes del regulador son publicados en el bus de información de alta tensión, la ECU reenvía alguna de esa información por el bus de baja tensión en su mensaje. Podemos

observar en la tabla 4 que la mayoría de la información se codifica en un solo byte. Esto se realiza para optimizar el traspaso de información. Sin embargo, la tensión y la corriente de la batería se codifican en dos bytes para aumentar la precisión de las muestras. La información emitida por el regulador no puede ser reajustada en tamaño, por lo que se distribuyen de forma que pueda ser transmitida en el menor numero de mensajes, en este caso dos mensajes.

ID	Asunto	Emisor	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x101	Mensaje ECU-REG	ECU	Mot_Temp	Vel_Value	Torque	Sec Status HV Status	Errors			
0x150	Mensaje ECU-Display	ECU	Vel_Value	Mot_Temp	Batt_Volt	SOC	HV_Enable Run_Allow			
0x201	Mensaje REG 1	REG	HS_Temp	Mot_Temp		Current		Batt_Volt		
0x202	Mensaje REG 2	REG	Vel RPM				Vel_value		Torque	
0x301	Mensaje BMS Gen	BMS	Total_V		Current		SOC	Bms_Fail TSMS Balance_Stat		
0x302	Mensaje BMS VT	BMS	Max_V	Min_V	Avg_V	Max_Cell_T	Avg_Cell_T	Max_Bal_T	Avg_Bal_T	
0x401	Mensaje TRC	TRC	Detector 1	Detector 2	Detector 3	IMU X	IMU Y	IMU Z	Errors	
0x501	Mensaje DIS	DIS	Buttons DisplayData							

Tabla 4. Mensajes de CANBus

5.4.1.2. Comunicaciones internas - NMEA0183

Para lograr que el microcontrolador y el microprocesador del Arduino Yún funcionen correctamente entre sí es necesario establecer un protocolo de comunicaciones entre ellos que nos permita transmitir información. En este caso se realizarán entre dos componentes muy próximos físicamente, por lo que no es necesario emplear un protocolo excesivamente robusto y fijo, en su lugar optamos por uno simple con una implementación más sencilla. El medio físico existente para realizar la comunicación es un puerto serie conectado a ambos dispositivos.

El protocolo que emplearemos entre ambos integrados es el NMEA0183. Este protocolo no limita la longitud de las tramas de datos; por lo tanto definiremos el concepto del mensaje, e incluiremos en el mismo toda la información que consideremos oportuna.

Fijamos cuatro mensajes que discurran de forma cíclica desde el microcontrolador al microprocesador: mensaje de estado general, de estado extendido, de relés accionados y de telemetría. No se establecen mensajes cíclicos

enviados desde el microprocesador al microcontrolador; por lo tanto se decide que, en el caso de querer enviar algún mensaje extraordinario en este sentido, no se empleará una estructuración fija del mensaje. Resultará necesario haber preparado previamente el microcontrolador para reaccionar de la forma adecuada ante la recepción de cada mensaje del microprocesador.

El mensaje de telemetría contendrá todos los datos que consideremos oportuno monitorizar de la motocicleta. Se registrarán datos del estado general de la batería (voltaje, corriente y SOC), así como datos de celdas específicas (celda más caliente, más cargada y menos cargada), datos del motor recibidos desde el regulador (Tª, Par, RPMs y Km/h) y del propio regulador (corriente).

Además existen otros mensajes que se han ido empleando en pruebas y que se encuentran implementados en la versión final, pero que no se encuentran activos. Por ejemplo existe un mensaje implementado de Heartbeats omitido en la versión final.

5.4.1.3. Comunicaciones hacia el exterior - TCP

Las comunicaciones que se establecerán entre el microprocesador y el usuario se realizarán de forma inalámbrica mediante un sistema de sockets vía TCP. El microprocesador actuará como repetidor entre los mensajes del microcontrolador y el exterior. Por lo tanto, los mensajes recibidos en un ordenador conectado al microprocesador serán nuevamente los del apartado anterior. Los mensajes enviados del usuario al microprocesador tampoco estarán estructurados de una forma concreta, en su lugar se emplearán comandos predefinidos que generarán la respuesta deseada.

Además existirán una serie de comandos entre el usuario y el microprocesador para lograr comportamientos concretos; estos funcionarán de forma similar a los mensajes enviados desde el microprocesador al microcontrolador.

En el anexo 2, Diagramas de funcionamiento software, se muestra el diagrama del flujo de innovación software.

5.4.2. Software del microcontrolador

Antes de comenzar con la programación del microcontrolador debemos conocer cuáles serán las actividades que desarrollará la aplicación. También resultará importante determinar el paradigma de programación que emplearemos: programación estructurada, programación funcional o programación orientada a

objetos. Sin embargo, este vendrá determinado por las actividades realizadas y el acercamiento que deseemos realizar.

Durante la mayoría de apartados se hará referencia a la página y la línea de código sobre la que se está centrando la atención. Debido a la gran cantidad de código se ha decidido que no se mostrará durante la explicación del mismo.

5.4.2.1. Cometidos del programa

Las funciones que llevará a cabo el microprocesador pueden agruparse en cinco bloques: configuraciones, recopilación de información, análisis de la información, actuación y envío de información.

Las funciones de configuración se ejecutan tan solo una vez, al principio del programa; preparan los elementos y las variables para que las funciones cíclicas puedan trabajar de forma correcta. Estas funciones de configuración se emplean para la inicialización de los canales de comunicación y para la configuración de los pines GPIO (General Purpose Input Output). En nuestro caso emplearemos cuatro funciones de configuración destinadas a: comunicación serie, comunicación con el IMD, comunicación CANBus y entradas-salidas de propósito general.

Las funciones de recopilación de información se encargarán de obtener toda la información necesaria para tomar las decisiones adecuadas. En esta categoría podemos diferenciar a su vez dos formas de recopilación de información; en primer lugar destacan las funciones encargadas de controlar la entrada de información por los canales de comunicación (Serie, CANBus y PWM). También debemos tener en cuenta la información obtenida directamente de la lectura de pines digitales o analógicos, de este modo obtendremos la información sobre el estado del sistema de seguridad, la alimentación, los supercondensadores o las entradas auxiliares.

Las funciones de análisis de información se encargan de tomar decisiones, modifican los estados y variables internas, respecto al estado actual de la ECU y la información obtenida mediante las funciones de recopilación. Se trata de la parte fundamental del programa, pues es aquella que definirá el comportamiento y las reacciones del sistema frente a los estímulos externos. Algunas de estas funciones se encargarán de analizar los mensajes recibidos y actuar en consecuencia, mientras que otras se encargaran de modificar el estado de acuerdo a las lecturas de la maniobra de seguridad y de la alimentación.

Las funciones de actuación son aquellas encargadas de, en función del estado actual de la ECU y del resto de sistemas, activar o desactivar los relés. Se trata de la etapa de salida de la ECU, entre estas funciones se encuentran el encendido o apagado del regulador y la activación o desactivación de la maniobra de seguridad.

Finalmente las funciones de envío de información, como su propio nombre indica, estarán encargadas de comunicar el estado actual de la ECU y datos importantes al resto de sistemas o al microprocesador y al usuario. Desarrollaremos funciones para enviar información mediante por el bus CAN, el puerto serie y al LED.

5.4.2.2. Paradigma de programación

Una vez definidas todas las funciones que deberá llevar a cabo el microcontrolador, debemos fijar un paradigma de programación. Existen diferentes opciones entre las que destacamos tres: la programación orientada a objetos, la funcional y la estructurada.

La programación estructurada se basa en un cuerpo principal de código que emplea bucles y estructuras para alcanzar sus objetivos. Resulta muy rápido programar, a la vez que se facilita la lectura del código, ya que es una lectura lineal y sin saltos incondicionales. Su principal inconveniente es la imposibilidad de reaprovechar código ya escrito, esto puede llevar a códigos excesivamente largos en los que el mismo conjunto de instrucciones se repite en diferentes apartados.

La programación funcional al contrario de la estructurada, se basa en el empleo de subrutinas simples que, agrupadas entre sí o de forma individual, alcancen el objetivo deseado. Al simplificar los problemas complejos en una sucesión o combinación de tareas más simples, se logra una gran claridad en el código. Además al emplear subrutinas simples se permite el reaprovechamiento de las mismas en diferentes apartados del código sin necesidad de volver a escribirlas. Uno de los problemas de la programación funcional radica en la complejidad de su seguimiento, especialmente para alguien no acostumbrado a ella, pues en este tipo de programación se realizan saltos incondicionales a subrutinas declaradas en partes diferentes del código.

La programación orientada a objetos (POO) es un paradigma de programación bastante diferente a los anteriores que se basa en la creación de unas estructuras complejas llamadas objetos, que cuentan con atributos y métodos. Pueden crearse infinidad de objetos similares, donde cada uno contará con los mismos métodos y

atributos, si bien sus valores y comportamientos diferirán en función de como se haya construido y su uso. La POO resulta de gran utilidad para muchas aplicaciones modernas, pues da acceso a unas características muy interesantes como la abstracción, la herencia, el polimorfismo o el encapsulamiento de objetos, etc.

Tras un estudio de las necesidades del microcontrolador y las ventajas de cada paradigma de programación se han llegado a algunas conclusiones: en primer lugar podemos afirmar que en este caso la programación orientada a objetos no supondría ninguna ventaja significativa, por lo que se descarta el uso debido a su costosa implementación. La programación estructurada podría resultar la más clara visualmente. Sin embargo, es probable que resulte en un código poco optimizado y excesivamente largo. Por su parte la programación funcional se adapta considerablemente bien a las necesidades de la ECU, ya que puede desarrollarse una función para cada tipo de acción y llamarlas en el momento deseado.

La solución que se va a implementar trata de unir las ventajas de la programación funcional y de la estructurada. En una primera fase se desarrollará una programación estructurada de alto nivel, en la que se definirá el flujo principal del programa. Es decir, qué funciones serán de obligada ejecución, cuales serán condicionales y cuales dependerán de bucles. Estas funciones serán saltos a otras partes del código que desarrollarán las funciones. En estas partes del código se desarrollará un paradigma funcional, donde se dividirán los objetivos complejos en varios objetivos simples y se crearán pequeñas funciones destinadas a resolverlos. Dichas funciones podrán ser empleadas siempre que se desee. En esta primera página se deberá hacer referencia a todas las librerías que empleemos, ya sean internas de Arduino, creadas por nosotros para organizar las funciones o de terceros, obtenidas de repositorios online. Estas referencias ocupan desde la primera línea del archivo Main_V4.cpp hasta la décima línea del mismo.

5.4.2.3. Estructura base del programa

Como se ha comentado previamente se empleará una primera etapa de programación de alto nivel. En la ilustración 25 se muestra dicha etapa en el entorno de programación Arduino IDE. El esquema base de programación de Arduino, formado por una función *setup* de configuración y una función *loop* de repetición constante se adapta perfectamente nuestras necesidades. En la parte central de la ilustración 25 podemos observar que se emplea la función *setup* para realizar todas las configuraciones iniciales: Serie, IMD, CANBus y I/O.

Main_V4 § | CANBus.hpp | CANBus_Def.hpp | Definitions.hpp

```
#include <SPI.h>
#include <EEPROM.h>
#include "Definitions.hpp"
#include "CANBus_Def.hpp"
#include "Gen_Funcions.hpp"
#include "IMD_Interface.hpp"
#include "Sec_Man_Interface.hpp"
#include "CANBus.hpp"
#include "Serial_Interface.hpp"
#include "Status.hpp"

void setup() {
  Serial1.begin(115200);
  IMD_Config();
  CAN_Config();
  InputOutputConfig();
}

void loop() {
  while(Buff_Save!=Buff_Index)CAN_Watch();
  if(Serial1.available())Rd_Serial1();
  if(IMD_Flag) IMD_Fail=IMD_Conv();
  if(Comms_Active)  Comms_Fail=Comms_Heartbeat(); else Comms_Heartbeat();
  Sec_Fail=WatchSecMan();
  UpdateStatus();
  WatchReload();
  UpdateDisplay();
  Status_LED(1000-(Status*400));
  if(LIN_HBT!=0)ECU_Stat();
}

```

Ilustración 25. Código principal del programa del microcontrolador

En la parte inferior de la ilustración 25 encontramos la función base *loop* de Arduino, en el que incluiremos todo el código que queramos ejecutar de forma cíclica. Se puede observar que se trata de llamadas a funciones declaradas en otras partes del programa.

Todas las funciones desarrolladas para el programa se han agrupado en hojas adicionales de formato hpp y son gestionadas automáticamente por el Arduino IDE. Debemos indicar al comienzo del código que deseamos incluirlas, así como el resto de librerías necesarias (SPI, CANBus). Separar las funciones en hojas de acuerdo a su aplicación otorga una mayor organización y simplicidad para programar y realizar modificaciones sobre el código.

También se han incluido un par de hojas, Definitions.hpp y CANBus_Def.hpp, donde se encuentran agrupadas todas las definiciones, tanto de constantes como de variables. Podemos observar en la ilustración 26 que se definen constantes que relacionan los pines del Arduino con sus respectivas funciones en el sistema. De esta forma resultará más intuitivo el código desarrollado.

```
1  //----------------Pinout----------
2  #define CAN_INT_HV    2
3  #define CAN_INT_GLVS  3
4  #define CAN_CS_HV     4
5  #define CAN_CS_GLVS   5
6  #define PIN_CAPS      6
7  #define PIN_IMD       7
8  #define CAN_RESET     8
9  #define PIN_AUX1      9
10 #define PIN_AUX2      10
11 #define SEC_MAN_OUT   11
12 #define PIN_POWER     12
13 #define PIN_STATUS    13
14 #define SEC_MAN_IN    A0
15 #define SEC_MAN_IMD   A1
16 #define CAN_POWER     A2
17 #define KEY_SWITCH    A3
18 #define START_RELAY   A4
19 #define SEC_RELAY     A5
```

Ilustración 26. Definición de los pines de Arduino de acuerdo a sus funciones

5.4.2.4. Código del Setup

En el Setup encontramos las funciones de configuración desde la línea 12 hasta la línea 17 de la página principal del programa, Main_V4.cpp. A continuación explicaremos el objetivo concreto de cada una de ellas, y el proceso empleado para lograrlo:

5.4.2.4.1. Serial1.begin

Llamada al método begin del objeto Serial1. Tanto el objeto como el método, están implementados en las librerías base de Arduino. La instanciación de este objeto se realiza de forma automática, no es necesario emplear el constructor previamente en el programa. Esta orden inicializa el puerto serie Serial1 con una velocidad de 115200 baudios (bits por segundo). El puerto serie Serial1 une el microprocesador y el microcontrolador del Arduino Yún. Siempre que queramos realizar alguna operación sobre este puerto emplearemos algún método del objeto Serial1.

5.4.2.4.2. IMD_Config

Esta llamada agrupa dos funciones base de Arduino empleadas para configurar la lectura de la onda PWM procedente del IMD. Podemos observar a partir la línea 30 de la página IMD_Interface.hpp que primero se configura el pin PIN_IMD como entrada digital, y posteriormente se configura como interrupción. Dicha interrupción se inicializa para actuar cada vez que el pin PIN_IMD cambie de estado y llamará a la función IMD_Call, entre las líneas 8 y 15, que se encargará de salvar los datos del PWM cada vez que cambie de nivel lógico.

Autor: **Javier Martínez Lahoz**

424.16.93

Podemos observar en el código referente al IMD_Call que en primer lugar se registrará el nivel lógico actual del PWM, alto o bajo, e inmediatamente después se calculará el tiempo transcurrido desde el cambio previo. Con estos dos datos se podrá reconstruir la onda PWM original sin perder información en el proceso. Cada dos iteraciones de guardado, indicador de un ciclo completo de la onda, se desactivará la interrupción y se activará una bandera a la espera de que se analice la información salvada, IMD_Flag. Esto se produce líneas 15 y 14 respectivamente de la hoja IMD_Interface.hpp.

5.4.2.4.3. CAN_Config

Esta orden realiza llamadas a todas las funciones necesarias para activar y configurar los dos canales aislados CANBus de la placa. Podemos diferenciar cinco pares de ordenes agrupadas dentro de esta función, entre las líneas 75 y 88 de la página CANBus.hpp. Los dos primeros pares de ordenes se encargan de configurar los pines del Arduino CAN_POWER y CAN_RESET como salida en estado alto (HIGH), estos pines se emplean para activar el Smart-Switch y los controladores de CANBus respectivamente.

Las cuatro siguientes órdenes sirven para configurar las interrupciones de recepción de mensajes. Las dos primeras configuran los pines, mientras que las dos segundas configuran las llamadas de las interrupciones. Cada vez que un controlador reciba un mensaje válido forzará el pin a un nivel bajo de tensión. En ese momento esta interrupción actuará y se procederá a guardar el mensaje en un buffer circular para su posterior lectura. El guardado de los mensajes difiere en el chip select (CS) según el canal, por lo que resulta necesario emplear dos funciones distintas: CAN_GLVS_Read y CAN_HVS_Read para los canales de baja tensión y alta tensión respectivamente.

Las dos funciones de guardado de mensajes, líneas 51-61 y 63-73, funcionan de forma similar. Se parte de dos elementos creados previamente: un objeto de tipo MCP_CAN que representa al controlador, líneas 55 y 56 de CANBus_Def.hpp, y un buffer circular para salvar los mensajes, líneas 3-5, 51 y 53 de CANBus_Def.hpp. El objeto MCP_CAN es individual para cada canal y pertenece a la librería CANBus de coryfowler (usuario de GitHub). El buffer circular es común para ambos canales con el objetivo de facilitar la lectura de mensajes. Dispone de 32 posiciones de guardado y dos indicadores de posición: Buff_Save y Buff_Index para escritura y lectura respectivamente.

La función de guardado comienza anulando todas las interrupciones para evitar problemas durante la lectura. A continuación, se comprueba si existe algún mensaje pendiente, en el caso de la primera iteración existirá uno al menos, por lo que se procederá a aumentar el indicador de posición de escritura del buffer circular para que apunte a una nueva posición. En este momento se guarda el mensaje mediante un método del objeto MCP_CAN. Este requiere que se indiquen tres argumentos que guardarán la ID del mensaje, su longitud y sus datos, para salvar los datos se indicará el puntero a un vector, de tamaño igual o superior al del mensaje donde se guardará. Después se realizará una comprobación al comparar los dos indicadores de posición del buffer circular; si son similares implica que se han producido demasiadas escrituras en el mismo y muy pocas lecturas, por lo que se ha producido exceso de información (overflow) que será indicada mediante un mensaje de error por pantalla.

En este punto se volverá comprobar si existe algún mensaje, en caso afirmativo, se repetirá el proceso anterior, sino se reactivarán las interrupciones y se continuará con el programa.

Retornamos a la función original de CAN_Config para tratar la última pareja de órdenes: CAN_begin, líneas 4-8 de CANBus_Def.hpp. Esta orden llama al método begin de los objetos MCP_CAN, encargada de configurar el controlador de CANBus de acuerdo al circuito oscilador empleado, la velocidad del bus al que se conecta y sus funciones. No profundizaremos en el funcionamiento interno de la misma, ya que está desarrollada por un tercero y no es el objeto de este trabajo.

5.4.2.4.4. InputOutputConfig

Emplearemos esta función, líneas 40-56 de Gen_Funcs.hpp, para llamar a las órdenes encargadas de realizar la configuración de los pines restantes. En primer lugar se selecciona el modo de los pines: entrada o salida. Lógicamente todos los pines conectados a relés se configurarán como salida, y todos los optoacopladores como entrada (en modo pull-up). También se fijará el estado inicial de todos los relés de forma que comiencen abiertos.

5.4.2.5. Código del Loop

Una vez finalizadas la configuración, el programa procederá a ejecutar las funciones nombradas en el loop de forma cíclica. Aquí encontraremos las que deban ser ejecutadas constantemente todas las funciones excepto las configuraciones. Ocupa desde la línea 19 hasta la 30 en la página principal del programa, Main_V4.cpp.

5.4.2.5.1. CAN_Watch

La primera orden empleada en el loop tiene como objetivo analizar los mensajes entrantes de CANBus. Al comparar los indicadores de posición del buffer circular, Buff_Index y Buff_Save, podemos saber si se han recibido mensajes nuevos. Si ambos indicadores son iguales implica que han sido leídos, en el caso de que sean diferentes, implicará que hay mensajes no leídos en el buffer circular, y se analizarán. El análisis de mensajes se producirá entre las líneas 104 y 132 de la página CANBus.hpp.

Para comenzar la lectura se anularán las interrupciones durante este proceso para mejorar la estabilidad del sistema, de esta forma se evita la recepción de un mensaje mientras se analiza otro ya que causaría problemas. El siguiente paso será mover una unidad el indicador de posición de lectura del buffer circular, esta nueva posición contiene el mensaje que vamos a leer.

En el caso de que la bandera Sniffer este activada, se llamará a la función CAN2SER, definida entre las lineas 91 y 102 de esta misma página. Esta función genera un string de caracteres en formato NMEA0183 que contiene el identificador del mensaje de CANBus, la longitud del campo de datos y todos sus datos, para enviarlos por el puerto serie al microprocesador mediante la orden Send_Data.

La orden Send_Data, encontrada en la hoja Gen_Funcs entre la líneas 2 y 9, se encarga de enviar cadenas de caracteres por el puerto serie. En un primer momento se calculará el Checksum correspondiente del mensaje. Se enviarán por el puerto serie Serial1 añadiendo un retorno de carro al final del mensaje, mediante un método del objeto existente en las librerías originales de Arduino, *println*. Emplearemos esta función cada vez que queramos enviar mensajes al microprocesador.

De vuelta en la función principal se realiza una comparación entre el identificador del mensaje recibido, y los IDs de los mensajes existentes, líneas 108-130. Cuando ambos identificadores coinciden, se copia la secuencia de datos entrante en una estructura de datos interna. Ambas contienen los mismo parámetros, del mismo tamaño y en el mismo orden, por lo que el traspaso de información resulta directo mediante la orden memcpy.

También se modifica una variable de Heartbeat para señalar que el emisor del mensaje se encuentra funcional y conectado al bus. En el caso de que el identificador de mensaje entrante no coincida con ninguno de esta lista, se omite. Esto ocurrirá con algunos mensajes del regulador que no contendrán datos.

5.4.2.5.2. Rd_Serial1

La siguiente función del loop también se emplea para leer mensajes entrantes, en esta ocasión se analizarán los entrantes por el puerto serie Serial1. Debido al funcionamiento del puerto serie, en esta llamada se agrupan las funciones de lectura y análisis de los mensajes.

Para leer la información existente en el puerto, líneas 27 a 35 de la página Serial_Interface.hpp, recurriremos a dos métodos básicos de Arduino Serial1.available y Serial1.read. Estas funciones se encargan de comprobar si existe algo en el puerto serie y leerlo respectivamente. Guardaremos toda la información recibida en un buffer, Buff_Serial, para su posterior análisis. Además, se esperará un mínimo de dos milisegundos desde que se lea un carácter hasta que se de por concluido el mensaje. De esta forma se evita que se pueda interpretar que un mensaje ha acabado antes de tiempo. Cuando no se reciba más caracteres se dará por concluido y se procederá su análisis mediante la llamada a la función Ev_Serial()

La función Ev_Serial(), definida entre las líneas 12 y 24 de la página Serial_Interace.hpp compara el mensaje recibido, y almacenado en el buffer, con una serie de vectores que corresponden con los mensajes esperados. Se emplea la función Cmp_Arrays, líneas 11-15 de Gen_Funcs.hpp, para determinar si son idénticos.

La función Cmp_Arrays compara individualmente cada carácter de dos vectores. En el caso de que los caracteres de ambos vectores no coincidan, se indicará y se finalizará la función. Tan solo se indicara que ambos mensajes son similares si tienen los mismos caracteres y finalizan a la vez.

Cuando la función Cmp_Arrays determine que ambos vectores son idénticos se ejecutará el comando adecuado en cada caso de la función Ev_Serial. En el caso de que el mensaje recibido no coincida con ningún mensaje predefinido, se devolverá un error de comando serie no válido.

5.4.2.5.3. IMD_Conv

Esta función se encarga de gestionar la información recibida del IMD; tan solo se activará cuando la bandera IMD_Flag se encuentre activada. En este momento se llamará a la función IMD_Conv, y se asignará su retorno a la variable IMD_Fail. Esta función devolverá un 1 cuando las medidas realizadas indiquen un estado anómalo o peligroso; mientras que un 0 indicará que todo funciona correctamente.

La función IMD_Conv, líneas 49-93 de la página IMD_Interface, comienza realizando una llamada a la función IMD_Read, en las líneas 36-47 de la misma página. Esta última se encarga de convertir las medidas de tiempo y nivel realizadas en la función IMD_Call, explicada en el apartado 5.4.2.4.2, a frecuencia y porcentaje de trabajo, los dos principales atributos de una onda PWM. Con la suma de tiempos conseguimos el periodo de la onda, y podemos deducir la frecuencia. Para obtener el ciclo de trabajo comparamos el tiempo durante el cual la onda fue positiva con el tiempo en el que fue negativa.

Una vez contamos con los valores de frecuencia y ciclo de trabajo, podemos proceder a analizar la información en la función IMD_Conv. En un primer momento fijamos la variable del valor de aislamiento a 0Ohm, esto se realiza para no generar medidas falsas en los casos en los que el valor de aislamiento no pueda ser calculado, línea 53. A continuación, realizamos un proceso de determinación de estado y aislamiento en dos fases. En primer lugar se empleará el valor de frecuencia, comparándolo mediante una serie de estructuras de tipo if-else con los esperables, líneas 55, 67 y 77. Posteriormente se empleará una combinación de estructuras if-else dependientes tanto de la frecuencia como del ciclo de trabajo para obtener los valores de situación de aislamiento (IMD_situation) y valor de aislamiento (IMD_isolation).

Una vez calculado ambos valores se reactivará la interrupción del IMD, que se había desactivado en la función IMD_Read, apartado 5.4.2.4.2. Finalmente se retornará un valor positivo o negativo en función de la situación y el aislamiento registrados, líneas 90-92.

5.4.2.5.4. Comms_Heartbeat

Esta función se encarga de controlar el estado de los sistemas de la moto. Al igual que en el caso anterior, se retornará un valor en función de si existe algún fallo y se almacenará en la variable Comms_Fail.

La función, líneas 77-101 de la página Gen_Funcs.hpp, hace uso de las variables de Heartbeat nombradas en la función CAN_Watch. Esta función emplea el comando millis junto con una comparación para ejecutarse tan solo una vez cada cierto tiempo, línea 78. En el caso de que no se ejecute debido a dicha comparación, se retornará el valor actual de Comms_Fail, línea 100.

Cuando la función se ejecute se tendrán en cuenta los valores de las variables de Heartbeat de los distintos sistemas. Se determina un límite del valor del Heartbeat

mediante una variable que indica cuándo se considera que un sistema ha perdido las comunicaciones, en este caso si su variable de Heartbeat es superior al límite; nuestro valor es cero. Si este es inferior al límite y distinto de cero, lo cual indicaría fallo, se considera que el sistema se encuentra conectado y se aumenta el valor de la misma en una unidad.

Por lo tanto si se ejecuta esta función el número de veces que indica el límite, sin que la variable de Heartbeat sea modificada por algún elemento externo, acabará valiendo cero. Esto es lo que sucede si un sistema deja de responder y se relaciona con que no esta funcionando correctamente. Cuando funciona correctamente envía mensajes que son analizados por la función CAN_Watch, apartado 5.4.2.5.1, y se modifica la variable Heartbeat correspondiente, retornando al valor uno.

El valor retornado por la función será cero, indicando un estado correcto, en el caso de que ninguna variable de Heartbeat sea nula, y por lo tanto todos se hayan comunicado recientemente. En el momento que uno de los sistemas no se haya comunicado, y su variable de Heartbeat sea cero, se retornará un uno, indicando un error. El error será diferente cuando se encuentre la alta tensión activada y cuando no; esto se debe al regulador, que no se considera cuando no existe alta tensión en la motocicleta. Si lo tuviéramos en cuenta, nunca desaparecería dicho fallo, y seria un gran inconveniente para la secuencia de arranque.

5.4.2.5.5. WatchSecMan

Esta función se encarga de estudiar el estado actual de la maniobra de seguridad. Al igual que en los dos casos anteriores, la función retornará un valor de error que se guardará en una variable, Sec_Fail. La función, definida entre las líneas 1 y 34 de la página Sec_Man.hpp, empleará un proceso basado en una lectura secuencia de los optoacopladores pertinentes para identificar el estado actual de la maniobra de seguridad. Estas lecturas, combinadas con una estructura if-else nos servirán para identificar en qué punto se encuentra la maniobra de seguridad.

Se considerará que la maniobra se encuentra en un estado correcto si se obtiene señal a la salida de la maniobra de seguridad cuando la moto se encuentre en marcha, y no haya pasado menos de un segundo desde el arranque, líneas 20-21. Se retornará un cero en este caso y un uno en cualquier otro caso. Además se emplea una variable adicional que sirve para indicar el estado concreto de la maniobra, Sec_Man_Status.

Si la moto no se encuentra en marcha, se considerará que la maniobra de seguridad no presenta ningún fallo cuando se obtenga señal justo antes del relé controlado por el regulador, líneas 13-14. Al igual que en el caso anterior, se retornará un cero si no existe fallo, y un uno si la maniobra no alcanza al relé mencionado.

5.4.2.5.6. UpdateStatus

La toma principal de decisiones de la ECU se realiza mediante esta función, que ocupa desde la línea 1 hasta la línea 68 de la página Status.hpp. Se encarga de generar los cambios de estado y activar o desactivar los relés en función de las circunstancias.

Antes de tomar ninguna decisión se realiza una medida del optoacoplador auxiliar de alta tensión, esta medida servirá para conocer si la existe alta tensión en el sistema, línea 2. A continuación se efectúa una segunda medida procedente de un optoacoplador, en este caso sirve para determinar si existe una alimentación externa, línea 3. Si dicha alimentación no existe, se activará el modo de alimentación auxiliar, que desactivará todos los relés a la espera de que se agoten las reservas de energía.

A continuación encontramos una estructura de tipo Switch-Case, en la que se ejecutará una parte del código u otro en función del estado de la ECU. Dicha estructura contará con tres opciones: modo de energía auxiliar, baja tensión y alta tensión; líneas 12, 16 y 52 respectivamente.

El modo de energía auxiliar se activará de la forma previamente nombrada. En este modo no se controlarán los relés; la única acción que se realizará será monitorizar el optoacoplador que indica si existe alimentación auxiliar. En caso de detectar alimentación externa, se procederá a reactivar el modo de baja tensión, línea 13.

El modo de baja tensión será el sistema primario de funcionamiento, y el encargado de gestionar el inicio del arranque de la moto. Su comportamiento esta definido entre las líneas 16 y 50. El primer paso que se realizará en el modo de baja tensión consistirá en llevar algunos parámetros a un punto conocido y aceptable. Estos parámetros son la variable Running, que indica la marcha y se fija al valor false; el pin auxiliar 2, que controla el Forward Switch (posteriormente se explicará el motivo de esta decisión) y que se dejará en modo abierto. Y finalmente la variable que representa el valor actual de velocidad fijado a cero, pues la moto no puede moverse

en este estado y el regulador que si se encuentra apagado, no puede proporcionar esta información.

La tomas de decisiones en este modo van agrupadas en dos bloques: una primera estructura condicional, líneas 35-40, analiza si existen errores en algún subsistema principal. Si no se detecta ningún error y todo funciona correctamente se activarán los dos relés de la maniobra de seguridad y el del Key-Switch. En el segundo bloque, líneas 21-34, se emplea otra estructura de tipo if-else que se ejecutará si se ha recibido alguna información del regulador, o se tiene constancia de que la alta tensión se encuentra activada. Cualquiera de estos indicadores implica que se ha activado la alta tensión de la moto, y por lo tanto se cambiará el funcionamiento del sistema a este modo. En este momento se registrará el tiempo del cambio para realizar correctamente la secuencia del encendido de la alta tensión.

Finalmente nos encontramos con el modo de alta tensión. En un primer momento se realiza una activación del relé auxiliar que controla el Forward-Switch en el momento oportun; y se le da el valor positivo a la variable Running, que indica si la motocicleta se encuentra en disposición de moverse. Más tarde este modo se encarga monitorizar el sistema y, en caso de detectar alguna anomalía, realizar un cambio de modo a baja tensión.

5.4.2.5.7. Watch_Reload

> *D.6.1.4 Una vez abierto el circuito de desconexión (contactores abiertos) por la actuación de cualquiera de los dispositivos previstos (TSMS, Interruptor de Emergencia, BMS o IMD) el sistema quedará en estado "no listo para conducir", y será necesario que el piloto lo reactive manual y voluntariamente (p.e. reiniciando el controlador), antes de que el circuito de desconexión vuelva a cerrarse.*
>
> *MEF (2015) Motostudent, Reglamento de la Competición*

De acuerdo a la normativa de la competición, D.6.1.4 citada anteriormente, en caso de que se produzca algún fallo en el sistema que desactive los contactores principales, se debe realizar un rearme voluntario y manual del sistema. Se ha decidido que se realizará mediante la pulsación y liberación del pulsador de emergencia, primer elemento de la maniobra de seguridad. La función Watch_Reload,

descrita entre las lineas 30 y 39 de la página Gen_Funcs, se basa en dos pasos y una variable auxiliar para lograr este comportamiento.

En un primer paso, líneas 36-39, se emplea una estructura condicional dependiente del estado de la maniobra de seguridad para comprobar si se puede permitir el rearme. Se fija la variable auxiliar en el estado positivo si no se detecta la maniobra de seguridad en ningún optoacoplador. Esto indica que la seta de emergencia se encuentra pulsada. En este momento se permite el rearme, pero no se efectúa.

El segundo paso condicional, líneas 31-35, analiza si el rearme está permitido y si la seta de emergencia está liberada; esto se deduce si algún optoacoplador de la maniobra de seguridad recibe señal. En caso de que ambos se cumplan, se lleva a cabo el rearme del sistema que viene determinado por la variable Run_Allow, que afecta a la función UpdateStatus del apartado 5.4.2.5.6.

5.4.2.5.8. UpdateDisplay

Esta función se define entre las líneas 58 y 74 de la página Gen_Funcs, y está destinada a enviar la información pertinente al display. Esta información estará codificada en el mensaje de CANBus con indicador 0x150, cuyos datos se han mostrado previamente. Esta función empleará una variable temporal y la función millis() para ejecutarse cada cierto tiempo, de la misma forma que la función Comms_Heartbeat del apartado 5.4.2.5.4.

Se comenzará rellenando un vector de bytes con los datos del mensaje. Todos se codificarán en un único byte para facilitar su descodificación posterior en el display. Una vez los campos del mensaje se han rellenado se procederá a enviar el mensaje. Se anularán las interrupciones en este proceso para aumentar la estabilidad del sistema. El mensaje se enviará mediante el método sendMsgBuff del objeto MCP_CAN, el cual requerirá que se le indique el identificador del mensaje, el tipo de mensaje (normal o extendido), la longitud del campo de datos y el puntero al comienzo vector de datos, línea 70.

5.4.2.5.9. Status_LED

El LED del interface parpadeará con una frecuencia determinada por el estado en el que se encuentre el sistema, esta función se encarga de gestionar este comportamiento. Esta función, definida entre las líneas 17 y 23 de la página

Gen_Funcs, alternará el estado del LED cada vez que haya transcurrido cierto tiempo. Este tiempo de parpadeo viene determinado por el estado de la ECU, el tiempo exacto se cálcula en el argumento de la llamada a la función, página principal Main_V4 línea 28. El parpadeo más rápido se producirá cuando la alta tensión se encuentre activa, y el más lento cuando no se detecte alimentación externa.

Para lograr el parpadeo se realizará una comparación entre el tiempo del último cambio de estado, el actual y el tiempo de parpadeo, línea 19. Cuando sea necesario se alternará el estado del LED, y se guardará el tiempo de dicho cambio.

5.4.2.5.10. ECU_Stat

La última función del loop, y por lo tanto la última que queda por estudiar del programa, es la encargada de transmitir la información desde el microcontrolador al microprocesador. Esta función, que engloba desde la línea 165 hasta la línea 186 de la página Serial_Interface, incluye tres bloques similares que siguen la misma estructura: el primer bloque, líneas 167-174, sirve para enviar los mensajes de estado básico y relés activados; el segundo, líneas 175-179, envía la información de telemetría y el tercero, líneas 180-186, envía la información de estado extendida. Los tres bloques contarán con temporización independiente para ejecutarse cada cierto tiempo.

El primer bloque, contiene llamadas a dos funciones distintas. La primera orden realiza una llamada a la función Status_Msg, líneas 40-74 de esta misma página. Esta función se encarga de enviar los datos de estado básicos. Comenzará generando un String con los primeros caracteres que formarán el mensaje: emisor y motivo del mensaje, línea 41. A continuación se rellenarán los campos de datos, líneas 42-17, con la información de estado. En el primer campo de datos se indicará la fase actual de funcionamiento del sistema mediante dos caracteres. En el segundo campo de datos se indicara si la marcha esta permitida o es necesario realizar un rearme manual y si los subsistemas principales (IMD, Comunicaciones y Maniobra de seguridad) funcionan correctamente o han detectado algún fallo. En el caso de que se haya detectado un fallo se proporcionará información relevante de ese subsistema para poder identificar la fuente del problema. Finalmente se llamará a la función Send_Data en la línea 73, previamente explicada en el apartado 5.4.2.5.1, que enviará el mensaje por el puerto serie.

Autor: **Javier Martínez Lahoz**
424.16.93

La segunda función, líneas 85-94, se emplea para conocer en todo momento el estado de los relés controlados por la ECU. Para ello se empleará una función similar a la anterior; en primer lugar se genera un String con la cabecera de mensaje correspondiente en la línea 86. Más tarde se irán añadiendo los nombre de los relés junto con sus estados en los campos de datos, líneas 87-91. Y por último se empleará la función Send_Data para enviar el mensaje por el puerto.

El segundo bloque de la función ECU_Stat transmite la información de telemetría del sistema al microprocesador. Se emplea una única llamada para retransmitir todos los datos deseados. Esta función, al igual que las anteriores, establecerá en primer lugar la cabecera del mensaje durante la creación del String. A continuación se irá insertando en los campos de datos la telemetría del sistema, líneas 138-146, separándolos con comas. Finalmente se enviará la información por el puerto serie mediante la orden Send_Data.

El último bloque se encarga del mensaje de estado extendido; este mensaje es similar al de estado normal, pero indica en todo momento el estado concreto de cada sistema, y no solo cuando se producen errores.

5.4.3. Software del microprocesador

5.4.3.1. Cometidos del microprocesador

La función principal del microprocesador será actuar como interface entre el microprocesador y el usuario, además de gestionar la base de datos con la información de telemetría del sistema. Podemos clasificar sus funciones de comunicaciones de dos formas; en primer lugar según el sentido de la información (lectura o escritura) y en segundo lugar según el canal comunicación (serie, TCP, base de datos).

Todas las funciones de escritura se ejecutarán como resultado de alguna de lectura. La lectura de datos por el puerto serie se redirigirá, tras un análisis del mensaje, a la base de datos y/o al usuario mediante TCP. La lectura de datos vía TCP, enviada por el usuario, también generará un envío de información, la cuál podrá ser directamente por serie hacia el microcontrolador, o de vuelta hacia el usuario. En algunos casos se realizará primero una petición a la base de datos, para redirigir la respuesta hacia el usuario vía TCP.

5.4.3.2. Lenguaje y Paradigma de programación

La programación del sistema operativo OpenWRT se puede realizar mediante una gran cantidad de lenguajes e interfaces de programación. Se ha optado por emplear el lenguaje de programación Python por la gran velocidad de desarrollo que proporciona frente a otros lenguajes como C++ y Java, los cuales son mucho más estrictos sintácticamente.

El programa se desarrollará en u editor de texto en un ordenador externo al sistema. Posteriormente se realizará un volcado al microprocesador mediante un servicio FTP. Una vez allí será ejecutado mediante ordenes remotas cuando queramos que entre en funcionamiento.

Este programa deberá ser capaz de realizar distintas funciones a la vez mientras espera información externa. La mayoría de las funciones de lectura de datos en Python son bloqueantes, lo que quiere decir que la función para la ejecución del programa y el resto del código no se puede ejecutar correctamente. Emplear funciones no bloqueantes resulta considerablemente más complejo, por lo que no resulta interesante. En su lugar se opta por otra solución que permite emplear funciones bloqueantes sin alterar el funcionamiento continuo del programa mediante el uso de hilos de ejecución.

Los hilos de ejecución permiten desarrollar dos tareas de forma simultanea sin que la ejecución de una afecte directamente a la otra. Para lograr esto el procesador desarrolla un hilo mientras el resto de hilos están en espera. Cuando se han ejecutado una cantidad determinada de órdenes en ese hilo, se encuentra una espera o una bloqueante se para la ejecución del hilo actual y se continua con otro hilo. De esta forma se pueden tener funciones bloqueantes en un hilo, mientras otro sigue ejecutándose de forma constante.

También se definirán funciones externamente a los hilos principales, esto se realizara para mantener una organización en el código y facilitar su lectura.

Finalmente se ha decidido también que el programa sea capaz de recibir ordenes directamente a través de la conexión ssh, la cual lo ejecutara. Estas se emplearán para modificar el comportamiento del sistema, realizar peticiones o cerrar el programa.

Autor: **Javier Martínez Lahoz**
424.16.93

5.4.3.3. Estructura base del programa

El programa, como hemos comentado en el apartado anterior, empleará diversos hilos de ejecución para llevar a cabo todas sus funciones. Se desarrollará de forma que su estructura sea similar a la de Arduino empleada en el microcontrolador. Aunque en este caso solo se empleará un archivo donde se desarrollará todo el código. Se toma esta decisión ya que se considera que el código final no resultará demasiado extenso.

La mayoría de lenguajes de programación solo cargan por defecto algunas librerías muy esenciales; Arduino en su compilación carga algunas librerías más específicas. Por lo tanto en las primeras líneas de código del programa, al igual que en el microcontrolador, se emplean para cargar las librerías que posteriormente se emplearán. Entre estas se encuentran, por ejemplo, la llamada a la librería serie, ya que ninguna se encuentra cargada por defecto. Esto ocupa desde la línea 3 hasta la 8 del código Main_Linino.py.

A continuación, encontramos las definiciones de variables, entre la línea 12 y la línea 45. Destaca especialmente, frente a la programación en C++ de Arduino, que en este caso no se especifica el tipo de cada variable; sin embargo, les retribuimos un valor representativo para que la asignación de tipos de python funcione como queremos. En este punto también generamos los objetos y punteros que nos servirán para establecer la comunicación SQL (líneas 20-21), serie (líneas 53-59) y TCP (líneas 47-49).

La parte más extensa del código, entre las líneas 65 y 470, esta formada por las definiciones de las funciones que se emplearán durante el programa. Todas las ellas, excepto la parte principal del código, se definirán y desarrollarán aquí. Se incluye entre estas funciones todo el código de los hilos de ejecución que no sean el principal.

Tras la definición de las funciones encontramos una parte de código equivalente al setup de Arduino. Este código solo se ejecutará una vez e incluye la configuración e inicialización del socket TCP (líneas 472-482), la creación e iniciación de los hilos de ejecución (líneas 484-492) y el código que queramos que solo se ejecute una vez (líneas 494-503).

El código situado entre las líneas 505 y 536 contiene el código del hilo principal, desde este se hacen llamadas a las funciones pertinentes en cada caso.

Finalmente se valora la opción, al contrario que Arduino, de que el programa acabe. En este caso el usuario puede finalizar el programa para poder acceder al microprocesador, ya que capturará la mayoría de recursos y E/S del microprocesador. Cuando el usuario dé la orden de finalizar se ejecutarán una serie de ordenes finales para asegurar el correcto cierre de todos los canales de comunicaciones empleados, líneas 538-541.

5.4.3.4. Código de Python

5.4.3.4.1. Librerías, Variables y Configuración

En primer lugar tendremos que declarar librerías especificas para casi cualquier función mínimamente compleja que vayamos a emplear, ya que no hay declaraciones implícitas.

En primer lugar cargamos la librería serial, que permite un sencillo acceso al puerto serie y todas sus tareas relacionadas. Tras ello cargamos la librerías socket, para trabajar con los que emplearemos en la comunicación TCP. A continuación se carga la librería sys, que permite acceder a determinadas funciones del microprocesador, como por ejemplo el control de la salida de datos o el control de las GPIOs. La librería threading permite crear y gestionar distintos hilos de ejecución. Para facilitar el acceso a la base de datos de telemetría se carga la librería sqlite3. Y finalmente se carga la librería time, para poder emplear funciones con control de tiempo.

Las declaraciones de variables se realizan fundamentalmente para fijar los tipos de las variables que se emplearan. Para crear una variable numérica se asigna el valor 0 a esa variable, y para asignarle una cadena de caracteres se le asignan dos comillas simples sin nada entre ellas. Se puede observar que la mayoría de declaraciones son de variables que se emplearán posteriormente para el salvado de datos a la base de datos, de la línea 24 a la 40.

En la creación del socket TCP, líneas 47-50, debemos definir un par de variables; en primer lugar se debe definir el nombre del Host al que nos deseamos conectar, en caso de dejarlo vacío, como ocurre en este caso, se conectará a cualquier Host. En segundo lugar debemos decidir en que puerto generaremos la conexión TCP, se ha escogido el 2002 porque no esta ocupado por ningún servicio de los implementados, ni se espera que se vaya a emplear, evitando los problemas derivados de que dos servicios traten de emplear el mismo puerto.

La inicialización de la comunicación serie, de la línea 53 a la 59, también requiere de algunos parámetros de configuración. Debemos definir la velocidad de transmisión, y debe coincidir con la velocidad del otro extremo, definida en el apartado 5.4.2.4.1, que es 115200 baudios. El resto de configuración debemos declarar si los paquetes contarán con bit de paridad, que no cuentan, si los paquetes cuentan con bit de stop, que no cuentan, y el tamaño de los bytes, 8 bits.

El resto de la configuración inicial se realiza después de las definiciones de funciones, líneas 472 a 482. En primer lugar se trata de establecer la conexión TCP, en caso de que el puerto este ocupado se devolverá un error. Si la conexión es correcta se preparará el socket para vigilar el puerto 2002 a la espera de información entrante.

A continuación, líneas 484 a la 492, se generan los diferentes hilos de ejecución que controlará el programa de forma simultánea. Finalmente se decide emplear tres hilos de ejecución secundarios, además del principal. Emplearemos un hilo de ejecución para realizar las transmisiones de información mediante TCP, apartado 5.4.3.4.3, otro para las recepciones de información vía TCP, apartado 5.4.3.4.4, y un último para recibir las ordenes del teclado, apartado 5.4.3.4.5.

Cada hilo se asigna a su función inicial, líneas 484-486, se configura como demonio, líneas 487-489, y se inicia, líneas 490-492. La configuración como demonio quiere decir que el programa finalizará cuando lo haga el hilo principal de ejecución, no esperará a que los secundarios finalicen.

Finalmente se ejecutan una serie de comandos para asegurar el funcionamiento correcto del sistema, líneas 494-503, que activan como salida la GPIO18 y le dan el valor lógico alto. Rápidamente se retorna al valor lógico bajo y se desactiva dicha salida. Esta serie de comandos fuerza el reinicio del microcontrolador , de esta forma nos aseguramos de que ambos comiencen su ejecución a la vez y no se produzcan problemas.

5.4.3.4.2. Hilo principal de ejecución

El hilo principal de ejecución se ejecuta de forma cíclica mientras una variable booleana se encuentre desactivada, línea 505. Esa variable solo puede ser modificada mediante una orden directa a través del teclado, apartado 5.4.3.4.5.

Al principio del código encontramos dos estructuras if que se ejecutarán periódicamente, líneas 513-515 y 516-518. La primera de estas estructura se

ejecutará diez veces por segundo, y sirve para indicarle al microcontrolador que el microprocesador se encuentra activo. Para ello envía un mensaje de cinco caracteres por el puerto serie.

La segunda estructura sirve para guardar los datos de telemetría de forma definitiva en la base de datos cada 20 segundos. De esta forma un apagado inesperado no eliminará toda la información de la última sesión, como mucho se eliminarán los últimos 20 segundos de datos. La parte principal del código se encuentra tras estas dos primeras estructuras.

Las lecturas del puerto serie recibidas se van almacenando en un vector a la vez que se va analizando. El salvado de datos al vector solo comenzará una vez se haya recibido el identificador de inicio de mensaje, el símbolo del dólar. Cada vez que se registra un nuevo carácter se comprueba si se trata del fin del mensaje, línea 520. En caso de que no haya acabado el mensaje se ejecuta el código declarado entre las líneas 528 y 536.

En primer lugar se guarda el dato entrante, siempre y cuando el mensaje ya haya comenzado, o el dato entrante sea el símbolo de inicio de mensaje. También se realiza la búsqueda del símbolo del asterisco para indicar el fin de los datos y el inicio del CheckSum. Una vez encontrado, se guarda la posición en la que se encuentra, línea 531. Para evitar problemas de saturación se desecharan los mensajes de más de 255 caracteres, líneas 533-536. No enviaremos mensajes de tal longitud, por lo que se trata de una función de seguridad.

Cuando el dato recibido sea un cambio de línea o un retorno de carro, y el mensaje sea mayor de cinco caracteres, se considerará que el mensaje ha acabado. Se escribirá el mensaje por la pantalla, línea 521, para poder realizar un seguimiento de la información recibida, se activará una variable para indicar que se ha recibido un mensaje, relevante para el funcionamiento del hilo de ejecución de transmisión TCP, y se llamará a la función getMSG, que se encarga de analizar los mensajes y actuar de acuerdo a cada uno.

La función getMSG, líneas 141-176, se encarga de gestionar los mensajes. Se comienza comprobando que el mensaje recibido es correcto, para ello se llama a la función getCheckSum. Esta función, definida entre las líneas 451 y 469, se encarga de obtener el valor del CheckSum explícito del mensaje, líneas 460-463; a la vez que se calcula nuevamente, líneas 456-459. Se comparan ambos valores en la línea 464 y se

devuelve el resultado de la comparación. En caso de que los valores no sean iguales, se descarta el mensaje, al tomarse como erróneo.

El siguiente paso, si el mensaje es aceptado, consiste en obtener el emisor del mensaje, de vuelta en la función getMSG. Esto se realiza mediante una comparación de los caracteres correspondientes, primero y segundo tras el dólar, entre las líneas 419 y 434.

A continuación debemos averiguar el tipo de mensaje recibido, para ello se realiza otra comprobación similar, en este caso empleando los caracteres que indican el tipo de mensaje, tercero a quinto. Esta comparación se realiza entre las líneas 436 y 449.

Una vez averiguado el tipo de mensaje que se ha recibido se puede proceder a actuar en consecuencia. Hay tres mensajes implementados en la versión actual del código: el mensaje de estado general, el mensaje de estado extendido y el mensaje de telemetría. En los tres casos se sigue una estructura similar para analizar el mensaje.

El mensaje de estado general llama a la función Msg_CUSUP, líneas 242-412, con el argumento 0, indicando que es un mensaje normal, no extendido. La función Msg_CUSUP se encarga de comparar cada campo de datos entrante con los posibles valores, y guarda en una variable los datos obtenidos en cada caso. Esto se realiza tanto para los mensajes de estado general (líneas 264-291), como para los mensajes de estado extendido (líneas 292-382), que emplean una llamada con el argumento 1.

De vuelta en la función get_MSG, se rellena una tupla, similar a un vector, con las variables de la función Msg_CUSUP, líneas 148-150 y 158-162 para los mensaje de estado general y extendido respectivamente. Durante muchas versiones estos vectores se guardaban en la base de datos para su posterior análisis, sin embargo en la versión final se descartó, para dar preferencia a los datos de telemetría.

El mensaje de telemetría funciona de forma similar a los mensajes de estado. En primer lugar se llama a la función Msg_CUTLM para separar los datos recibidos y asignarlos a variables, todo ello entre las líneas 178 y 240, y con la ayuda de la función getSubstring, declarada entre las líneas 414 y 418. De vuelta en get_MSG se insertan todos los valores en una tupla, que se empleará posteriormente para el guardado en la base de datos, líneas 172 y 173.

Tras esto se da por concluido el análisis y guardado del mensaje. Se retorna al código principal del hilo, línea 525, donde se vacían los buffers empleados para leer el mensaje, y se espera a la recepción de un nuevo mensaje.

En caso de recibir la orden de cerrar el programa, proveniente del apartado 5.4.3.4.5, se procederá a la ejecución de las ordenes finales del código, líneas 538 a 541. Estas cuatro líneas se encargan, en este orden, de: salvar la base de datos definitivamente, cerrar la conexión serie, cerrar la conexión TCP y cerrar la base de datos. De esta forma se realiza un fin de programa controlado y correcto para todos los métodos de comunicación y la base de datos.

5.4.3.4.3. Hilo de transmisión TCP

Este hilo se encarga de gestionar una conexión TCP y reenviar los mensajes recibidos del microcontrolador; se encuentra definido entre las líneas 94 y 121 del código. Es necesario trabajar de forma independiente al hilo principal porque la función empleada para establecer la conexión, s.accept en la línea 101, es bloqueante y molestaría al flujo principal del programa.

Una vez se logre establecer comunicación emplearemos una estructura de tipo try-except para gestionarla. Emplearemos un bucle while dentro del apartado try para que el código contenido se ejecute de forma constante mientras el servidor se encuentre activo. Cuando este se cierre desde el otro extremo de la conxión se producirá un error de envío, qué provocará que entre en funcionamiento el apartado except. Este apartado cerrará la conexión y, mediante un bucle while, se volverá a la línea 101 a la espera de una nueva conexión.

Cabe destacar que en las líneas 104 y 119 se activa y desactiva una variable que, como veremos en el apartado 5.4.3.4.4, sirve para permitir la ejecución del hilo de recepción TCP.

La parte de ejecución cíclica del apartado try de esta función, líneas 106 a 117, se divide a su vez en dos estructuras if. La primera de ellas, líneas 106-111, está destinada a enviar mensajes indicando que el microprocesador, así como la conexión TCP, se encuentran activos y funcionando correctamente; estos mensajes se envían de forma cíclica cada segundo. La segunda parte, líneas 112-117, se activa cuando se acaba de recibir un nuevo mensaje, y se encarga de reenviar ese mensaje por la conexión TCP al usuario o a otro programa, para que pueda ser analizado mediando medios más especializados.

5.4.3.4.4. Hilo de recepción TCP

La función ejecutada en este hilo es TCP_Read, descrita entre las líneas 123 y 139. Esta función ejecuta de forma cíclica una serie de comprobaciones para identificar los comandos entrantes mediante la conexión TCP. En primer lugar, línea 126 se comprueba si el servidor se encuentra activo mediante la variable asignada en el apartado anterior: 5.4.3.4.3. A continuación se espera a la recepción de un mensaje de hasta 128 bytes. En caso de recibir un mensaje no nulo se procederá a un pequeño análisis. Si el mensaje recibido es "Update" se sacará toda la información de la base de datos y se imprimirá por pantalla, líneas 130 y 131. En caso contrario se redirigirá el mensaje al puerto serie, pues se tratará de un mensaje para el microcontrolador, líneas 135 y 136. De esta forma un programa externo conectado por TCP puede solicitar una descarga de la información contenida en la base de datos

5.4.3.4.5. Hilo de lectura de teclado

Este hilo de ejecución se concentra en la función KeyboardRead, entre las líneas 65 y 88. Su función es identificar las ordenes recibidas por teclado y ejecutar los comando correspondientes. Para la lectura se emplea el comando raw_input, en la línea 74; este simplifica en gran medida el hilo, pues la salida puede compararse directamente con las órdenes esperadas. El gran problema de esta función reside en que es bloqueante, por eso se encuentra en un hilo de ejecución independiente al resto del código. Entre las líneas 75 y 86 se encuentran los comandos esperables, entre los que destacamos el comando exit, que se emplea para finalizar la ejecución del hilo principal.

En el anexo 2, Diagrama de funcionamiento Software, se muestra un esquema simplificado del flujo de información y de trabajo entre el microcontrolador, el microprocesador y el usuario.

6. RESULTADOS

Una vez el sistema se encuentra completamente diseñado tanto a nivel hardware como software, debemos realizar pruebas para asegurarnos de que el funcionamiento es el correcto. Se han ido realizando pruebas durante el desarrollo para comprobar su funcionamiento adecuado, sin embargo debemos realizar pruebas globales. Los resultados de estas pruebas de diversa índole sirven para evaluar la calidad del diseño global que se ha llevado a cabo.

6.1. RESULTADO DE LAS COMUNICACIONES

El objetivo del sistema desarrollado consiste en obtener información y gestionarla, por lo tanto una de las características más importantes es la cantidad de información que puede llegar a gestionar.

El primer paso consiste en obtener la velocidad de trabajo del Arduino. Para ello emplearemos una función que colocaremos en el loop, y calculará el tiempo transcurrido en cada ciclo. Para este cometido emplearemos la función micros, cuya resolución es mucho mayor a millis; resolución de microsegundos frente a resolución de milisegundos.

Evento	t	t	Δt	f	t tot[us]
Sin eventos	-	150-	-	4	794.900(res
Msg. Estado	157	2000	1850	1	1.850
Msg.	290	3650	3500	5	15.100
Msg. Relés	100	1400	1250	1	1.250
Lectura serie	230	3100	2950-	1	31.550
Envío CAN	130	2000	2750	5	7.950
IMD	200	800	650	5	10.450
CAN	310	3800	3650-	3	136.950

Tabla 5. Tiempos de trabajo de las funciones

Se realizan diversas pruebas, activando y desactivando diversas funciones. Se toman medidas de cuanto tiempo se tarda en realizar cada acción concreta, cuanto tiempo resultante emplea el loop, cuanto tiempo se añade al loop, con cuanta frecuencia ocurre ese evento y cuanto tiempo supone aproximadamente por segundo.

En la tabla 5 podemos observar que el método de comunicaciones que más tiempo de media consume a la ECU es el bus CAN, empleando una media de casi 137ms para la lectura de mensajes por cada segundo de ejecución. Esto se debe a la gran cantidad de mensajes, 35 por segundo que se reciben en total (14 del regulador, 6 de la batería, 10 del detector de pilotos y 5 del display).

También se invierte una gran cantidad de tiempo en la comunicación serie, tanto de entrada como de salida. La recepción de mensajes serie ocupa más de 31ms de cada segundo. El envió de mensajes serie, combinando los tres implementados, ocupa apenas 18ms, si bien la mayor parte de este tiempo es consumido exclusivamente por el envió de telemetría, ya que es el más largo y con más frecuencia.

Tras el resto de eventos obtenemos que aun se pueden emplear casi 800ms de cada segundo para realizar tareas de análisis, no comunicación. La optimización en las tareas fundamentales del loop logra que este se ejecute en apenas 180us, por lo que en 800ms podrá ejecutarse todo el ciclo más de 4000 veces.

Cabe destacar que al comienzo de las primeras pruebas el regulador se encontraba parametrizado de tal forma que emitía 600 mensajes por segundo. Esta ingente cantidad de mensajes requería de más recursos de los que se disponía, en concreto hacían falta 2'15 segundos para analizar los mensajes que se recibían cada segundo. A raíz de aquello se redujo la velocidad y cantidad de mensajes del regulador hasta los valores definitivos, que permite que la ECU funcione correctamente.

También podemos calcular los mensajes máximos de CANBus que sería capaz de administrar la ECU, manteniendo un margen de seguridad. Si no se recibiera ningún mensaje de CANBus dispondríamos de 932ms para cálculos. Podemos asumir que el límite máximo de trabajo sea de 900ms empleados para CANBus, dejando disponibles 32ms de margen. Teniendo en cuenta un tiempo medio de 3500us por mensaje, podríamos gestionar algo más de 250 mensajes cada segundo. Esta cifra deja en evidencia que la ECU esta trabajando por debajo de sus capacidades de movimiento

de información; si bien los parámetros originales del regulador exceden completamente estas capacidades.

Resulta necesario también destacar, que si bien la velocidad de trabajo del Arduino hubiera permitido trabajar con más mensajes, la memoria de programa y de variables del Arduino se encuentran al borde de la inestabilidad. Se están empleando casi 25000 bytes de memoria de programa con una capacidad máxima de 28672, se trata de un 86%, todos estos datos han sido obtenidos mediante la interfaz de programación Arduino IDE y se muestran en la ilustración 27. Sin embargo el punto más crítico se da en la memoria de variables, donde las variables globales emplean más de 2000 bytes, un 80%, y apenas dejan libres 501 bytes para las variables locales. Esta memoria tan reducida en un programa donde se emplean variables locales estáticas puede llegar a causar problemas de inestabilidad.

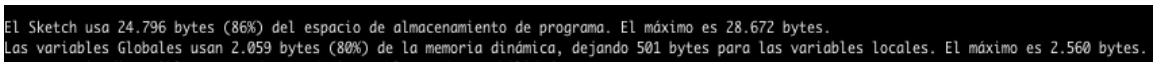

Ilustración 27. Límites de memoria según el Arduino IDE

A continuación calcularemos el ancho de banda ocupada en cada canal de comunicaciones. Para ello emplearemos el número de mensajes que se envían por segundo, la tasa de transferencia de datos y haremos una estimación de los datos enviados en cada mensaje. Todo ello se muestra en la tabla 6.

Canal	Nº	bits/	Nº bits	Bau	Ancho
CANB	21	110	2310	500.	0'462%
CANB	11	110	1210	500.	0'242%
NMEA	7	560	3920	115.	3'402%

Tabla 6. Ocupación de canales de comunicación

El número de mensajes de la tabla 6 se ha obtenido de la programación de los distintos sistemas. Para los mensajes de CANBus se ha supuesto que todos los mensajes van con todos los campos de datos ocupados, unos 110bits de acuerdo al apartado 4.5.3. Mientras que para los mensajes NMEA0183 se ha realizado una media ponderada de los diferentes mensajes y sus longitudes aproximadas, más un margen de seguridad. Consideramos que el mensaje de estado normal suele contar con unos 50 caracteres, el mensaje de relés con unos 45 y el mensaje de telemetría con unos 80, por lo tanto, y teniendo en cuenta que el mensaje de telemetría se emite a mayor

Autor: **Javier Martínez Lahoz**
424.16.93

frecuencia (5Hz frente a 1Hz), se considera un tamaño medio de 70 caracteres, teniendo en cuenta 8bits por carácter.

En la última columna de la tabla se puede observar que todos los canales de comunicaciones están preparados para trabajar con flujos de información mucho mayores a los empleados. Esto se debe principalmente a que se trata de estándares de comunicación muy empleados y preparados para mucha información, pero nuestro hardware, Arduino, no permitía mover flujos de información tan grandes (algo más grandes sí), si bien tampoco era necesario dada nuestra aplicación.

De acuerdo a estos valores de ancho de banda podríamos aumentar a información en el CANBus de baja tensión hasta los 4500 mensajes por segundo, al igual que el canal CANBus de alta tensión. En el canal de comunicaciones NMEA0183 podríamos aumentar, siguiendo el mismo formato, hasta los 205 mensajes. Una forma de alcanzar esto sería aumentar la frecuencia de tomas de muestras para telemetría, que se podría aumentar hasta los 200Hz, o bien mantener la velocidad y aumentar el número de variables.

6.2. RESULTADO DE LA PLACA

Durante las pruebas intermedias de integración de sistemas, entre las que se incluyeron las pruebas definitivas de la maniobra de seguridad, se detectó otro problema. Este problema viene relacionado con la secuencia de arranque de la maniobra de seguridad, concretamente el relé del regulador que se debe puentear temporalmente, apartado 5.2.2.4. Durante las primeras pruebas no se programó el regulador para que controlará su relé como "contactor de línea", como debería haberse hecho, en su lugar se programo como una salida digital normal, como por ejemplo las luces o la bocina del vehículo. En este estado la maniobra de seguridad funcionaba perfectamente; sin embargo este relé no se desactivaría cuando hubiera un problema, para eso deberá estar programado como contactor de línea. A continuación procedemos a realizar una prueba de funcionamiento de la maniobra de seguridad completa.

Programamos la salida del regulador para que actué como contactor de línea, sin embargo observamos que no se activa. Esto provoca que la secuencia de arranque falle cuando el relé de Start-Up se abre para ceder el control al relé del regulador. El regulador emite un fallo a través de CANBus, este fallo se define como "Welded

Contactors", que podemos traducir como contactores soldados. Este fallo se debe a que, cuando el regulador esta programado para controlar los contactores, no espera alta tensión en sus bornes de potencia hasta que él activa su relé. Sin embargo en este caso si recibe dicha tensión, cosa que no esta preparado para asumir, y genera un fallo. Probamos diferentes configuraciones y tiempos de relés, sin embargo en ningún momento conseguimos que la maniobra funcione completamente. Finalmente se opta por puentear completamente el relé del regulador mediante el relé de Start-Up para que la moto pueda arrancar.

Resulta interesante estudiar el consumo de la placa en cada estado, de esta forma podemos saber en que régimen de trabajo se encuentra la fuente de alimentación en cada momento, así como los consumos. Se realizarán mediciones de consumo en diversos estados del microcontrolador y del microprocesador. Todas las mediciones se realizarán empleando un amperímetro de laboratorio en serie con la alimentación. Los estados en los que se me mide el consumo son progresivos, esto quiere decir que las mediciones de consumo se realizan manteniendo todos los estados anteriores activos. De esta forma se observa el consumo progresivo del sistema según se le van añadiendo servicios o funciones. Las mediciones del primer estado, estado mínimo, se realizan con los sistemas de comunicaciones desactivados, y a continuación se añaden los servicios y comunicaciones de baja tensión, alta tensión, el script de python de la base de datos y finalmente la comunicación TCP con un usuario externo.

La primera columna de la tabla 7 indica la corriente medida, aquella que llega a la placa. A continuación se calcula la corriente de salida de la fuente de alimentación, teniendo en cuenta la relación de reducción (24/5) y la eficiencia (78%). Finalmente se expresa este consumo en potencia y se calcula en que régimen esta trabajando la F.A., columnas cuarta y quinta de la tabla 7. Finalmente se obtiene, de acuerdo a la eficiencia de la fuente, la disipación de energía en forma de calor, última columna de la tabla 7.

Podemos observar en la tabla 7 que cada activación de una parte del sistema, GLVS y HVS, consume aproximadamente 110mA. También destaca el consumo máximo, de 843mA, que se acerca a los niveles máximos de trabajo de la fuente (1A). Sin embargo la fuente no llega en ningún momento a su límite de trabajo, por lo que se consideran correctas la aproximación de consumos y la elección realizadas durante el diseño, concretamente en el apartado 5.2.2.6.1. Podemos destacar que su

disipación máxima de potencia no llega a 1W, por lo que no resulta necesario emplear un disipador, si bien el componente ganará algo de temperatura durante el trabajo.

Estado	Consumo (antes de F.A)[mA]	Consumo (después de F.A) [mA]	Potencia (después de F.A.) [W]	Trabajo de la F.A. [%]	Disipación de calor [W]
Mínimo	110 mA	412 mA	2'06 W	41'2 %	0'45 W
GLVS	140 mA	525 mA	2'63 W	52'5 %	0'58 W
HVS	170 mA	637 mA	3'19 W	63'7 %	0'70 W
Database	190 mA	712 mA	3'56 W	71'2 %	0'78 W
TCP	225 mA	843 mA	4'22 W	84'3 %	0'92 W

Tabla 7. Consumos

Durante las pruebas de consumo del sistema se detecto que, pasado medio minuto aproximadamente, el consumo de la placa aumenta de forma rápida y progresiva, por lo que es necesario desactivar la alimentación. Era de suponer que el fallo provenía del subsistema Power Manager, en concreto del circuito del acumulador, por lo que se anula este circuito quitando la resistencia de limitación de corriente. El subsistema queda en circuito abierto y procedemos a reconectar el sistema. Comprobamos que en esta ocasión no se produce ningún aumento de consumo, por lo que queda patente que el problema procede de este subsistema. De entre todos los componentes de este sistema procedemos a investigar el regulador de corriente, puesto que el funcionamiento del resto de componentes (resistencias, condensadores y diodos) es mucho más intuitivo.

Realizamos pruebas de carga mientras monitorizamos el comportamiento del regulador de tensión, y observamos un rápido aumento de la temperatura del mismo durante la carga. Además se comprueba que el aumento de consumo empieza a producirse cuando el regulador de tensión se acerca sus límites nominales de temperatura, 125ºC. Este problema se debe a que, durante la etapa de diseño, no se tuvo en cuenta la potencia que disiparía este componente. Esta potencia viene determinada por la caída de tensión entre la entrada y la salida, y la corriente que lo atraviesa; se trata de una ecuación muy estándar para el cálculo de potencia eléctrica, mostrada en la ecuación 14.

Ecuación 14

$$P=\left(V_{IN} \quad V_{OUT}\right)\times I$$

Sustituimos nuestros valores en la ecuación 14, suponiendo una tensión de salida media de 12, al ser el punto medio entre la descarga completa y la carga completa y tratarse de una carga lineal. La ecuación 15 muestra que este regulador de tensión disipa durante la carga una media de 8W, llegando a ser el doble justo al comienzo de la misma (0V de Vout).

Ecuación 15

$$\left(24 \quad 12\right)\times 0'66 = 8W$$

Esta disipación de potencia resulta excesiva para el componente, por lo que resultará necesario realizar modificaciones sobre la placa para reducir la potencia a disipar, a la vez que ayudamos a la disipación de la misma. Finalmente se opta por duplicar el valor de la resistencia de limitación (de 1R8 a 3R6), reduciendo la corriente y la potencia a la mitad, y emplear un disipador para disipar mejor el calor generado.

Tras las modificaciones se vuelven a realizar pruebas de carga y descarga. Con las modificaciones se logra que el sistema funcione correctamente, eliminando los consumos excesivos que se daban anteriormente. Aun así, la temperatura que alcanza el disipador del componente durante la carga de los condensadores es de casi 90ºC. Se trata de una temperatura muy elevada, pero encaja dentro del rango máxima de temperatura admisible del componente (125ºC).

6.3. RESULTADOS FINALES DE COMPORTAMIENTO

El objetivo final del sistema era controlar y registrar los datos de una motocicleta de competición durante la competición internacional Motostudent IV. Durante toda la competición la ECU cumplió con su trabajo al registrar de forma adecuada todos los parámetros deseados de los diferentes sistemas, contenidos en el mensaje de telemetría del apartado 5.4.1.2.

Estos mensajes se registraron con una frecuencia de 5Hz, permitiendo su recuperación posterior mediante la ejecución de un programa auxiliar, creado con este fin, en el microprocesador. Los datos se guardan en un archivo de texto plano,

Autor: **Javier Martínez Lahoz**
424.16.93

mostrado en la ilustración 28, para su posterior análisis mediante un software de instrumentación electrónica, como pueden ser LabView o Matlab.

```
Corriente       Cell_Temp       Mot_Temp        Par     Total_V SOC     RPMs    Vel
(-0.08, 29.0, 61.0, 0.0, 103.18000000000001, 80.0, 0.0, 0.0, 116368.0, 0.0, 3.98, 3.87)
(-0.1, 29.0, 61.0, 0.0, 103.18000000000001, 80.0, 0.0, 0.0, 116571.0, 0.0, 3.98, 3.87)
(-0.08, 29.0, 61.0, 0.0, 103.18000000000001, 80.0, 0.0, 0.0, 116774.0, 0.0, 3.98, 3.87)
(-0.08, 29.0, 61.0, 0.0, 103.18000000000001, 80.0, 0.0, 0.0, 116976.0, 0.0, 3.98, 3.87)
(-0.08, 29.0, 61.0, 0.0, 103.18000000000001, 80.0, 0.0, 0.0, 117180.0, 0.0, 3.98, 3.87)
(-0.1, 29.0, 61.0, 0.0, 103.18000000000001, 80.0, 0.0, 0.0, 117383.0, 0.0, 3.98, 3.87)
(-0.08, 29.0, 61.0, 0.0, 103.18000000000001, 80.0, 0.0, 0.0, 117586.0, 0.0, 3.98, 3.87)
```

Ilustración 28. Datos recibidos de la base de datos

Empleamos Matlab para representar estos datos de forma gráfica y así poder analizarlos más fácilmente. Gracias a estas gráficas, como la mostrada en la ilustración 29, se pudieron realizar ajustes de parámetros del regulador y controlar el estado de la batería, considerado el elemento más crítico de la moto. En la ilustración 29 se muestran la velocidad del motor, en color rojo; el par generado por el motor, en color verde; la velocidad de la motocicleta, en rojo; la corriente de la batería, en azul; y la tensión de la batería, en amarillo. Con esta gráfica podemos relacionar rápidamente el par generado con la corriente consumida y la bajada de tensión que produce dicha corriente en la batería, debido a la impedancia interna de la misma. También se puede observar como asciende la temperatura del motor debido a la gran corriente empleada.

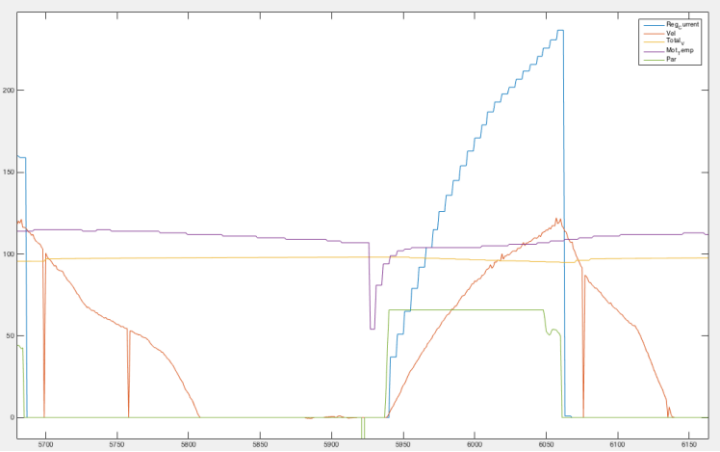

Ilustración 29. Gráficas obtenidas con MatLab con los datos de telemetría

Además podemos observar que algunas gráficas son más escalonadas que otras, esto se debe a que la frecuencia de los mensajes es distinta en función de su contenido, por lo que algunos datos, como la velocidad, se actualizan a una velocidad mucho mayor que otros, como por ejemplo la temperatura.

7. CONCLUSIONES

Podemos concluir que el diseño del sistema, tanto a nivel hardware como software ha sido un éxito general, ya que se han cumplido los objetivos propuestos, al lograr controlar y registrar la motocicleta durante las pruebas y la competición desarrolladas. El procesador y el controlador se han comportado de forma correcta trabajando de forma conjunta. El microcontrolador se encargaba de las tareas de bajo nivel y el microprocesador de las tares de alto nivel, la comunicación entre ambos ha sido fluida y se han podido transmitir los datos de telemetría para su salvado sin ningún problema.

Existen diversos aspectos del sistema que, en caso de realizar una revisión y/o actualización completa, sería conveniente modificar para optimizar el funcionamiento. Los problemas que han surgido durante el desarrollo del proyecto ya han sido investigados y se han propuesto una solución. Tras el análisis de los resultados también se podrían plantear una serie de mejoras de base sobre el sistema. Estas mejoras no consistirán en una sustitución o modificación directa, como las desarrolladas anteriormente, en su lugar se trata de cambios de planteamiento o tecnología que afectarían a todo el diseño.

Ilustración 30. Motocicleta de EUPLA Racing Team donde se instaló la ECU

A nivel hardware el sistema se ha comportado correctamente, todos los subsistemas se han comportado de la forma esperada, si bien ha sido necesario realizar algunas modificaciones. Todos los subsistemas han sido compatibles entre si y

Autor: **Javier Martínez Lahoz**
424.16.93

no ha sido necesario realizar grandes modificaciones, por lo que se puede concluir que el proceso general de la placa ha sido correcto. Podemos asegurar también que el diseño de los subsistemas auxiliares han sido todo un éxito. Ambos componentes auxiliares de alta tensión, el optoacoplador y el relé, han sido empleado y han facilitado el control de la motocicleta. Podemos concluir que el sistema a nivel hardware a funcionado correctamente, sin embargo cabe destacar que se trata de un sistema muy específico, que esta exclusivamente diseñado para nuestra motocicleta, mostrada en la ilustración 30, y probablemente no funcionaría de forma óptima en otro entorno.

A nivel software el diseño ha cumplido con sus funciones de control del sistema hardware. Los problemas que han ido surgiendo en el diseño software se han ido reparando rápidamente gracias a la gran velocidad de desarrollo que se logra al emplear el Arduino IDE y la programación en Python. El control del sistema ha resultado correcto, si bien como ya se vio anteriormente, el microcontrolador Atmega32U4 se encuentra cerca de los límites de saturación de memoria. La frecuencia de ejecución del programa era buena, sin embargo fue reducir los mensajes del bus CAN para que el funcionamiento fuera correcto. Además no se realizaba ningún análisis en tiempo real de la tendencia de datos de telemetría, tan solo de datos instantáneos.

Es lógico pensar que en ediciones posteriores de la competición Motostudent se volverá a diseñar un sistema de control electrónico, dicho sistema será una evolución del sistema desarrollado en este trabajo. Sería recomendable realizar algunas modificaciones para mejorar el funcionamiento y las capacidades del sistema en general.

Existen algunas modificaciones que se podrían realizar directamente sobre la placa. En primer lugar se debería idear un método para unir ambos canales de comunicaciones CANBus para que todos los sistemas estén conectados entre si, de esta forma la ECU no tendría que actuar como espejo entre ambos canales empleando dos controladores. Esto podría realizarse empleando dos transceptores contrapuestos para aislar las señales y unirlas, si bien sería necesario realizar pruebas.

Los dos componentes auxiliares de alta tensión han sido empleados, por lo tanto podemos asumir que en caso de realizar otra motocicleta, podrían surgir otras necesidades de entrada y salida. La conclusión que podemos obtener es que la adaptabilidad y la versatilidad del sistema resulta de vital importancia en este sistema.

Por lo tanto una de las opciones que se valora es la inclusión de una mayor cantidad de componentes auxiliares de propósito general.

Una de las modificaciones de base más importante que se valora para las siguientes versiones consiste en modificar los núcleos de procesamiento. En primer lugar, debido a la saturación del Arduino, se puede valorar su sustitución. Una de las opciones mejor valoradas en este caso pasa por la separación del procesamiento de bajo y alto nivel, empleando núcleos más específicos. Se podría emplear otro microcontrolador, por ejemplo un PIC, con una mayor cantidad de GPIOs y canales de comunicación para el control en bajo nivel, permitiendo una mayor cantidad de módulos adicionales. También se valora positivamente la opción de incluir un procesador de más potencia que permita desarrollar un interface intuitivo con el usuario, además de poder realizar un primer análisis de los datos en tiempo real, sin necesidad de exportar todos los datos a un software externo.

Autor: **Javier Martínez Lahoz**
424.16.93

8. BIBLIOGRAFÍA

8.1. REFERENCIAS BIBLIOGRÁFICAS

Hudson, J & Luecke, J. (1999). Basic Communications Electronics. Lincolnwoods, Ilinois.

Vodovozov, V. (2010). Introduction to Electronic Engineering. Bookboon

Weiji Wang. (2012). Introduction to Digital Signal and System Analysis. Bookboon

Pleite Guerra J., Vergaz Benito R., & Ruiz de Marcos JM.. (2009). ELECTRÓNICA ANALÓGICA PARA INGENIEROS. Aravaca (Madrid): McGraw-Hill/Interamericana de España, S.A.U.

Carter, R. (2010). Electromagnetism for Electronic Engineers. Bookboon

Dr. Naeem, W. (2009). Concepts in Electric Circuits. Bookboon.

8.2. OTRAS FUENTES CONSULTADAS

Firesmith, D. (11 nov 2013). Using V Models for Testing. Software Engineering Institute, Carnegie Mellon University. Recuperado de: https://insights.sei.cmu.edu/sei_blog/2013/11/using-v-models-for-testing.html

National Instruments (17 dic 2010). Isolation and Safety Standards for Electronic Instruments. Recuperado de: http://www.ni.com/white-paper/2827/en/

National Instruments (04 may 2015). Isolation Types and Considerations when Taking a Measurement. Recuperado de: http://www.ni.com/white-paper/3410/en/

Battery University (13 may 2016) How does a Supercapacitor Work? Recuperado de: http://batteryuniversity.com/learn/article/whats_the_role_of_the_supercapacitor

Comunidad Arduino (s.f.) Información acerca de Arduino. Recuperado de: https://www.arduino.cc/

Comunidad OpenWRT (s.f.): Información acerca de OpenWRT. Recuperado de: https://openwrt.org/

Comunidad SQLite (s.f.): Información acerca de SQLite. Recuperado de: https://sqlite.org/

Asociación Python España (s.f.) Información acerca de Python. Recuperado de: http://www.es.python.org/

Prometec.net (s.f.) Arduino y el Serial Peripheral Interface. Recuperado de: http://www.prometec.net/bus-spi/

National Marine Electronics Assotiation (s.f.) Información acerca de NMEA-0183. Recuperado de: https://www.nmea.org/

CAN in Automation (s.f.) Información de los protocolos CANBus. Recuperado de: https://www.can-cia.org/

CCM Benchmark Group (oct 2016) Las características del protocolo TCP. Recuperado de: http://es.ccm.net/contents/281-protocolo-tcp

8.3. DATASHEETS CONSULTADOS

Panasonic. (2008). LNJ926W8CRA, Light Emitting Diode

Panasonic. (2010). LNJ437W84RA, Hight Bright Surface Mounting Chip LED

Panasonic. (2011). LNJ237W82RA Hight Bright Surface Mounting Chip LED

Panasonic. (2008). LNJ326W83RA Ultra High Bright Surface Mounting Chip LED

Panasonic. (2012). ST RELAYS, 1a1b/2a 8A polarized power relays

Panasonic Electric Works Co. (s.f.). TX-S RELAYS

Multicomp. (2013). Thick Film Chip Resistor 0805

Multicomp. (2012). 2211S - MC34 Series

Microchip. (2012). MCP2515 Stand-Alone CAN Controller with SPI Interface

Texas Instruments. (2015). ISO1050 Isolated CAN Transceiver

Texas Instruments. (2015). LM117, LM317-N Wide Temperature Three-Pin Adjustable Regulator

Traco Power. (2013). DC/DC Converters TMA Series, 1 Watt

Tracopower. (2015). DC/DC Converters TEN 5WI Series, 6 Watt

Walsin Technology Corporation. (2015). MULTILAYER CERAMIC CAPACITORS General Purpose Series

TXC. (s.f.). Quartz Crystals SMD HC-49S 9C SERIES

NXP Semiconductors N.V. (2014). BC847 series

Avago Technologies. (2010). Very High CMR, Wide VCC Logic Gate Optocouplers

Diodes Incorporated. (2015). SBR1A40S1, 1A SBR® SUPER BARRIER RECTIFIER, DS33306

Diodes Incorporated. (2014). S1A/B - S1M/B 1.0A SURFACE MOUNT GLASS PASSIVATED RECTIFIER

TT Electronics (s.f.). Resistors, Anti Sulphur Chip Resistors, ASC Series

Tyco Electronics. (2005). Miniature PCB Relay PCJ

TE Connectivity Ltd. (2012). Miniature PCB Relay PCH

TE Connectivity. (2011). SMD Power Resistors Type 3520 Series

TE Connectivity. (2012). SMD High Power Precision Resistors Type RP73 Series

Vishay BCcomponents. (2016). 220 EDLC ENYCAP

STMicroelectronics. (2013). Enhanced single channel power switches TMPS2141, STMPS2151, STMPS2161, STMPS2171

Relación de documentos

(x) Memoria .....129 páginas

(_) Planos ..... 13 páginas

(_) Anexos ..... 17 páginas

La Almunia, a 29 de noviembre de 2016

Firmado: Javier Martínez Lahoz