UNIVERSITAT POLITÈCNICA DE VALÈNCIA

DEPARTAMENTO DE INGENIERÍA ELECTRÓNICA

DISEÑO, ANÁLISIS Y MONITOREO DE LA RED CON PROTOCOLO ISO 11992-2 EN REMOLQUES CON TERMÓGRAFOS APACHE

Autor: Jorge Andres Collaguazo Enriquez

Tutor: José Pelegrí Sebastiá

Tutor de empresa: Edwin Ordóñez Jiménez, PHD

Trabajo Fin de Máster presentado en el Departamento de Ingeniería Electrónica de la Universitat Politècnica de València para la obtención del Título de Máster Universitario en Ingeniería de Sistemas Electrónicos

Curso 2019-20

Valencia, Septiembre de 2020

Resumen

Hoy en día la adquisición, el control y el análisis de datos tiene muchas connotaciones, es un recurso que para cualquier empresa se muestra como una fuente sostenible de competitividad, para el caso, la red CAN-BUS que se implementa en los vehículos, maneja gran cantidad de datos tanto para diagnóstico, mantenimiento y centralización de sensores. Toda esta comunicación de datos robustece el correcto funcionamiento de cada sistema, ahora, para la comunicación es necesario un protocolo, el cual norme todo su funcionamiento, entre los cuales tenemos la ISO 11898 o la 11992, en los que nos centraremos en la investigación.

Se realizará un estudio comparativo para tener conocimiento acerca de los dispositivos similares en el mercado y compatibilidad con el equipo desarrollado en la empresa. En base a esto se pretende, como primera etapa, que el diseño del circuito tenga las características de leer el protocolo de la norma, poderse configurar en aquel modo o en lector de trama de un identificador en específico, todo por comandos AT. Este dispositivo debe ser capaz de comunicase con una trama de información definida compatible con el hardware Termógrafo APACHE Cold Tracer, mediante comunicación RS232 con el microprocesador principal. En el este dispositivo se debe leer las características de la trama, así como poderlo configurar, lo que implica una configuración en los servidores, programación web e interfaz gráfico para mostrar datos obtenidos en página web del monitoreo. Este trabajo va a permitir monitorizar valores como el activado de ABS, activado VDC, velocidad, alerta roja, alerta ámbar, Eje de Carga, Presión en los Neumáticos, revestimiento en el Freno, identificador en la rueda, temperatura del neumático, detección de fugas de aire, umbral de presión en los neumáticos, todos estos datos serán trasmitidos a la plataforma de visualización de la empresa para conocer el estado de los remolques.

Tras realizar todo el proceso, se ha podido demostrar las ventajas de nuestro sistema Apache para suministrar información del tráiler, para agilizar procesos y control de flota. Sin embargo, el sistema puede mejorarse con la actualización de software, ya que se encuentra preparado en hardware para la implantación de un GPS, lo que incrementaría las funciones y la versatilidad del sistema.

Resum

A dia de hui l'adquisició, el control i l'anàlisi de dades té moltes connotacions, és un recurs que per a qualsevol empresa es mostra com una font sostenible de competitivitat, per al cas, la xarxa CA-BUS que s'implementa en els vehicles, maneja gran quantitat de dades tant per a diagnòstic, manteniment i centralització de sensors. Tota aquesta comunicació de dades enrobusteix el correcte funcionament de cada sistema, ara, per a la comunicació és necessari un protocol, el qual regeix tot el seu funcionament, entre els quals tenim l'ISO 11898 o la 11992, en els quals ens centrarem en la investigació.

Es realitzarà un estudi comparatiu per a tindre coneixement sobre els dispositius similars en el mercat i compatibilitat amb l'equip desenvolupat en l'empresa. Sobre la base d'això es pretén, com a primera etapa, que el disseny del circuit tinga les característiques de llegir el protocol de la norma, poder-se configurar en aquella manera o en lector de trama d'un identificador en específic, tot per comandos AT. Aquest dispositiu ha de ser capaç de comunicara amb una trama d'informació definida compatible amb el maquinari Termògraf APACHE Cold Tracer, mitjançant comunicació RS232 amb el microprocessador principal. En l'est dispositiu s'ha de llegir les característiques de la trama, així com poder-ho configurar, la qual cosa implica una configuració en els servidors, programació web i interfície gràfica per a mostrar dades obtingudes en pàgina web del monitoratge. Aquest treball permetrà monitorar valors com l'activat d'ABS, activat VDC, velocitat, alerta roja, alerta ambre, Eix de Càrrega, Pressió en els Pneumàtics, revestiment en el Fre, identificador en la roda, temperatura del pneumàtic, detecció de fugides d'aire, llindar de pressió en els pneumàtics, totes aquestes dades seran transmesos a la plataforma de visualització de l'empresa per a conéixer l'estat dels remolcs.

Després de realitzar tot el procés, s'ha pogut demostrar els avantatges del nostre sistema Apatxe per a subministrar informació del tràiler, per a agilitar processos i control de flota. No obstant això, el sistema pot millorar-se amb l'actualització de programari, ja que es troba preparat en maquinari per a la implantació d'un GPS, la qual cosa incrementaria les funcions i la versatilitat del sistema.

Abstract

Today the acquisition, control and analysis of data has many connections, it is a resource that for any company is shown as a sustainable source of competitiveness, for that matter, the red CAN-BUS that is implemented in vehicles, manages large amount of data for diagnosis, maintenance and centralization of sensors. All this data communication strengthens the correct operation of each system, now, for communication, a protocol is necessary, which regulates all its operation, among which we have ISO 11898 or 11992, in which we will focus on research.

A comparative study will be carried out to gain knowledge about similar devices on the market and the compatibility with the equipment developed in the company. Based on this, it is intended as a first stage that the circuit design has the characteristics of reading the protocol of the standard, being able to configure in that mode or in a frame reader of a specific identifier, all by AT commands. This device must be communication layers with a defined transmission of information compatible with the APACHE Cold Tracer Thermograph hardware, through RS232 communication with the main microprocessor. In this device the characteristics of the plot must be read, as well as being able to configure, which implies a configuration on the servers, web programming and graphical interface to display selected data on the monitoring website. This work will allow monitoring of values such as active ABS, active VDC, speed, red warning, amber warning, axle load, tire pressure, brake lining, tire identification, tire temperature, air leak detection, tire pressure, to the company's viewing platform to find out the status of the trailers.

After completing the entire process, the advantages of our Apache system can be demonstrated to provide trailer information, to streamline processes and fleet control. However, the system can improve with the software update, since it is prepared in hardware for the implementation of a GPS, which increases the functions and versatility of the system.

Índice General

Índice de figuras.

Índice de Tablas.

1. Introducción

1.1 Motivación

En la actualidad las empresas que buscan establecerse en el nicho de algún servicio de forma exitosa, se enfrentan a una alta competitividad y a una constante toma de decisiones estratégicas, haciendo que la optimización de recursos sea daca vez más valioso. Tomando en cuenta que uno de los cimientos en el que se fundamenta la toma de decisiones es el conocimiento, la adquisición de INFORMACION toma un papel de alto precio.

La importancia en cuanto se habla de procesos operativos en las empresas, nos referimos a que cada parte de la misma ayuda a las en ventas, produzca, facture y crezca, lo que realza más la importancia de tener datos correctos, ya que de esto depende una decisión que lleve a la empresa a un alto grado de utilidades o a la quiebra.

En el contexto, para el proyecto se pretende implementar un dispositivo capas de adquirir información específica del estado de un remolque de frio, como el monitoreo del ABS, velocidad, presión en los neumáticos, carga en los ejes, alarmas, entre otras. Todos los datos adquiridos bajos las normas requeridas, se muestran acoplándolos al interfaz de usuario de la plataforma APACHE Cold Tracer.

El presente documento se estudia paso a paso como se ha desarrollado el mecanismo de adquisición de datos, su comprobación y vinculación al procesador central de información, el dispositivo será capaz de adquirir información de la trama CAN-BUS, procesarla, y estructurar una trama compuesta por los requerimientos de la mayoría de los usuarios a las que va dirigido según el estándar. La trama se acoplará al sistema ya realizado de Termografía Pache Cold Tracer, en consecuencia, la comunicación pasará de serie a RS232, lo primera que realizará sara una configuración por defecto y su comprobación, para después identificar el tipo de trama, procesarla, enviarla a los servidores y visualizarla dentro de la plataforma de la empresa.

Es importante recalcar que como punto de partida ya existe un sistema al cual le vamos a añadir la adquisición de los datos, lo que conlleva que debemos localizar la estructura de programación y un estructurar una trama, para el mejor acoplamiento a este dispositivo, teniendo en cuenta todos los parámetros antes mencionados, su implementación genera varias interrogantes, experimentos, co-validación e investigación propias para un el desarrollo de este trabajo de fin de Máster.

1.2 Objetivos

1.2.1 Objetivo Principal

- Diseñar un sistema de adquisición y monitoreo de datos de la Red CAN-Bus con Protocolo ISO 11992-2 en remolques con termógrafos APACHE.

1.2.2 Objetivos Específicos.

- Realizar una investigación acerca de mecanismos en el mercado, con funciones parecidas o que están en camino de desarrollo, que puedan suministrar el mismo servicio de adquisición de datos.
- Desarrollar un dispositivo de adquisición de tramas configurable por comandos AT, tal que tenga la opción de entregar una trama estructurada ISO11992-2 o configurar un identificador para obtener los datos de la misma.
- Implementar el hardware para la comunicación entre línea CAN-bus a serie TTL microprocesador, serie TTL microprocesador a RS232 del termógrafo APACHE y estar preparado para la escalabilidad de proyecto.
- Realizar el software de configuración automática y comprobación, en el dispositivo de Termógrafos Apache, además de la estructurar la trama que se envía a los servidores.
- Incorporar un sistema de visualización de las características de la trama, acoplado a los servicios que suministra la empresa de frio.

2. Fundamentos CAN-BUS

2.1 Aspectos generales

Por sus siglas Controller Area Network es un protocolo de comunicación tipo bus, que aplica su funcionamiento similar a la capa física y enlace de datos, comparándola con el modelo OSI. Entre sus características principales tenemos un sistema multimaestro, con una velocidad máxima a 1Mbps, el mensaje es enviado a todos los nodos, además de ofrecer una alta inmunidad a las interferencias, prioridad de mensajes, detección y señalización de errores [1].

2.1.1 Conceptos

El sistema utiliza un arbitraje basado en bits para resolver las colisiones, también la prioridad del identificador de un mensaje, para un mejor entendimiento, en la siguiente sección se explican conceptos básicos del funcionamiento del CAN, así como su clasificación según su extensión de trama en los siguientes grupos.

2.1.1.1 CAN Base

Como primer tipo de trama nos encontramos con el siguiente contenido:

SOF.- Bit de inicio de trama, el cual se utiliza para sincronizar el inicio del mensaje.

Identificador. - su función es la de establecer una prioridad del mensaje en el BUS, en cuanto más bajo sea el valor binario, mayor prioridad.

RTR. - Repetición de transmisión Remota establece un estado de Dominante (0) para trama de datos y recesivo (1) para tramas de peticiones remotas.

IDE. – es un bit de extensión de identificador que dominante (0) para el formato base.

r0.- bit reservado para su posible uso en futuras modificaciones.

DLC. - Es la longitud de datos de 4 bits

DATOS. – Transmitir 64 bits de información

CRC. – Comprobación de redundancia cíclica 16 bits

ACK. – El trasmisor emite Recesivo y el receptor emite dominante, más el delimitador

EOF. -Campo de fin de Trama, 7 bits

IFS. – Espacio Entre tramas de 7 Bits

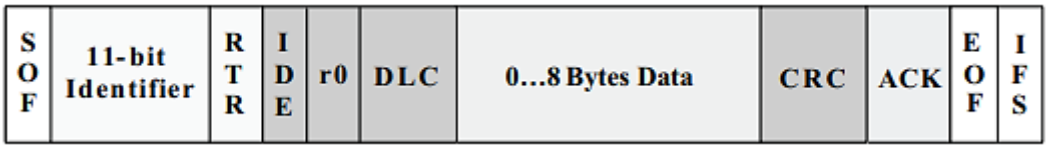


Figura 1.CAN base: Identificador 11-Bit [1].

2.1.1.2 CAN extendido

Para este tipo de trama se añaden algunos parámetros a la trama base, como:

SRR. – El bit de solicitud remota reemplaza al RTR para marcador de posición del formato extendido.

IDE. – Un bit en recesivo que indica que todavía le siguen 18 bits más al identificador

r1.- Incluye un bit de reserva adicional

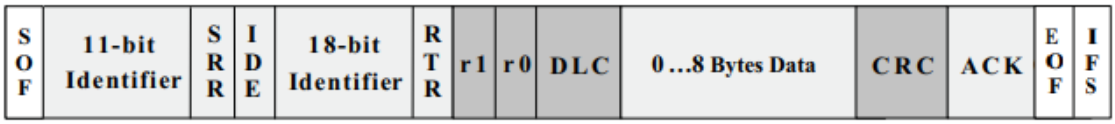


Figura 2.CAN extendido: Identificador 29-Bit [1].

2.1.2 Arquitectura.

Como se puede observar en la figura 2.2, se realiza una relación a los procesos de comunicación de datos con el modelo OSI, en el cual la capa aplicación se puede realizar con un DSP o un microcontrolador, la capa de enlace de datos con un controlador embebido, cabe mencionar que en el cual se debe configurar los filtros, máscaras y la velocidad de transmisión, la capa Física se relaciona con el transceiver can el cual se encarga de transformar las señales eléctricas para acoplarse al BUS.

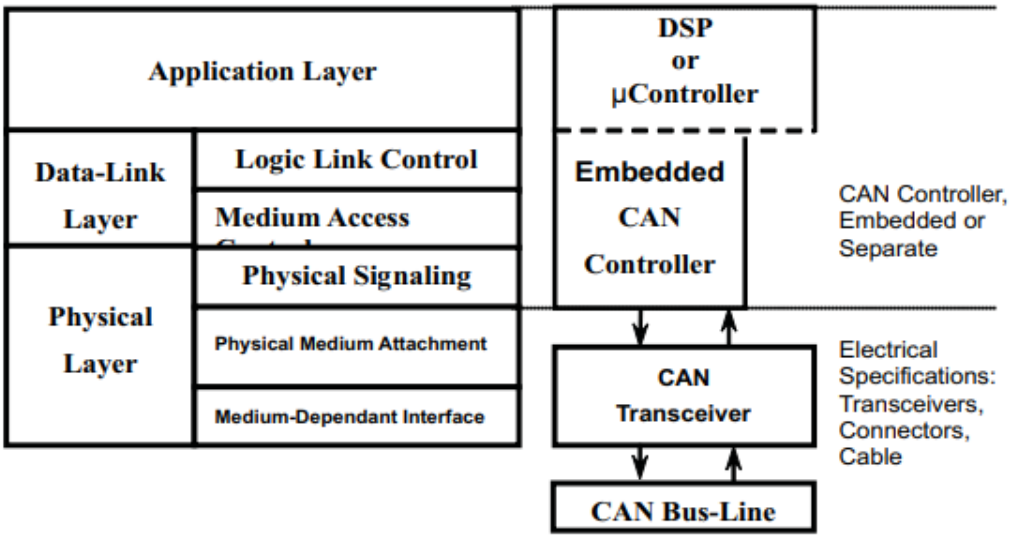


Figura 3.Arquitectura estándar de capas según ISO11898 [1].

2.1.3 Estándares

2.1.3.1 ISO 11898

Cubre la capa física y la capa de enlace de datos para las conexiones al sistema, controla el área de red, que admite control y multiplexación de su uso en de automóviles. La norma específica el formato de trama estándar, velocidad hasta 1Mbit/s, en la siguiente tabla se muestra la distribución de la especificación [2].

Tabla 1.Normativa y contenido desarrollado [2].

Estándar	Contenido
ISO 11898-1:2015	Parte1: Capa de enlace de datos y señalización física
ISO 11898-2:2016	Parte 2: unidad de acceso medio de alta velocidad
ISO 11898-3:2006	Parte 3: interfaz de baja velocidad, tolerante a fallas y dependiente del medio. Este estándar ha sido revisado y confirmado en 2015
ISO 11898-4:2004	Parte 4: Comunicación activada por tiempo. Este estándar ha sido revisado y confirmado en 2013
ISO 11898-5:2007	Parte 5: Unidad de acceso medio de alta velocidad con modo de bajo consumo
ISO 11898-6:2013	Parte 6: Unidad de acceso medio de alta velocidad con funcionalidad de activación selectiva
ISO 16845:2016	Plan de prueba de conformidad

2.1.3.2 ISO 11992-2

El estándar consta de las especificaciones para intercambio de información digital de las conexiones eléctricas entre remolques y vehículos remolcados, como punto de partida tenemos la capa física y enlace de datos, a continuación, se menciona la capa de aplicación para frenos y tren de rodaje, en la siguiente parte se especifica sobre los equipos que no sean los anteriores y finaliza con un diagnóstico.

El identificador que se utiliza es de 29 bits, la PDU (Protocol Data Unit) tiene siete campos además de los específicos para el CAN que se pueden ver en la figura 4.

	P	R	DP	PF	PS	SA		Data field
Bits	3	1	1	8	8	8		0 to 64

Figura 4. PDU Extendido [3].

Los Campos son la Prioridad(P), Reservado(R), Pagina de Datos (DP), Formato de PDU(PF), PDU Specifico (PS), dirección de origen (SA) y campó de datos.

2.1.3.2.1 Prioridad

Estos tres bits se utilizan para optimizar la latencia de los mensajes. La prioridad más alta es 0 (000_2) y la más baja 7 (111_2), para los mensajes orientados al control es 3 (011_2), y los mensajes informativos son 6 (110_2).

2.1.3.2.2 Bit Reservado

Este bit se encuentra reservado para futuras expansiones, será cero para los mensajes transmitidos.

2.1.3.2.3 Página de Datos

Selecciona una página auxiliar de los grupos de parámetros

2.1.3.2.4 Formato de PDU

Este campo de ocho bits determina el formato de PDU, determinando el número de grupo de parámetros asignados al campo de datos.

2.1.3.2.5 PDU especifico

El campo específico de PDU es un campo de ocho bits y depende del formato de PDU. Dependiendo del formato PDU, puede ser una dirección de destino o una extensión de grupo. Si el valor del campo de formato PDU (PF) es inferior a 240, entonces el campo específico de PDU es una dirección de destino. Si el valor del campo PF es 240 a 255, entonces la PDU específica.

	PDU format (PF) field	PDU-specific (PS) field
PDU 1 field	0 to 239	Destination address
PDU 2 field	240 to 255	Group extension

Figura 5. Campo de PDU específico [3].

2.1.3.2.6 Dirección de origen

Tiene una longitud de ocho bits, esta dirección debe ser única en la red, lo que asegura que el identificador será único.

2.1.3.2.7 Campo de Datos

La longitud de datos son ocho bytes de datos, dentro de los cuales se menciona todas las características requeridas en el sistema dentro del estándar, y que se detallan en el estándar ISO mencionado [3].

2.2 Dispositivos en competencia directa

2.2.1 Thermo King

Es una de las marcas competidoras, sus características empresariales en servicios son similares a nuestra empresa APACHE Cold Tracer, capaz de entregar gestión de temperatura homologada con dos dispositivos, denominados registrador y recopilador de datos, los cuales están enfocados al almacenamiento de temperatura y humedad relativa [4]. Con mucha diferencia, el proyecto desarrollado como TFM, que toma la red CAN-BUS con la normativa ISO 11992-2 para adquirir las tramas y monitorizar el estado del vehículo, amplia las ventajas de la empresa sobre la competencia.

Figura 6. Registrador de datos USB[4].

2.2.2 ColdTrans

La marca posee un dispositivo capaz de realizar comunicación remota de temperatura homologas, configuración de horas UTC, informes de temperatura, posee características iguales hasta el momento que APACHE [5], con variación al momento de servicio al cliente, sin embargo, para ampliar el servicio, no adquiere datos de la columna principal del sistema CAN, es decir no implementa un lector de trama, de esta manera el proyecto gana innovación en el mercado respecto a su competencia directa.

Figura 7.Campo de PDU [5].

2.2.3 Ruptela

La empresa se dedica por completo a telemática, por ende, ofrecen una amplia gama de servicios base, estos se pueden acoplar de forma puntual a los servicios que se pretenden implementa en el proyecto, sin embargo, existen varios matices a tomar en cuenta, la necesidad de implementar, el microprocesador encargado de la tabulación-Contador de las tramas, que conlleva un tiempo de proceso, además también incrementa los costes, tanto en hardware como en software [6].

Figura 8. Dispositivo FmPRO4 [6].

3. Recursos y Metodología

3.1 Recursos

Dentro del siguiente apartado se detalla los materiales que se necesitan en mínima instancia en el desarrollo del proyecto.

3.1.1 Software

3.1.1.1 QT Creator

Es un IDE multiplataforma, que posee el editor de código C++, herramientas de administración, depurador, cómoda configuración para incorporar el programador de microcontrolador, en sí, es una herramienta sumamente versátil e intuitiva, en el programa es donde se desarrolla el algoritmo de configuración automática del lector CAN.

3.1.1.2 KICAD

Es un programa de código abierto para la automatización de diseño electrónico, en el cual se realiza la esquematización, con la ventaja de la fácil incorporación de símbolos, huellas y un entorno de rápido aprendizaje, reúne características de rutado de pistas y de capas basto, lo suficiente para el desarrollo del instrumento de medición CAN, una función muy interesante es la incorporación del visor 3D, para inspeccionar detalles del lienzo, obteniendo una idea final del producto muy confiable.

3.1.1.3 Arduino

El entorno de código abierto donde se incorpora las librerías y el programa principal para adquirir y procesar las tramas de CAN con el estándar, su interfaz de fácil configuración, alta flexibilidad y sus múltiples versiones de hardware, hacen que sea una herramienta eficaz para manipular código.

3.1.1.4 Termite

El terminal permite la configuración de un puerto serial, lo utilizamos para poder visualizar los resultados en la trama, testear la trama, además de poder configurar los parámetros del Apache Box por medio de comandos AT.

3.1.1.5 Programación por ISP

Viene de las siglas “In Systen Programming”, es la habilidad de algunos microcontroladores para ser programado mientras está instalado en un sistema completo. De esta forma podemos se puede programar el microcontrolador instalado en la placa de conexión.

3.1.2 Hardware

3.1.2.1 Termógrafo Apache (STM32)

El dispositivo muestra la temperatura de dos sondas, contiene un microcontrolador STM32, impresión de las temperaturas, guardado de temperatura durante un año, configuración del tiempo de recolección de datos, visualización de parámetros, su programación es homologada para registradores de temperatura para el trasporte, almacenamiento y distribución de productos sensibles, norma UNE-EN 12830, dentro de sus configuraciones existen 3 tipos, la primera, tipo Core, que se enfoca a los vehículos frigoríficos de poco tamaño, puede ser colocado en el salpicadero de la cabina, el segundo es tipo tráiler el cual se emplea en camiones en la parte exterior, por lo que necesita protección IP68, en sí tiene las mismas características que el anterior pero mayor protección contra golpes y un diseño más elaborado, el tercer tipo se encuentra en desarrollo para poder medir un mayor número de sondas de temperatura y de presión, con protocolo one wire

3.1.2.2 ATMEGA328P

El microcontrolador tiene como características, 8 bits, una arquitectura RISC, interface ISP, 1Kbyte de EEPROM, ADC de 10 bits, Watchdog programable, voltaje de operación 2.7 hasta 5.5, un puerto USART [7]. Consigo tiene las características suficientes para implementar el lector de trama CAN por ISP y trasmitirlo hacia el termógrafo por serial.

3.1.2.3 ATMEGA2560

El microcontrolador posee similares características, pero ampliando sus funciones tales como EEPROM de 4 Kbytes, 4 USARTs la cual es la principal característica por la que fue escogido, en consecuencia, obtenemos un dispositivo capaz de gestionar varias entradas y salidas seriales.

3.1.2.4 Programador USBASP ISP

Es un programador para microcontroladores Atmel, su funcionamiento es a través de un ATMEGA 8, es multiplataforma y fácil de utilizar. Entre sus especificaciones se menciona la lectura y escritura de EEPROM, Firmware, fuse bits y bits de bloqueo.

3.2 Metodología

3.2.1 Punto de Partida

Como punto de partida tenemos que investigar el concepto de los estándares que se desea establecer en el dispositivo lector, se debe generar una red CAN BUS en donde se simula la transferencia del estándar, debemos comprobar que la red simulada contiene todos los requerimientos del bus CAN.

3.2.2 Esquematización de la estructura

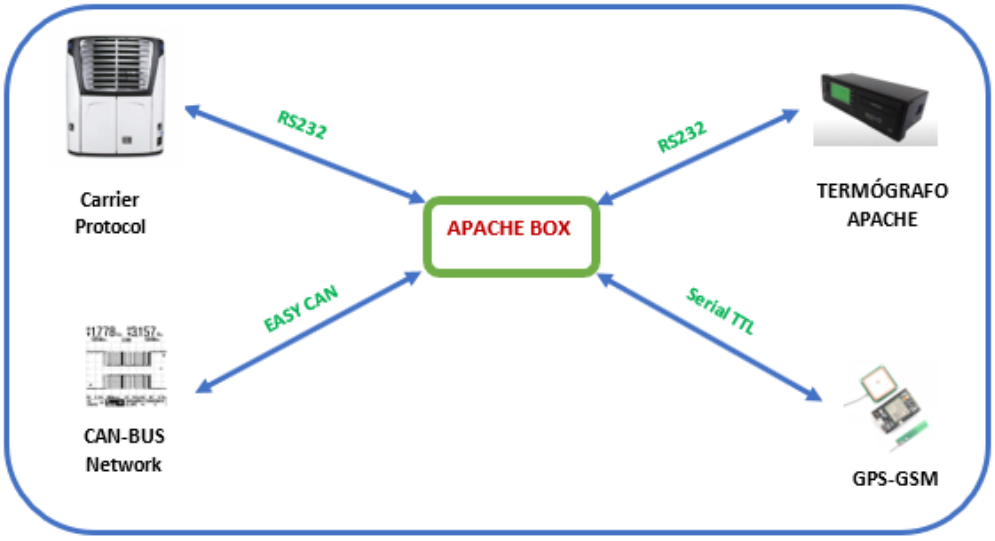


Figura 9. Mecanismo de conexión hardware.

Apache BOX establece las conexiones básicas, en el presente trabajo se desarrolla la vinculación Can Bus con el protocolo 11992-2. Sin embargo, el hardware se encuentra preparado para añadir varios servicios tales como como la conexión a GPS, GSM y lectura de la trama de protocolos Carrier.

4. Desarrollo

4.1 Red para simulación

4.1.1 Librería CAN

De forma que se pueda aprovechar la adquisición de datos se toma como ejemplos los programas implementados dentro de la librería, de esta forma podemos editarlos para trabajar de acuerdo a las necesidades del sistema.

En la tabla 2 nos muestra una calculadora para determinar los registros de configuración para el integrado, para facilidad, pero también se debe tener en cuenta las especificaciones del data sheet.

Tabla 2. Configuración del registro en MCP2515.

MCP2515 Configuration Register Calculator											
Crystal Frequency	16	MHz	Baud Register Values					Misc Register Values			
Baud Rate	250	kbs	Baud Rate Prescaler (BRP)			1		PS2 Bit Time Mode (BTLMODE)			1
			Propogation Segment			1		Sample Mode (SAM)			1
Time Quantum	0,25	microseconds	Phase Segment 1 (PS1)			6		Start-of Frame Signal (SOF)			1
Bit Time	4	microseconds	Phase Segment 2 (PS2)			5		Wake-Up Filter (WAKFIL)			0
Time Quantum Max	16		Synchronization Jump Width (SJW)			1					
Prescaler Setting	1		Adjust BRP To Get A Nice Whole Time Quantum Max Value								
SJW	2	x T_Q	PropSeg + PS1 >= PS2					Bit Sample Point Occurs Between PS1 and PS2			
SyncSeg	1	x T_Q	PropSeg + PS1 >= T_{DELAY}					Typically, the T_{DELAY} is 1-2 T_Q			
PropSeg	2	x T_Q	PS2 > SJW								
PS1	7	x T_Q	MCP2515 Baud Registers								
PS2	6	x T_Q	msb								lsb
	16	Total T_Q	7	6	5	4	3	2	1	0	
Register CNF1	41	01000001	0	1	0	0	0	0	0	1	
			SJW		BRP						
Register CNF2	F1	11110001	1	1	1	1	0	0	0	1	
			BTLMODE	SAM	PS1			Prop			
Register CNF3	85	10000101	1	0	0	0	0	1	0	1	
			SOF	WAKFIL	Reserved “000”			PS2			

En la figura 10, se muestran las diferentes funciones que la librería nos ofrece, de esta forma podemos editar el algoritmo para nuestro fin.

```
*  CAN operator function
*********************************************************************************************************/

    INT8U setMsg(INT32U id, INT8U rtr, INT8U ext, INT8U len, INT8U *pData);        // Set message
    INT8U clearMsg();                                                              // Clear all message to zero
    INT8U readMsg();                                                               // Read message
    INT8U sendMsg();                                                               // Send message

public:
    MCP_CAN(INT8U _CS);
    INT8U begin(INT8U idmodeset, INT8U speedset, INT8U clockset);       // Initialize controller parameters
    INT8U init_Mask(INT8U num, INT8U ext, INT32U ulData);               // Initialize Mask(s)
    INT8U init_Mask(INT8U num, INT32U ulData);                          // Initialize Mask(s)
    INT8U init_Filt(INT8U num, INT8U ext, INT32U ulData);               // Initialize Filter(s)
    INT8U init_Filt(INT8U num, INT32U ulData);                          // Initialize Filter(s)
    INT8U setMode(INT8U opMode);                                        // Set operational mode
    INT8U sendMsgBuf(INT32U id, INT8U ext, INT8U len, INT8U *buf);      // Send message to transmit buffer
    INT8U sendMsgBuf(INT32U id, INT8U len, INT8U *buf);                 // Send message to transmit buffer
    INT8U readMsgBuf(INT32U *id, INT8U *ext, INT8U *len, INT8U *buf);   // Read message from receive buffer
    INT8U readMsgBuf(INT32U *id, INT8U *len, INT8U *buf);               // Read message from receive buffer
    INT8U checkReceive(void);                                           // Check for received data
    INT8U checkError(void);                                             // Check for errors
    INT8U getError(void);                                               // Check for errors
    INT8U errorCountRX(void);                                           // Get error count
    INT8U errorCountTX(void);                                           // Get error count
    INT8U enOneShotTX(void);                                            // Enable one-shot transmission
    INT8U disOneShotTX(void);                                           // Disable one-shot transmission
    INT8U abortTX(void);                                                // Abort queued transmission(s)
    INT8U setGPO(INT8U data);                                           // Sets GPO
    INT8U getGPI(void);                                                 // Reads GPI
};

#endif
```

Figura 10. Opciones de Can-BUS [10].

En la figura 11 se muestra la conexión de forma se pueda establecer la conexión CAN, importante que el valor de las resistencias de terminación coincida con la impedancia característica del bus, definida en 120 Ω, para evitar reflexiones en la línea que podrían perturbar la comunicación. Con esta configuración la velocidad del bus es de un máximo de 1 Mbit/s.

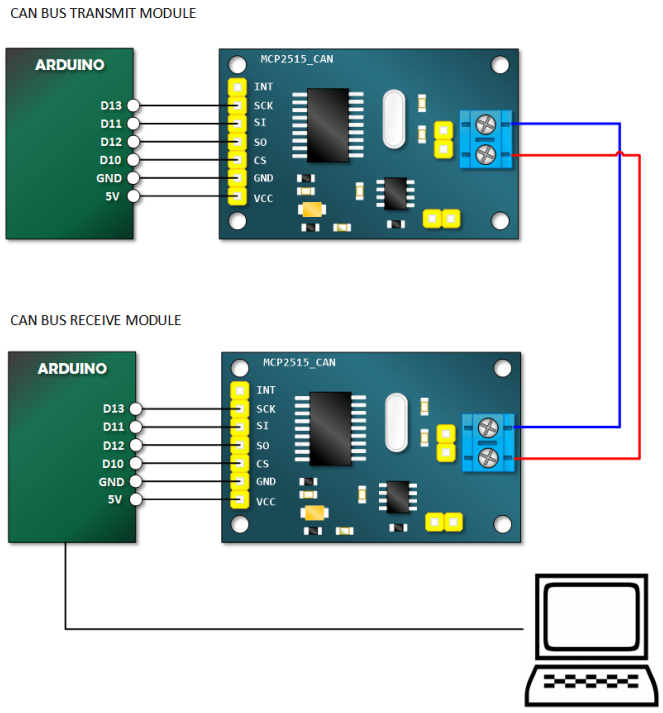


Figura 11. Hardware Red CAN BUS[10].

4.1.1.1 Verificación

Para el proceso de Verificación se realiza el análisis de la trama de forma visual, conectado las líneas CAN al osciloscopio, de esta forma se comprueba que la red trabaja a los voltajes correctos (CAN-H 3.5v y CAN-L 2.5v), ya que CAN trabaja en líneas diferenciales la codificación dentro de la red se muestra en la figura 12:

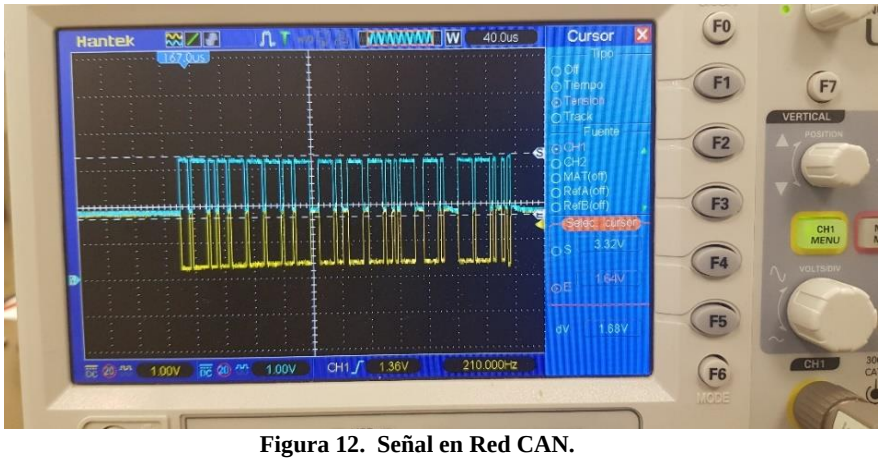


Figura 12. Señal en Red CAN.

Para realizar la adquisición de datos se tomó la decisión de utilizar 2 transceiver para la capa física del controlador y el controlador CAN.

El primer transceiver se encarga de transformar los voltajes de la red CAN a valores TTL observados en la figura 13 y el segundo de realizar la transformación de TTL a valores de RS232

observados en la figura 14, de esta manera podemos obtener todos los datos de la red, sin embargo, necesitamos desarrollar el control perfecto de la trama ya que estarían llegando al termógrafo todos los bits de tramas sin discriminar ninguno, en las imágenes a continuación se muestra como se ha realizado las pruebas con el dispositivo.

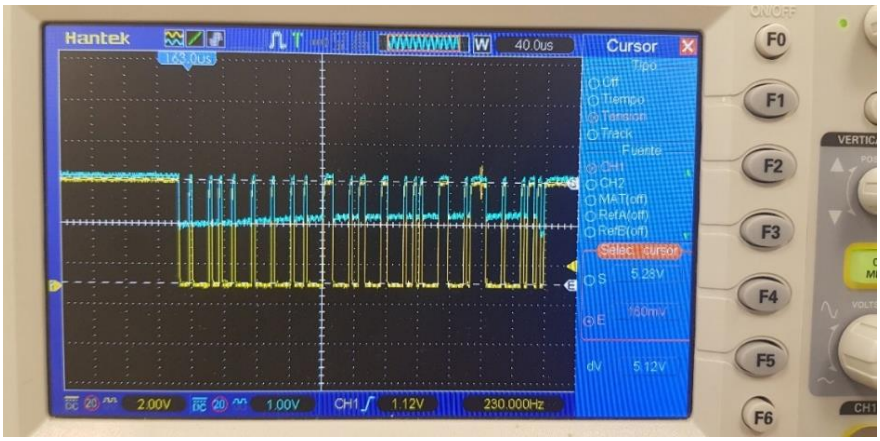


Figura 13. Señal TTL 0v - 5v.

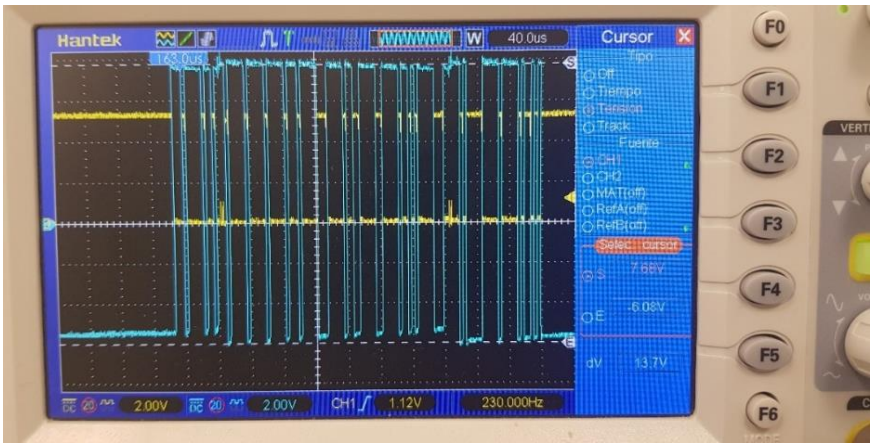


Figura 14. Señal RS232.

4.1.2 Emisor

En la figura 15 se muestra la configuración necesaria para el envío de la trama. Como punto inicial tenemos la inclusión de las librerías tanto para la configuración del lector CAN por el bus SPI (Serial Peripheral Interface), establecido como master, dentro del Setup, inicializamos la comunicación y enviamos la configuración básica, para que el periférico esclavo actúe con una velocidad de transmisión de 500 Kbs y la velocidad del oscilador de 16MHz. Como constantes declaramos vectores que contienen datos reales el estándar según la ISO 11992-2 adquiridos en la prueba de Campo, según los identificadores de trama se asigna el contenido y lo enviamos a la red cada 100ms, el código se observa en el anexo 8.1.

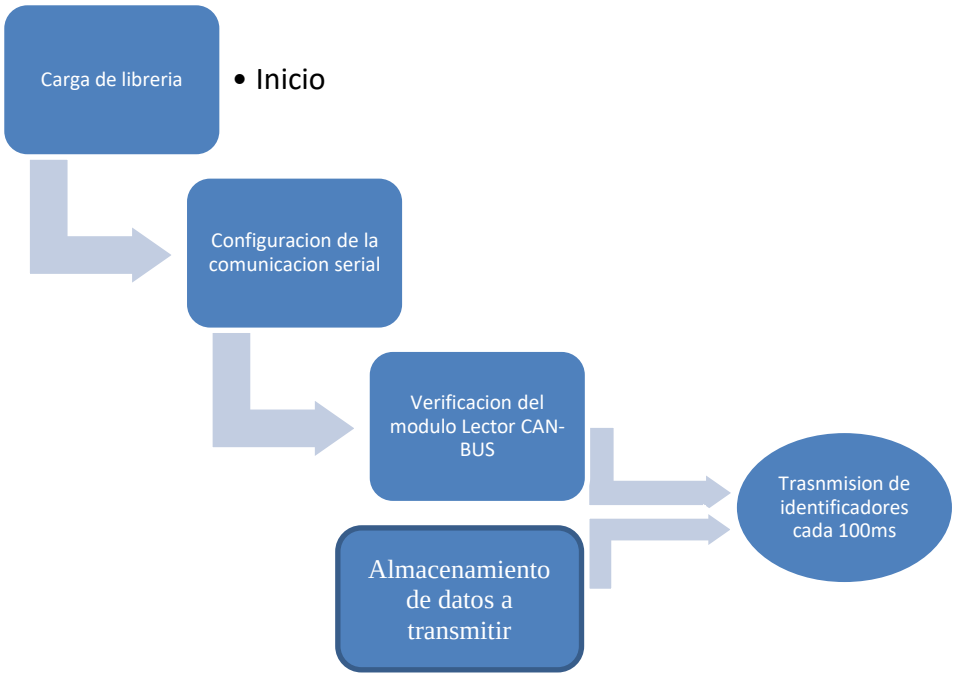


Figura 15. Algoritmo de envío a la red.

4.1.3 Receptor

En la parte de setup es idéntica a la del emisor, con respecto al bloque de repetición, comienza con la lectura del puntero al identificador, longitud y el buffer, de esta forma se puede identificar qué tipo de trama corresponde, la de 29 bits (extendida) o la de 11 bits (estándar), para finalizar mostramos el resultado del contenido de los datos, como respuesta tenemos el contenido de todas las tramas con sus respectivos identificadores, el código se observa en el anexo 8.2 y que se muestran en la pantalla del terminal en la figura 17.

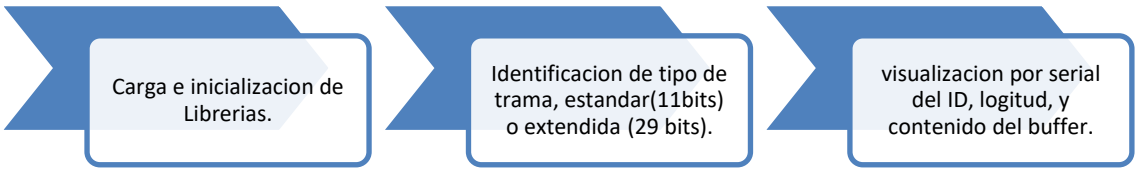


Figura 16. Algoritmo de recepción de la trama.

```
Extended ID: 0x18FE5EC8     DLC: 8  Data: 0x17 0x40 0x25 0xFF 0xFF 0xEA 0x00 0x00
Extended ID: 0x18FEC4C8     DLC: 8  Data: 0x00 0x14 0x00 0x00 0xA5 0x39 0x00 0x00
Extended ID: 0x18FEC6C8     DLC: 8  Data: 0x05 0x17 0x00 0x00 0x4E 0x00 0x00 0x88
Extended ID: 0x18FE5CC8     DLC: 8  Data: 0x00 0x00 0x00 0x00 0x19 0xAB 0xCD 0x00
Extended ID: 0x18FE5EC8     DLC: 8  Data: 0x17 0x40 0x25 0xFF 0xFF 0xEA 0x00 0x00
Extended ID: 0x18FEC4C8     DLC: 8  Data: 0x00 0x14 0x00 0x00 0xA5 0x39 0x00 0x00
Extended ID: 0x18FEC6C8     DLC: 8  Data: 0x05 0x17 0x00 0x00 0x4E 0x00 0x00 0x88
Extended ID: 0x18FE5CC8     DLC: 8  Data: 0x00 0x00 0x00 0x00 0x19 0xAB 0xCD 0x00
Extended ID: 0x18FE5EC8     DLC: 8  Data: 0x17 0x40 0x25 0xFF 0xFF 0xEA 0x00 0x00
Extended ID: 0x18FEC4C8     DLC: 8  Data: 0x00 0x14 0x00 0x00 0xA5 0x39 0x00 0x00
Extended ID: 0x18FEC6C8     DLC: 8  Data: 0x05 0x17 0x00 0x00 0x4E 0x00 0x00 0x88
Extended ID: 0x18FE5CC8     DLC: 8  Data: 0x00 0x00 0x00 0x00 0x19 0xAB 0xCD 0x00
Extended ID: 0x18FE5EC8     DLC: 8  Data: 0x17 0x40 0x25 0xFF 0xFF 0xEA 0x00 0x00
Extended ID: 0x18FEC4C8     DLC: 8  Data: 0x00 0x14 0x00 0x00 0xA5 0x39 0x00 0x00
Extended ID: 0x18FEC6C8     DLC: 8  Data: 0x05 0x17 0x00 0x00 0x4E 0x00 0x00 0x88
Extended ID: 0x18FE5CC8     DLC: 8  Data: 0x00 0x00 0x00 0x00 0x19 0xAB 0xCD 0x00
Extended ID: 0x18FE5EC8     DLC: 8  Data: 0x17 0x40 0x25 0xFF 0xFF 0xEA 0x00 0x00
Extended ID: 0x18FEC4C8     DLC: 8  Data: 0x00 0x14 0x00 0x00 0xA5 0x39 0x00 0x00
Extended ID: 0x18FEC6C8     DLC: 8  Data: 0x05 0x17 0x00 0x00 0x4E 0x00 0x00 0x88
Extended ID: 0x18FE5CC8     DLC: 8  Data: 0x00 0x00 0x00 0x00 0x19 0xAB 0xCD 0x00
Extended ID: 0x18FE5EC8     DLC: 8  Data: 0x17 0x40 0x25 0xFF 0xFF 0xEA 0x00 0x00
Extended ID: 0x18FEC4C8     DLC: 8  Data: 0x00 0x14 0x00 0x00 0xA5 0x39 0x00 0x00
Extended ID: 0x18FEC6C8     DLC: 8  Data: 0x05 0x17 0x00 0x00 0x4E 0x00 0x00 0x88
Extended ID: 0x18FE5CC8     DLC: 8  Data: 0x00 0x00 0x00 0x00 0x19 0xAB 0xCD 0x00
Extended ID: 0x18FE5EC8     DLC: 8  Data: 0x17 0x40 0x25 0xFF 0xFF 0xEA 0x00 0x00
```

Figura 17. Visualización de identificadores.

4.2 Dispositivo Lector Comercial

4.2.1 Descripción y resultados Longan Serial CAN BUS

Una vez analizada la documentación, se procede a verificar el funcionamiento, se analiza los requerimientos para adquirir datos de la red CAN bus, también se evalúa el funcionamiento de dispositivo comercial, configurando para acoplase, entender y visualizar todos los identificadores de trama dentro de la red.

El dispositivo comercial tiene las siguientes funciones: Comunicación UART a CAN Bus, compatibilidad multiplataforma hardware, configuración de UART hasta 115200 baudios, velocidad CAN hasta 1 Mb/s, máscaras y filtros reconfigurables. Todas sus configuraciones se realizan a través de comandos AT, como lo muestra la tabla 3.

Tabla 3. Comandos de configuración AT.

CMD	Descripción
+++	Cambiar del modo normal al modo de configuración
AT + S = [valor]	Establecer velocidad de transmisión en serie
AT + C = [valor]	Establecer la velocidad de transmisión del bus CAN
AT + M = [N] [EXT] [valor]	Establecer máscara
AT + F = [N] [EXT] [valor]	Establecer filtro
AT + Q	Cambiar al modo normal

Como podemos comprobar tiene una fácil configuración, de esta forma se realizan las pruebas Base de lectura CAN Bus en la empresa Knorr-Bremse, su estructura se puede observar en la figura [18], se configura el dispositivo como un filtro PASA todo, de esta forma se tiene acceso a todos los datos de la trama, en lo posterior se tabulan los datos de trama a través de filtros, configurando el Identificador se hallan los datos de la tabla 4.

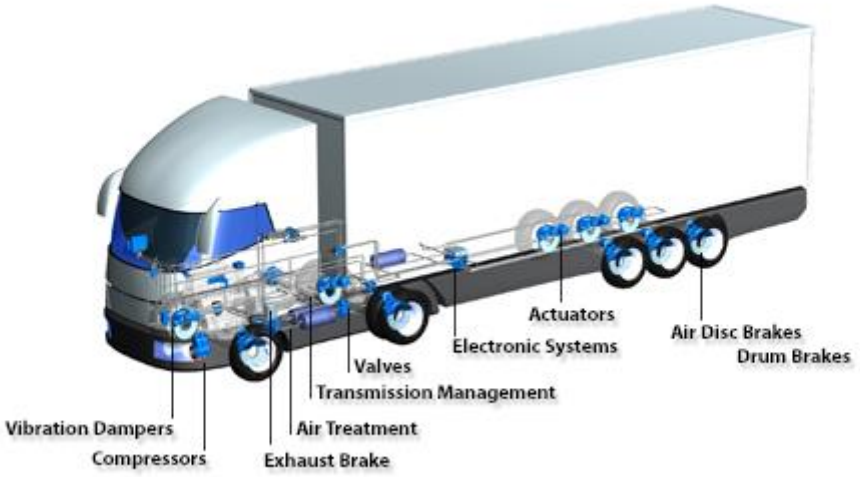


Figura 18. Maquina Knorr- Bremse [12].

Tabla 4. Resumen de los datos adquiridos de la trama CAN-BUS.

DATOS EBS 21	DATOS EBS 22	DATOS EBS 23
Search "0C 03 20 C8" (7478 hits in 1 files)	**Search "18 fe c4 c8" (750 hits in 1 files)**	**Search "18 fe c6 c8" (746 hits in 1 files)**
Line 4: 0c 03 20 c8 cc f0 00 00 ff 00 7d ff	Line 7: 18 fe c4 c8 cc d4 c1 43 ff ff ff ff	Line 253: 18 fe c6 c8 7f ff ff ff ff ff ff 97
Line 19: 0c 03 20 c8 cc f0 00 00 ff 00 7d 7d	Line 94: 18 fe c4 c8 cc d5 c5 43 90 34 ff ff	Line 409: 18 fe c6 c8 7f ff ff ff ff ff ff 97
Line 29: 0c 03 20 c8 cc f0 00 00 ff 00 7d 7d	Line 173: 18 fe c4 c8 cc d5 c5 43 aa 34 ff ff	Line 486: 18 fe c6 c8 7f ff ff ff ff ff ff 97
Line 36: 0c 03 20 c8 cc f0 00 00 ff 00 7d 7d	Line 251: 18 fe c4 c8 cc d5 c5 43 b3 34 ff ff	Line 563: 18 fe c6 c8 7f ff ff ff ff ff ff 97
Line 43: 0c 03 20 c8 cc f0 00 00 ff 00 7d 7d	Line 330: 18 fe c4 c8 cc d5 c5 43 b3 34 ff ff	Line 640: 18 fe c6 c8 7f ff ff ff ff ff ff 97
Line 51: 0c 03 20 c8 cc f0 00 00 ff 00 7d ff	Line 407: 18 fe c4 c8 cc d5 c5 43 aa 34 ff ff	Line 717: 18 fe c6 c8 7f ff ff ff ff ff ff 97
Line 60: 0c 03 20 c8 cc f0 00 00 ff 00 7d ff	Line 484: 18 fe c4 c8 cc d5 c5 43 aa 34 ff ff	Line 797: 18 fe c6 c8 7f ff ff ff ff ff ff 97
Line 68: 0c 03 20 c8 cc f0 00 00 ff 00 7d ff	Line 561: 18 fe c4 c8 cc d5 c5 43 99 34 ff ff	Line 884: 18 fe c6 c8 7f ff ff ff ff ff ff 97
Line 75: 0c 03 20 c8 cc f0 00 00 ff 00 7d ff	Line 638: 18 fe c4 c8 cc d5 c5 43 99 34 ff ff	Line 967: 18 fe c6 c8 7f ff ff ff ff ff ff 97

4.3 Diseño primer prototipo

4.3.1 Estructura Software

Dentro de la estructura para el desarrollo software se implementa una máquina de estados, el estado inicial A, establece la configuración inicial de dispositivo lector, como la velocidad de transmisión CAN-BUS, tipo de configuración, tipo de Oscilador utilizado y la verificación de que retorne un ok para guardar la configuración en la EEPROM.

El estado B es el encargado de procesar el identificador de trama, realizar las condiciones necesarias para escoger los bits con los que se estructura la trama "T0", además de realizar los cálculos de la decodificación según el estándar ISO 11992-2, para observarlos en una forma adecuada por el puerto serie.

El estado C tiene la función de administrar los comandos AT y determinar la configuración deseada, los comandos se mencionan en la tabla 5.

Tabla 5. Comandos de configuración del Módulo.

Comando AT	Acción
+++	Ingresa al modo Configuración
AT+OP=	0→(trama tipo ISO 91122) 1→(trama datos ID extendido)
AT+ID=	0x00000000(29bits)
AT+CK	Comando de conexión
AT+RST	Reseteo del componente
AT+Q	A modo datos

El estado D contiene la opción para adquirir la trama datos con ID extendido y visualizar su contenido. En la figura 19 se muestra el diagrama de la máquina de estados con sus transiciones

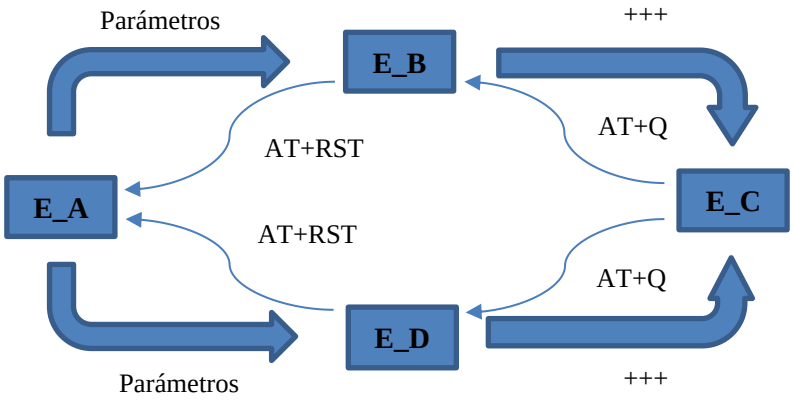


Figura 19. Diagrama de estados.

4.3.1.1 Construcción de trama

Para construir la trama debemos guiarnos en el PDU del estándar, de esta forma se puede codificar el identificador, adquirir los datos de trama entrante y decodificarlos para un mejor entendimiento.

A continuación, se menciona la estructura de cálculo de los valores de constructor de trama.

- **Identificador de trama EBS21**

Se refiere al mensaje del Vehículo remolcado, el sistema electrónico de frenos muestra las siguientes características

- Tiempo de repetición de transmisión: 10 ms ± 1 ms
- Longitud de datos: 8 bytes
- Página de datos: 0
- Formato PDU: 3
- PDU específica: dirección del predecesor
- Prioridad predeterminada: 3
- Prioridad = 0C
- PDU formato = 03
- PDU específico = 20
- Destino = C8

En la tabla 6 se muestra un ejemplo de trama CAN-BUS extendida, a la cual se realiza los caculos pertinentes, para que el microprocesador entregue datos entendibles de forma comprensible.

Tabla 6. Ejemplo resumen de trama EBS21.

IDENTIFICADOR (29 BITS)	**Data 1 (8 Bytes)**							
0C 03 20 C8	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**	**Byte 8**
HEX	cc	f0	00	00	ff	00	7d	7d
BIN	1100 1100	1111 0000	0000 0000	0000 0000	1111 1111	0000 0000	0111 1101	0111 1101

-Byte 1 Estado del sistema del vehículo remolcado 1

Bits 1 a 2 ABS del vehículo activo / pasivo = Pasivo

Definición parámetro Vehículo ABS: Señal que indica que el ABS está activo / pasivo. La señal se activa cuando el ABS comienza a modular la presión del freno de la rueda, y se restablece a pasivo cuando todas las ruedas están en condiciones estables durante un cierto período de tiempo.

La señal también se puede activar cuando las ruedas motrices están en alto deslizamiento (por ejemplo, causadas por el retardador).

00 - ABS del vehículo pasivo, pero instalado

01 - Vehículo ABS activo

-Byte 2 Estado del sistema de vehículo remolcado 2

Bits 1 a 2 VDC activo = pasivo

Definición parámetro VDC: Señal que indica que el Control Dinámico del Vehículo (VDC) está activo / pasivo. VDC contiene Prevención de vuelco (ROP) o Control de guiñada (YC) o ambos.

00 - VDC pasivo, pero instalado

01 - VDC activo

-Bytes de 3 a 4 ruedas basadas en la velocidad del vehículo

Bytes 3 a 4 = 0

Definición parámetro velocidad del vehículo basada en ruedas (del sistema de frenos): Velocidad real del vehículo se calculada como el promedio de las velocidades de las ruedas de un eje influenciado por el deslizamiento y filtrado por un rango de frecuencia de 5 Hz a 20 Hz.

Longitud de datos: 2 bytes

Resolución: 1/256 km/h / bit de ganancia, 0 km/h de compensación

Rango de datos: 0 km/h a 250 km / h

- **Identificador de trama EBS22**

- Tiempo de repetición de transmisión: 100 ms ± 10 ms
- Longitud de datos: 8 bytes
- Página de datos: 0
- Formato PDU: 254
- PDU específica: 196
- Prioridad predeterminada: 6
- Prioridad = 18
- PDU_format = FE
- PDU_especific = C4
- Destino = C8

En la tabla 7 se muestra la trama del identificador EBS22.

Tabla 7. Ejemplo resumen de trama EBS22

IDENTIFICADOR (29 BITS)	**Data 1 (8 Bytes)**							
18 FE C4 C8	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**	**Byte 8**
HEX	cc	d5	c5	43	89	32	ff	ff
BIN	1100 1100	1101 0101	1100 0101	0100 0011	1000 1001	0011 0010	1111 1111	1111 1111

-Byte 2 Estado del vehículo remolcado 3

Bits 3 a 4 Advertencia roja = señal de advertencia roja encendida

Definición parámetro advertencia roja: Solicitud del vehículo remolcado al vehículo comercial para activar la señal de advertencia roja en el vehículo comercial, que indica ciertas fallas específicas dentro del equipo de frenado de los vehículos remolcados.

00 - No se indica la falla del vehículo remolcado por la señal de advertencia roja

01 - El vehículo remolcado no se indica mediante la señal de advertencia roja

Bits 5 a 6 Advertencia ámbar = señal de advertencia en ámbar encendida

Definición parámetro advertencia ámbar: Solicitar del vehículo remolcado al vehículo comercial para activar la señal de advertencia ámbar en el vehículo comercial.

00 - No se indica la falla del vehículo remolcado por la señal de advertencia ámbar

01 - Falla del vehículo remolcado a ser indicada por la señal de advertencia ámbar

-Bytes 5 a 6 Suma de carga por eje

Definición de la suma de las cargas verticales estáticas de los ejes del vehículo:
Longitud de datos: 2 bytes
Resolución: ganancia de 2 kg / bit, desplazamiento de 0 kg
Rango de datos: 0 kg a 128 510 kg

HEX = 3289

DEC = 12937 * 2 kg / bit = 25874 kg

➢ **Identificador de trama EBS23**

- Se requiere enviar este mensaje.
- Tiempo de repetición de transmisión: 100 ms ± 10 ms
- Longitud de datos: 8 bytes
- Página de datos: 0
- Formato PDU: 254
- PDU específica: 198
- Prioridad predeterminada: 6
- Prioridad = 18
- PDU_format = FE
- PDU_especific = C6
- Destino = C8

En la tabla 8 se muestra la trama del identificador EBS23.

Tabla 8. Ejemplo resumen de trama EBS23.

IDENTIFICADOR (29 BITS)	**Data 1 (8 Bytes)**							
18 FE C6 C8	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**	**Byte 8**
HEX	7C	17	FF	FF	FF	FF	FF	95
BIN	0111 1100	0001 0111	1111 1111	1111 1111	1111 1111	1111 1111	1111 1111	1001 0101

-Byte 1 Estado del sistema del vehículo remolcado

Bits 1 a 2 Presión de los neumáticos suficiente / insuficiente = Presión de los neumáticos insuficiente

Definición del parámetro de la Presión de los neumáticos: Señal que indica que la presión de los neumáticos es insuficiente, es decir, fuera del rango de presión recomendado por el fabricante del neumático o del vehículo para garantizar una operación optimizada con respecto al consumo de combustible del vehículo y la vida útil del neumático.

00 - Presión de los neumáticos insuficiente

01 - Presión de los neumáticos suficiente

Bits 3 a 4 Recubrimiento de freno suficiente / insuficiente = No disponible

Definición de recubrimiento de frenos: Señal que indica que el recubrimiento del freno es suficiente / insuficiente.

00 - Recubrimiento de freno insuficientes

01 - Recubrimiento de freno suficientes

-Byte 2 Identificación de llanta / rueda (presión)

RESULTADO = 1,7

El número de identificación especifica la posición del neumático o la rueda en cada eje (Bit 1 a Bit 4) y el número de ejes que comienzan desde la parte delantera del vehículo remolcado respectivo (Bit 5 a Bit 8).

El número de identificación se usa junto con toda la información relacionada con el neumático, la rueda o el extremo de la rueda en el mensaje. El número de identificación "0" se utilizará si no se puede identificar la posición del neumático, la rueda, el extremo de la rueda o el eje. Como se puede observar en la figura 20.

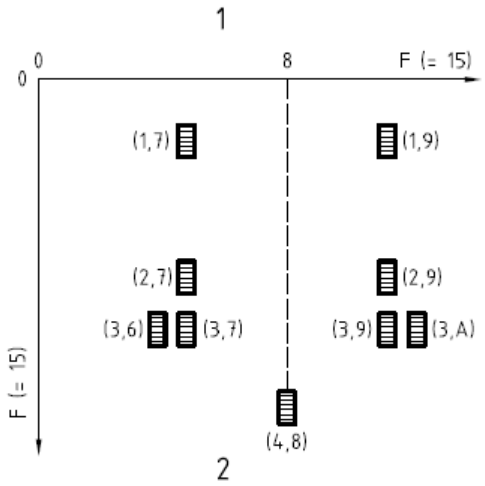


Figura 20. Identificación de rueda.

-Byte 5 Presión de los neumáticos

Presión real de los neumáticos sin correcciones.

Longitud de datos: 1 byte

Resolución: ganancia de 10 kPa / bit, desplazamiento de 0 kPa

Rango de datos: 0 kPa a 2 500 kPa

RESULTADO = no disponible

-Byte 8 Presión de suministro neumático

Presión de suministro real del depósito del sistema de frenado.

Longitud de datos: 1 byte

Resolución: ganancia de 5 kPa / bit, desplazamiento de 0 kPa

Rango de datos: 0 kPa a 1 250 kPa

RESULTADO

HEX = 95

DEC = 149 * 5 kPa / bit = 745KPa = 7.45bar

- **Identificador de trama RGE22**

- Tiempo de repetición de transmisión: 100 ms ± 10 ms
- Longitud de datos: 8 bytes
- Página de datos: 0
- Formato PDU: 254
- PDU específica: 92
- Prioridad predeterminada: 6
- Prioridad = 18
- PDU_format = FE
- PDU_especific = 5C
- Destino = C8

En la tabla 9 se muestra la trama del identificador RGE22.

Tabla 9. Ejemplo resumen de trama RGE22.

IDENTIFICADOR (29 BITS)	**Data 1 (8 Bytes)**							
18 FE 5C C8	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**	**Byte 8**
HEX	FF	FF	01	7D	10	9A	4D	FF
BIN	1111 1111	1111 1111	0000 0001	0111 1101	0001 0000	1001 1010	0100 1101	1111 1111

-Byte 5 Identificación de llanta / rueda (carga)

RESULTADO = 1,0

-Bytes 6 a 7 carga por eje

Longitud de datos: 2 bytes

Resolución: ganancia de 2 kg / bit, desplazamiento de 0 kg

Rango de datos: 0 kg a 128 510 kg

RESULTADO

HEX = 4D9A

DEC = 19866 * 2 kg / bit = 39 732 kg

- **Identificador de trama RGE23**

- Tiempo de repetición de transmisión: 1000 ms ± 100 ms
- Longitud de datos: 8 bytes
- Página de datos: 0
- Formato PDU: 254
- PDU específica: 94
- Prioridad predeterminada: 6
- Prioridad = 18
- PDU_format = FE
- PDU_especific = 5E
- Destino = C8

En la tabla 10 se muestra la trama del identificador EBS23.

Tabla 10. Ejemplo resumen de trama EBS23.

IDENTIFICADOR (29 BITS)	Data 1 (8 Bytes)							
18 FE 5E C8	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**	**Byte 8**
HEX	19	00	25	FF	FF	EA	07	49
BIN	0001 1001	0000 0000	0010 0101	1111 1111	1111 1111	1110 1010	0000 0111	0100 1001

-Byte 1 Identificación de llanta / rueda (temperatura)

RESULTADO = 1,9

-Bytes 2 a 3 Temperatura del neumático

La temperatura medida por el módulo del neumático.

Longitud de datos: 2 bytes

Resolución: 0,03125 ° C / ganancia de bit, - 273 ° C de compensación

Rango de datos: - 273 ° C a 1 735 ° C

RESULTADO

HEX = 2500

DEC = (9472 * 0,03125 ° C / ganancia de bit) - 273 ° C = 23 ° C

-Bytes 4 a 5 detección de fugas de aire

La pérdida de presión de un neumático.

Longitud de datos: 2 bytes

Resolución: 0,1 Pa /s/bit de ganancia, 0 Pa /s de compensación

Rango de datos: 0 Pa /s a 6 425,5 Pa /s

RESULTADO = NO disponible

-Byte 6 Función de tren de rodaje del vehículo remolcado 5

Bits 1 a 3 Detección del umbral de presión de los neumáticos = 010

Señal que indica el nivel de presión del neumático.
000 - Sobrepresión extrema
001 - Sobrepresión
010 - Sin advertencia de presión
011 - Bajo presión
100 - Extremo bajo presión
101 - No definido
110 - Indicador de error
111 - No disponible

4.3.1.2 Construcción de trama con identificadores

En la figura 21 se muestra el diagrama de flujo de la construcción de la trama, de forma que exista orden y coherencia entre el proceso de adquisición de datos, proceso y envió de la trama final para gestionarlo al termógrafo APACHE.

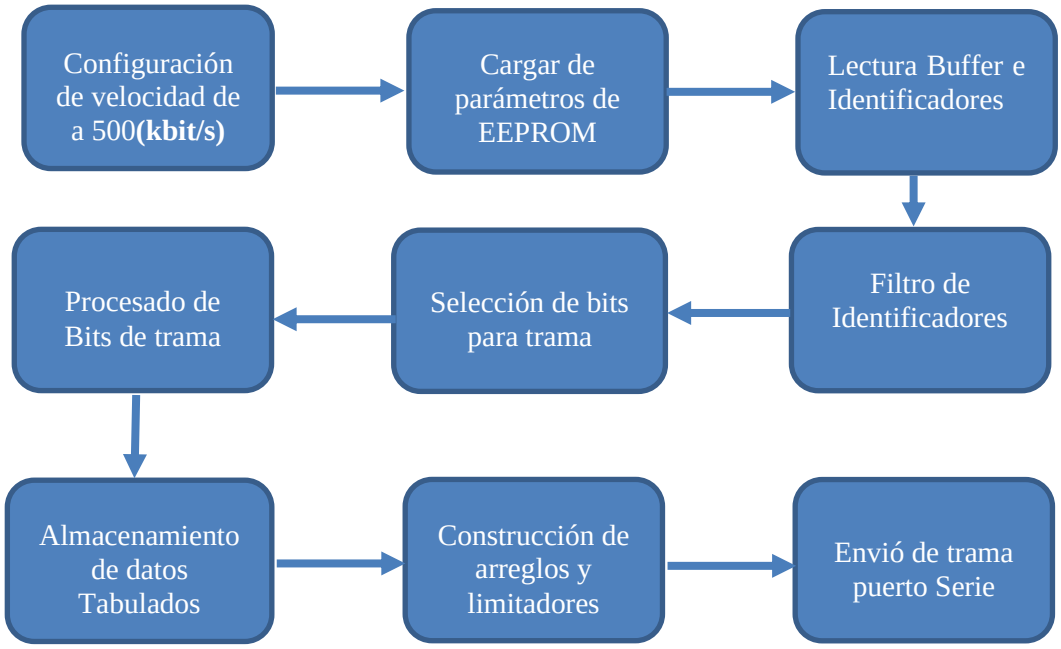


Figura 21. Diagrama de la construcción de trama.

Para construir la trama se procede a implementar un mecanismo para procesar el contenido de los identificadores, visto en la figura 22.

```
void switch_IDs()
{
 switch (rxId & 0x1FFFFFFF)
    {
    case 0x0C0320C8:
      ID1_EBS21();
      break;
    case 0x18FEC4C8:
      ID2_EBS22();
      break;
    case 0x18FEC6C8:
      ID3_EBS23();
      break;
    case 0x18FE5CC8:
      ID4_RGE22();
      break;
    case 0x18FE5EC8:
      ID5_RGE23();
      break;
    default:
      //Serial.print("otros");
      break;
      }
}
```

Figura 22. Estructura de sección de identificador.

Para el procesado de los bits que conforman la trama, las operaciones de filtros y mascaras se pueden visualizar en el Anexo 8.3, en él se establece el algebra necesaria para cada identificador, seleccionar los bits del buffer y añadirlos a la trama principal.

Cuando ser a realizado la adquisición de datos de la red, se procede a delimitar cada uno de los parámetros y preparar la trama de salida, como lo muestra la figura 23.

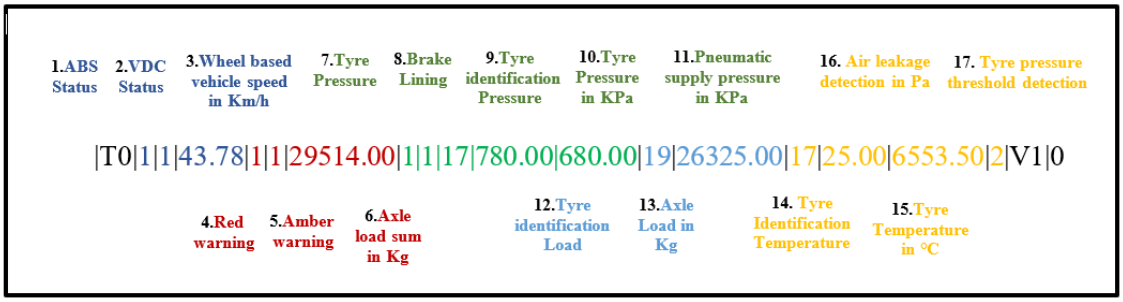


Figura 23. Estructura de trama entregada.

4.3.2 Diseño Hardware

En la figura 24 se realiza una placa de conexión entre el microcontrolador Arduino nano y el módulo que adquiere de datos CAN-BUS, la comunicación se realiza a través de ISP, de esta manera se envía errores en conexión.

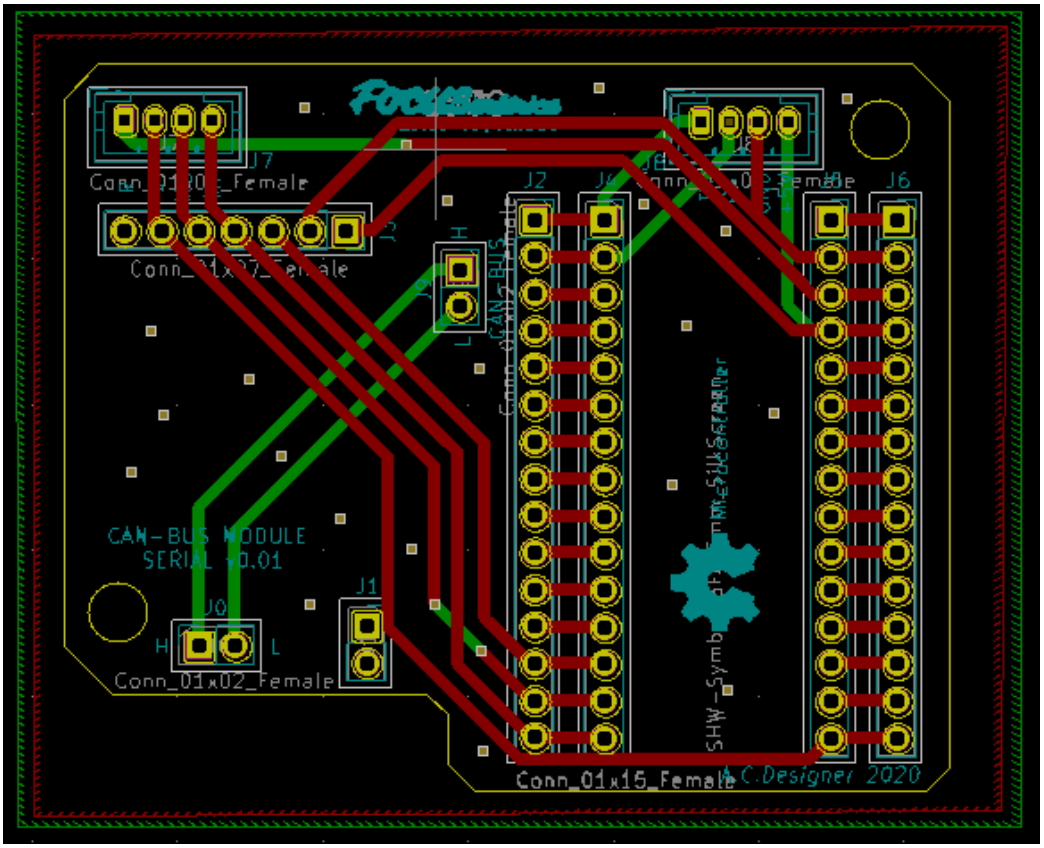


Figura 24. Procesador y módulo CAN- BUS.

4.4 Diseño Segundo prototipo

4.4.1 Diseño Hardware

Para el rediseño, se trata de incorporar en una misma PCB el mecanismo de adquisición de la trama CAN y un módulo GPS, también se quita por USB, reemplazándola por ISP, también se incorpora la activación por del GPS por código, en la figura 25.

Figura 25. Esquemático Lector CAN-Bus desarrollado.

Para quemar el firmware en el microcontrolador, configuramos el IDE Arduino para que genere el archivo con extensión hex con el contenido del bootloader y el programa principal. En la figura 36 podemos observar la configuración bits de fuses, su significado se menciona en la tabla 11, para profundizar en la configuración es conveniente revisar siempre el datasheet.

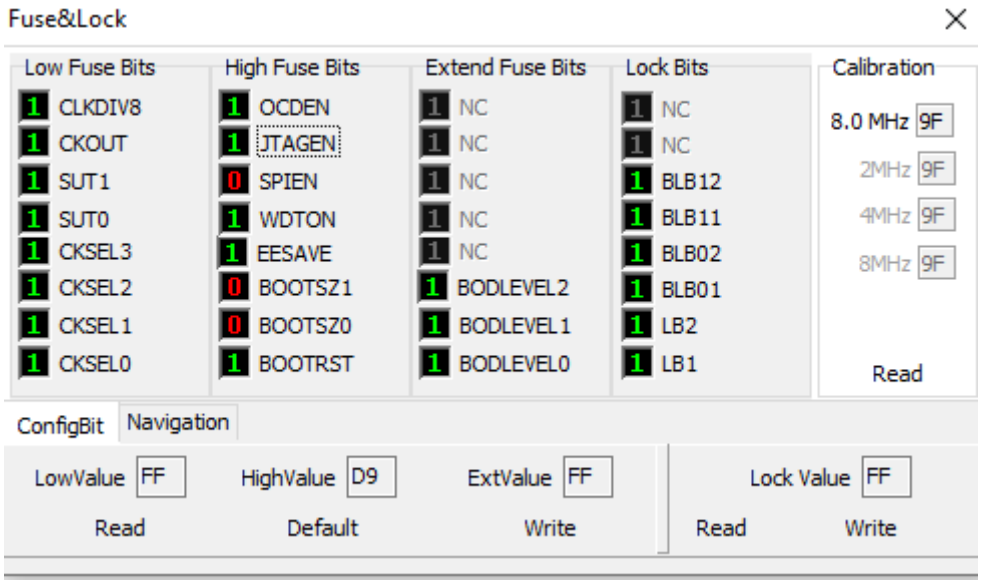


Figura 26. Fuses de Configuración ATMEGA328p.

Tabla 11. Configuración de fuses.

FUSES	ATmega328p
Low Fuse	
CLKDIV8	Desactivada la división del reloj para 8.
CKOUT	Desactivada la salida del reloj.
SUT1, SUT0	Tiempo de inicio en 14ck+65ms.

CKSEL3, CKSEL2, CKSEL1, CKSEL0	Se encuentra configurado con oscilador de cristal externo, con estos bits se puedo configurar el tipo de reloj.
High Fuse	
OCDEN	Debug system desactivado.
JTAGEN	Programación por JTAG desactivada.
SPIEN	Programación por SPI Activada.
WDTON	Watchdog Timer desactivado.
EESAVE	Guardar el contenido de la EEPROM
BOOTSZ1, BOOTSZ0	2048 words Flaz aplication: 0x0000-0x37FF Flaz BootLoader: 0x3800-0x3FFF
BOOTRST	Aplicación reset en la dirección 0x0000.
Extend Fuse	
BODLEVEL2, BODLEVEL1, BODLEVEL0	Desactivamos la protección de rangos de voltajes para reinicia.
Lock	Todos desactivados.

En la figura 27 se puede observar la distribución de la configuración ISP, salidas RS232, entrada para el GPS/GSM, 2 salidas de interrupciones, entrada CAN-BUS. De esta manera se puede incorporar la conexión al termógrafo.

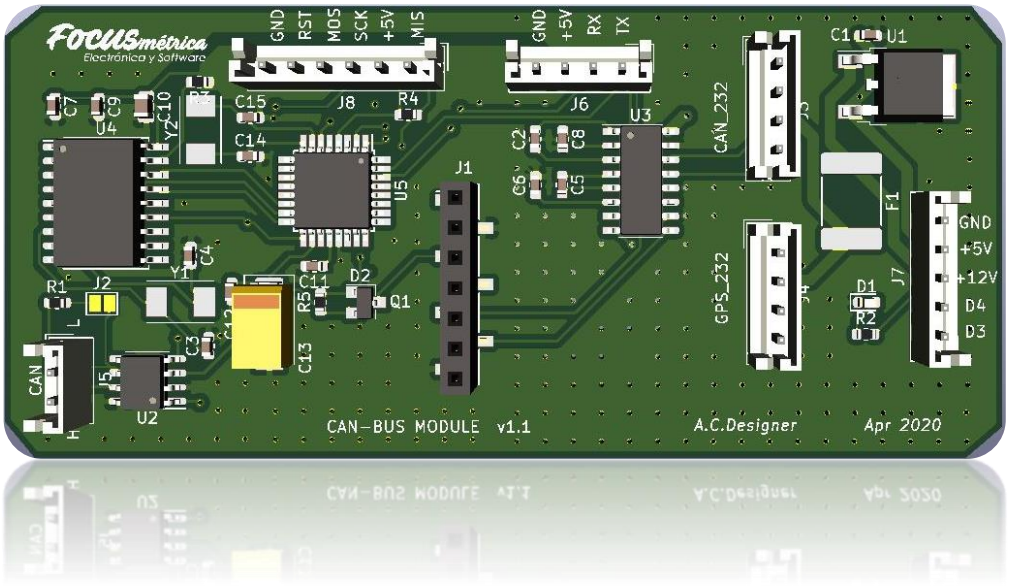


Figura 27. Rediseño configuración por ISP.

4.5 Diseño final con cambio de microcontrolador

4.5.1 Diseño Hardware

Para el diseño final se trata de incorporar el hardware necesario para implementar varias funciones a futuro, como los datos de la trama CARRIER, TERMO KING, un puerto RS232 puente para comunicación directa con el termógrafo, lector de voltaje suministrado, lectura de GPS/GSM y la Lectura de la trama CAN-BUS ISO 11992-2. Para lo cual se decidió escoger una placa

controladora que tenga varios puertos de comunicación serie vista en la figura 28, las características se muestran en la Tabla 12.

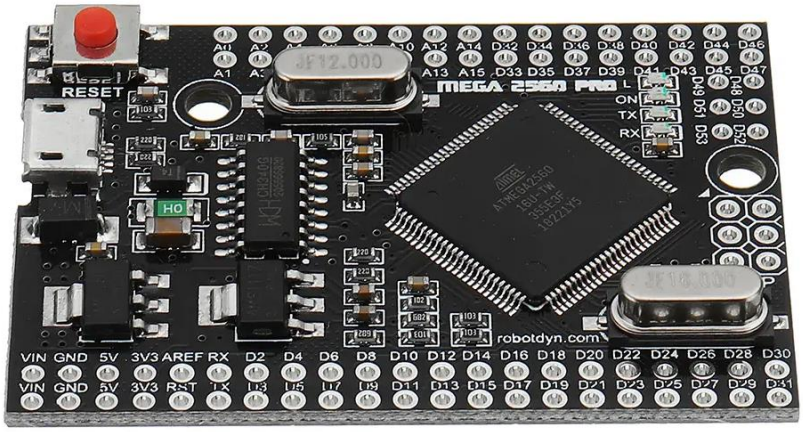


Figura 28. ATMEGA 2560 PRO.

Tabla 12. Característica de la Placa controladora.

Microcontrolador	ATmega2560
Convertidor USB-TTL	CH340
Sin electricidad	5V-800mA
Potencia en.	5V
Potencia en. VIN / DC Jack	5V
El consumo de energía	5V 220mA
Nivel lógico	5V
Puertos Serie	4 Usarts
USB	Micro USB
Reloj Frecuencia	16MHz
Voltaje de suministro operativo	5V
E / S digital	54
E / S analógica	16
Tamaño de la memoria	256kb
Datos RAM Tipo / Tamaño	8Kb
Datos ROM Tipo / Tamaño	4Kb
Interfaz Tipo	ISP
Temperatura de funcionamiento	−40 ° / + 85 °
longitud x Ancho	38 × 54mm

La configuración de fuses para el nuevo microcontrolador varia un poco, hay que tener en cuenta la distribución en la memoria flash como dirección de memoria, donde se coloca el bootloader y el programa principal, la configuración se puede observar en la en la figura 29.

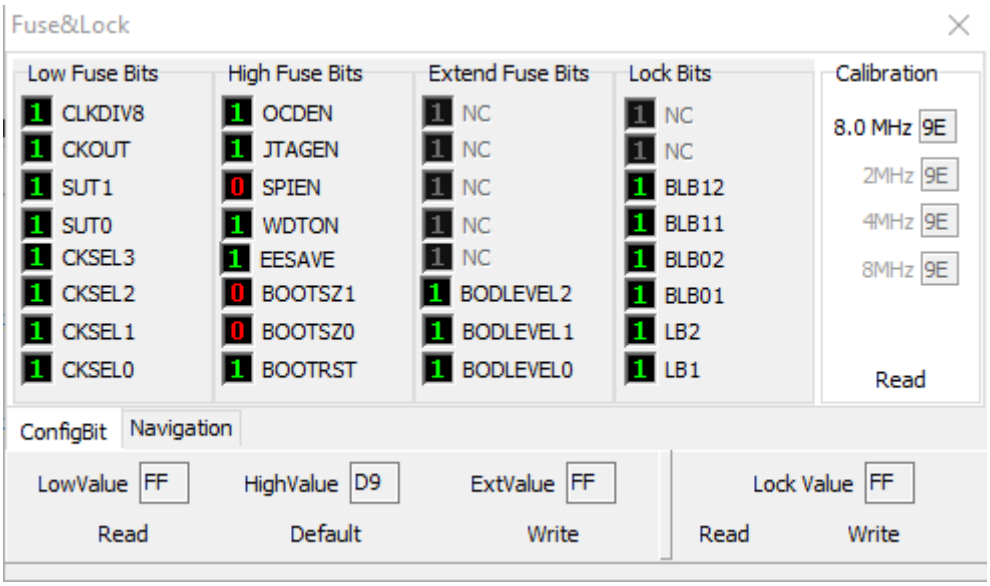


Figura 29. Fuses de configuración ATMEGA2560.

Para la fácil depuración del programa, se configura el programa Atmel Studio, de forma que se puedan abrir, compilar y cargar archivos de Arduino dentro de su interfaz. Para esto, se ingresa al IDE, al menú Tools, clicamos en External Tools,visto en la figura 30, en el apartado de comandos se debe colocar la dirección donde se encuentra el programa con el que quema la memoria flash del microcontrolador, en el apartado de argumentos se coloca las órdenes de configuración de proceso, tales como el path de dirección al cual le configuramos la versión del microcontrolador, el puerto COM, la velocidad de comunicación, el proceso de escritura, y se finaliza dando el path del archivo compilado, los parámetros a insertar se encuentran en la tabla 28.

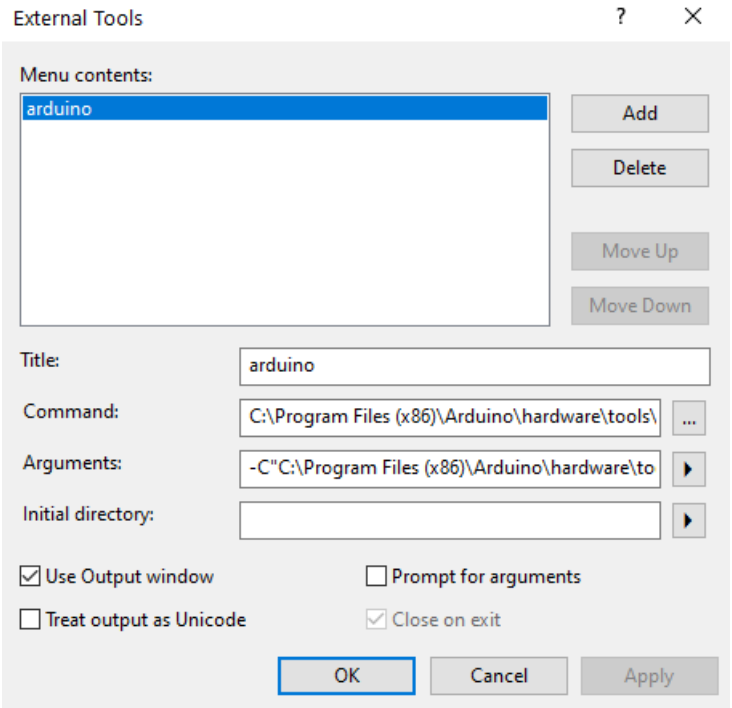


Figura 30. Configuración de AVRDUDE.

Tabla 13. Configuración de AVRDUDE para AVR Studio.

Apartado	Instrucción
Comando	C:\Program Files (x86)\Arduino\hardware\tools\avr\bin\avrdude.exe
Argumento	-C"C:\Program Files (x86)\Arduino\hardware\tools\avr/etc/avrdude.conf" -v -patmega2560 -cwiring -PCOM12 -b115200 -D -Uflash:w:"$(ProjectDir)\debug\$(TargetName).hex":i

Para el diseño hardware se procede a la distribución de los componentes, primero se identifica que pines se van a ocupar, los cuales se mencionan en la tabla 14.

Tabla 14. Conexión del microcontrolador y placa.

Características	Conexión
Rx0 y Tx0	Comunicación hacia el termógrafo a 115200 baudios
Rx1 y Tx1	Comunicación con GPS a 115200 baudios
Rx2 y Tx2	Puerto que se utilizaría como puente para comunicación con el termógrafo, o para maquinas con estándar Carrier o termo King
Pin A0	Sensor de voltaje
Pin D2	Control de encendido por bajo del GPS
Miso, Mosi, SCK, SS	Conexión necesaria para la comunicación CAN-BUS
Voltaje	Suministro del regulador de tensión
GND	Tierra común para toda la circuitería.

En la Figura 31 se muestra la estructura hardware lista para la inserción de la tarjeta controladora

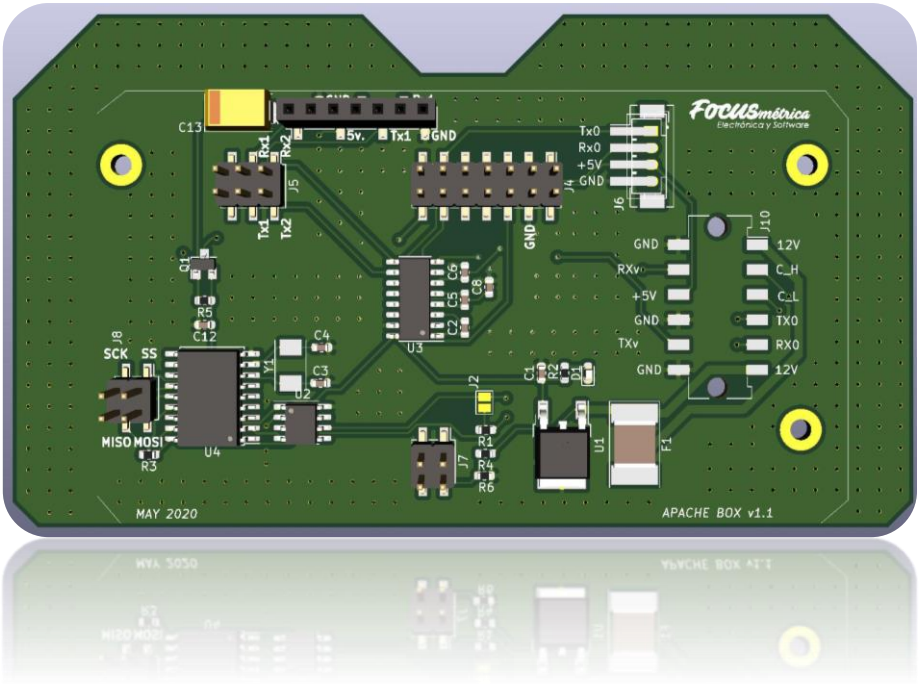


Figura 31. PCB para la comunicación con el Termógrafo.

4.6 Código para Configuración Automática (Termógrafo Apache)

4.6.1 Lógica de programación

Para el desarrollo de la siguiente etapa se debe considerar el funcionamiento y la estructura de programación del dispositivo que se tiene como punto de partida, de esta forma se implementa varias subetapas, las cuales se detallan en los siguientes apartados.

4.6.1.1 Máquina de estados de Comunicación.

La declaración se realiza en el archivo can_module.h, de esta forma se puede acceder de una forma más ordenada, la idea es que cada estado se realice de forma correcta, una buena práctica es el test en cada fase y mostrarlo por pantalla. En la figura 32 se muestra el proceso a realiza en el programa.

```
typedef enum {
    SM_SEARCH_DISCONNETED = 0x01,
    SM_SEARCH_CONNETED = 0x02,
    SM_SEARCH_CONFIGURED = 0x03,
    SM_SEARCH_DATA0 = 0x04,
    SM_SEARCH_DATA1 = 0x05,
    SM_SEARCH_ERROR = 0x00
} SM_SEARCH;
```

Figura 32. Estructura de estados.

4.6.1.2 Lógica para conexión y desconexión

En el siguiente apartado se implementa un mecanismo denominado flag de conexión para realizar el control, primero se escribe por el puerto serial el comando “AT+CK” cada 3 segundo, guardamos el contenido de la respuesta en un buffer, si responde con la palabra “CONECTION” el flag cambia y pasa al siguiente estado de configuración. Como se observa en la figura 33.

```
// If disconnected, check if it is connected again every CAN_TM_CHECK ms
if (g_flagCANModule==0){
    onDisconnect_CANmodule();
    if (halSysTick_GetMilliDiff(Canbus.dwStatusMilliRef) > CAN_TM_CHECK){
        Can_TxWrite(TOCHECK, strlen(TOCHECK));
        CANISO.sm_search=SM_SEARCH_DISCONNETED;
        Canbus.dwStatusMilliRef= halSysTick_GetMilliCount();
    }
}
```

Figura 33. Detección del dispositivo.

4.6.1.3 Procesos de configuración

A continuación, se realiza una segunda máquina de estados para el testeo de la configuración de Lector CAN-BUS, para lo cual se establece la estructura observable en la figura 34.

```
typedef enum {
    SM_STATUS_INITIAL = 0x01,
    SM_STATUS_START = 0x02,
    SM_STATUS_CONFIG_OP = 0x03,
    SM_STATUS_CONFIG_ID = 0x04,
    SM_STATUS_READ = 0x05,
    SM_STATUS_DATA = 0x06,
    SM_STATUS_ERROR = 0x00
} SM_STATUS;
```

Figura 34. Estructura de configuración.

Una vez que se detectó el estado de conexión es envía el comando “+++\n”, para seguir con el proceso debe responder con “Enter Setting Mode”, para en lo posterior asignar el estatus de inicio.

```
Canbus.dwAliveMilliRef = halSysTick_GetMilliCount();
if (strcmpi(szCmd, "Enter Setting Mode") == 0) {
    onConfig_CAN();
    CANISO.sm_status=SM_STATUS_START;
    char szBuffer[512];
    siprintf(szBuffer, "1--->%s<---\r\n", szCmd);
    Can_TxWrite(szBuffer, strlen(szBuffer));
    CANModule_CONFIG( );
```

Figura 35. Ingreso a modo de configuración.

Es necesario preparar una flag para mantener en el estado de configuración, se realiza una estructura switch case, primero se envía el comando AT+OP=0\n, esperamos que el mensaje de respuesta sea “MODE 0 :ISO 91122 configured”, de esta forma se confirma que se ha configurado el dispositivo para trasmitir la trama con el modelo estándar ISO y se actualiza la máquina de estados, en lo posterior se configura el en Identificador extendido por defecto, en consecuencia se envía el comando “AT+ID=0x0C0320C8\n”, se actualiza la máquina de estados solo cuando el mensaje de respuesta es “New ID Configurad”, para salir de modo de configuración es necesario enviar el comando “AT+Q\n” y esperar la respuesta “Exit Setting Mode”, la estructura de configuración se puede observar en la figura 36 y el diagrama de estados en la figura 37.

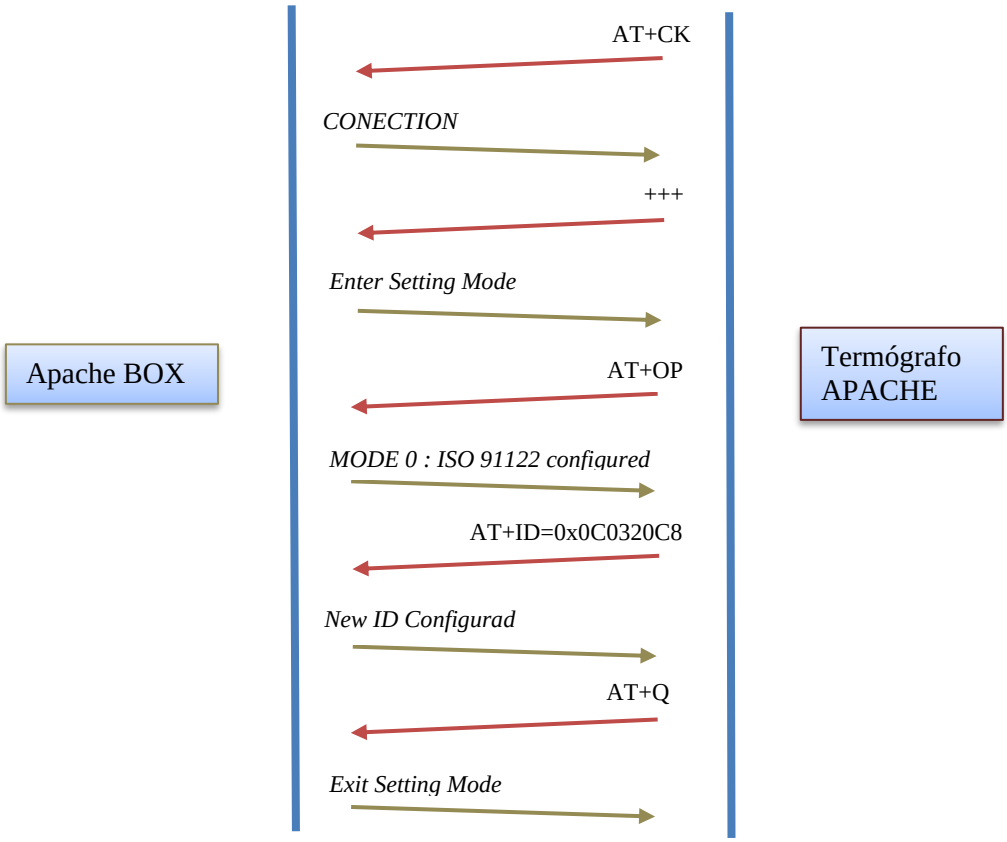


Figura 36. Estructura de Configuración.

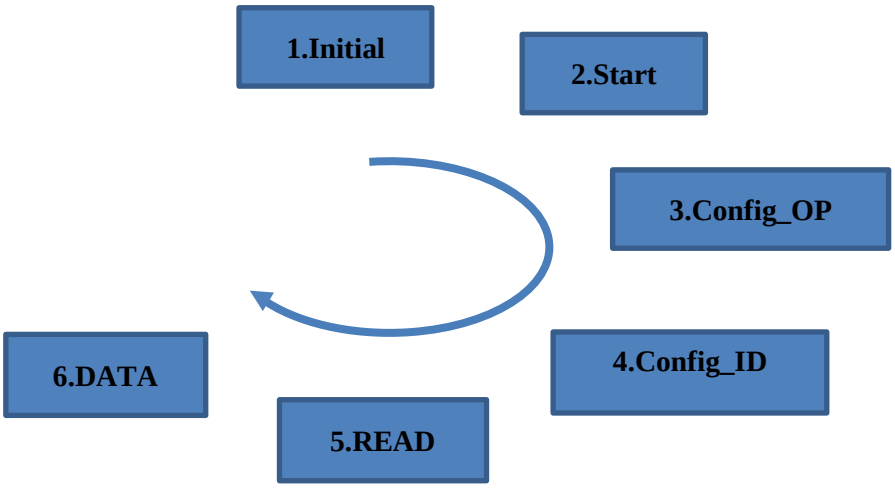


Figura 37. Máquina de estados de Configuración.

4.6.1.4 Procesado del contenido de trama.

En la etapa se implementa el algoritmo capaz de leer el buffer, separarlos en por medio de los delimitadores, el resultado es transformado a caracteres tipo int o unsigned long dependiendo el caso y se los asigna a la estructura que se va refrescando cada que pasa por el main. En el anexo 8.3 se muestra la estructura de la trama CAN-ISO 11992-2.

4.7 Comunicación con el servidor

4.7.1 Estructura de programación

En cuanto a lo relacionado al servidor es necesario una estructura para refrescar el contenido de la trama ISO11992-2 cada vez que se ejecute en el main, para lo cual se verifica que el flag de conexión se encuentre activo y que el dispositivo debe estar conectado a una red wifi, se puede visualizar en el anexo 8.4.

Dentro de la función, se define una estructura para mantener el orden e ir desarrollando por etapas el envío y la verificación de recibo por parte del servidor, los estados se muestran en el anexo 8.4.

Primero se configura el soket de conexión, que para el caso es 5, a continuación, se realiza un switch que se acciona en cada estado, el algoritmo de cada estado es el siguiente:

- SM_SERVER_IDLE

En el primer estado se realiza la verificación del nombre del servidor y el time out del envío, una vez que los dos son validados, se asigna al siguiente estado.

- SM_SERVER_CONNECT

En el segundo estado se pregunta por los flags del servidor, se abre el socket de conexión wifi, dependiendo que exista el enlace a una red para proceder al estado siguiente, es importante tener una variable que indique la conexión y desconexión de la validación del servidor.

- SM_SERVER_AWAIT_CONNECT

Para el tercer estado se verifica que la referencia temporal tenga coherencia, es decir que no se sobrepase el ACK del servidor y cambiar de estado de la conexión del servidor.

- SM_SERVER_SEND

Para el cuarto estado se realiza la construcción de la trama de envío, en este paso se añaden varios parámetros, primero el header que se compone de un número de trama, tipo de trama, tipo de dispositivo y longitud de encriptado, visto en el anexo 8.4.

Para construir la trama comienza incorporando a tiempo de la trama, la estructura de los datos y un CRC como validación de trama.
A continuación, se realiza la escritura de la trama al servidor y se comprueba que el ACK funcione, para proceder al siguiente estado.

- SM_SERVER_AWAIT_SEND_OK

En el quinto estado se realiza time out de espera, que sobrepase los 5 segundos y pase al siguiente estado.

- SM_SERVER_ACK

El sexto estado es el encargado de obtener la respuesta del servidor y comprobar que según el paquete se realice la verificación del tipo de trama, el ACK y el wifi activo. Por otra parte, también se implementa la verificación del tiempo de respuesta, se establece los criterios de espera y comprobación por medio de referencias temporales, de manera que se valide el envío y recepción de paquetes al servidor.

- SM_SERVER_DISCONNECT

Dentro del séptimo paso debemos tener una opción para la desconexión del Socket y también establecer una variable temporal para gestionar el control de la misma.

- SM_SERVER_DELAY

Se establece un tiempo de espera para volver al estado IDLE, que lo gestionamos dentro del estado y si lo cumple continuamos al inicio de la máquina de estados para la conexión al servidor.

4.7.2 Comunicación Trama CAN Termógrafo al servidor

4.7.2.1 Descripción General

La transmisión de los datos del termógrafo APACHE hacia el servidor se realiza sobre el protocolo UDP con reconocimiento ACK, por defecto envía el termógrafo el primer dato almacenado en memoria, y queda a la espera de la respuesta por parte del servidor. Si dentro de un determinado tiempo no hay respuesta, el envío esta invalidado y vuelve a enviar la misma trama.

Cuando dicha trama es recibida y reconocida por el servidor, automáticamente el equipo envía el siguiente dato. Todo se gestiona con un puntero que indica la dirección de memoria del último dato enviado y reconocido por el servidor, sin que se requiera un control adicional por parte del cliente.

4.7.2.2 Estructura de trama Temperatura.

La trama emitida por el Termógrafo consta de una cabecera propia de 9 bytes, que sigue la estructura de la tabla 15.

Tabla 15. Estructura encabezada de trama.

FRAME_NUMBER	FRAME_TYPE	DEVICEID	PAYLOAD_LENGTH
2 bytes	1 byte	4 bytes	2 bytes

Para la descripción de cada campo se visualiza en la tabla 16.

Tabla 16. Descripción de la estructura de cabecera.

Campo	Representación	Tamaño	Descripción
FRAME_NUMBER	UINT16 (LE)	2 bytes	Número de trama. Se corresponde con el identificador de la trama de datos que contiene el payload.
FRAME_TYPE	UINT8 (LE)	1 byte	**Tipo de trama** - 0: Trama de datos Temp - 1: Trama de ACK Temp
DEVICEID	UINT32 (LE)	4 bytes	S/N del termógrafo
PAYLOAD_LENGTH	UINT16 (LE)	2 bytes	Tamaño en bytes del payload

Para la trama de datos también se incluye un payload de 52 bytes que sigue la estructura de la tabla 17.

Tabla 17. Estructura payload de trama de datos.

TIME	T1	T2	STATUS	COMPANY_ VAT	VEHICLE _PLATE	COUNT	CRC
4 bytes	2	2	4 bytes	16 bytes	16 bytes	4 bytes	4 bytes

La descripción de cada campo se realiza en la tabla 18.

Tabla 18. Descripción de la estructura del payload.

Campo	Representación	Tamaño	Descripción
TIME	UINT32 (LE)	4 bytes	Timestamp del momento de medición de la trama (UTC) en formato exFAT.
T1	INT16 (LE)	2 bytes	Temperatura de la **sonda 1** expresada en coma fija (x10)
T2	INT16 (LE)	2 bytes	Temperatura de la **sonda 2** expresada en coma fija (x10)

STATUS	UINT32 (LE)	4 bytes	Estado de las entradas, salidas y avisos del equipo. - Bit0: Flag E1 – Contacto ON - Bit1: Flag E2 – Motor ON - Bit2: Flag E3 – Puerta abierta - Bit3: Flag E4 – Otra entrada - Bit4: Flag S1 – Aviso externo - Bit5: Flag Temp1 alta - Bit6: Flag Temp1 baja - Bit7: Flag Temp2 alta - Bit8: Flag Temp2 baja - Bit9: Flag aviso horas motor - Bit10: Flag aviso fecha revisión - Bits 11-31: En desuso
COMPANY_VAT	CHAR	16 bytes	String con el CIF de la empresa
VEHICLE_PLATE	CHAR	16 bytes	String con la matrícula/ID del vehículo
COUNT	UINT32 (LE)	4 bytes	Tiempo de funcionamiento del motor frigorífico, en minutos. Incrementos de 5 minutos.
CRC	UINT32 (LE)	4 bytes	CRC32 de todos los campos anteriores.

Para el ACK se envía el encabezado correspondiente y un payload de 2 bytes, su descripción se muestra en la tabla 19.

Tabla 19. Descripción de la estructura del ACK.

Campo	**Representación**	**Tamaño**	**Descripción**
FRAME_NUMBER	UINT16 (LE)	2 bytes	Número de la trama emitida por el termógrafo a la que se responde con el ACK.

4.7.2.3 Estructura de trama ISO 11992-2.

Para la trama CAN ISO 11992-2 se establece una estructura de comunicación similar a la de temperatura, se transmite de forma se asíncrona. Cada paquete de datos enviado al servidor contendrá una cadena de caracteres (o un String) de 72 bytes: 9 bytes para la cabecera y 63 bytes para el Payload. La frecuencia de transmisión de los datos es independiente de los de temperatura, la cabecera se muestra en la tabla 20 y la descripción del payload en la tabla 21.

Tabla 20. Descripción de la estructura de cabecera ISO 11992-2.

Campo	Representación	Tamaño	Descripción
FRAME_NUMBER	UINT16 (LE)	2 bytes	**Número de trama**. Se corresponde con el identificador de la trama de datos que contiene el payload.
FRAME_TYPE	UINT8 (LE)	1 byte	**Tipo de trama** - 7: Trama de datos CAN - 8: Trama de ACK CAN
DEVICEID	UINT32 (LE)	4 bytes	**S/N** del termógrafo
PAYLOAD_LENGTH	UINT16 (LE)	2 bytes	Tamaño en **bytes del payload**

Tabla 21. Descripción de la estructura de Payload ISO 11992-2.

N°	Campo	Representación	Tamaño	Descripción
1	ABS Status	UINT8 (LE)	1 byte	Activo o pasivo, Anti-lock Braking System
2	VDC Status	UINT8 (LE)	1 byte	Activo o pasivo, Vehicle Dynamic Control.
3	Speed vehicle	UINT64 (LE)	8 bytes	Velocidad real. Rango:0 km/h a 250 km / h.
4	Red Warning	UINT8 (LE)	1 byte	Activo o pasivo, señal de advertencia roja.
5	Amber Warning	UINT8 (LE)	1 byte	Activo o pasivo, señal de advertencia Ambar.
6	Axle load sum	UINT64 (LE)	8 bytes	Suma de las cargas verticales estáticas de los ejes del vehículo. Rango:0 kg a 128 510 kg
7	Tyre Pressure Status	UINT8 (LE)	1 byte	Suficiente o insuficiente la presión de los neumáticos.
8	Brake Lining	UINT8 (LE)	1 byte	Suficiente o insuficiente el recubrimiento de los frenos.
9	Tyre Identification Pressure	UINT32 (LE)	4 bytes	Identificador de la rueda para presión.
10	Tyre Pressure	UINT32 (LE)	4 bytes	Presión del neumático. Rango: 0 kPa a 2 500 kPa
11	Pneumatic Supply Pressure	UINT32 (LE)	4 bytes	Presión de suministro real del depósito del sistema de frenado. Rango: 0 kPa a 1 250 kPa
12	Tyre Identification Load	UINT32 (LE)	4 bytes	Identificador de la rueda de carga.

13	Axle Load	UINT32 (LE)	4 bytes	Carga por Eje. Rango: 0 kg a 128 510 kg
14	Tyre Identification Temperature	UINT32 (LE)	4 bytes	Identificador de la rueda de Temperatura.
15	Tyre Temperature	UINT32 (LE)	4 bytes	Temperatura por eje. Rango: - 273 ° C a 1 735 ° C
16	Air leakage detection	UINT32 (LE)	4 bytes	La pérdida de presión de un neumático. Rango:0 Pa /s a 6 425,5 Pa /s
17	Tyre Pressure threshold Detection	UINT8 (LE)	1 byte	Detección del umbral de presión. Rango: Entre 7 valores posibles.
18	CRC	UINT32 (LE)	4 bytes	**CRC32** de todos los campos anteriores.

4.8 Visualización en pantalla en el interfaz Apache

4.8.1 Estructura general

El sistema consta de tres servidores, los cuales se encargan de la gestión de los datos de las temperaturas y de la trama CAN, en la figura 38 se muestran los identificadores de los mismos.

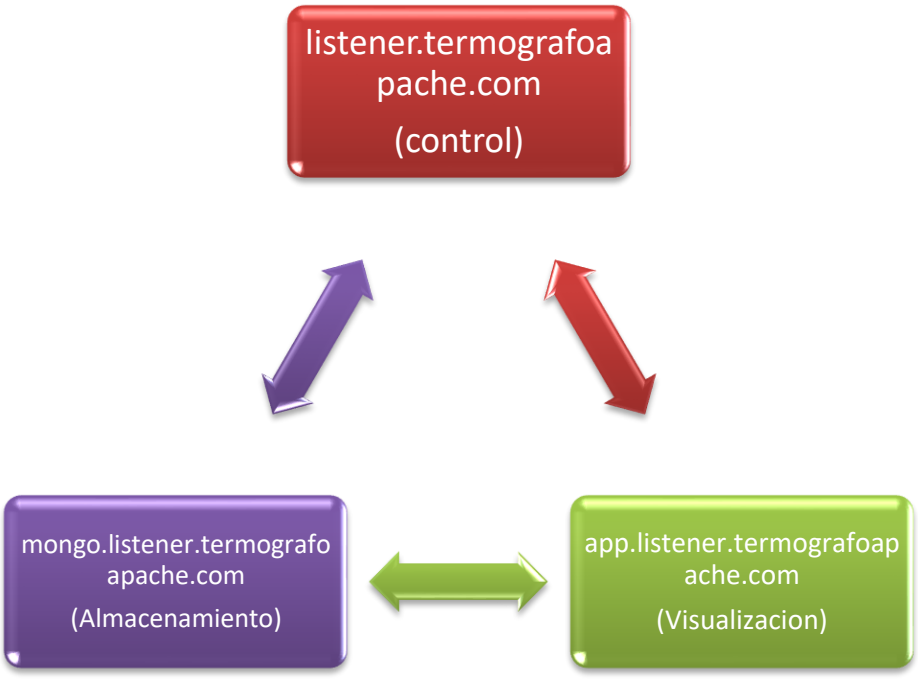


Figura 38. Estructura de servidores Apache.

4.8.2 Servidor de Control

Se utiliza el dominio listener.termografoapache.com, se establece una estructura capas de recibir en un buffer la trama CAN-ISO por el puerto 7020, descomponerla en los diferentes atributos, calcular el CRC y compararlo para la validación de datos, escrito en JavaScript. Se puede observar en la figura 39.

```
// CAN
data.payload = buffers.read(data.payloadDecrypted, {
    date: 'readUInt32LE',
    ABS_status: 'readUInt8',
    VDC_status: 'readUInt8',
    SpeedVehicle: ['slice', 8, ['$offset', '$offset+size']],
    Red_War:'readUInt8',
    Amber_War:'readUInt8',
    Axle_load_Sum: ['slice', 8, ['$offset', '$offset+size']],
    tyre_pressu_status: 'readUInt8',
    Brake_Lining: 'readUInt8',
    Tyre_Iden_Pressure: 'readUInt32LE',
    Tyre_Pressure:'readUInt32LE',
    Pneumatic_Pressure:'readUInt32LE',
    Tyre_Iden_Load:'readUInt32LE',
    Axle_Load:'readUInt32LE',
    Tyre_Iden_Temp:'readUInt32LE',
    Tyre_Temp:'readUInt32LE',
    Air_leakage_detec:'readUInt32LE',
    Tyre_threshold_Detec:'readUInt8',
    crcReceived: 'readUInt32LE'
});
```

Figura 39. Descomposición de la trama.

4.8.3 Servidor de Almacenamiento

El dominio es mongo.listener.termografoapache.com, una vez que se encuentra validada la trama, se almacena y se puede visualizar los atributos de la trama ISO como se muestra en la figura 40, la clasificación de los datos adquiridos ordenados por campo los podemos observar en anexo 8.5

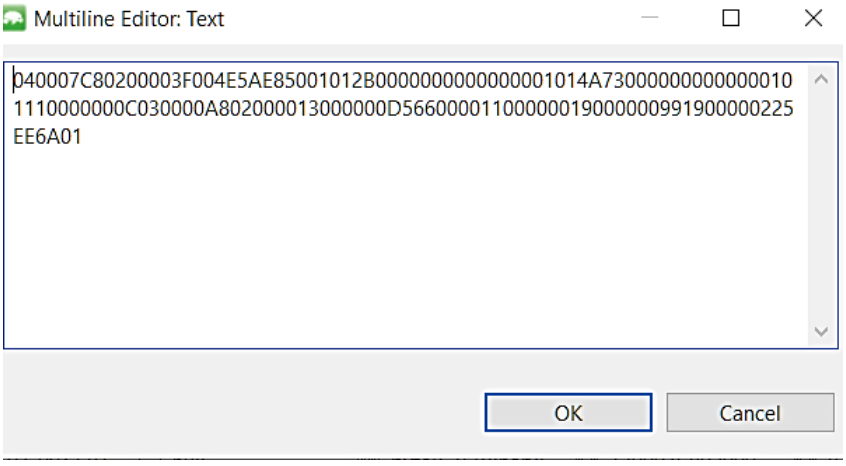


Figura 40. Trama válida Adquirida.

4.8.4 Servidor de Visualización

En la siguiente etapa se realiza la correlación de los datos recolectados y guardados en el servidor, para empezar, se establece el lenguaje de programación con el cual se va a trabajar, que para el caso es Pug, antes llamado Jade, es un motor de plantillas, con él se escribe código HTML con una sintaxis más sencilla, directa y clara. Para visualizar el código realizado por los desarrolladores del termógrafo se utiliza sublime Text, al cual tenemos que añadirle la sintaxis Pug e ir construyendo la página web de resultados.

Para mostrar los valores de la trama guardados en el servidor, es necesario direccionar donde se encuentran almacenados, así como crear o vincular los archivos de construcción de la cabecera, el cuerpo y los estilos. Una de las ventajas de Pug es que existe una amplia gama de platillas libres, para el caso se utiliza una tabla de valores, con el motivo de que la programación sea acorde y escalable se utiliza un direccionamiento dinámico de nombres, ya que la pagina es traducida a tres idiomas. En la figura 41 se muestra la construcción de la tabla de nombres y en la figura 42 se muestra la construcción de la tabla de datos.

```
30          if frames
31              table.table
32                  thead: tr
33                      th= __('-common.frameNum')
34                      th= __('-common.rxDate')
35                      th= __('-common.dateUTC')
36
37                      th ABS_status
38                      th VDC_status
39                      th SpeedVehicle
40                      th Red_War
41                      th Amber_War
42                      th Axle_load_Sum
43                      th tyre_pressu_status
44                      th Brake_Lining
45                      th Tyre_Iden_Pressure
46                      th Tyre_Pressure
47                      th Pneumatic_Pressure
48                      th Tyre_Iden_Load
49                      th Axle_Load
50                      th Tyre_Iden_Temp
51                      th Tyre_Temp
52                      th Air_leakage_detec
53                      th Tyre_threshold_Detec
54
55                      th Raw Frame
```

Figura 41.Construccion de la tabla de nombres.

```
57              tbody
58                  each frame in frames
59                      tr(id=frame._id)
60                          td= frame.header.frameNum.toString(16)
61                          td.nowrap!= printDate(frame.createdAt)
62                          td.nowrap!= printDate(frame.payload.date)
63
64                          td= frame.payload && frame.payload.ABS_status
65                          td= frame.payload && frame.payload.VDC_status
66                          td= frame.payload && formatBytes(frame.payload.SpeedVehicle)
67                          td= frame.payload && frame.payload.Red_War
68                          td= frame.payload && frame.payload.Amber_War
69                          td= frame.payload && frame.payload.Axle_load_Sum
70                          td= frame.payload && frame.payload.tyre_pressu_status
71                          td= frame.payload && frame.payload.Brake_Lining
72                          td= frame.payload && frame.payload.Tyre_Iden_Pressure
73                          td= frame.payload && frame.payload.Tyre_Pressure
74                          td= frame.payload && frame.payload.Pneumatic_Pressure
75                          td= frame.payload && frame.payload.Tyre_Iden_Load
76                          td= frame.payload && frame.payload.Axle_Load
77                          td= frame.payload && frame.payload.Tyre_Iden_Temp
78                          td= frame.payload && frame.payload.Tyre_Temp
79                          td= frame.payload && frame.payload.Air_leakage_detec
80                          td= frame.payload && frame.payload.Tyre_threshold_Detec
81                          //td= frame.input
82                          td: button(onclick="showExtraInfo('" + JSON.stringify(frame) + "')") Ver Raw
```

Figura 42. Construcción de la tabla de datos.

De esta forma se guardan los archivos programados, se compilan, se reinicia el servidor para que nuevamente se reestablezca y se encuentre pendiente para mostrar los datos.

5. Resultados

5.1 Test 1 module Serial CAN-Bus

5.1.1 Trama de salida CAN-Serial

Los resultados que se muestran en este capítulo, hacen referencia a la recolección, filtrado y tabulación de datos de la Red CAN con el estándar ISO11992-2, para comenzar en la figura 43 se muestra la trama obtenida de la red con datos tabulados, acorde a los requerimientos y necesidades del cliente. Una de las ventajas de producto es su enfoque, ya que solo se conecta a la red CAN y muestra los datos en del estándar en un idioma fácil de interpretar, de esta forma la versatilidad para acoplarse a otros sistemas se convierte en un punto fuerte.

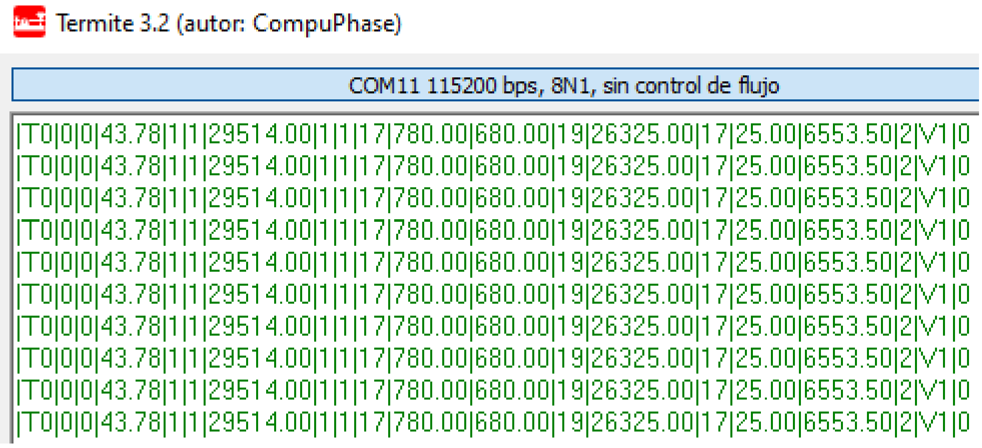


Figura 43. Trama construida APACHE-CAN.

Por defecto se encuentra configurada la trama APACHE-CAN pero para el test de funcionamiento se implementan algunos textos para depurar el código, que inicia con el envío del comando que testea si hay o no un dispositivo conectado y procede a la configuración, el termógrafo envía el comando y siempre espera confirmación para pasar al siguiente, con "+++" espera "Enter Setting Mode", con AT+OP=0, espera a "MODE 0 : ISO 91122 configured", envía "AT+ID=0x0C03020C8", espera a "New id Configured ", envía a "AT+Q", y espera a "Exit Setting Mode", de forma que siempre se autoconfigure a lo deseado, el proceso de configuración y validación se visualiza en las figura 44 y 45.

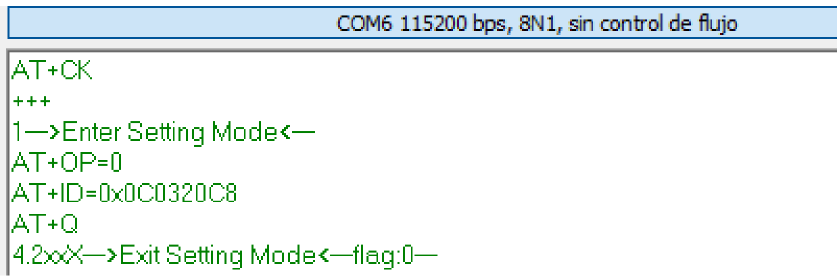


Figura 44. Envió de configuración de Apache Box por Termógrafo.

```
Entering Configuration Mode Successful!
Setting Baudrate Successful!
CONECTION
|T0|0|0|0.00|0|0|0.00|0|0|0|0|0.00|0.00|0|0.00|0|0.00|0.00|0|V1|0
Enter Setting Mode
CMD ERROR
MODE 0 :ISO 91122 configured
New ID Configured
Exit Setting Mode
|T0|0|0|0.00|0|0|0.00|0|0|0|0|0.00|0.00|0|0.00|17|25.00|6553.50|2|V1|0
|T0|0|0|0.00|0|0|0.00|1|1|17|780.00|680.00|0|0.00|17|25.00|6553.50|2|V1|0
|T0|0|0|0.00|0|0|0.00|1|1|17|780.00|680.00|0|0.00|17|25.00|6553.50|2|V1|0
```

Figura 45. Envío de validación de configuración al Termógrafo.

En el diseño del dispositivo lector de la red, ha cumplido con todos los parámetros requeridos para la su correcta implementación y escalabilidad, debido a ello se elige un procesador mayor al de inicio, que tenga mayor número de puertos de comunicación, pines de entrada y salida, velocidad de procesamiento, mayor tamaño de EEPROM, lo que conlleva el aumento de la complejidad en diseño hardware a la vez que aumenta la rentabilidad a futuro. También se ha conseguido implementar una forma para que solo con el archivo con extensión HEX se pueda quemar la memoria flash, lo cual tiene mucha relevancia en el ambiente de producción. En la figura 46 y 47 se muestra el dispositivo final.

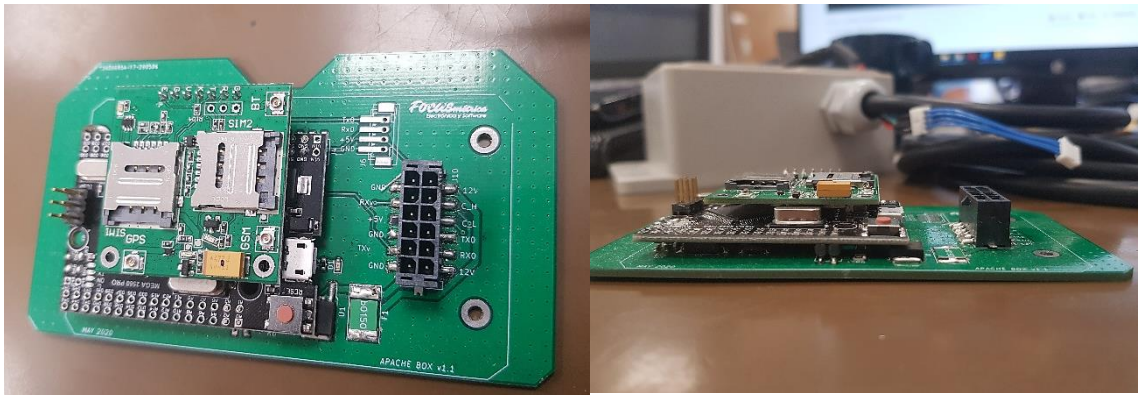


Figura 46. Dispositivo Can Module.

Figura 47. Can Module con la envolvente.

5.2 Test 2 Construcción de la trama al servidor

5.2.1 Verificación de funcionamiento

Para el test de funcionamiento se incorpora la función que muestra la trama completa que se envía al servidor, a la cual se le somete a un estímulo de encendido y apagado a las dos primeros componentes de trama, de esta forma podemos observar el correcto cambio del mismo, en el Anexo 8.6 se encuentran algunas tramas de ejemplo. A continuación, en la tabla 22 se realiza la decodificación de las tramas enviadas al servidor.

DEBUG_FRAME_CAN: 2020-06-22 / 6:40:15 frame=11

0B0007C80200003F000F35D65001012B0000000000000001014A730000000000001011100
00000C030000A802000013000000D566000011000000190000009919000002EDC73779

Cabecera→ **0B00 07 C8020000 3F00**

- FRAME_NUMBER (2 bytes): 0x0B00 → 0x000B → Trama núm. 11
- FRAME_TYPE (1 byte): 0x07 → 0x07 → Trama de datos CAN
- DEVICEID (4 bytes): 0xC8020000 → 0x000002C8 → S/N 000712
- PAYLOAD_LENGTH (2 bytes): 0x3F00 → 0x003F → Payload de 63 bytes

Tabla 22. Análisis de la trama de datos enviada al servidor.

N°	Campo	Tamaño	Descripción (LE)	Decodificar	Valor
0	Fecha	4 bytes	0x0F35D650	0x50D6350F	2020-06-22 / 6:40:15
1	ABS Status	1 byte	0x01	0x01	Activado
2	VDC Status	1 byte	0x01	0x01	Activado
3	Speed vehicle	8 bytes	0x2B000000 00000000	0x00000000 0000002B	43 Km/h
4	Red Warning	1 byte	0x01	0x01	Activado
5	Amber Warning	1 byte	0x01	0x01	Activado
6	Axle load sum	8 bytes	0x4A730000 00000000	0x00000000 0000734A	29514 Kg
7	Tyre Pressure	1 byte	0x01	0x01	Suficiente
8	Brake Lining	1 byte	0x01	0x01	Suficiente
9	Tyre Identification Pressure	4 bytes	0x11000000	0x00000011	1,7
10	Tyre Pressure	4 bytes	0x0C030000	0x030C	780 kPa

11	Pneumatic Supply Pressure	4 bytes	0xA8020000	0x02A8	680 kPa
12	Tyre Identification Load	4 bytes	0x13000000	0x00000013	1,9
13	Axle Load	4 bytes	0xD5660000	0x000066D5	26325 kg
14	Tyre Identification Temperature	4 bytes	0x11000000	0x00000011	1,7
15	Tyre Temperature	4 bytes	0x19000000	0x00000019	25 ° C
16	Air leakage detection	4 bytes	0x99190000	0x00001999	6553 Pa
17	Tyre Pressure threshold Detection	1 byte	0x02	0x02	0x02→ Sin advertencia de presión
18	CRC	4 bytes	0xEDC73779	0x7937C7ED	CRC32 de todos los campos anteriores

5.3 Pruebas de Campo

Las pruebas del dispositivo se realizan en las instalaciones de KNOR-BRENSE en Madrid, ya que cuentan con un simulador el cual contempla todos los imprevistos o estímulos para poner a prueba el sistema. Se conecta el lector el CAN-BOX y el termógrafo APACHE por el puerto serie como se observa en la figura 48, en lo posterior se ejecuta los estímulos y se verifican uno a uno los parámetros de la trama, primero en el CAN-BOX, luego en la memoria del Termógrafo y para finalizar los datos en el servidor mostrados en la página web de APACHE.

Figura 48. Conexión CAN BOX al Termógrafo Apache.

En la figura 49 se muestra el simulador de todos los parámetros de lectura del CAN en la empresa KNOR-BREMSE.

Figura 49. Pruebas de CAN BOX.

5.4 Test 3 (datos en la web)

En concreto, el sistema para visualizar las características del vehículo se muestra en la página web de Apache, en la figura 50 se muestra las características de la trama en la página del termógrafo Apache.

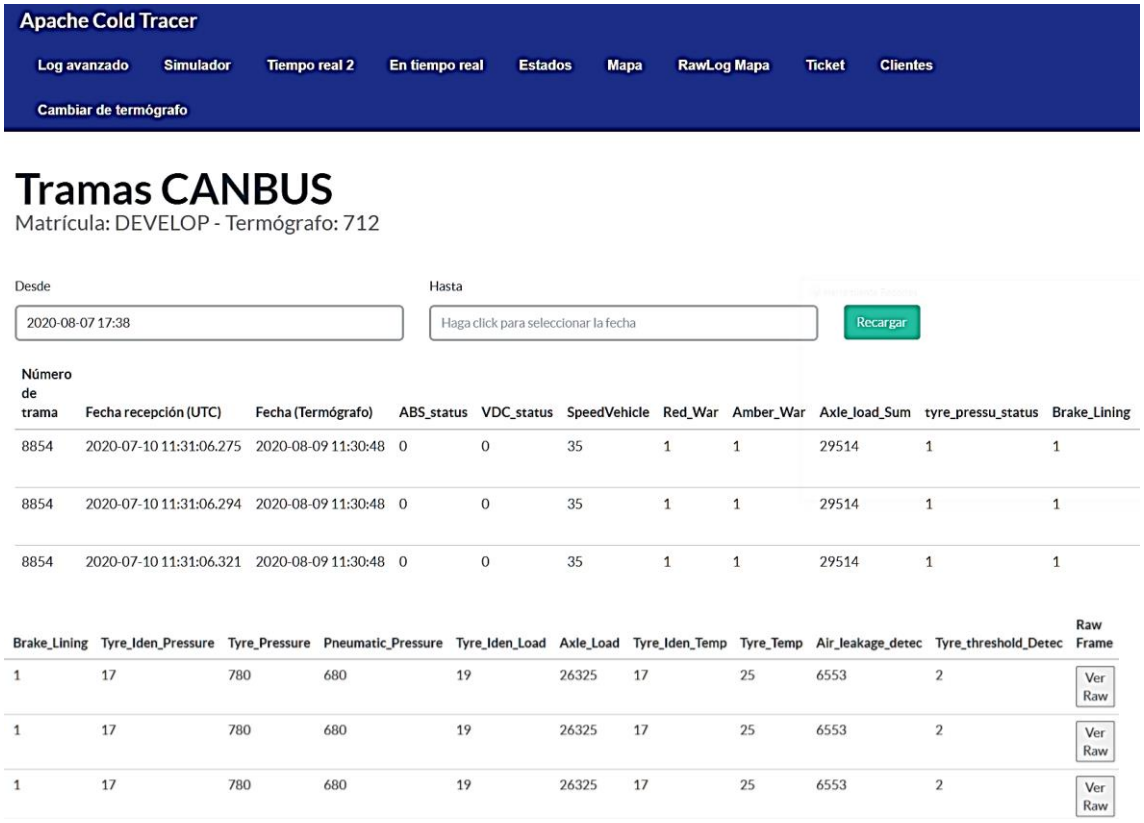


Figura 50. Visualización de Datos en página web del Termógrafo Apache

6. Conclusiones

Una vez finalizado por completo las etapas del proyecto, se ha conseguido todos los objetivos planteados, además resulta interesante realizar una breve exposición de conclusiones conseguidas:

- El objetivo principal se ha cumplido, debido a que se implementa el sistema diseñado, además, la adquisición y monitoreos de la Red CAN-Bus con protocolo ISO11992-2 en remolques con termógrafos APACHE es correcta.
- Se ha realizado una investigación del estado del arte de la temática, así se puede evidenciar que el desarrollo del producto se encuentra bien encaminado y se puede insertar en el mercado competitivo.
- Después de realizar las evaluaciones correspondientes, se ha cumplido con el desarrollo de un dispositivo configurado por comandos AT, que entrega una trama con estructura ISO 11992-2 o los datos de un identificador en específico.
- El hardware de entre las diferentes líneas de comunicación se ha realizado con éxito, de forma que se encuentren acorde con las buenas prácticas de diseño y funcionalidad, es importante mencionar que el hardware se encuentra preparado para su próxima etapa de desarrollo, contando con características pensadas en su futura implementación.
- Debido a que la mayor compatibilidad, debe establecerse con APACHE, se ha implementado con éxito la programación necesaria en el termógrafo para la configuración y comprobación automática que permite adquirir datos del dispositivo desarrollado con el estándar ISO11992-2.
- Para el sistema de visualización se incorpora una página dedicada a los parámetros del estándar, las tramas almacenadas en el servidor se extraen y se muestran en la página de Apache, acoplado a la interface gráfico con el que cuenta la empresa.

6.1 Escalabilidad y Proyección

Habiendo aprendido sobre las características y usos de la placa desarrollada, se puede trabajar sobre la elaboración proyectos futuros, entre los cuales se tiene:

- Discretización de tramas. Según el cliente se podría elegir que parámetros se desea mostrar, todo esto se configuraría de forma remota, sin borrar datos, es decir, que los servidores se encargarían de adaptar esta información para cada cliente.
- Conmutador de Protocolos. Si bien es cierto que el sistema se encuentra preparado para adquirir datos de la trama ISO, pero también se puede diseñar o construir protocolos de máquinas de frio, que ayudarían a rentabilizar más el producto.
- Conexión a la red CAN sin cortes. En un futuro se piensa implementar un circuito lector CAN-BUS, de forma que no se corten cables, ya sea por inducción o comparadores de voltaje para CAN LOW y CAN HIGH.
- Generación de informes en PDF. La información cada vez se hace más relevante en el momento de tomar decisiones, por lo cual, a una empresa con los datos exactos de sus camiones, le da una ventaja estratégica frente a cualquier eventualidad.
- La ampliación del proyecto con GPS y GSM. Es uno de los puntos trascendentales para los planes a futuro, ya que la temática es una rama donde se puede aplicar métodos como Machine learning y Big Data.

7. Referencias

[1] Introduction to the Controller Area Network (CAN). En: (05/2020). Available online: http://www.ti.com/lit/an/sloa101b/sloa101b.pdf

[2] ISO 11898-1:2015(en) Road vehicles Controller area network (CAN)Part 1: Data link layer and physical signalling En: (05/2020). Available online: https://www.iso.org/obp/ui/#iso:std:iso:11898:-1:ed-2:v1:en

[3] Road vehicles — Interchange of digital information on electrical connections between towing and towed vehicles. En: (05/2020). Available online: http://read.pudn.com/downloads662/sourcecode/others/2687337/ISO%2011992-2-2003.pdf

[4] Registrador de datos USB. En: (05/2020). Available online: https://europe.thermoking.com/es/solutions/connected-solutions/usb-datalogger/

[5] COLDTrans, the GPRS communications. En: (05/2020). Available online: https://www.carrier.com/truck-trailer/en/eu/products/eu-truck-trailer/telematics/coldtrans/

[6] FM-Pro4/Pro4 3G datasheet, manuals & peripherals. En: (05/2020). Available online: https://doc.ruptela.lt/pages/viewpage.action?pageId=884776

[7] ATMega328P, Microcontroller with 32K Bytes In-System. En: (05/2020). Available online: http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf

[8] ATMega2560, 8-bit Atmel Microcontroller with 16/32/64KB In-System Programmable Flash. En: (05/2020). Available online: https://ww1.microchip.com/downloads/en/devicedoc/atmel-2549-8-bit-avr-microcontroller-atmega640-1280-1281-2560-2561_datasheet.pdf

[9] Introducción a CAN bus: Descripción, ejemplos y aplicaciones de tiempo real. En: (05/2020). Available online: http://oa.upm.es/48061/

[10] Arduino CAN Bus Module 1st Network. En: (05/2020). Available online: https://www.pinterest.ca/pin/286471226284021229/

[11] Modulo Can Bus Serie introducción. En: (05/2020). Available online: https://docs.longan-labs.cc/1030001/

[12] Producto de simulación kanorr-bremse. En: (05/2020). Available online: https://www.knorr-bremsecvs.com/en/products_1/products_1.jsp

[13] Safe-Guarding CAN FD for applications in trucks. En: (04/2020). Available online: https://pdfs.semanticscholar.org/3849/321cb38f8472b7d6bbf361116c58b87ae541.pdf?_ga=2.181553507.57673679.1598396749-2029047490.1598396749

8. Anexos

8.1 Código de envió de tramas a la red.

```
#include <mcp_can.h>
#include <SPI.h>
MCP_CAN CAN0(10);     // Set CS to pin 10
void setup()
{
  Serial.begin(115200);
  // Initialize MCP2515 running at 16MHz with a baudrate of 500kb/s and the masks and filters disabled.
  if(CAN0.begin(MCP_ANY,CAN_500KBPS, MCP_16MHZ) == CAN_OK) //CAN_500KBPS  MCP_STDEXT
      Serial.println("MCP2515 Initialized Successfully!");
  else
      Serial.println("Error Initializing MCP2515...");
  CAN0.setMode(MCP_NORMAL);   // Change to normal mode to allow messages to be transmitted
}

byte data1[8] = {0x00, 0b00010100, 0x00, 0x00, 0xA5, 0x39, 0x00, 0x00};//EBS 22
byte data2[8] = {0b00000101, 0x17, 0x00, 0x00, 0x4E, 0x00, 0x00, 0x88};//EBS 23
byte data3[8] = {0x00, 0x00, 0x00, 0x00, 0x19, 0xAB, 0xCD, 0x00};//RGE22
byte data4[8] = {0x17, 0x40, 0x25, 0xFF, 0xFF, 0xEA, 0x00, 0x00};//RGE23

void loop()
{
  byte sndStat = CAN0.sendMsgBuf(0x18FEC4C8, 1, 8, data1);
  delay(100);   // send data per 100ms
   sndStat = CAN0.sendMsgBuf(0x18FEC6C8, 1, 8, data2);
  delay(100);
   sndStat = CAN0.sendMsgBuf(0x18FE5CC8, 1, 8, data3);
  delay(100);
   sndStat = CAN0.sendMsgBuf(0x18FE5EC8, 1, 8, data4);
  delay(100);
```

8.2 Código de recepción de tramas en la red

```
void loop()
{
    CAN0.readMsgBuf(&rxId, &len, rxBuf);              // Read data: len = data length, buf = data byte(s)
    if((rxId & 0x80000000) == 0x80000000)             // Determine if ID is standard (11 bits) or extended (29 bits)
      sprintf(msgString, "Extended ID: 0x%.8lX       DLC: %1d  Data:",(rxId & 0x1FFFFFFF), len);
    else
      sprintf(msgString, "Standard ID: 0x%.3lX       DLC: %1d  Data:",rxId , len);

    Serial.print(msgString);

    if((rxId & 0x40000000) == 0x40000000){    // Determine if message is a remote request frame.
      sprintf(msgString, " REMOTE REQUEST FRAME");
      Serial.print(msgString);
    } else {
      for(byte i = 0; i<len; i++){
        sprintf(msgString, " 0x%.2X", rxBuf[i]); //Determine if message is a remote request frame
        Serial.print(msgString);
      }
    }
    Serial.println();
  delay(100);
}
```

8.3 Procesado de Bits para construcción de Trama.

```
char ID1_EBS21()
{
  double Wheel_speed;
  uint16_t P_Entera;
  double fl_Wheel_speed;
  double fl_Wheel_speed1;
  rxBuf[0]=(rxBuf[0]&0b00000011);
  rxBuf[1]=(rxBuf[1]&0b00000011);
  if(rxBuf[0]==0b00000001)
    {TramaISO[0]= 0x01;}
    else
    {TramaISO[0]= 0x00;}
  if(rxBuf[1]==0b00000001)
    {TramaISO[1]= 0x01;}
    else
    {TramaISO[1]= 0x00;}

    P_Entera=rxBuf[3]<<8;
    Wheel_speed=((P_Entera)|(rxBuf[2]));
    fl_Wheel_speed=Wheel_speed/256;

    TramaISO_d[0]=fl_Wheel_speed;
    return TramaISO,TramaISO_d;

}
```

Procesado de los datos con identificador EBS21.

```
char ID2_EBS22()
{
  //rxBuf[1]=(rxBuf[1]&0b00111100);
  byte Red  =rxBuf[1]&0b00001100;
  byte Amber=rxBuf[1]&0b00110000;
  double AxleLoadSum;
  uint16_t Axle;

 if(Red==0b00000100)
    {TramaISO[3]= 0x01;}
    else
    {TramaISO[3]= 0x00;}
  if(Amber==0b00010000)
    {TramaISO[4]= 0x01;}
    else
    {TramaISO[4]= 0x00;}

    Axle=rxBuf[5]<<8;
    AxleLoadSum=(Axle)|(rxBuf[4]);
    AxleLoadSum=AxleLoadSum*2;
    TramaISO[5]=AxleLoadSum;
    TramaISO_d[1]=AxleLoadSum;

    return TramaISO,TramaISO_d;
}
```

Procesado de los datos con identificador EBS22.

```
char ID3_EBS23()
{
  //rxBuf[0]=(rxBuf[1]&0b00111100);
  byte Tyre_Pressure =rxBuf[0]&0b00000011;
  byte Brake_Lining  =rxBuf[0]&0b00001100;
  byte wheel_identifi,Tyre_pressure,Pneumatic_supply_pressure;
  if(Tyre_Pressure==0b00000001)
    {TramaISO[6]= 0x01;}
    else
    {TramaISO[6]= 0x00;}
  if(Brake_Lining==0b00000100)
    {TramaISO[7]= 0x01;}
    else
    {TramaISO[7]= 0x00;}

wheel_identifi=rxBuf[1];
Tyre_pressure=rxBuf[4];
Pneumatic_supply_pressure=rxBuf[7];

TramaISO[8]=wheel_identifi;
TramaISO_d[2]=Tyre_pressure*10;
TramaISO_d[3]=Pneumatic_supply_pressure*5;

  return TramaISO,TramaISO_d;
}
```

Procesado de los datos con identificador EBS23.

```
char ID4_RGE22()
{
  byte wheel_identifi;
  uint16_t AxleLoad;
  uint16_t Axle;
    wheel_identifi=rxBuf[4];
    Axle=rxBuf[6]<<8;
    AxleLoad=(Axle)|(rxBuf[5]);
    TramaISO_d[4]=AxleLoad/2;
    TramaISO[11]=wheel_identifi;

    return TramaISO,TramaISO_d;
}
```

Procesado de los datos con identificador RGE22.

```
char ID5_RGE23()
{
  byte wheel_identifi, Tyre_pressure_threshold;
  uint16_t Tyre_Temperature,Tyre_t;
  uint16_t Air_leakage_detec,Air;
  wheel_identifi=rxBuf[0];
  Tyre_t=rxBuf[2]<<8;
  Tyre_Temperature=(Tyre_t)|(rxBuf[1]);
  Air=rxBuf[4]<<8;
  Air_leakage_detec=((Air)|(rxBuf[3]));
  Tyre_pressure_threshold=rxBuf[5]&0b00000111;

  TramaISO[13]=wheel_identifi;
  TramaISO_d[5]=((Tyre_Temperature*0.03125)-273);
  TramaISO_d[6]=Air_leakage_detec*0.1;
  TramaISO[16]=Tyre_pressure_threshold;

  return TramaISO,TramaISO_d;
}
```

Procesado de los datos con identificador RGE23.

8.4 Construcción de trama hacia el servidor

```
#pragma pack(1)
typedef struct {
    uint8_t bABS_Status:1;//0:
    uint8_t bVDC_Status:1;//1:
    uint16_t dwSpeed:16; //2:0-250km
    uint8_t bRed_warning:1;//3:
    uint8_t bAmber_warning:1;//4:
    uint32_t dwLoad_Axle_sum:17;//5:0 kg to 128 510 kg(3 byte)
    uint8_t bTyre_Pressure:1;//6:
    uint8_t bBrake_Lining:1;//7:
    uint16_t dwIdentifi_Pressu_Tyre:15;//8: 1,7 - 5,A
    uint16_t dwPressure_Tyre:12;       //9: 0 to 2 500 kPa
    uint16_t dwPneumatic_supply:11;    //10:0 to 1 250 kPa
    uint16_t dwIdentifi_Load_Tyre:15;  //11:1,7 - 5,A
    uint32_t dwLoad_Tyre:17;           //12:0 to 128 510 kg
    uint16_t dwIdentifi_Tempera_Tyre:15;//13:1,7 - 5,A
    uint16_t dwTemperat_Tyre:15;        //14:-273 to 1 735 °C C
    uint16_t dwAir_leakage_detecti:15;  //15:0 to 6 425,5 Pa/s
    uint8_t dwThreshold_pressu_Tyre:3;  //16:0 to 7 options
} Mode0_ISO_def; /*  bytes () */
#pragma pack()
```

Estructura de trama ISO 11992.

```
/* Refresh Can Server. */
if (g_CurrentMode != g_LastMode){
    can_server_reset(CAN_SLOT);
} else {
    if (g_flagCANModule && (flagWifiUp || gprs_isup())) {
        can_server_refresh(CAN_SLOT);
    } else {
        can_server_reset(CAN_SLOT);
    }
}
```

Refresco del servidor.

```
typedef enum {
    SM_SERVER_IDLE,
    SM_SERVER_CONNECT,
    SM_SERVER_AWAIT_CONNECT,
    SM_SERVER_SEND,
    SM_SERVER_AWAIT_SEND_OK,
    SM_SERVER_ACK,
    SM_SERVER_COMMAND,
    SM_SERVER_DISCONNECT,
    SM_SERVER_DELAY
} SM_SERVER;
```

Estructura de control del servidor.

```
//Build  frame header
memset(&Packet, 0, sizeof(Packet));
Packet.Header.wFrameNum = Servers[bSocket].wFrameNum;
Packet.Header.bFrameType = FRAME_TYPE_CANDATA;
Packet.Header.dwDeviceId = g_Settings.Ident.dwDevice;
Packet.Header.wPayloadLen = wCryptLen;
```

Construcción de Header.

```
//Build date frame
 CanPayload.dwFatTime=get_fattime();
//Build data frame
 CanPayload.bABS=Mode0_struct.bABS_Status;
 CanPayload.bVDC=Mode0_struct.bVDC_Status;
 CanPayload.dwWSpeed=Mode0_struct.dwSpeed;
 CanPayload.bRED=Mode0_struct.bRed_warning;
 CanPayload.bAMBER=Mode0_struct.bAmber_warning;
 CanPayload.dwAxleLoadSum=Mode0_struct.dwLoad_Axle_sum;
 CanPayload.bTyrePressure=Mode0_struct.bTyre_Pressure;
 CanPayload.bBrankeLining=Mode0_struct.bBrake_Lining;
 CanPayload.dwTyreIdentiPressure=Mode0_struct.dwIdentifi_Pressu_Tyre;
 CanPayload.dwTyrePressure=Mode0_struct.dwPressure_Tyre;
 CanPayload.dwPneumaticSupplyPressure=Mode0_struct.dwPneumatic_supply;
 CanPayload.dwTyreIdentiLoad=Mode0_struct.dwIdentifi_Load_Tyre;
 CanPayload.dwAxleLoad=Mode0_struct.dwLoad_Tyre;
 CanPayload.dwTyreIdentiTemperature=Mode0_struct.dwIdentifi_Tempera_Tyre;
 CanPayload.dwTyreTemperatura=Mode0_struct.dwTemperat_Tyre;
 CanPayload.dwAirLeakage=Mode0_struct.dwAir_leakage_detecti;
 CanPayload.dwTyrePressureThreshold=Mode0_struct.dwThreshold_pressu_Tyre;
//Build CRC
 CanPayload.dwCRC=halCRC_Calculate32(&CanPayload, sizeof(PayloadCanFrameISO_def)
```

Construcción de la trama CAN-Bus de envió al servidor.

8.5 Tramas enviadas al Servidor

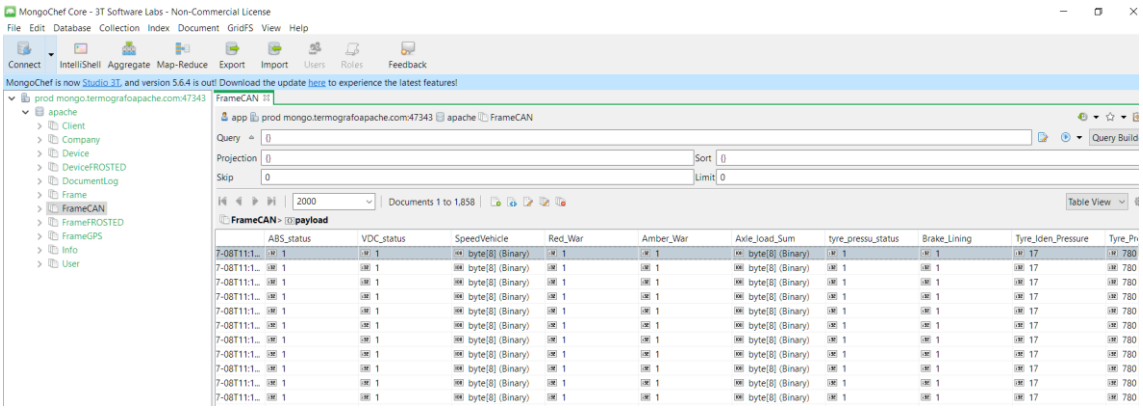
Figura para test de envío al servidor.

```
AT+CK
+++
1—>Enter Setting Mode<—
AT+OP=0
AT+ID=0x0C0320C8
AT+Q
4.2xxX—>Exit Setting Mode<—flag:0—
DEBUG_FRAME_CAN: 2020-06-22  6:31:22 frame=4    payload=040007000000003F00F633D650000000000000000000000001014A73000000000000001011100000000C030000A80200001300000D5660000110000001900000099190000002F97B8C95
DEBUG_FRAME_CAN: 2020-06-22  6:33:00 frame=5    payload=050007000000003F002034D650000000000000000000000001014A73000000000000001011100000000C030000A80200001300000D5660000110000001900000099190000002CE0C0A75
DEBUG_FRAME_CAN: 2020-06-22  6:34:07 frame=6    payload=060007000000003F004734D650000000000000000000000001014A73000000000000001011100000000C030000A80200001300000D5660000110000001900000099190000002A692A07D
DEBUG_FRAME_CAN: 2020-06-22  6:35:15 frame=7    payload=070007000000003F006F34D65000002B0000000000000001014A73000000000000001011100000000C030000A80200001300000D56600001100000019000000991900000002265369DD
DEBUG_FRAME_CAN: 2020-06-22  6:36:22 frame=8    payload=080007000000003F009634D65001012B0000000000000001014A73000000000000001011100000000C030000A80200001300000D566000011000000190000009919000000239AA904C
DEBUG_FRAME_CAN: 2020-06-22  6:38:00 frame=9    payload=090007000000003F00C034D65001012B0000000000000001014A73000000000000001011100000000C030000A80200001300000D566000011000000190000009919000000025FF1E94B
DEBUG_FRAME_CAN: 2020-06-22  6:39:07 frame=10   payload=0A0007000000003F00E734D65001012B0000000000000001014A73000000000000001011100000000C030000A80200001300000D56600001100000019000000991900000002636E96CA
DEBUG_FRAME_CAN: 2020-06-22  6:40:15 frame=11   payload=0B0007000000003F000F35D65001012B0000000000000001014A73000000000000001011100000000C030000A80200001300000D5660000110000001900000099190000002EDC73779
```

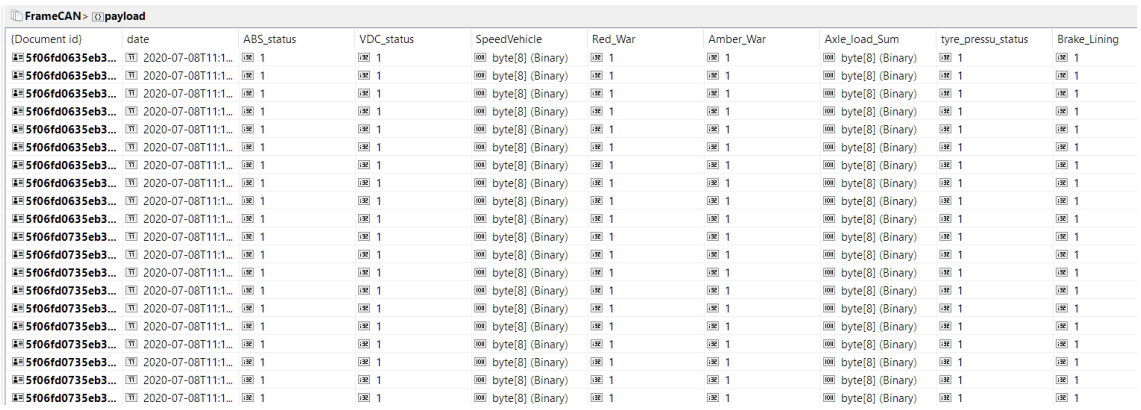
DEBUG_FRAME_CAN: 2020-06-22 6:31:22 frame=4
payload=040007000000003F00F633D650000000000000000000000001014A730000000000000101110000
000C030000A80200001300000D5660000110000001900000099190000002F97B8C95
DEBUG_FRAME_CAN: 2020-06-22 6:33:00 frame=5
payload=050007000000003F002034D650000000000000000000000001014A730000000000000101110000
000C030000A80200001300000D5660000110000001900000099190000002CE0C0A75
DEBUG_FRAME_CAN: 2020-06-22 6:34:07 frame=6
payload=060007000000003F004734D650000000000000000000000001014A730000000000000101110000
000C030000A80200001300000D5660000110000001900000099190000002A692A07D
DEBUG_FRAME_CAN: 2020-06-22 6:35:15 frame=7
payload=070007000000003F006F34D65000002B0000000000000001014A730000000000000101110000
000C030000A80200001300000D56600001100000019000000991900000002265369DD
DEBUG_FRAME_CAN: 2020-06-22 6:36:22 frame=8
payload=080007000000003F009634D65001012B0000000000000001014A730000000000000101110000
000C030000A80200001300000D566000011000000190000009919000000239AA904C
DEBUG_FRAME_CAN: 2020-06-22 6:38:00 frame=9
payload=090007000000003F00C034D65001012B0000000000000001014A730000000000000101110000
000C030000A80200001300000D566000011000000190000009919000000025FF1E94B
DEBUG_FRAME_CAN: 2020-06-22 6:39:07 frame=10
payload=0A0007000000003F00E734D65001012B0000000000000001014A730000000000000101110000
000C030000A80200001300000D5660000110000001900000099190000002

8.6 Visualización de la trama guardada en mongo

Descomposición global de la trama, en el servidor de APACHE.



Descomposición en parte 1:



Descomposición en parte 2: