Universidad Del Azuay

Facultad de Ciencia y Tecnología

Escuela de Ingeniería Electrónica

Diseño y Construcción de un equipo para diagnóstico de sensores, actuadores y ECU's automotrices para vehículos *KIA* y *HYUNDAI*

Trabajo de graduación previo a la obtención del título de
Ingeniero Electrónico

Autores

John Eduardo Llivicura Ávila
Edgar Geovanny Lupercio Jimbo

Director

Ing. Efrén Esteban Fernández Palomeque

Cuenca, Ecuador
2014

DEDICATORIA

Me permito dedicar el siguiente trabajo de graduación **a Dios**, **a mi esposa Melissa, a mis padres Enrique y Rosa, a mis hermanos Henry y Anthony** y a todas aquellas personas que con su apoyo y sus consejos hicieron posible la culminación de un gran paso más en mi vida.

John

A Dios Padre, que por su voluntad me ha permitido alcanzar una aspiración anhelada.

Doy gracias de manera especial a mis padres y hermanos, por que sembraron en mí, el deseo de superación, por brindarme su apoyo incondicional durante mis estudios y por estar a mi lado en los momentos más difíciles de mi vida.

Agradezco mi esposa e hijos, por apoyarme durante el trayecto de mi carrera, gracias por aceptarme, por comprenderme y por estar siempre a mi lado.

A mis maestros, por ser un pilar fundamental en mi formación académica, supieron transmitir sus conocimientos a lo largo de mi carrera.

Finalmente a compañeros y amigos que me brindaron su amistad, ya q en ellos encontré, el verdadero significado del trabajo en equipo.

Geovanny

AGRADECIMIENTOS

La elaboración de este proyecto para nuestra graduación es consecuencia del esfuerzo de muchos años, y no solo de quienes realizamos el proyecto, sino de todas aquellas personas que desinteresadamente nos supieron apoyar en este largo camino hacia la meta llegar a ser “Ingenieros”. Queremos agradecer a todas esas personas, pero en especial a nuestras familias que con mucha abnegación supieron regalarnos tiempo y sobre todo paciencia durante todo el largo período del pregrado y en el desarrollo del presente trabajo.

Además a todos nuestros profesores que con la mayor dedicación y profesionalismo supieron impartir sus conocimientos; no deseamos nombrar a ninguno de los mismos porque podríamos olvidar a alguno, siendo injusto, ya que todos y cada uno de ellos colocaron un granito de arena en este nuestro trabajo final.

John y Geovanny

RESUMEN

DISEÑO Y CONSTRUCCION DE UN EQUIPO PARA DIAGNOSTICO DE SENSORES, ACTUADORES Y ECU'S AUTOMOTRICES PARA VEHICULOS *KIA* Y *HYUNDAI*

En el presente trabajo de grado se plantea el diseño y construcción de un equipo capaz de diagnosticar y simular sensores automotrices de varios tipos, comprobando además el funcionamiento de algunos actuadores y de la unidad electrónica de control. En principio fueron considerados algunos aspectos fundamentales de funcionamiento de los sistemas: sus sensores, sus actuadores y de la misma ECU para ser simulados, captados y visualizados; para luego diseñar paralelamente el *software* y *hardware* que nos permitirá realizar el diagnóstico y comprobación de los mismos en automóviles. Como paso final se realizó la construcción del equipo, fusionando las tarjetas de adquisición de señales, procesamiento y potencia, obteniendo un equipo de diagnóstico de sensores, actuadores y ECU's automotrices para automóviles *KIA* y *HYUNDAI*.

PALABRAS CLAVES: Sensor, Actuador, ECU (Unidad Electrónica de Control), Arduino, *Labview*, Diagnóstico, Señal, Procesamiento.

DIRECTOR DE ESCUELA
INGENIERIA ELECTRONICA
Ing. Francisco Eugenio Vásquez Calero

DIRECTOR DEL TRABAJO DE GRADO
Ing. Efrén Esteban Fernández Palomeque

AUTOR
John Eduardo Llivicura Ávila

AUTOR
Edgar Geovanny Lupercio Jimbo

ABSTRACT

DESIGN AND CONSTRUCTION OF SENSORS, ACTUATORS AND AUTOMOTIVE ECU's DIAGNOSTIC EQUIPMENT FOR KIA AND HYUNDAI CARS

This graduation paper discusses the design and construction of equipment capable of diagnosing and simulating automotive sensors of various types, checking in addition, the operation of some actuators and of the electronic control unit. At the beginning some fundamental aspects of system performance were considered: its sensors, actuators as well as ECUs (engine control unit) to be simulated, captured and displayed; and then design simultaneously the software and hardware that will allow us to make the diagnosis and their testing in cars. As a final step the construction of the equipment was performed by merging signal acquisition cards, processing and power obtaining a diagnostic equipment of sensors, actuators and automotive ECU's for HYUNDAI and KIA cars.

KEYWORDS: Sensor, Actuator, ECU (Electronic Control Unit), Arduino, Labview, Diagnosis, Signal Processing.

Ing. Francisco Eugenio Vásquez Calero
ELECTRONICS ENGINEERING
SCHOOL DIRECTOR

Ing. Efrén Esteban Fernández Palomeque
THESIS DIRECTOR

John Eduardo Llivicura Ávila
AUTHOR

Edgar Geovanny Lupercio Jimbo
AUTHOR

Translated by,
Lic. Lourdes Crespo

INDICE DE CONTENIDOS

CAPÍTULO 2: DISEÑO Y PROGRAMACION DEL *FIRMWARE* DE CONEXIÓN CON LOS SENSORES, ACTUADORES Y UNIDAD ELECTRONICA DE CONTROL

CAPÍTULO 3: DISEÑO Y CONSTRUCCION DE LAS INTERFACES APLICADAS A LOS SENSORES, ACTUADORES Y UNIDAD ELECTRONICA DE CONTROL

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

Llivicura Ávila John Eduardo

Lupercio Jimbo Edgar Geovanny

Trabajo de graduación

Ing. Efrén Fernández Palomeque

Mayo 2014

DISEÑO Y CONSTRUCCIÓN DE UN EQUIPO PARA DIAGNÓSTICO DE SENSORES, ACTUADORES Y ECU'S AUTOMOTRICES PARA VEHÍCULOS *KIA* Y *HYUNDAI*

INTRODUCCION

En la actualidad el diagnóstico de los dispositivos y sistemas electrónicos en el automóvil es mucho más complejo, ya que la tecnología avanza a pasos agigantados día a día. Al ser dispositivos más complejos, las herramientas para diagnosticar son mucho más costosas, además que se debe poseer el conocimiento adecuado no solo del sistema sino de la herramienta para poder hacer el diagnóstico correcto.

En nuestro medio en muchos lugares donde se realiza trabajos de reparación de estos dispositivos y sistemas, no cuentan con herramientas adecuadas para hacer este diagnóstico, por lo cual recurren a herramientas muchas veces ortodoxas o al empirismo a cerca de un determinado fallo en un dispositivo o sistema, sin poder garantizar 100% la reparación o arreglo del dispositivo o sistema averiado.

Entonces lo que deseamos plantear en el desarrollo de este trabajo de graduación es elaborar una herramienta capaz de diagnosticar algunos tipos de sensores, actuadores

y la unidad electrónica de control del automóvil, en este caso de loa vehículos *KIA* y *HYUNDAI*. La herramienta a obtener será de fácil manipulación, amigable con el operario, lo más compacta posible, además de garantizar 100% el diagnóstico que está realizando, generando beneficios para el operario y el propietario de automóvil averiado.

Para el desarrollo de este trabajo de graduación se empleará el método experimental, pues se requiere de una serie de pruebas sobre un vehículo durante el proceso de construcción y acoplamiento para obtener un equipo eficaz y eficiente. Además se utilizará el tipo de investigación bibliográfica y documental, a fin de encontrar suficiente información que apoye nuestro trabajo para poder alcanzar nuestros objetivos.

CAPITULO 1

CARACTERISTICAS Y PRINCIPIO DE FUNCIONAMIENTO

SENSORES, ACTUADORES Y UNIDAD DE CONTROL ELECTRONICO

1.1. Generalidades

La electrónica aplicada al automóvil tiene su punto más relevante en la unidad de control electrónico (ECU), misma que recolecta la información obtenida de los diferentes sensores en el motor, para modificar el estado de los actuadores, modificando y corrigiendo el funcionamiento del mismo de acuerdo a sus necesidades.

Si bien es cierto, la unidad de control electrónico es la encargada de interpretar los valores enviados por los sensores, cada uno de estos están constituidos y construidos de acuerdo a la variable que vayan a medir. He aquí donde entra la electrónica en los sensores, ya que serán estos los dispositivos encargados de convertir la variable física en una señal electrónica.

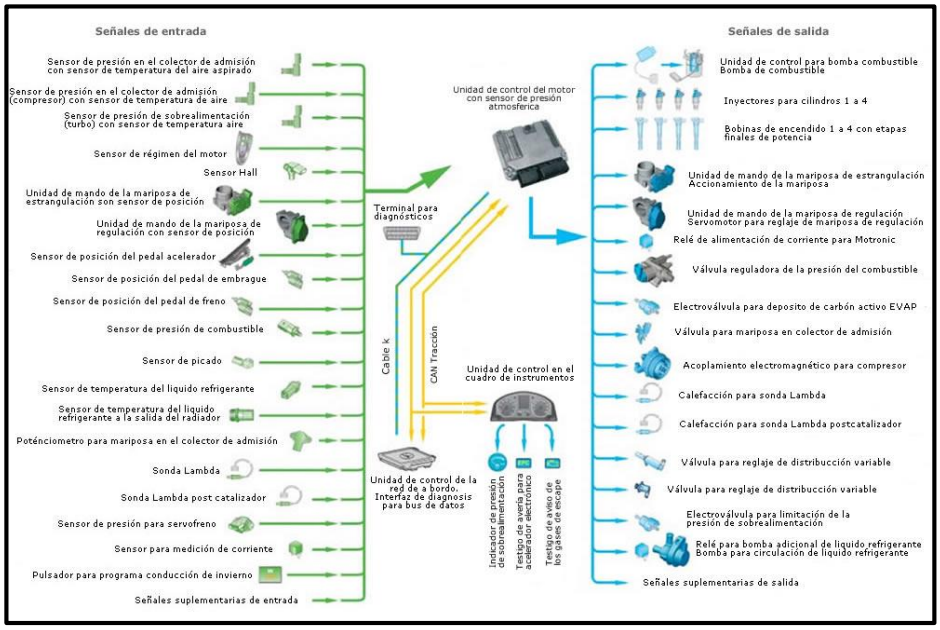
> Entonces: "los sensores convierte las condiciones de funcionamiento del motor (temperatura, presión absoluta del múltiple, movimientos mecánicos, etc.) en un voltaje eléctrico que es enviado a la computadora para ser procesado y comparado con datos de referencia grabados en sus memorias"[1]
>
> Y "los actuadores son dispositivos que realizan los cambios en la operación del vehículo, para adecuar su operación a diferentes condiciones específicas"[2]

Podemos definir la función de la unidad electrónica de control: "Es el dispositivo que recibe la información, misma que es comparada con datos grabados en sus memorias y, según su programa o *software* almacenado, ordena a los actuadores determinadas respuestas", es así como controla el funcionamiento del motor, a través de diferentes áreas de control como: entrega de combustible, velocidad en vacío, sincronización del avance de la chispa, dispositivos de emisiones, etc.

En la **figura 1.1.** se observa un esquema de funcionamiento en conjunto de sensores, actuadores y la unidad de control electrónico.

[1] BOLETIN TU TALLER MECANICO BOLETIN TP11-03 6
[2] BOLETIN TU TALLER MECANICO BOLETIN TP11-03 7

Figura 1.1. Esquema de funcionamiento general de los sensores (señales de entrada), actuadores (señales de salida) y la unidad de control electrónico,



Fuente: MOTOBITURBO "TSI". http://www.naikontuning.com/articulos/motor-bi-turbo-TSI/esquema-componentes-tsi.jpg
[Consulta: 04 de mayo de 2013].

1.2. Sensores

1.2.1. Sensor de masa de aire (MAF: Mass Air Flow Sensor)

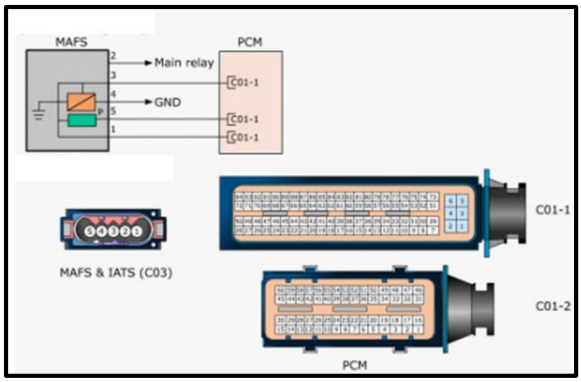
El sensor MAF (**figura 1.2.**) convierte la cantidad de aire aspirado por el motor en una señal de voltaje hacia la ECU. La ECU necesita saber la cantidad de aire ingresado para calcular la carga de éste. Esto es fundamental para calcular la cantidad de combustible a inyectar, cuando tiene que saltar la chispa en la bujía y cuando realizar los cambios en la A/T. Se utiliza un MAF tipo película caliente fabricado por *SIEMENS*. En ralentí el voltaje es aproximadamente 1V, el cual representa una cantidad de aire de 8,2 kg/h (kilogramos por hora). Recuerde que sólo MC 1.6L con CVVT usa MAF.

Figura 1.2. Sensor de Masa de Aire MAF

Fuente: Mercadolibre.com. http://img1.mlstatic.com/sensor-maf-para-accent-de-hyundai-o-flujo-de-aire_MLV-O-30987460_1836.jpg
[Consulta: 04 de mayo de 2013].

En la **Figura 1.3.** se observan los pines de conexión entre la ECU (PCM) y el sensor MAF y su configuración interna. Adicional el conexionado del sensor IAT hacia la ECU y sus respectivos pines.

Figura 1.3. Conexión y pines de los sensores MAF e IAT



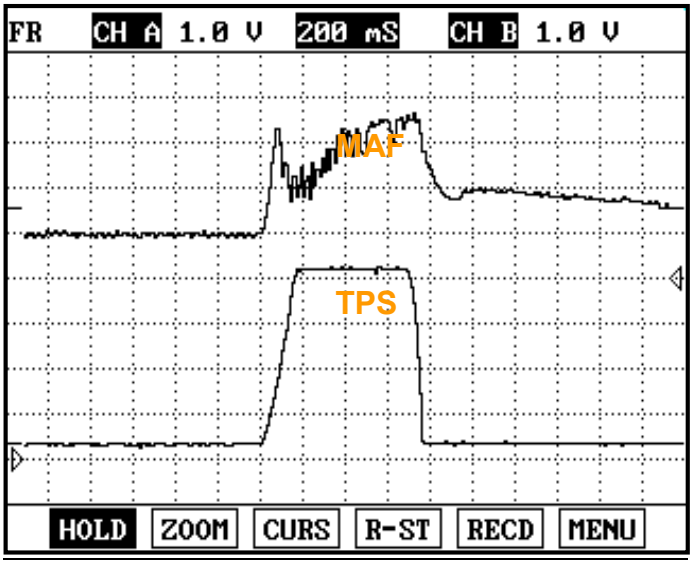
MAF	PCM Pin No.	Descripción
1	C01-1 8	masa
2	Relé principal	B+
3	C01-1 10	Señal MAF
4	GND	masa
5	C01-1 32	Señal IATS

Fuente: *MC Gasoline Engine Spanish Mau 1*
[Consulta: 04 de mayo de 2013].

Las variaciones en el sensor MAF, son generadas cuando existe un cambio en la posición de la mariposa de aceleración, es decir, cuando se da un cambio en la carga o estado de funcionamiento del motor. Al haber el cambio en la carga significa que se dio un cambio en la posición de la

mariposa de aceleración que está conecta al sensor TPS. En la **figura 1.4.** se representa el cambio que existe en el sensor MAF cuando se da un cambio en el TPS.

Figura 1.4. Forma de onda del sensor MAF y TPS



Item	Idle	WOT
MAF(V)	1	3.5
Aire (kg/h)	8.3	227.5
TPS(V)	0.3	3.9

Fuente: *MC Gasoline Engine Spanish Mau 1*
[Consulta: 04 de mayo de 2013].

1.2.2. Sensor de temperatura del refrigerante del motor (ETC: Engine Coolant Temperature Sensor. Figura 1.5.)

El ECT responde a los cambios en la temperatura del motor. Este sensor es crítico para muchas funciones de la ECU, como la inyección de combustible, punto de encendido, CVVT, patrón de cambio de la A/T, etc.

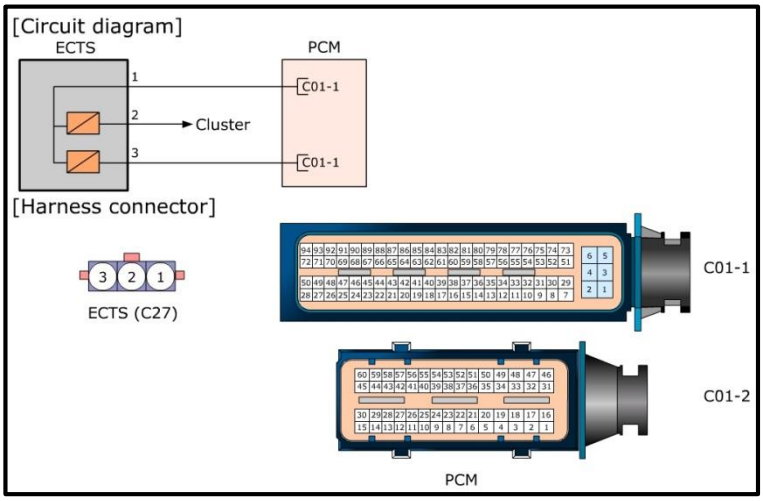
Figura 1.5. Sensor de temperatura del refrigerante del motor

Fuente: http://www.copartes.com/img/6500

[Consulta: 04 de mayo de 2013].

La **figura 1.6.** muestra el diagrama del circuito para la ETC, así como sus pines de conexión con la ECU.

Figura 1.6. Conexionado y pines para la ETC



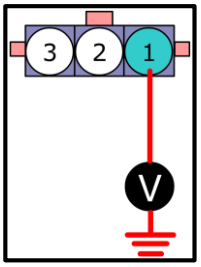
ECT	PCM Pin No.	Descripción
1	C01-1 77	Señal
2	Tablero	Medidor
3	C01-1 73	Masa

Fuente: *MC Gasoline Engine Spanish Mau 1*

[Consulta: 04 de mayo de 2013].

Con temperatura baja, el tiempo de abertura del inyector se incrementa para compensar el fenómeno de combustible adherido a las paredes del cilindro (*wall film*). Si este sensor no funciona, la ECU no corregirá la inyección por lo que el arranque será muy difícil. La variación del voltaje y la resistencia se presentan en la **figura 1.7.** según la temperatura de funcionamiento de la ECT.

Figura 1.7. Variación de voltaje y resistencia según temperatura de la ECT

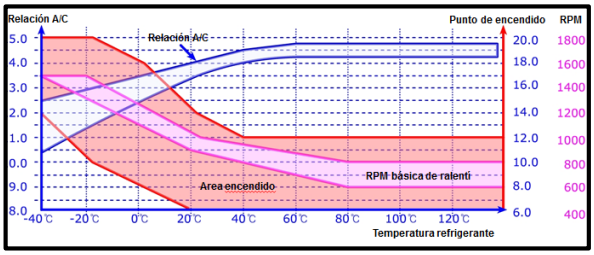


ECT	20°C	80°C
salida (V)	3.6	1.2
kΩ	2.4	0.32

Fuente: *MC Gasoline Engine Spanish Mau 1*
[Consulta: 04 de mayo de 2013].

En la **figura 1.8.** se muestra el mapa interno de la ECU que relaciona la temperatura del refrigerante del motor con la relación aire/combustible (A/C), el punto de encendido y las rpm en ralentí. Cuando la temperatura es baja, el punto de ignición es avanzado resultando en el incremento de las rpm. Si usted tiene 1°de ignición avanzado, significa un aumento de la velocidad del motor de aprox. 50 rpm. Para la relación A/C, cuando el motor está frío la mezcla se enriquece. Por ejemplo, para una temperatura del refrigerante de 0°C, tenemos una relación A/C de entre 12,5:1 a 13,5:1; un punto de encendido que oscila entre 8° y 18°; y un ralentí de entre 1300 y 1500 rpm

Figura 1.8. Relación entre la temperatura del refrigerante del motor y la relación estequiométrica

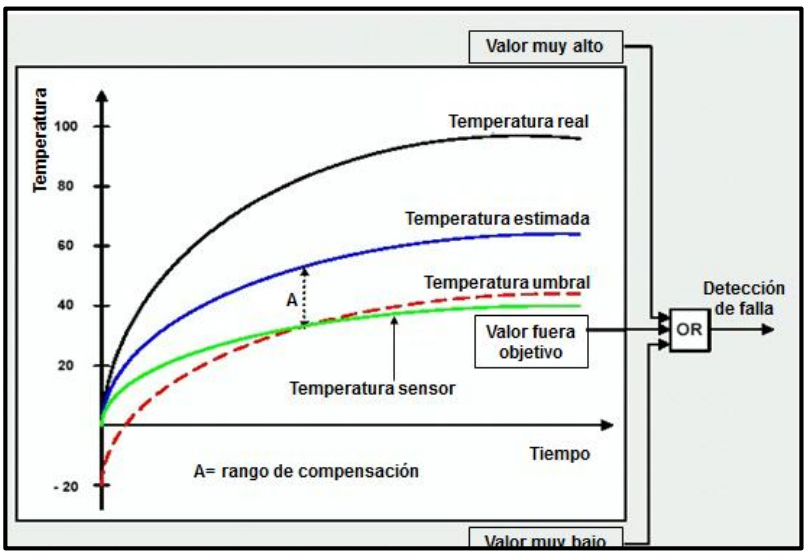


Fuente: *MC Gasoline Engine Spanish Mau 1*

[Consulta: 04 de mayo de 2013].

La **figura 1.9** muestra la verificación de racionalidad (para el sistema OBD II). Después de conducir cierta distancia, la temperatura del motor debería estar dentro de cierto rango, pero si no es así, la ECU lo reconocerá como una falla en el ECT (la temperatura medida está fuera de la temperatura objetivo (target)).

Figura 1.9. Verificación de racionalidad en la ECT



Fuente: *MC Gasoline Engine Spanish Mau 1*

[Consulta: 04 de mayo de 2013].

1.2.3. Sensor de posición de la mariposa (TPS: Throttle Position Sensor. Figura 1.10.)

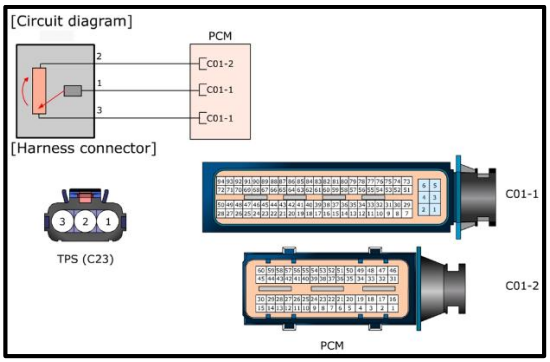
El TPS está montado en el cuerpo del obturador y convierte el ángulo de la válvula mariposa en una señal eléctrica usando una resistencia variable (potenciómetro). A medida que la mariposa se abre, la señal de voltaje aumenta. La ECM usa esta señal para saber la intención del conductor y con ello realizar las correcciones correspondientes principalmente de combustible y encendido. El rango de voltaje normal es de 0,2V a 4,8V.

Figura 1.10. Sensor de posición de la mariposa (TPS)

Fuente: EBAY PRODUCTS.
http://i.ebayimg.com/00/s/MzAwWDMwMA==/$(KGrHqF,!hMFCw59TVjGBQ9P9()pLw~~60_35.JPG
[Consulta: 04 de mayo de 2013].

En la **figura 1.11.** se muestra el diagrama de conexión del TPS con la ECU, así como su diagrama de circuito interno.

Figura 1.11. Conexionado y pines del TPS



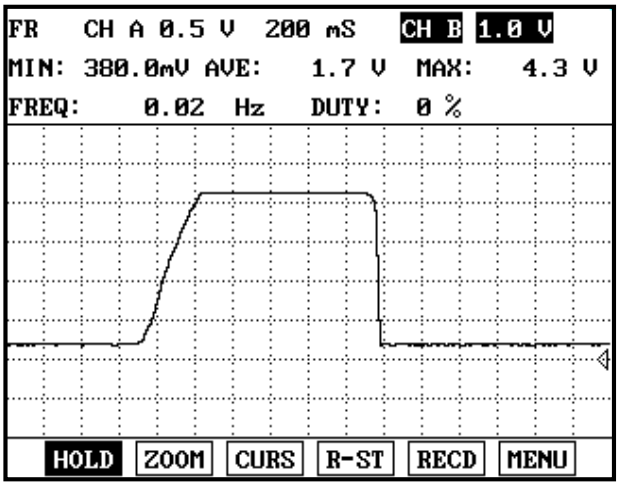
TPS	PCM Pin No.	Descripción
1	C01-1 75	Señal
2	C01-2 58	Referencia (5V)
3	C01-1 51	masa

Fuente: *MC Gasoline Engine Spanish Mau 1*

[Consulta: 04 de mayo de 2013].

La **figura 1.12.** indica la variación de voltaje en la señal del TPS cuando se encuentra en posición ralentí y luego con el acelerador a fondo (WOT).

Figura 1.12. Variación de voltaje de TPS



TPs	Ig. On	WOT
salida (V)	0.3	3.9

Fuente: *MC Gasoline Engine Spanish Mau 1*

[Consulta: 04 de mayo de 2013].

1.2.4. Sensor de posición del cigüeñal (CKP: Crankshaft Position Sensor. Figura 1.13)

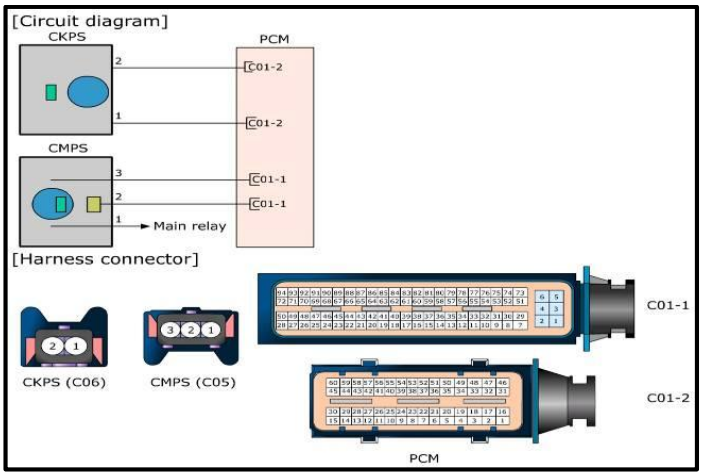
La ECU calcula el ángulo exacto del cigüeñal y las rpm del motor usando el CKP, el cual es del tipo inductivo. Con ello, se determina el tiempo de inyección, avances del encendido, detectar pérdidas de encendido y calcular el ángulo optimo del eje de levas de admisión en el CVVT.

Figura 1.13. Sensor de posición del cigüeñal (CKP)

Fuente: http://mlv-s1-p.mlstatic.com/sensor-de-posicion-ciguenal-ckp-hyundai-accent-3-pines-7017-MLV5153521985_102013-O.jpg
[Consulta: 04 de mayo de 2013].

En la **figura 1.14** se puede observar el conexionado del CKP y CMP (sensor de la posición del árbol de levas), así como sus diagramas internos.

Figura 1.14. Conexionado y pines del CKP y CMP



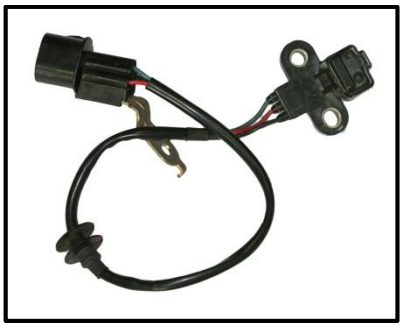
CKP	PCM Pin No.	Descripción
1	C01-2 55	señal
2	C01-2 40	señal

Fuente: *MC Gasoline Engine Spanish Mau 1*
[Consulta: 04 de mayo de 2013].

1.2.5. Sensor de posición del eje de levas (CMP: Cramshaft Position Sensor. Figura 1.15)

El CMP es de tipo efecto Hall y su señal sirve para determinar en que fase están los cilindros junto con el CKP, y en el caso determinar la posición del eje de levas para el control del CVVT.

Figura 1.15. Sensor de posición del eje de levas y pines de conexión



CMP	PCM Pin No.	Descripción
1	Relé principal	B+
2	C01-1 41	señal
3	C01-1 38	masa

Fuente: http://i0.wp.com/www.enauto.cl/wp-content/uploads/2012/06/sensor-de-posici%C3%B3n.jpg

MC Gasoline Engine Spanish Mau 1

[Consulta: 04 de mayo de 2013].

Syncro.State (CKP/CMP) es una forma de indicar que la correlación entre el cigüeñal y el eje de levas es el correcto. "**ON**" significa que no hay problema.

La posición del eje de levas es para determinar el manejo del CVVT. Inicialmente se muestra -8°, pero una vez arrancado variará.

1.2.6. Sensor de oxigeno calentado (H02S: Heated Oxygen Sensor. Figura 1.16)

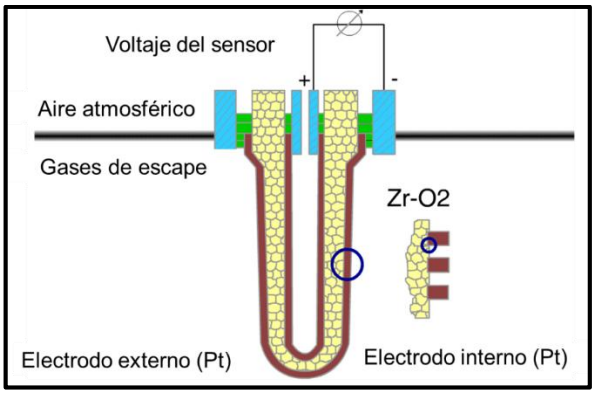
Figura 1.16. Sensor de Oxígeno

Fuente: http://www.elgenuinorepuestos.com/wp-content/uploads/2013/10/11049.jpg
[Consulta: 04 de mayo de 2013].

Se utiliza el sensor de oxígeno de zirconio, el cual está hecho de dióxido de zirconio, electrodos de platino y un calentador. Este sensor genera señal de voltaje basado en la cantidad de oxigeno contenido en los gases de escape comparado con el oxígeno de la atmósfera. El elemento de zirconio tiene un lado expuesto a los gases de escape y el otro lado hacia el aire de la atmosfera. Cada lado del elemento de zirconio tiene un electrodo de platino, el cual conduce el voltaje generado, **figura 1.17.**

La contaminación o la corrosión en los electrodos de platino o en el zirconio reducirán el voltaje de salida de la señal.

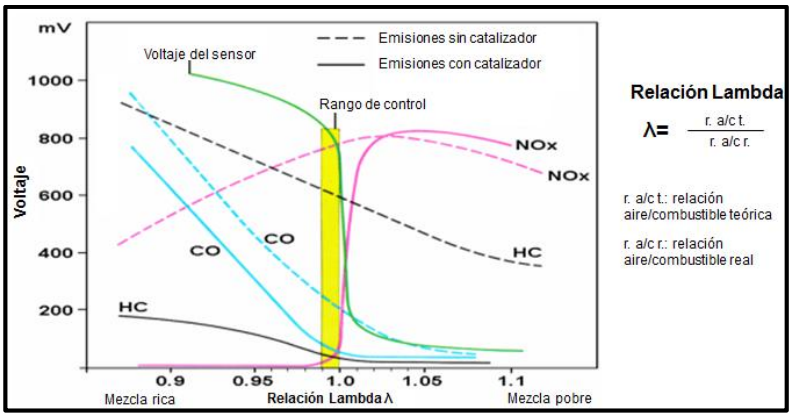
Figura 1.17. Esquema de funcionamiento de sensor de oxígeno



Fuente: *MC Gasoline Engine Spanish Mau 1*
[Consulta: 04 de mayo de 2013].

Cuando el contenido de oxígeno en el escape es alto (mezcla pobre) el voltaje del sensor es bajo y cuando el contenido de oxígeno en el escape es bajo (mezcla rica) el voltaje del sensor es alto. Gracias a esta señal, la ECU verifica la calidad de la mezcla y con ello realizar las correcciones correspondientes. Una mezcla rica consume casi todo el O2 en la combustión, entonces en los gases de escape existirá poco O2 residual, por lo que el voltaje de salida será alto en un rango de 0,6 a 1V. En una mezcla pobre "sobra" O2, por lo que provocará que después de la combustión exista bastante O2 residual, resultando un voltaje de salida bajo en un rango de 0,1 a 0,4V. Una mezcla estequiométrica de aire/combustible arrojará un voltaje cercano a 0,45V. **Figura 1.18.**

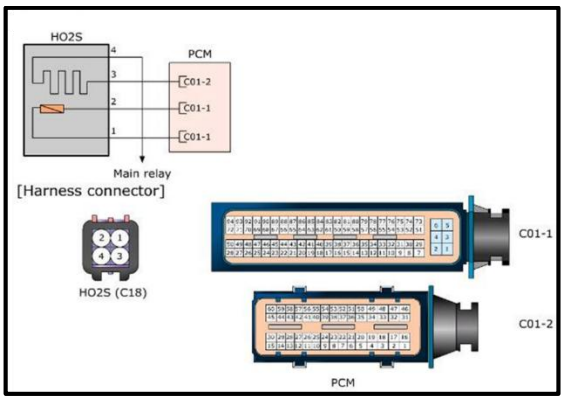
Figura 1.18. Relación estequiométrica variada por el sensor de oxígeno



Fuente: *MC Gasoline Engine Spanish Mau 1*
[Consulta: 04 de mayo de 2013].

El conexionado y la configuración interna del sensor de oxígeno se muestran en la **figura 1.19**

Figura 1.19. Conexión y diagrama interno del sensor de oxígeno



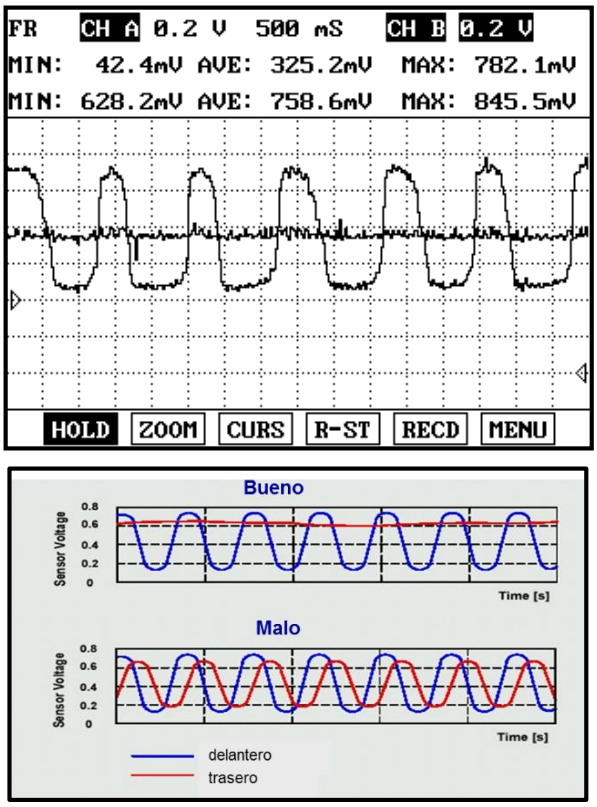
Sensor O2 (del.)	PCM Pin No.	Description
1	C01-2 (31)	Masa sensor
2	C01-1 (35)	señal
3	C01-2 (34)	calefactor
4	Relé principal	B+

Fuente: *MC Gasoline Engine Spanish Mau 1*
[Consulta: 04 de mayo de 2013].

El sensor de oxígeno trasero está después del catalizador y sirve para verificar el rendimiento de éste. La cantidad de oxígeno después de catalizar los gases de escape es relativamente baja (valor alrededor de 0,5V) y

constante en condición de no aceleración o no desaceleración. Si la señal del sensor O2 trasero varía en forma similar al delantero, significa que el catalizador no está trabajando adecuadamente. **Figura 1.20.**

Figura 1.20. Verificación del correcto funcionamiento de los sensores de oxigeno



Fuente: *MC Gasoline Engine Spanish Mau 1*

[Consulta: 04 de mayo de 2013].

1.2.7. Sensor de golpeteo (KS: Knock Sensor. Figura 1.21)

Figura 1.21. Sensor de Golpeteo

Fuente: Repuestos Hyundai. http://www.repuestoshyundai.cr/wp-content/uploads/2012/09/Sensor-golpeteo-para-Hyundai-Accent-94-99.jpg [Consulta: 04 de mayo de 2013].

El golpeteo es caracterizado por un ruido y vibración indeseables el cual puede causar daño al motor. El KS está montado en el bloque de cilindros y sensa el golpeteo del motor. La vibración/ruido del golpeteo se transmite por el bloque y se aplica como presión sobre el elemento del KS generando una señal de voltaje. La ECU controla el tiempo de encendido basado en la amplitud y frecuencia de la señal del KS. Por ejemplo: cuando el golpeteo es detectado, el tiempo de encendido es retrasado.

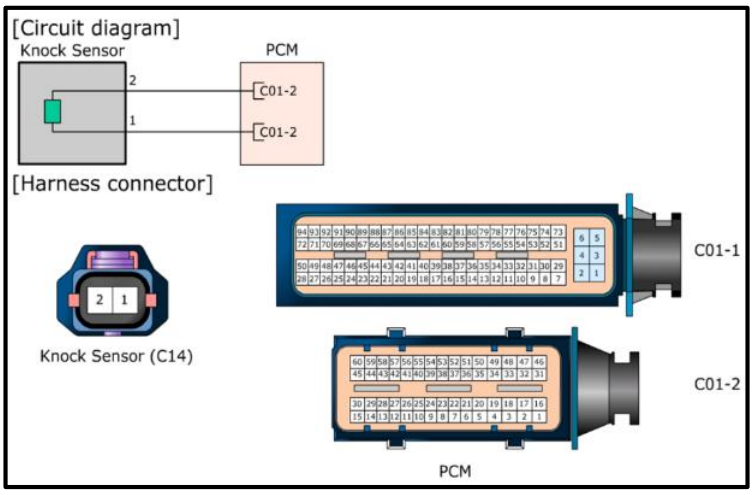
EL MTB (avance mínimo de chispa para mejor torque) es el punto de encendido en el cual se obtiene el máximo torque, y su posición es justo antes de que se produzca el golpeteo. Si no existiese control de golpeteo, el tiempo de encendido estaría preajustado un poco retardado del punto de máximo torque de manera de asegurar que el motor no golpetee, pero se obtendría menor torque. Por otro lado, el punto de ignición puede ser ajustado cerca del rango de golpeteo elevando el torque de salida del motor con el uso del KS.

El golpeteo genera una vibración de alta frecuencia (5 – 10 KHz) en el cilindro, la cual se transmite al KS por el bloque de cilindros. Como la señal de salida del KS contiene varias frecuencias mezcladas, el filtro "pasa banda" limpia la señal, la cual será usada para determinar si existe golpeteo.

El golpeteo solo ocurre durante un período de la fase de combustión, de esta forma se evita la detección errónea debido a posibles "ruidos" que puedan ocurrir. La ECU retarda el punto de ignición cuando se detecta golpeteo y luego lentamente avanza el punto de ignición una vez que ha cesado el golpeteo, como forma de retroalimentación.

En la **figura 1.22** se presenta el diagrama interno y de conexión del sensor de golpeteo.

Figura 1.22. Diagrama de interno y de conexión de sensor de golpeteo



KS	PCM Pin No.	Descripción
1	CO1-2 30	Señal alta
2	C01-2 15	Señal baja

Fuente: *MC Gasoline Engine Spanish Mau 1*
[Consulta: 04 de mayo de 2013].

1.2.8. Sensor de velocidad del vehículo (VSS: Vehicle Speed Sensor. Figura 1.23)

El sensado de la velocidad del vehículo es distinto según el caso. Cuando está el vehículo equipado con ABS o se aplica el sistema OBD, se utiliza el sensor de velocidad de la rueda (WSS) en lugar del sensor de velocidad del vehículo (VSS). El WSS también es usado para detectar situaciones incorrectas de pérdidas de chispa.

El VSS es instalado en la caja de cambios y en este caso es solo para el velocímetro.

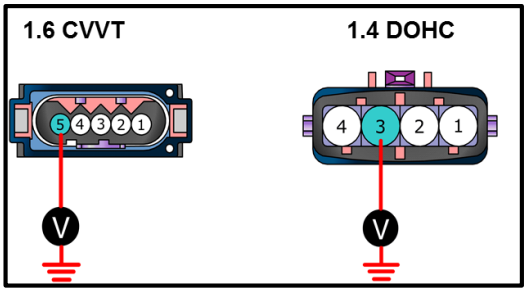
Figura 1.23. Sensor de velocidad del vehículo

Fuente: *MC Gasoline Engine Spanish Mau 1*
[Consulta: 04 de mayo de 2013].

1.2.9. Sensor de temperatura del aire de admisión (IAT: Intake Air Temperature Sensor. Figura 1.24)

El sensor IAT detecta la temperatura del aire entrante. Se utiliza para detección de la temperatura ambiente en un arranque en frío y la temperatura del aire de admisión mientras el motor calienta el aire entrante.

Figura 1.24. Diagrama de interno y de conexión del IAT



IAT	20°C	40°C
salida (V)	2.6	1.7
kΩ	2.4	1.1

Fuente: *MC Gasoline Engine Spanish Mau 1*
[Consulta: 04 de mayo de 2013].

1.2.10. Sensor de presión absoluta del múltiple (MAP: Mainfold Absolute Pressure Sensor. Figura 1.25)

En el sensor MAP existe un chip y un diafragma piezoeléctrico resistivo montado dentro de una cámara de referencia. A un lado del diafragma existe una presión de referencia la cual es un vacío perfecto y por el otro lado se expone a la presión del múltiple de admisión. El diafragma cambia su resistencia de acuerdo a los cambios de la presión del múltiple. Estos cambios

en la resistencia alteran el voltaje de la señal. La ECU interpreta estos cambios en el voltaje como cambios en la presión, en donde a mayor presión (o menor vacío), mayor es el voltaje de la señal, y a menor presión (o mayor vacío), mayor es el voltaje de la señal. Dentro de la ECU existe un convertidor de señal A/D para poder leer la señal.

La presión en el múltiple de admisión está directamente relacionado con la cantidad de aire que entra al motor y por ende la carga. Con ello se calculan parámetros como cantidad de inyección, avance de encendido, et. Con el interruptor de ignición en ON y motor detenido, en el múltiple de admisión existe presión atmosférica con un voltaje de señal de 4,3V. Cuando el motor está en ralentí la presión en el múltiple baja (o sube el vacío) a aproximadamente 35 KPa. Cuando existen problemas con esta señal, la ECU estima la cantidad de aire de admisión usando la señal del TPS y las rpm del motor.

Figura 1.25. Sensor MAP

Fuente: http://bimg1.mlstatic.com/hyundai-accent-9599-sensor-map_MLC-F-3044465452_082012.jpg
[Consulta: 04 de mayo de 2013].

1.3. Actuadores

1.3.1 Inyector de combustible. Figura 1.26.

Es una válvula solenoide controlada electrónicamente la cual suministra la cantidad de combustible exacta al motor para una óptima combustión bajo varias condiciones de carga y velocidad. Para alcanzar la relación aire-combustible requerida por el sistema, la ECU regula la cantidad de inyección de combustible operando sobre el tiempo de energizado del solenoide del inyector de acuerdo principalmente a la cantidad de aire de entrada al motor y corregido con la señal de retroalimentación del sensor O2. Para este preciso control se requiere un solenoide de rápida respuesta.

Figura 1.26. Inyector

Fuente: http://mpe-s1-p.mlstatic.com/inyector-para-hyundai-accent-tucson-inyector-para-kia-rio-478-MPE4532888237_062013-O.jpg

[Consulta: 04 de mayo de 2013].

1.3.2. Actuador de control de la velocidad de ralentí (ISCA: Idle Speed Control Actuator)

El flujo de aire necesario en ralentí es ajustado por la ISCA de acuerdo a las señales de la PCM de manera que se mantenga la velocidad en ralentí adecuado bajo variadas condiciones del motor. Estas condiciones varían debido a muchos factores, como la temperatura del motor, el aire acondicionado, cargas eléctricas, etc. La ISA es el tipo de dos bobinas, la cual una de ellas cierra la válvula y la otra la abre. En ralentí el porcentaje de activación para abrir puede ser entre 25 y 28%. Cuando existen más o menos 100 rpm de diferencia entre el ralentí real y el objetivo, la ECU activa un error y activa un DTC.

1.3.3. Válvula solenoide de control de purga (PCSV: Purge Control Solenoid Valve)

La válvula de solenoide de control de purga del canister es un artefacto que controla el flujo de vapores de combustible hacia el puerto de purga.

Figura 1.27. PCSV

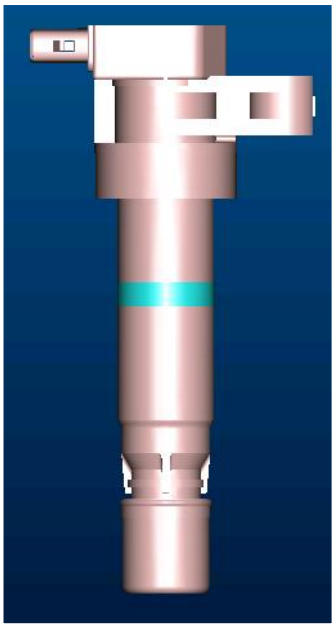
Fuente: http://i01.i.aliimg.com/wsphoto/v0/537252147/IDLE-AIR-CONTROL-VALVE-35150-22600-for-Hyundai-Accent.jpg

[Consulta: 04 de mayo de 2013].

1.3.4. Bobina de encendido

Las bobinas de encendido son las encargadas de generar alta tensión en las bujías para realizar la explosión de la mezcla aire combustible en el interior del cilindro.

Figura 1.28. Bobina de encendido



Fuente: *MC Gasoline Engine Spanish Mau 1*

[Consulta: 04 de mayo de 2013]

CAPITULO 2

DISEÑO Y PROGRAMACION DEL *FIRMWARE* DE CONEXIÓN CON LOS SENSORES, ACTUADORES Y UNIDAD ELECTRONICA DE CONTROL

En este apartado se llevará a cabo el diseño del *software* (programa) que realizará el control y la comunicación entre los diferentes elementos para obtener o generar señales con el fin de comprobar el correcto funcionamiento de un determinado elemento: sensor, actuador e incluso el correcto funcionamiento de la unidad electrónica de control.

Para ello vamos a comenzar planteando los puntos que van a ser tomados en cuenta para la selección de cada uno de las diferentes interfaces:

1. Las interfaces a diseñar interpretarán y generarán señales analógicas o digitales para los protocolos de comunicación OBD2 utilizados en los vehículos *KIA* y *HYUNDAI*.
2. Establecerá una conexión entre la ECU del automóvil y un sistema por diseñar para poder establecer comunicación entre estos dos dispositivos, con el fin de interpretar las señales de la ECU.

3. Generación e interpretación señales de tipo analógica para simular señales de sensores y actuadores.
4. Establecer una conexión entre la ECU del automóvil y una PC para visualizar los datos gráficamente a fin de interpretarlos de manera adecuada.

Los puntos señalados anteriormente han sido establecidos teniendo en cuenta que el objetivo del dispositivo es determinar, en un defecto electrónico con el automóvil, con mayor precisión la causa (sensor, actuador o unidad electrónica de control), para que sea reparada.

Empezaremos detallando los puntos 2 y 3, ya que el punto 4 se detalla por completo en el capítulo 4.

2.1. Conexión entre la ECU y la interface de generación e interpretación de señales

En el mercado automotriz y el desarrollo de las herramientas de diagnóstico de los vehículos, existen un sinnúmero de transceptores (dispositivos que realizan transmisión y recepción de datos), uno de ellos es el ELM327.

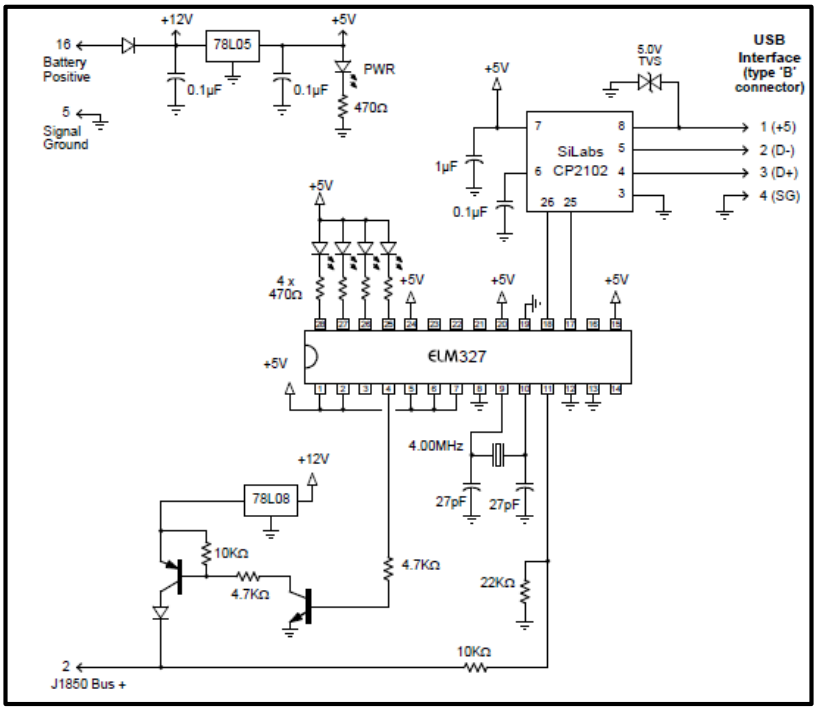
ELM327 es un microcontrolador programable producido por ELM Electrónica, montado sobre un microcontrolador 18F2480 de *Microchip Technology*. No es más que un transceptor del Diagnóstico de Interface a Bordo de segunda generación (OBD2) con la PC.

El microcontrolador ELM327 montado sobre la tarjeta de adaptación de señales y comunicación serial o usb es de bajo costo, este dispositivo se comercializa en nuestro mercado bajo el nombre ELMSCAN 5.

Al ser un dispositivo que presenta ya los pines para el ingreso de instrucciones y lectura de datos, hemos optado por la adquisición de uno de ellos, con el afán de abaratar los costos totales del proyecto, además de establecer adecuadamente la comunicación con la ECU del automóvil.

A continuación se presenta el diagrama interno del ELMSCAN 5 con un microcontrolador ELM327:

Figura 2.1. Configuración interna ELMSCAN 5 con ELM327



Fuente: http://www.elmelectronics.com/DSheets/ELM327DS.pdf

[Consulta: 25 de julio de 2013].

2.2. Diseño de interface de interpretación y generación de señales

Al igual que la interface de conexión con la ECU, existen en el mercado de la electrónica varias interfaces para transmisión y recepción de datos.

En nuestro caso vamos a acoplar una de ellas, para que el costo del proyecto cada vez se reduzca.

Lo que se necesita es una interface que sea capaz de realizar la transmisión, recepción y visualización datos, realice cálculos dentro de su programación y convierta datos hexadecimales a decimales.

Uno de ellos es la tarjeta **ARDUINO MEGA2560** que además de cumplir con las necesidades anteriormente mencionadas, tiene características adicionales como generación de PWM (modulación por ancho de pulso), característica muy importante ya que alguno de los actuadores o sensores funcionan en base a esta modulación.

Algunas características adicionales de la tarjeta se presentan a continuación:

Microcontroladores	ATmega2560
Tensión de funcionamiento	5V
Voltaje de entrada (recomendado)	7-12V
Voltaje de entrada (límites)	6-20V
Digital I / O Pins	54 (de los cuales 15 proporcionan salida PWM)
Pines de entrada analógica	16
Corriente continua para las E / S Pin	40 mA
Corriente de la CC para Pin 3.3V	50 mA
Memoria Flash	256 KB, 8 KB utilizado por el gestor de arranque
SRAM	8 KB
EEPROM	4 KB
Velocidad del reloj	16 MHz

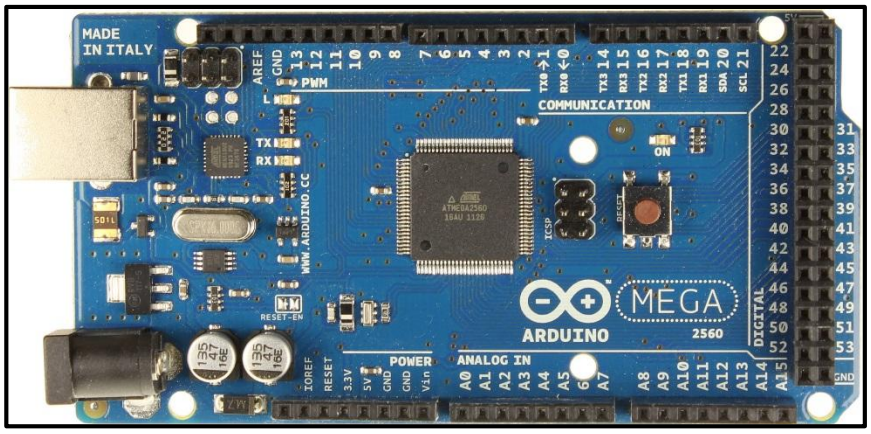
Tabla 2.1. Características de la Tarjeta Arduino Mega 2560
Fuente: ARDUINO http://arduino.cc/en/Main/arduinoBoardMega2560
[Consulta: 25 de julio de 2013].

2.2.1. Comunicación

Una tarjeta Arduino Mega 2560, posee varios pines para la comunicación con una PC, otra tarjeta Arduino o con otros microcontroladores. El *software* de programación incluye un monitor para control de los datos de transmisión TX y RX pines 0 y 1, dicha comunicación es de tipo serial. El conector USB no controla la comunicación por los pines 0 y 1.

Cabe recalcar que Arduino es una plataforma abierta, de programación de alto nivel, para simplificar los códigos de programación.

Figura 2.2. Tarjeta Arduino Mega 2560



Fuente: http://arduino.cc/en/uploads/Main/ArduinoMega2560_R3_Front.jpg
[Consulta: 25 de julio de 2013].

2.3. Diseño del *firmware* para la comunicación, interpretación y generación de señales.

Una vez seleccionados los dispositivos que nos ayudarán a comunicarnos con la ECU del automóvil, vamos a diseñar el *firmware*.

Paralelamente se va diseñando el hardware necesario para la adaptación de las señales generadas para comprobar sensores y actuadores, para leer los

datos de cada uno de los sensores y de la ECU. La construcción de dicho *hardware* se detalla en el siguiente capítulo.

La interface utilizada para la interacción entre el usuario y el dispositivo es un lcd matricial 16X4 como el que se muestra en la **figura 2.3**.

Figura 2.3. LCD matricial 16 X4

Fuente: es.aliexpress.com [Consulta: 25 de julio de 2013].

2.4. Diagrama de flujo del funcionamiento del dispositivo a diseñar

En los siguientes diagramas de flujo se observa cual será el funcionamiento de la interface del dispositivo que se está diseñado.

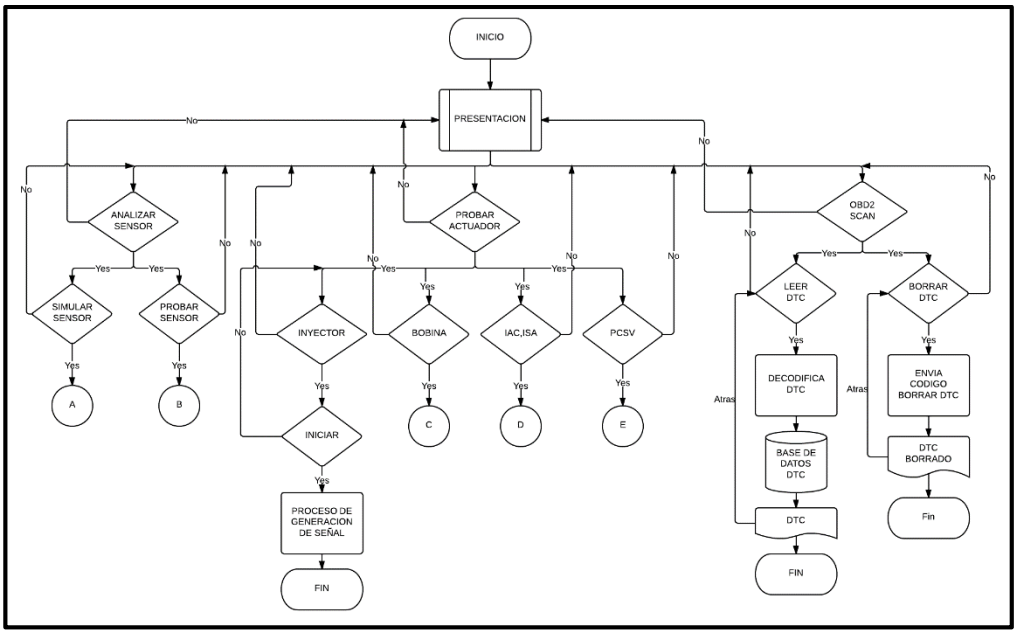


Figura 2.4. Diagrama de flujo inicial

En el diagrama de flujo inicial se puede observar que en principio se establece una presentación del nombre del dispositivo y su versión (PRESENTACION). Inmediatamente después de aquello, el usuario tiene las siguientes opciones:

- Analizar sensor: que a su vez ofrece dos opciones adicionales

  - Simular sensor: ofrece la posibilidad de simular señales mediante voltajes caracterizados, según el sensor que se elija en el siguiente menú.

    Simular sensor tiene en su característica visual la posibilidad de que el usuario simule el funcionamiento del sensor correspondiente.

    Es así que cuando se elige esta opción en el visualizador se observa una lista de sensores que pueden ser elegidos, entre ellos tenemos: VSS,

TPS, CKP, ECT, IAT, MAF, MAP, H02S. Cada una de las caracterizaciones de los sensores serán explicados en puntos posteriores.

- Probar sensor: esta parte del menú da la posibilidad al usuario, sin desmontar los sensores del vehículo, comprobar su funcionamiento. Mediante la inyección de voltajes para comprobar el correcto funcionamiento de los mismos.

Para activar a cada uno de los sensores se ha estudiado su comportamiento con el fin de poder establecer su rango de funcionamiento correcto. Para cada uno de los sensores se coloca el rango correcto de funcionamiento para calificarlo como correcto, y si se encuentra fuera como sensor defectuoso.

Cada una de estas relaciones se detalla en puntos posteriores.

Si no se desea realizar operaciones con los sensores, se selecciona retroceder, para dirigirnos al segundo ítem del primer menú, es decir, probar actuador.

- Probar Actuador: al ingresar a esta opción, el menú que aparece permite seleccionar el actuador a comprobar; los mismos son: Inyector, Bobina, IAC/ISA, PCSV.

-

Al seleccionar cualquiera de ellos, el programa deja visualizar en el LCD, las opciones: Iniciar o Atrás.

Iniciar: Genera la forma de onda necesaria para activar el actuador seleccionado. Más adelante se detalla cada uno de los procesos para generar las señales.

- La última opción que deja visualizar el dispositivo es OBDII SCAN.

Si se selecciona esta opción deja ver el siguiente menú en el cual se observa:

- Leer DTC (Código de Diagnóstico): Seleccionando esta opción, inmediatamente se visualiza el o los códigos de diagnóstico o más conocido como código de falla que posea el vehículo; o se indica que el vehículo no tiene DTC.
- Borrar DTC: En caso de existir código de diagnóstico y luego de haber sido reparada la falla que éste indicaba, se debe borrar el mismo de la memoria de la ECU; al seleccionar esta opción se borran los DTC existentes en el mismo. La comprobación de que esta opción funciona correctamente es el apagado de la luz testigo de *CHECK ENGINE*.

Ahora se detallarán los procesos internos programación para poder interpretar y generar las señales a fin de detectar errores en el sistema del vehículo.

Comenzaremos indicando el diagrama de flujo general para simular un sensor, para luego detallar, por sus características cada uno de ellos.

2.4.1. Simular sensor

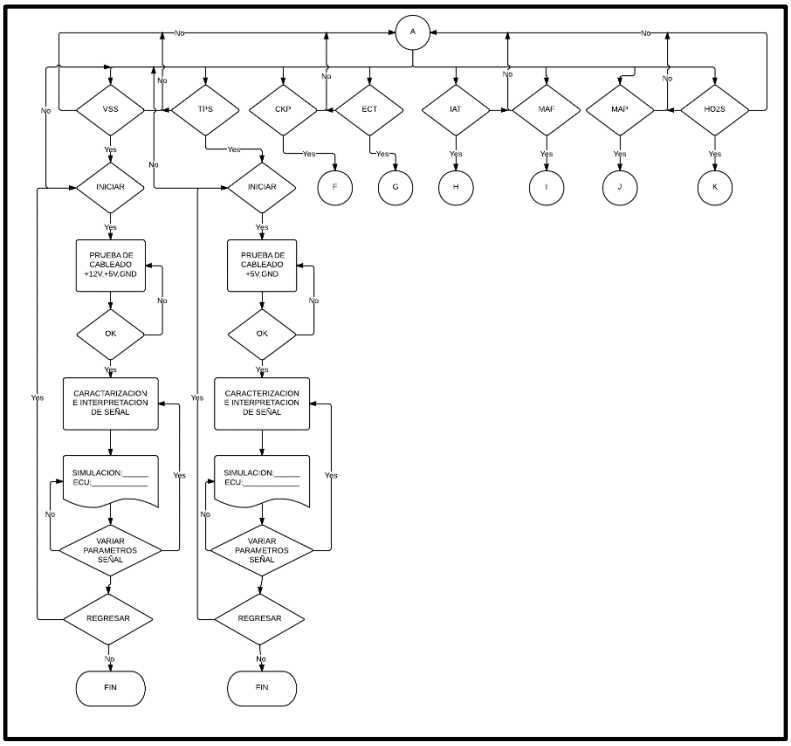


Figura 2.5. Diagrama de flujo para simular un sensor

Con la finalidad de poder simular correctamente una señal de sensor, se verificará la misma mediante la lectura que hace la ECU, indicando a la vez las dos señales.

Esto significa que para cada uno de los cambios de la señal generada, la ECU indicará el cambio que ha detectado en su entrada.

Además dependiendo del sensor que se esté simulando, se realiza previamente la comprobación de su cableado. Si el cableado no está funcionando correctamente, el programa no dejará simular el sensor.

Gracias a esta disposición se puede comprobar a la vez el cableado del sensor y el correcto funcionamiento de las salidas de la ECU para cada uno de los sensores analizados.

2.4.1.1. Diagrama de flujo para simular el sensor VSS

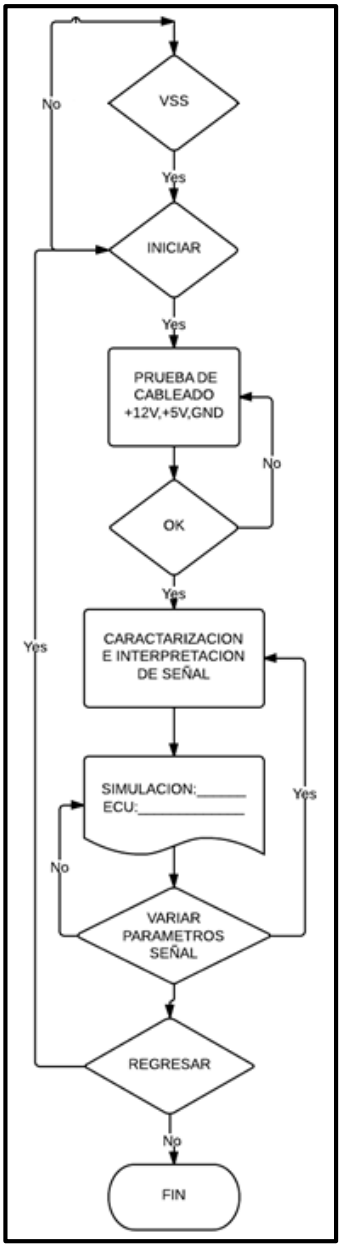


Figura 2.6. Diagrama de flujo para simular sensor VSS

Antes de iniciar la simulación, vamos a conectar el equipo al conector del sensor VSS hacia la ECU, el equipo diagnostica la conexión con la misma evaluando los voltajes en cada uno de los pines del conector.

Enseguida verifica que los voltajes son los correctos, el equipo empieza a simular los pulsos generados por el sensor, verificando el correcto funcionamiento de la ECU comparándola con el valor generado por el equipo.

En la figura 2.7 se muestra la programación para generar la señal análoga del sensor VSS.

```
//////////programa para simular vss//////////////
if (cartel==4)
   {
    delay(200);
    estadoVTPS = digitalRead(VTPS);
    if (estadoVTPS == HIGH)
    {
     lcd.setCursor(0, 2);lcd.print("B+  -> Error ---X---");
     delay(500);
     lcd.setCursor(0, 2);lcd.print("                    ");
    }
    else
    {
     delay(700);
     lcd.setCursor(0, 2);lcd.print("B+  -> Ok           ");
    }
    estadoGTPS = digitalRead(GTPS);
    if (estadoGTPS == HIGH)
    {
     lcd.setCursor(0, 3);lcd.print("GND -> Error ---X---");
     delay(500);
     lcd.setCursor(0, 3);lcd.print("                    ");
    }
    else
    {
     delay(700);
     lcd.setCursor(0, 3);lcd.print("GND -> Ok           ");
     delay(700);
    }
    if (estadoVTPS==LOW )
    {
     if (estadoGTPS==LOW)
     {
      digitalWrite(22, HIGH);
      cartel=12;CartelLCD();
      lcd.setCursor(0, 2);lcd.print("Simulado:");
      lcd.setCursor(0, 3);lcd.print("ECU:");
      TSensor=0;LVSS=11;
      ////////////////////////////////////////////////////
      //////////Configuracion de interrupciones Timer5//////
      cli();
      TCCR5A=0;
      TCCR5B=0;
      OCR5A=1000;
      TCCR5B |= (1<<WGM12);
      TCCR5B |= (1<<CS10);
      TCCR5B |= (1<<CS12);
      TIMSK5=(1<<OCIE1A);
      sei();
      ////////////////////////////////////////////////////
      ////////////////////////////////////////////////////
      Serial1.write("ATSP6\r"); ////ISO 15765-4 CAN 11/500
      while(cartel==12)
```

```
                                {
                                  adc_key_in = analogRead(0);
                                  key = get_key(adc_key_in);
                                  if(key==1)
                                  {
                                   if (OCR5A==100)
                                   {
                                   }
                                   else
                                   {
                                    OCR5A=OCR5A-10;
                                    LVSS=((0.000000001393*OCR5A*OCR5A*OCR5A*OCR5A)-
(0.000003561*OCR5A*OCR5A*OCR5A)+(0.003258*OCR5A*OCR5A)-(1.288*OCR5A)+210.3);
                                    lcd.setCursor(10, 2);lcd.print("    Km/h");
                                   }
                                  }
                                  if(key==2)
                                  {
                                   if (OCR5A==1000)
                                   {
                                   }
                                   else
                                   {
                                    OCR5A=OCR5A+10;
                                    LVSS=((0.000000001393*OCR5A*OCR5A*OCR5A*OCR5A)-
(0.000003561*OCR5A*OCR5A*OCR5A)+(0.003258*OCR5A*OCR5A)-(1.288*OCR5A)+210.3);
                                    lcd.setCursor(10, 2);lcd.print("    Km/h");
                                   }
                                  }
                                  if (cartel==3)
                                  {
                                   digitalWrite(SVSS, HIGH);
                                  }
                                  else
                                  {
                                   lcd.setCursor(10, 2);lcd.print(LVSS);
                                  }
                                  //////////////VSS OBDII//////////////////
                                  delay(250);
                                  Serial1.write("010D\r");
                                  SerialOBDII();
                                  ////////////////////////////////////
                                  if ( key == 0)
                                  {
                                   digitalWrite(22, LOW);
                                   cartel=3;CartelLCD();TVSS=1000;
                                   TSensor=0;Dato=0;PosDatoSerial=0;valVss=0;Dato1;Dato2=0;
                                   ///////Fin de interrupcion///////
                                   cli();
                                   TCCR5A=0;
                                   TCCR5B=0;
                                   sei();
```

Figura 2.7. Programa para simular sensor VSS

2.4.1.2. Diagrama de flujo para simular el sensor TPS

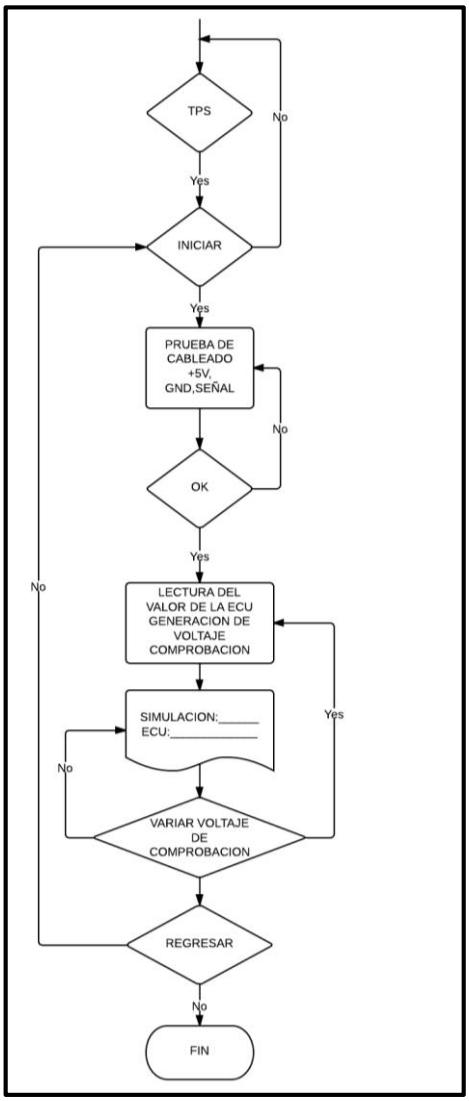
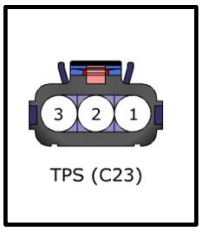


Figura 2.8. Diagrama de flujo simular sensor TPS

Para iniciar la simulación se debe conectar los cables al conector del sensor que va hacia la ECU, para recordar su configuración se presenta la figura 2.9:



TPS	PCM Pin No.	Descripción
1	C01-1 75	Señal
2	C01-2 58	Referencia (5V)
3	C01-1 51	masa

Figura 2.9. Pines de conexión sensor TPS

La opción simular TPS, en caso de iniciarlo, entra en un bucle, el cual mide la aplicación de voltaje de 5V sobre el mismo, así como su derivación a GND. En caso de no ser correcta la aplicación de estas referencias no se puede realizar la simulación del sensor.

Si los voltajes en el sensor son correctos, en el LCD se visualizará los datos generados de simulación, así como los valores leídos por la ECU.

Para generar una variación de voltaje en el sensor, como se indicó se utiliza un partidor de tensión, en el cual se genera una variación de voltaje para que sea medido en el sensor.

Para que la tarjeta Arduino Mega 2560 genere una variación de voltaje entre 0 y 5V se utiliza su salida analógica que posee una resolución de 255, es decir que para generar internamente una salida de 5V, la variación interna en la tarjeta es de 5/255=19,6 mV por división.

El circuito externo para el acondicionamiento de la señal, posee un integrado MCP41001, que no es más que una resistencia digital variable. Al

variar digitalmente una de sus entradas, su resistencia interna varía, generando una variación en la salida de voltaje en el partidor de tensión. Su resolución es igual al de la tarjeta Arduino.

Como la resolución de los dos dispositivos es de 255 y aprovechamos el PID del sensor TPS que señala que el voltaje en la salida de la ECU es igual a:

$$PID\ TPS = \frac{A * 100}{255} \quad (\%\ apertura)$$

Donde A en la salida de la ECU es el valor hexadecimal leído por la misma de la posición del TPS.

Ahora se utiliza la misma relación invertida para aproximar valores generados de voltaje. Relación invertida ya que el potenciómetro digital permite el paso de 5V cuando se le aplica 0V (0 de la salida analógica de tarjeta Arduino) y 0V cuando se aplica una tensión de 5V (255 en la salida analógica de la tarjeta Arduino).

Entonces la relación de generación de señal para simular el sensor TPS es:

$$Señal\ TPS = \frac{255 - X + 1}{255} * 100$$

Siendo X el valor generado en el programa para ser enviado al potenciómetro digital tendiendo en la salida un valor de voltaje entre 0 y 5V, referido a su PID correspondiente.

Para realizar la comparación correspondiente con el valor leído por la ECU, se envía hacia ELM327 la instrucción del PID del sensor TPS "0111\r", lo que devuelve el valor leído e interpretado por la ECU.

La respectiva programación se presenta en la figura 2.10.

```
                ////////////////PROGRAMA PARA SIMULAR SENSOR TPS/////////////
if (cartel==4)
                {
                 delay(200);
                 estadoVTPS = digitalRead(VTPS);
                 if (estadoVTPS == HIGH)
                 {
                   lcd.setCursor(0, 2);lcd.print("5v  -> Error ---X---");
                   delay(500);
                   lcd.setCursor(0, 2); lcd.print("                    ");
                 }
                 else
                 {
                   delay(700);
                   lcd.setCursor(0, 2);lcd.print("5v  -> Ok          ");
                 }

                 estadoGTPS = digitalRead(GTPS);
                 if (estadoGTPS == HIGH)
                 {
                   lcd.setCursor(0, 3);lcd.print("GND -> Error ---X---");
                   delay(500);
                   lcd.setCursor(0, 3);lcd.print("                    ");
                 }
                 else
                 {
                   delay(700);
                   lcd.setCursor(0, 3);lcd.print("GND -> Ok          ");
                   delay(700);
                 }
                 if (estadoVTPS==LOW )
                 {
                   if (estadoGTPS==LOW)
                   {
                     cartel=12;CartelLCD();
                     lcd.setCursor(0, 2);lcd.print("Simulado:");
                     lcd.setCursor(0, 3);lcd.print("ECU:");
                     TSensor=1;                          ///Selecciona tipo de sensor
                     Serial1.write("ATSP6\r");                ////ISO 15765-4 CAN 11/500

                     while(cartel==12)
                     {
```

```
                adc_key_in = analogRead(0);
                key = get_key(adc_key_in);
                if(key==1)
                {
                 if ( STPS == 0)
                 {
                 }
                 else
                 {
                  delay(10);
                  STPS= STPS-1;
                  LTPS=(((255-STPS+1)*100)/255);
                  //LTPS=LTPS+1;
                  lcd.setCursor(10, 2);lcd.print("   %");
                 }
                }
                if(key==2)
                {
                 if (STPS==255)
                 {
                 }
                 else
                 {
                  delay(10);
                  STPS= STPS+1;
                  LTPS=(((255-STPS-1)*100)/255);
                  //STPS=STPS+3;
                  //LTPS=LTPS-1;
                  lcd.setCursor(10, 2);lcd.print("   %");
                 }
                }
                if (cartel==5)
                {
                 digitalPotWrite();
                }
                else
                {
                 digitalPotWrite();
                 lcd.setCursor(10, 2);lcd.print(LTPS);
                }

                ////////////////////TPS OBDII////////////////////
                delay(250);
                Serial1.write("0111\r");///PID 0111 -->PETICION TPS
                SerialOBDII();
                ////////////////////////////////////////////////
                if ( key == 0)
                {
                 cartel=5;CartelLCD();STPS=255;
                 cont=0;ini=0;TSensor=0;Dato=0;PosDatoSerial=0;valTPS=0;LTPS=0;
                 delay(200);
           ////////////FIN PROGRAMA PARA SIMULAR SENSOR TPS/////////////
```

Figura 2.10. Programación de Simulación de Sensor TPS

2.4.1.3. Diagrama de flujo para simular el sensor CKP

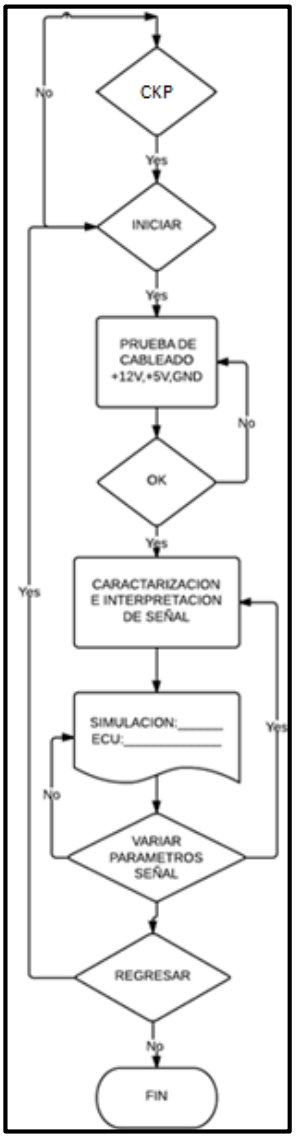


Figura 2.11. Diagrama de flujo de Simulación de Sensor CKP

Al igual que en los sensores anteriores, conectamos el soque del sensor CKP que va hacia la ECU al equipo, para generar la señal que interpretará la ECU para comprobar el correcto funcionamiento de la misma comparándola con el valor generado.

De idéntica manera el sistema primero evaluará los voltajes en el soque para poder realizar la simulación.

En la figura 2.12 se presenta la programación para la simulación del sensor CKP.

```
                    ////////////////////////////////////////////////////////
                    ///////////////PROGRAMA PARA SIMULAR SENSOR CKP///////////////
                    if (cartel==4)
                    {

                        digitalWrite(22, HIGH);
                        digitalWrite(35, HIGH);
                        cartel=12;CartelLCD();
                        lcd.setCursor(0, 2);lcd.print("Simulado:");
                        lcd.setCursor(0, 3);lcd.print("ECU:");
                        TSensor=2;                              ////Selecciona tipo de sensor
                        ////////////////////////////////////////////////////
                        /////////Configuracion de interrupciones Timer5//////
                        cli();
                        TCCR5A=0;
                        TCCR5B=0;
                        OCR5A=1000;
                        TCCR5B |= (1<<WGM12);
                        TCCR5B |= (1<<CS10);
                        TCCR5B |= (1<<CS12);
                        TIMSK5=(1<<OCIE1A);
                        sei();
                        ////////////////////////////////////////////////////
                        ////////////////////////////////////////////////////
                        Serial1.write("ATSP6\r");                   ////ISO 15765-4 CAN 11/500

                        while(cartel==12)
                        {
                           adc_key_in = analogRead(0);
                           key = get_key(adc_key_in);


                           ////////////////CKP OBDII/////////////////////////////
                           delay(250);
                           Serial1.write("010C\r");///PID Peticion CKP
                           SerialOBDII();
                           ////////////////////////////////////////////////////
                           if ( key == 0)
                           {
                            digitalWrite(22, LOW);
                            digitalWrite(35, LOW);
                            cartel=6;CartelLCD();TVSS=1000;
                            TSensor=0;Dato=0;Dato2=0;Dato1=0;PosDatoSerial=0;
                            analogWrite(PinPWM1, 0);

                            //////////Fin de interrupcion Timer 5=OFF//////////
                            cli();
                            TCCR5A=0;
                            TCCR5B=0;
                            sei();
                            ////////////////////////////////////////////////////
                           }
                        }
                    }
                    ///////////////FIN DE PROGRAMA PARA SIMULAR CKP//////////////////
                    ////////////////////////////////////////////////////////////////
                }
```

Figura 2.12. Programación de Simulación de Sensor CKP

2.4.1.4. Diagrama de flujo para simular el sensor ECT e IAT

Ahora se detalla el proceso para la programación de la simulación de los sensores ECT e IAT.

Su programación es idéntica, ya que son sensores que funcionan dentro de un mismo rango de temperatura.

La diferencia principal del ECT con la IAT, es su ensamblaje físico, ya que la IAT está ensamblada con el MAP, en el automóvil que se tiene disponible para el desarrollo del programa, además que el ECT necesita una alimentación de 5V para su funcionamiento.

Para la conexión del IAT, recordamos la conexión mediante la figura 2.13

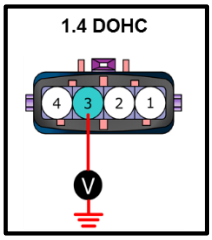
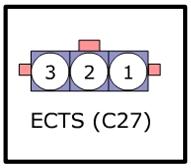


Figura 2.13. Pines de conexión IAT

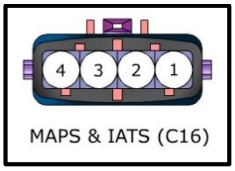
Donde el pin 3 es la señal del sensor y el pin 4 es la conexión GND.
Para la conexión del ECT se tiene la figura 2.14.



ECT	PCM Pin No.	Descripción
1	C01-1 77	Señal
2	Tablero	Medidor
3	C01-1 73	Masa

Figura 2.14. Pines de conexión ECT

Para la conexión del sensor IAT tenemos la siguiente configuración, recordando que esta ensamblado físicamente en conjunto con el MAP.



MAP	PCM Pin No.	Descripción
1	C01-1 10	Señal MAP
2	C01-2 43	Voltaje de referencia (5V)
3	C01-1 32	Señal IAT
4	C01-1 8	Masa

Figura 2.15. Pines de conexión MAP e IAT

El siguiente diagrama de flujo representa la programación para la simulación de los sensores ECT e IAT.

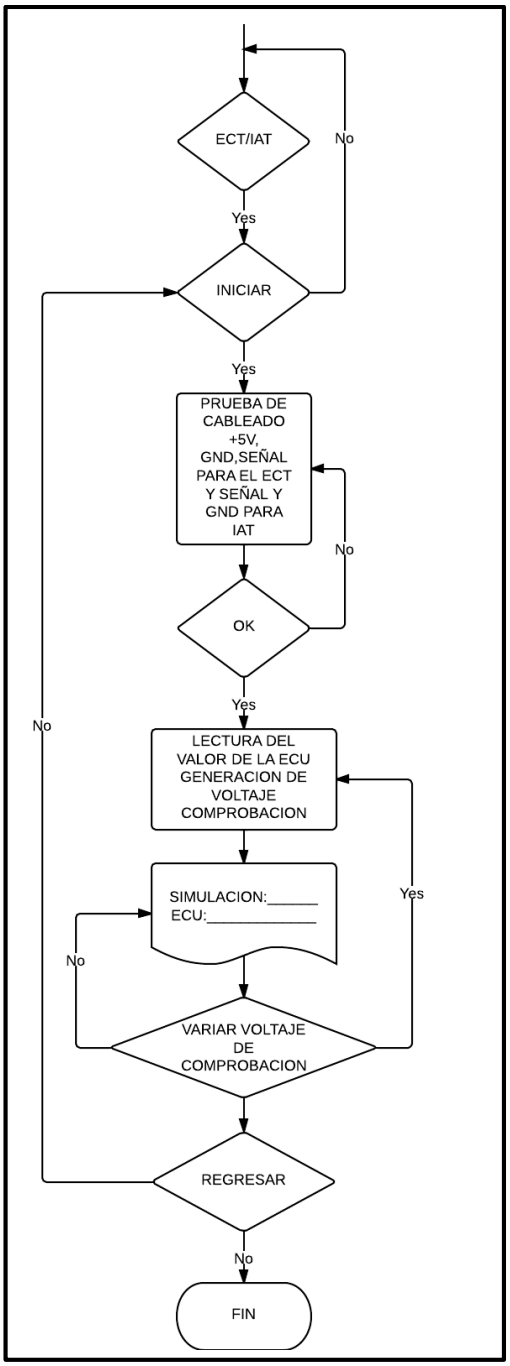


Figura 2.16. Diagrama de Flujo Programación ECT e IAT

Existe analogía de funcionamiento entre el programa de los sensores que anteriormente se han detallado con el que presentamos ECT, IAT.

A continuación se detalla la programación para los dos sensores haciendo diferencia cuando sea necesario.

Se selecciona la opción de iniciar, enseguida el dispositivo comprueba el cableado respectivo del sensor que este a prueba.

Finalizada la prueba de cableado, ingresa al bucle en el cual genera una señal que será leída por ECU para ser interpretada como señal del respectivo sensor.

La generación de la señal referida a los pulsos en la entrada y la visualización en incrementos en el LCD es más compleja que la anterior, pues en este caso los valores generados por cualquiera de los dos sensores es exponencial, de modo que para generar un valor de señal, haciendo un incremento de una unidad en LCD, representa un incremento exponencial en salida analógica de la tarjeta Arduino.

Entonces para poder visualizar paralelamente los valores simulados y valor de la ECU, recurrimos a ajustar una curva. Para la obtención de los puntos para el ajuste ingresamos valores a la resolución de la salida analógica de la tarjeta Arduino (entre 0 y 255 que representan 0 a 5V), obteniendo los valores respectivos de temperatura, con esos valores se obtiene la siguiente ecuación:

STPS= ((0.000000024067*LTPS*LTPS*LTPS*LTPS*LTPS) - (0.000010094*LTPS*LTPS*LTPS*LTPS) + (0.0016854*LTPS*LTPS*LTPS) - (0.14494*LTPS*LTPS) +6.8711*LTPS + 98.256)

Siendo STPS la variable que genera una variación en la salida analógica de la tarjeta Arduino y LTPS los valores que varía el usuario para comprobar el sensor.

Con esta relación se presentan con exactitud los valores simulados y ECU para comprobar exactamente el funcionamiento del sensor.

Tanto para el ECT, como para el IAT la programación es la misma puesto que los dos sensores funcionan en el mismo rango de temperatura y bajo el mismo principio, son sensores exponenciales.

Para obtener y comparar el valor que está siendo leído por la ECU se envía la instrucción AT **0105\r para el ECT o 010F\r para el IAT,** a la ECU mediante el ELM327, de manera que la programación es la siguiente:

```
////////////////PROGRAMA PARA SIMULACION DEL SENSOR ECT/////////////////
if (cartel==4)
{
 delay(100);
 estadoGTPS = digitalRead(GTPS);
 if (estadoGTPS == HIGH)
 {
  lcd.setCursor(0, 3);lcd.print("GND -> Error ---X---");
  delay(500);
  lcd.setCursor(0, 3);lcd.print("                    ");
 }
 else
 {
  delay(700);
  lcd.setCursor(0, 3);lcd.print("GND -> Ok          ");
  delay(800);
 }

  if (estadoGTPS==LOW)
  {
   digitalWrite(23, HIGH);
   cartel=12;CartelLCD();
   lcd.setCursor(0, 2);lcd.print("Simulado:");
   lcd.setCursor(0, 3);lcd.print("ECU:");

   TSensor=3;STPS=96;
   Serial1.write("ATSP6\r");                    ////ISO 15765-4 CAN 11/500
```

```
                        while(cartel==12)
                        {
                          adc_key_in = analogRead(0);
                          key = get_key(adc_key_in);
                          if(key==1)
                          {
                           if ( STPS == 255)
                           {
                           }
                           else
                           {
                            delay(20);
                            LTPS=LTPS+1;
                            STPS= ((0.000000024067*LTPS*LTPS*LTPS*LTPS*LTPS) -
(0.000010094*LTPS*LTPS*LTPS*LTPS) + (0.0016854*LTPS*LTPS*LTPS) - (0.14494*LTPS*LTPS) +6.8711*LTPS +
98.256);
                            lcd.setCursor(10, 2);lcd.print("   C");
                           }
                          }
                          if(key==2)
                          {
                           if (STPS==96)
                           {
                           }
                           else
                           {
                            delay(30);
                            LTPS=LTPS-1;
                            STPS= ((0.000000024067*LTPS*LTPS*LTPS*LTPS*LTPS) -
(0.000010094*LTPS*LTPS*LTPS*LTPS) + (0.0016854*LTPS*LTPS*LTPS) - (0.14494*LTPS*LTPS) +6.8711*LTPS +
98.256);
                            lcd.setCursor(10, 2);lcd.print("   C");
                           }
                          }
                          if (cartel==7)
                          {
                           digitalPotWrite();
                          }
                          else
                          {
                           digitalPotWrite();
                           //LTPS= (((STPS*80)/249)-30);
                           lcd.setCursor(10, 2);lcd.print(LTPS);
                          }

                          ////////////////////////ECT OBDII////////////////////////////////
                          delay(250);
                          Serial1.write("0105\r");  ///PID peticion ECT
                          SerialOBDII();
                          ////////////////////////////////////////////////////////////////
                          if ( key == 0)
                          {
                           cartel=7;CartelLCD();STPS=96;
                           cont=0;ini=0;TSensor=0;Dato=0;Dato1=0;Dato2=0;PosDatoSerial=0;valTEMP=0;LTPS=0;
                           digitalWrite(23, LOW);
                           delay(200)
```

Figura 2.17. Programación para simulación de sensor ECT e IAT

2.4.1.5. Diagrama de flujo para simular el sensor MAF

El vehículo en el cual estamos realizando las pruebas para el diseño y la construcción del equipo es un *KIA RIO XCITE 1.4* y posee únicamente sensor MAP, ya que el sensor MAF viene montado en otro modelo de motor, no hemos podido realizar las pruebas sobre este sensor, pero se deja un espacio en la programación para poder realizar mejoras a futuro del equipo.

2.4.1.6. Diagrama de flujo para simular el sensor MAP

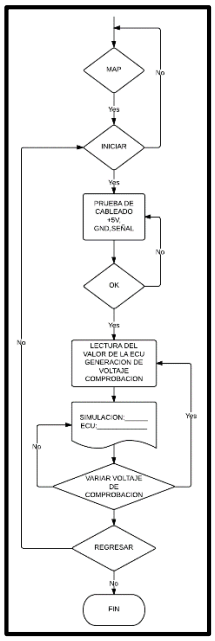


Figura 2.18. Diagrama de Flujo Programación MAP

Todos los análisis para la generación de las señales suponen la misma programación, con la diferencia de las relaciones para la visualización en el LCD, debido a los respectivos funcionamientos de los sensores.

En este caso del MAP, la generación de señal es mucho más sencilla, ya que el funcionamiento del sensor es lineal, así que un cambio en la salida de la tarjeta Arduino es proporcional al cambio en el ingreso del usuario. Tan proporcional como 2 a 1, quedando representado en la siguiente ecuación:

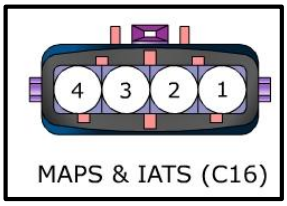
STPS=STPS-2;

LTPS=LTPS+1;

Donde SPTS en este caso enviado a la salida análoga de la tarjeta Arduino que genera un voltaje proporcional a la entrada LTPS manipulada por el usuario, para comprobar el sensor. El voltaje aplicado oscila entre 0 y 5V.

El valor recibido por la ECU será verificado mediante la instrucción AT de ELM327 ingresando **0110\r** comparándolo con el voltaje ingresado hacia la ECU, debiendo ser los mismos valores.

Para recordar la conexión del MAP para la simulación tenemos:



MAP	PCM Pin No.	Descripción
1	C01-1 10	Señal MAP
2	C01-2 43	Voltaje de referencia (5V)
3	C01-1 32	Señal IAT
4	C01-1 8	Masa

Figura 2.19. Pines de conexión sensor MAP

El programa para la simulación del sensor MAP se presenta en la figura 2.20:

```
//////////////////////PROGRAMA PARA SIMULAR SENSOR MAF/////////////////
 if (cartel==4)
 {
  delay(200);
  estadoVVSS = digitalRead(VVSS);
  if (estadoVVSS == HIGH)
  {
    lcd.setCursor(0, 2); lcd.print("B+  -> Error ---X---");
    delay(500);
    lcd.setCursor(0, 2); lcd.print("                    ");
  }
  else
  {
    delay(700);
    lcd.setCursor(0, 2);  lcd.print("B+  -> Ok           ");
  }

  estadoGVSS = digitalRead(GVSS);
  if (estadoGVSS == HIGH)
  {
    lcd.setCursor(0, 3); lcd.print("GND -> Error ---X---");
    delay(500);
    lcd.setCursor(0, 3); lcd.print("                    ");
  }
  else
  {
    delay(700);
    lcd.setCursor(0, 3); lcd.print("GND -> Ok           ");
    delay(700);
  }

  if (estadoVVSS==LOW )
  {
    if (estadoGVSS==LOW)
    {
      cartel=12;CartelLCD();
      lcd.setCursor(0, 2);lcd.print("Simulado:");
      lcd.setCursor(0, 3);lcd.print("ECU:");
      TSensor=4;                          ///Selecciona tipo de sensor
      Serial1.write("ATSP6\r");                ////ISO 15765-4 CAN 11/500

      while(cartel==12)
      {
         adc_key_in = analogRead(0);
         key = get_key(adc_key_in);
         if(key==1)
         {
          if ( STPS == 0)
          {
          }
          else
          {
            delay(20);
            STPS=STPS-2;
            LTPS=LTPS+1;
            lcd.setCursor(10, 2);lcd.print("   grams/s");
          }
         }
         if(key==2)
```

```
{
 if (STPS==255)
 {
 }
 else
 {
  delay(30);
  STPS=STPS+2;
  LTPS=LTPS-1;
  lcd.setCursor(10, 2);lcd.print("   grams/s");
 }
}
if (cartel==5)
{
 digitalPotWrite();
}
else
{
 digitalPotWrite();
 lcd.setCursor(10, 2);lcd.print(LTPS);
}

//////////////////////MAF OBDII//////////////////////
delay(250);
Serial1.write("0110\r");///PID 011O -->PETICION MAF
SerialOBDII();
////////////////////////////////////////////////////
if ( key == 0)
{
 cartel=9;CartelLCD();STPS=255;
 cont=0;ini=0;TSensor=0;Dato=0;PosDatoSerial=0;valTPS=0;LTPS=0;
 delay(200);
}
```

Figura 2.20. Programación simulación sensor MAP

2.4.1.7. Diagrama de flujo para simular el sensor H02S

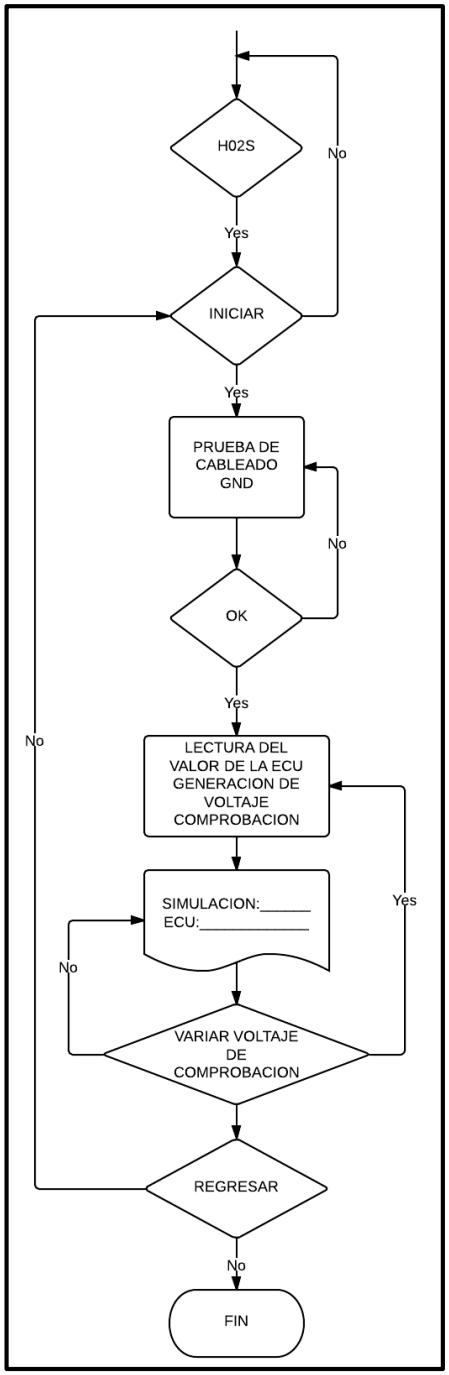


Figura 2.21. Diagrama de Flujo Programación HO2S

La conexión para simular el sensor de oxígeno se puede referir mediante la figura 2.22:

Sensor O2 (del.)	PCM Pin No.	Description
1	C01-2 (31)	Masa sensor
2	C01-1 (35)	señal
3	C01-2 (34)	calefactor
4	Relé principal	B+

Figura 2.22. Pines de conexión sensor HO2S

La programación del programa para el caso de la comprobación del cableado prueba la conexión únicamente a GND, ya que el voltaje es generado por el propio sensor con el paso de los gases de escape por el transductor del propio sensor.

Una vez comprobado el cableado, ingresa al bucle donde se realiza la simulación; al igual que en varios de los sensores, ya indicados, para la visualización de los valores ingresados a la ECU (en voltios) se obtiene una relación mediante el ajuste de curvas tomando puntos de ingreso en el programa y valores obtenidos y leídos por la ECU, es así que se comprueba que el sensor es lineal, teniendo como ecuación:

$$LHO2S = (-0.019116*STPS+2.3349);$$

Donde STPS es el valor generado y cambia la salida analógica de la señal hacia la ECU y LH02S es el valor que será presentado en el LCD.

Para obtener la lectura que está percibiendo la ECU del automóvil se envía el código AT **0114\r** por el ELM327, para realizar la comparación entre el ingreso y la lectura a la ECU.

De modo que el programa que simula es sensor de oxígeno es:

```
//////////////////////PROGRAMA PARA SIMULAR SENSOR HO2S/////////////////
if (cartel==4)
{
 delay(200);
 estadoGTPS = digitalRead(GTPS);
 if (estadoGTPS == HIGH)
 {
  lcd.setCursor(0, 3);lcd.print("GND -> Error ---X---");
  delay(500);
  lcd.setCursor(0, 3);lcd.print("                    ");
 }
 else
 {
  delay(700);
  lcd.setCursor(0, 3);lcd.print("GND -> Ok          ");
  delay(800);
 }

  if (estadoGTPS==LOW)
  {
   cartel=12;CartelLCD();
   lcd.setCursor(0, 2);lcd.print("Simulado:");
   lcd.setCursor(0, 3);lcd.print("ECU:");

   TSensor=6;STPS=95;LHO2S= 0.51;
   Serial1.write("ATSP6\r"); ////ISO 15765-4 CAN 11/500

   while(cartel==12)
   {
      adc_key_in = analogRead(0);
      key = get_key(adc_key_in);
      if(key==1)
      {
       if ( STPS == 62)
       {
       }
       else
       {
        delay(20);

        STPS=STPS-1;
        LHO2S=(-0.019116*STPS+2.3349);
        lcd.setCursor(10, 2);lcd.print("     v");
       }
      }
      if(key==2)
      {
       if (STPS==114)
       {
       }
       else
       {
        delay(30);
        STPS=STPS+1;
        LHO2S=(-0.019116*STPS+2.3349);
        lcd.setCursor(10, 2);lcd.print("     v");
       }
      }
      if (cartel==11)
      {
       digitalPotWrite();
```

```
                }
                else
                {
                 digitalPotWrite();
                 lcd.setCursor(10, 2);lcd.print(LHO2S);
                }

                //////////////////////HO2S OBDII//////////////////
                delay(250);
                Serial1.write("0114\r");///PID peticion HO2S
                SerialOBDII();
                ////////////////////////////////////////////////

                if ( key == 0)
                {
                 cartel=11;CartelLCD();
                 cont=0;ini=0;TSensor=0;Dato=0;Dato1=0;Dato2=0;PosDatoSerial=0;LTPS=0;
                 delay(200);
                }
              }
            }
          }

        ////////////////////FIN DE PROGRAMA SIMULAR SENSOR HO2S////////////////
```

Figura 2.23. Diagrama de Flujo Programación H02S

Luego de simular todos los sensores con el afán de demostrar que las entradas de la unidad electrónica de control y el cableado respectivo de cada uno de los sensores simulados se encuentran en correcto estado, procedemos a comprobar cada uno de los sensores, interpretando cada una de las señales y representándola en el LCD.

2.4.2. Diagrama de flujo para probar un sensor

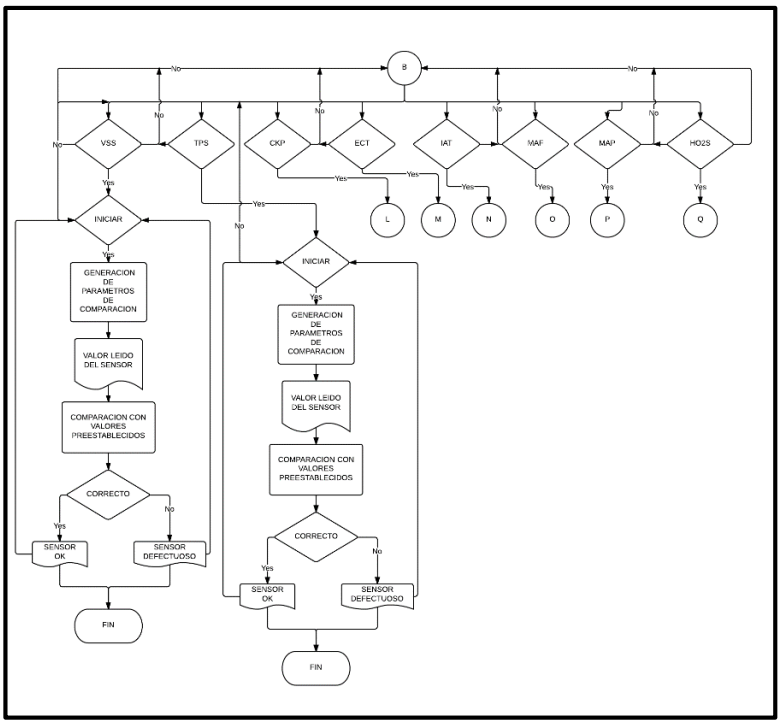


Figura 2.24. Diagrama de flujo probar un sensor

4.2.2.1. Diagrama de flujo para probar sensor VSS o CMP

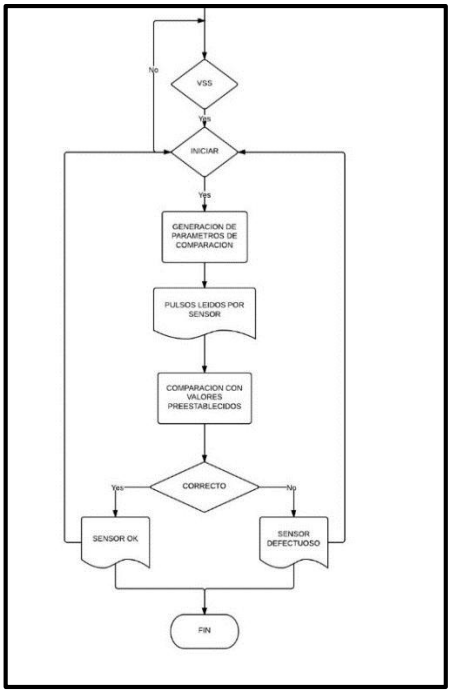


Figura 2.25. Diagrama de flujo probar sensor VSS o CMP

Para comprobar el sensor VSS y cualquier sensor a comprobar se deberá seguir el mismo procedimiento básicamente con ciertas diferencias.

Para comprobar el sensor VSS, el equipo genera el parámetro de comparación haciendo sensor defectuoso en el caso que no se detecten pulsos o sensor ok en el caso que se fuesen detectados pulsos en el equipo.

La programación específica del sensor se presenta en la figura 2.26:

```
//////////////////////////PROGRAMA PARA PROBAR VSS//////////////////////


lcd.setCursor(15, 2);lcd.print(val);
diag = digitalRead(PSENS);
if (diag == LOW)
{
  val=val+1;
  while (diag == LOW)
  {
    diag = digitalRead(PSENS);
    adc_key_in = analogRead(0);
    key = get_key(adc_key_in);
    if ( key == 0)
    {
      if (val==1)
        {
          lcd.setCursor(0, 2);lcd.print("SENSOR DEFECTUOSO");
          delay(1500);
        }
       else
       {
         lcd.setCursor(0, 2);lcd.print("SENSOR OK        ");
          delay(1500);
       }
      cartel=20;
      diag=HIGH;
      val=0;
      CartelLCD();
      delay(200);

    }
  }
}
adc_key_in = analogRead(0);
key = get_key(adc_key_in);
if ( key == 0)
{
  if (val==0)
    {
      lcd.setCursor(0, 2);lcd.print("SENSOR DEFECTUOSO");
      delay(1000);
    }
  else
    {
```

```
    lcd.setCursor(0, 2);lcd.print("SENSOR OK       ");
    delay(1500);
   }
  digitalWrite(28, LOW);                  //Desactiva rele 3 12V
  cartel=20;CartelLCD();
  val=0;
  delay(200);
 }
////////////////////////////FIN DE PROGRAMA PARA PROBAR VSS//////////////////////////
```

Figura 2.26. Programación para probar sensor VSS o CMP

La comprobación del sensor CMP, se consigue con la misma lógica, ya que los dos sensores generan pulsos enviados a la ECU para determinar parámetros característicos de cada uno de los sensores.

2.4.2.2. Diagrama de flujo para probar sensor TPS

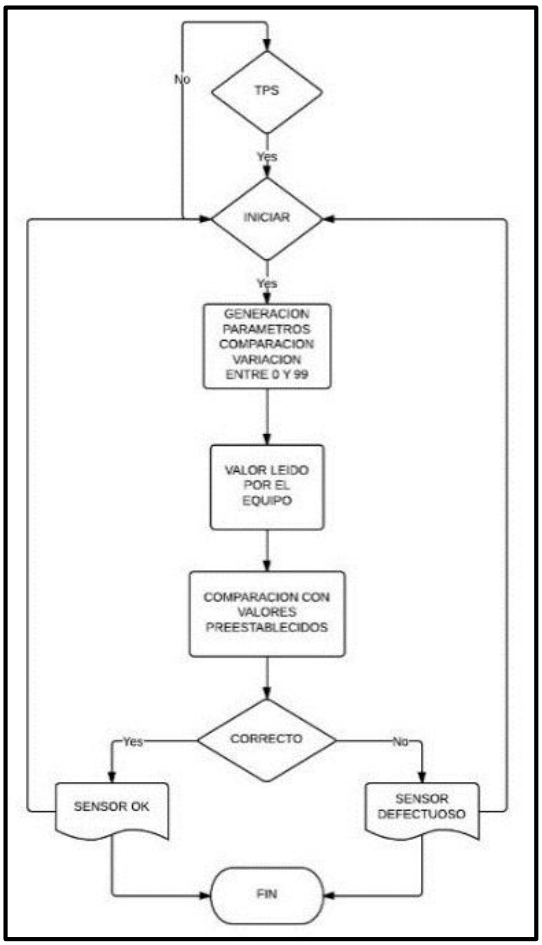


Figura 2.27. Diagrama de flujo probar sensor TPS

Para el caso en el que se diagnostica el sensor TPS, los valores a comparar estarán entre 0 y 99 %, de no ser así el mensaje será sensor defectuoso.

La programación para probar el sensor se encuentra en la figura 2.28:

```
////////////////////////////////////////////////////////////////////////////
//////////////////////PROGRAMA PARA PROBAR TPS/////////////////////////////////
 lcd.setCursor(13, 2);lcd.print("  % ");
 lcd.setCursor(13, 2);lcd.print(val);
 val = analogRead(sensorPin);
 val=((val*100)/1024);
 delay(200);

 adc_key_in = analogRead(0);
 key = get_key(adc_key_in);
 if ( key == 0)
 {
  if (val>90)
  {
   lcd.setCursor(0, 2);lcd.print("SENSOR DEFECTUOSO");
   val=0;
   delay(2000);
  }
  if (val<3)
  {
   lcd.setCursor(0, 2);lcd.print("SENSOR DEFECTUOSO");
   delay(2000);
  }
  if (val>3)
  {
   lcd.setCursor(0, 2);lcd.print("SENSOR OK         ");
   delay(2000);
  }
  digitalWrite(30, LOW);                    //Desactiva rele #5 5V
  cartel=21;
  CartelLCD();
  val=0;
  delay(200);
 }

 /////////////////FIN DE PROGRAMA PARA PROBAR TPS/////////////////////////////////
 ////////////////////////////////////////////////////////////////////////////
```

Figura 2.28. Programación para probar sensor VSS

2.4.2.3. Diagrama de flujo probar sensor ECT e IAT

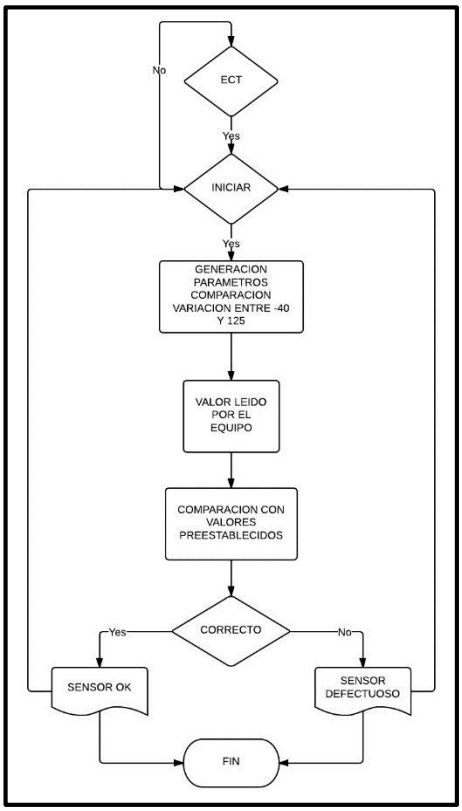


Figura 2.29. Diagrama de flujo probar sensor ECT e IAT

El diagnóstico de los sensores ECT e IAT es el mismo ya que los dos funcionan bajo los mismos parámetros, diferenciándose en el rango de temperatura de funcionamiento normal para el sensor. Se diagnosticará como sensor ok si se encuentra dentro de estos parámetros.

La programación se muestra en la figura 2.30:

```
////////////////////////////////////////////////////////////////////////////////
//////////////////////PROGRAMA PARA PROBAR ECT/////////////////////////////////

lcd.setCursor(0, 2);lcd.print("Tem. Actual:");
lcd.setCursor(13, 2);lcd.print("    C");
val = analogRead(sensorPin);
val = (((-0.0000211)* val* val*val)+(0.00899*val*val) - (1.4384 * val) +116.1);
lcd.setCursor(13, 2);lcd.print(val);
delay(200);
```

```
adc_key_in = analogRead(0);
key = get_key(adc_key_in);
if ( key == 0)
{
 if (val>100)
 {
  lcd.setCursor(0, 2);lcd.print("SENSOR DEFECTUOSO   ");
  delay(2000);
 }
 if (val<3)
 {
  lcd.setCursor(0, 2);lcd.print("SENSOR DEFECTUOSO   ");
  delay(2000);
 }
 if (val>3)
 {
  if( val<100)
  {
   lcd.setCursor(0, 2);lcd.print("SENSOR OK           ");
   delay(2000);
  }
 }
 digitalWrite(30, LOW);                 //Desactiva rele #5 5V
 digitalWrite(31, LOW);                 //Desactiva rele #6
 cartel=23;CartelLCD();
 val=0;
 delay(200);
}
//////////////////////FIN PROGRAMA PARA PROBAR ECT/////////////////////////////////
//////////////////////////////////////////////////////////////////////////////
```

Figura 2.30. Programación probar sensor ECT e IAT

2.4.2.4. Diagrama de flujo para probar sensor MAF

Como se indicó en un inciso anterior, este espacio queda libre para la mejora del equipo que se está diseñando.

2.4.2.5. Diagrama de flujo probar sensor MAP

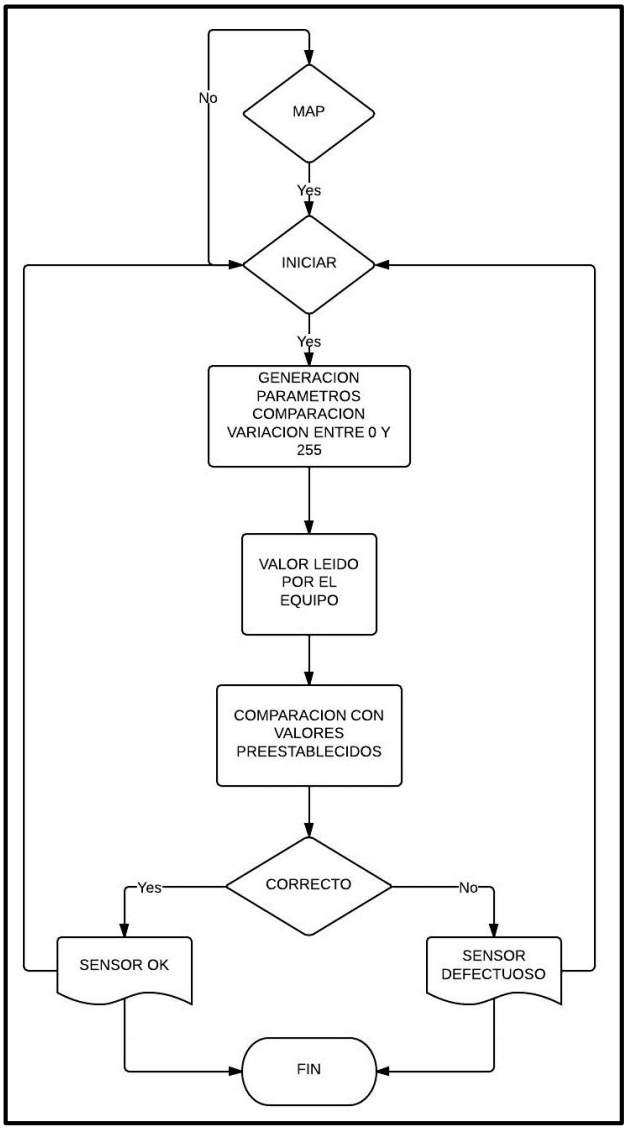


Figura 2.31. Diagrama de flujo probar sensor MAP

El sensor MAP se diagnostica en base a sus parámetros de funcionamiento, pudiendo tener valores entre 0 y 255, para generar mensaje de sensor ok, caso contrario sensor defectuoso.

La programación se presenta en la figura 2.32:

```
////////////////////////////////////////////////////////////////////////
                    ////////////////////////PROGRAMA PARA PROBAR MAP/////////////////////////////////
                     lcd.setCursor(13, 2);lcd.print("   kPa ");
                     lcd.setCursor(13, 2);lcd.print(val);
                     val = analogRead(sensorPin);
                     val=((val*127)/1024);
                     delay(200);

                     adc_key_in = analogRead(0);
                     key = get_key(adc_key_in);
                     if ( key == 0)
                     {
                      if (val>75)
                      {
                        lcd.setCursor(0, 2);lcd.print("SENSOR DEFECTUOSO ");
                        val=0;
                        delay(2000);
                      }
                      if (val<20)
                      {
                        lcd.setCursor(0, 2);lcd.print("SENSOR DEFECTUOSO ");
                        delay(2000);
                      }
                      if (val>20)
                      {
                        lcd.setCursor(0, 2);lcd.print("SENSOR OK         ");
                        delay(2000);
                      }
                      digitalWrite(30, LOW);                    //Desactiva rele #5 5V
                      cartel=26;CartelLCD();
                      val=0;
                      delay(200);
                     }
                     ////////////////////FIN PROGRAMA PROBAR MAP//////////////////////////
                     ////////////////////////////////////////////////////////////////////
```

Figura 2.32. Programación para probar sensor MAP

2.4.2.6. Diagrama de flujo probar sensor HO2S

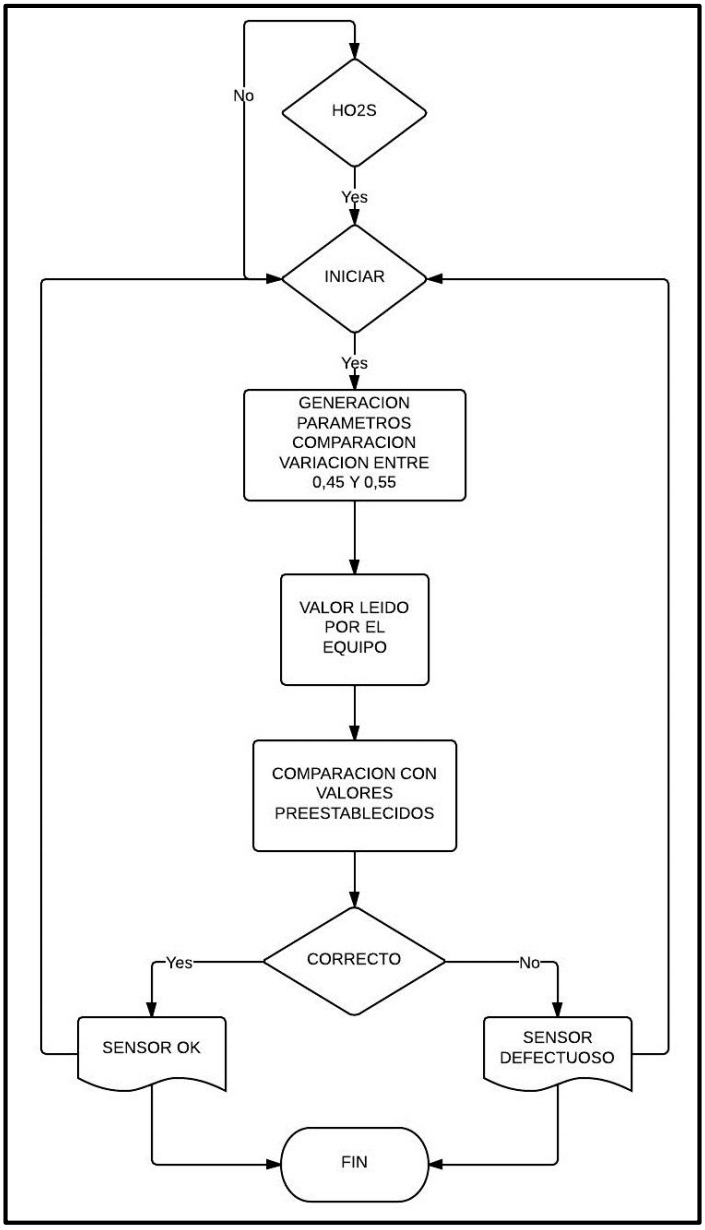


Figura 2.33. Diagrama de flujo probar sensor HO2S

La operación del sensor de oxígeno debe oscilar entre los valores de 0,45 y 0,55V para que se determine sensor ok, caso contrario determinará sensor defectuoso.

La programación para determinar el estado del sensor de oxígeno es la siguiente:

```
////////////////////////////////////////////////////////////////////////////
//////////////////////PROGRAMA PARA PROBAR HO2S////////////////////////////////
 lcd.setCursor(0, 3);lcd.print("Lectura:");
 //lcd.setCursor(13, 2);lcd.print(val);
 //val = analogRead(sensorPin);
 //delay(200);
 //////////////////////HO2S OBDII//////////////////
 delay(250);
 Serial1.write("0114\r");///PID peticion HO2S
 SerialOBDII();
  ////////////////////////////////////////////////
 adc_key_in = analogRead(0);
 key = get_key(adc_key_in);
 if ( key == 0)
 {
  lcd.setCursor(0, 1);lcd.print("Sensor OK, si:");
  lcd.setCursor(0, 2);lcd.print("Lectura entre:");
  lcd.setCursor(0, 3);lcd.print("0.450v - 0.550v");
  delay(8000);
  cartel=27;CartelLCD();TSensor=0;
  val=0;
  delay(200);
 }
```

Figura 2.34. Programación probar sensor HO2S

2.4.3. Diagrama de flujo para probar actuador

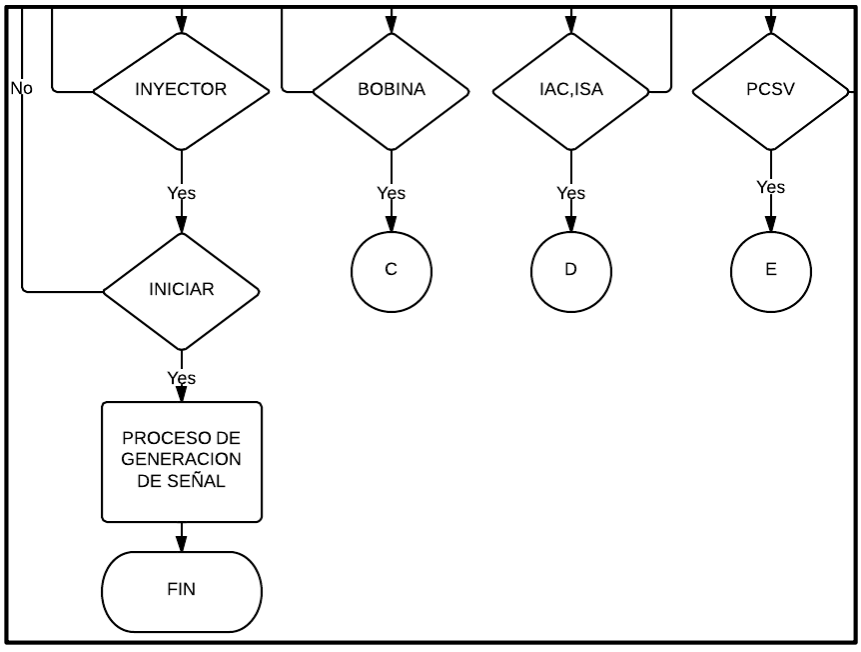


Figura 2.35. Diagrama de flujo probar actuador

El diagrama de flujo para probar los actuadores es muy sencillo, ya que se selecciona el actuador a comprobar, a continuación iniciar y se realiza un proceso de generación de señal dependiente del tipo de actuador en diagnosis. En las siguientes figuras se presenta la programación para la comprobación de cada uno de los actuadores indicados.

```
if (numCols==0)
{
  cartel=18;CartelLCD();
  digitalWrite(28, HIGH);                    //Activa rele 3 12V
  while (cartel==18)//////////////////////////Cartel 18: INYECTOR  CTR  B+
  {              ////////////////////////      INICIO    ATRAS
    adc_key_in = analogRead(0);
    key = get_key(adc_key_in);
    if ( key == 0)
    {
      cartel=15;CartelLCD();cursorLCD();
      digitalWrite(28, LOW);                  //Desactiva rele 3 12V
      delay(200);
     }
     if ( key == 4)
    {
      while (key == 4)
      {
       digitalWrite(27, HIGH);
       delay(20);
       digitalWrite(27, LOW);
       delay(300);
       adc_key_in = analogRead(0);
       key = get_key(adc_key_in);
```

Figura 2.36. Programación para probar inyector

```
 if (numCols==1)
{
  cartel=19;CartelLCD();
  digitalWrite(28, HIGH);                    //Activa rele 3 12V
  while (cartel==19)//////////////////////////Cartel 18: BOBINA  CTR  B+
  {              ////////////////////////      INICIO    ATRAS
    adc_key_in = analogRead(0);
    key = get_key(adc_key_in);
    if ( key == 0)
    {
      cartel=15;CartelLCD();cursorLCD();
      digitalWrite(28, LOW);                  //Desactiva rele 3 12V
      delay(200);
     }
     if ( key == 4)
    {
      while (key == 4)
      {

       digitalWrite(27, HIGH);
       delay(10);
```

```
                digitalWrite(27, LOW);
                delay(300);

                adc_key_in = analogRead(0);
                key = get_key(adc_key_in);
```

Figura 2.37. Programación para probar bobina

```
            if (numCols==2)
            {
             cartel=28;CartelLCD();
             while (cartel==28)//////////////////////////Cartel 18: IAC  CO  B+  CC
             {              ////////////////////////              INICIO    ATRAS
               adc_key_in = analogRead(0);
               key = get_key(adc_key_in);
               if ( key == 0)
               {
                 cartel=15;CartelLCD();cursorLCD();
                 delay(200);
               }
               if ( key == 4)
               {
                 cartel=33;
                 CartelLCD();
                 digitalWrite(28, HIGH);                  //Activa rele 3 12V
                 digitalWrite(29, HIGH);                  //Activa rele 4 Control IAC
                 val1=1;
                 ////////////////////////////////////////////////////////////////////
                 ////////////////CAMBIA LA FRECUECIA PWM//////////////////////////////
                 int myEraser = 7;              // this is 111 in binary and is used as an eraser
                 TCCR3B &= ~myEraser;           // this operation (AND plus NOT),  set the three bits in TCCR2B to 0
                 int myPrescaler = 4;           // this could be a number in [1 , 6]. In this case, 3 corresponds in binary to 011.
                 TCCR3B |= myPrescaler;         //this operation (OR), replaces the last three bits in TCCR2B with our new
value 011
                 ////////////////////////////////////////////////////////////////////
                 ////////////////////////////////////////////////////////////////////

                 while (cartel==33)//////////////////////////Cartel 18: IAC  CO  B+  CC
                 {              ////////////////////////              INICIO    ATRAS
                   adc_key_in = analogRead(0);
                   key = get_key(adc_key_in);

                   if (key == 1)
                   {
                     if (val1==1)
                     {
                     }
                     else
                     {
                       val1 = val1 - 1;
                       val2 = ((val1*100)/255);
                       lcd.setCursor(1, 2);lcd.write("   %");
                       lcd.setCursor(1, 2);
                       lcd.print(val2);
                       delay(50);
                     }
                   }
                   if (key == 2)
                   {
                     if (val1==255)
                     {
                     }
                     else
                     {
                       val1 = val1 + 1;
```

```
                    val2 = ((val1*100)/255);
                    lcd.setCursor(1, 2);lcd.write("   %");
                    lcd.setCursor(1, 2);
                    lcd.print(val2);
                    delay(50);
                   }
                  }

                  analogWrite(PinPWM2, val1);
                  if ( key == 0)
                  {
                    lcd.clear();
                    cartel=28;CartelLCD();
                    val1=1;
                    val2=1;
                    analogWrite(PinPWM2, val1);
                    digitalWrite(28, LOW);              //Desactiva rele 3 12V
                    digitalWrite(29, LOW);              //Desactiva rele 4 Control IAC
                    delay(200);
```

Figura 2.38. Programación para probar IAC o ISCA

```
if (numCols==3)
{
 cartel=29;CartelLCD();
 digitalWrite(28, HIGH);                  //Activa rele 3 12V
 while (cartel==29)//////////////////////////Cartel 18: PCSV  B+  CTR
 {              ////////////////////////      INICIO    ATRAS
   adc_key_in = analogRead(0);
   key = get_key(adc_key_in);
   if ( key == 0)
   {
     cartel=15;CartelLCD();cursorLCD();
     digitalWrite(28, LOW);                 //Desactiva rele 3 12V
     delay(200);
   }
    if ( key == 4)
   {
     while (key == 4)
     {
      digitalWrite(27, HIGH);
      delay(150);
      digitalWrite(27, LOW);
      delay(300);
      adc_key_in = analogRead(0);
      key = get_key(adc_key_in);
     }
```

Figura 2.39. Programación para probar PCSV

2.4.4. Diagrama de flujo para la lectura de códigos de falla

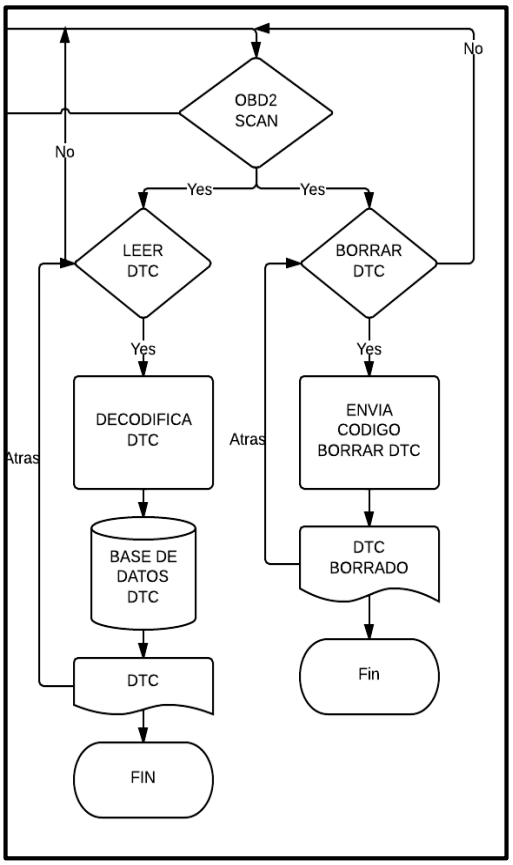


Figura 2.40. Diagrama de flujo par lectura DTC

Para la lectura de los códigos de error generado y almacenado en la unidad electrónica de control, se utilizarán las instrucciones AT de comunicación del ELM327de manera que se puedan detectar la cantidad de errores y poder presentarlos mediante el LCD.

Además se pueden borrar los códigos, siempre y cuando el origen del error haya sido corregido.

Así los códigos AT necesarios para la lectura es el código **03** y para borrar los códigos es **04**, el resto de la programación que se presenta en la siguiente figura es utilizada para mostrar e indicar el código de falla, además de generar una pequeña base de datos para la lectura de dichos códigos.

```
////////////////////////////////OBDII SCAN/////////////////////////////////////////////////////
if (numCols==2)
{
 cartel=17;CartelLCD();
 while (cartel==17)
 {
    adc_key_in = analogRead(0);            // Leemos el valor de la pulsación
    key = get_key(adc_key_in);             // Obtenemos el boton pulsado
    if ( key == 0)
    {
     cartel=0;CartelLCD();cursorLCD();
     delay(200);
    }
     if ( key == 4)
    {
     cartel=30;CartelLCD();encero();selecCursor2();cursorLCD();
     delay(200);
     //Serial1.write("0101\r"); ////ISO 15765-4 CAN 11/500
     while (cartel==30)
     {
       adc_key_in = analogRead(0);          // Leemos el valor de la pulsacion
       key = get_key(adc_key_in);           // Obtenemos el boton pulsado
       if ( key == 0)
       {
        cartel=17;CartelLCD();
        delay(200);
       }
       selecCursor2();
       if ( key == 4)
       {
        delay(200);
        if (numCols==1)
        {
           cartel=31;CartelLCD();
           while (cartel==31)
           {
            delay(300);
            Serial1.write("03\r"); ////ISO 15765-4 CAN 11/500
            diag=4;
            SerialOBDII2();

            adc_key_in = analogRead(0);    // Leemos el valor de la pulsacion
            key = get_key(adc_key_in);     // Obtenemos el boton pulsado
            if ( key == 0)
            {
             cartel=30;CartelLCD();cursorLCD();
             delay(200);
        if (numCols==2)
        {
           cartel=32;CartelLCD();
           while (cartel==32)
           {
            delay(250);
            Serial1.write("04\r"); ////ISO 15765-4 CAN 11/500
            SerialOBDII2();
            cartel=30;CartelLCD();cursorLCD();
            delay(200);
           }
```

Figura 2.41. Programación para lectura de DTC

CAPITULO 3

DISEÑO Y CONSTRUCCION DE LAS INTERFACES APLICADAS A LOS SENSORES, ACTUADORES Y UNIDAD ELECTRONICA DE CONTROL

3.1. Interfaces aplicadas a los sensores

Luego de diseñar cada una de las subrutinas para simular y comprobar el funcionamiento de los sensores, se procede con el diseño de potencia para adecuar la señal de simulación o generación de la señal para comprobar un determinado sensor.

La característica principal de la interface diseñada es que para analizar los sensores se utiliza tres cables, para probar los sensores tres cables más y para probar los actuadores tres cables más.

Por lo tanto la tarjeta de acoplamiento que se diseña está dividida en tres secciones:

a. Simular sensor

b. Probar sensor

c. Probar actuador

3.1.1 Simular sensor

La sección simular sensor es la encargada de acoplar el equipo a las entradas de la unidad electrónica de control, para que sean utilizados únicamente tres cables para la verificación de cada una de las entradas.

Este acoplamiento se da en base a conmutación mediante relés conectados al equipo, que se activarán dependiendo del sensor que se esté deseando simular.

En la figura se puede observar los acoplamientos de los relés, misma que será construida en la placa.

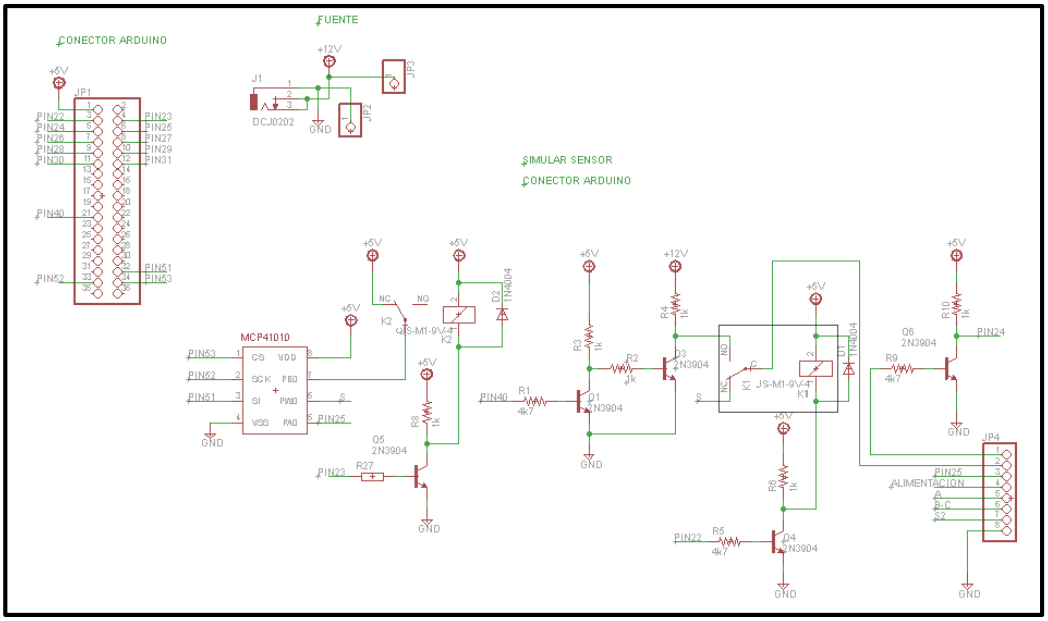


Figura 3.1. Esquemático de circuito para simular sensor

3.1.2 Probar sensor

Cada uno de los sensores que vamos a poder diagnosticar con el equipo que se está diseñando, serán conectados a esta parte de la placa de acoplamiento, que conmuta mediante relés la lectura de cada una de las señales (una por una) de los sensores a probar.

En la figura se observa el esquemático del circuito para probar sensor que será construido en la tarjeta.

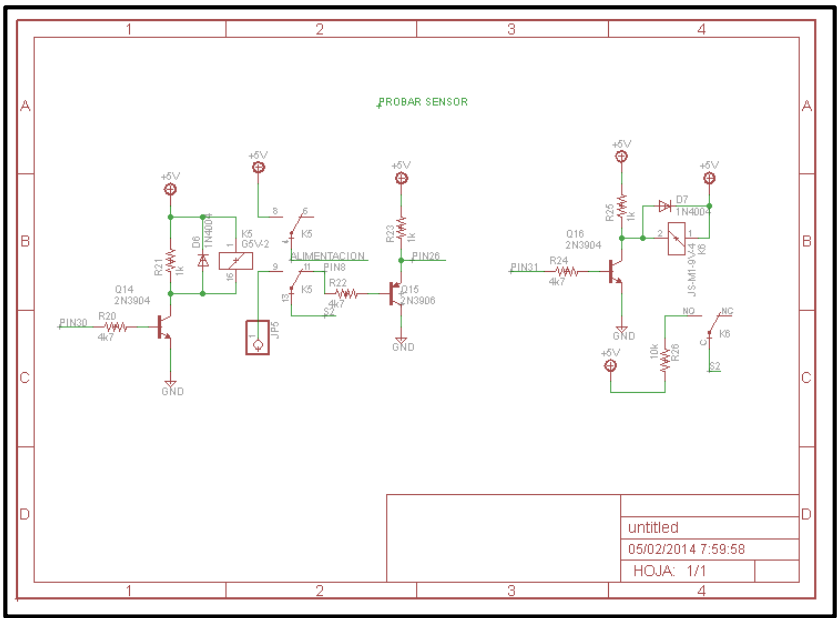


Figura 3.2. Esquemático de circuito para probar sensor

3.1.3 Probar actuador

Al igual que en los puntos anteriores está compuesto por una serie de relés principalmente que hacen conmutar las salidas de la tarjeta para utilizar únicamente tres cables y poder comprobar los 4 actuadores planteados.

En la figura se observa el circuito a construir sobre la tarjeta de acoplamiento de señales.

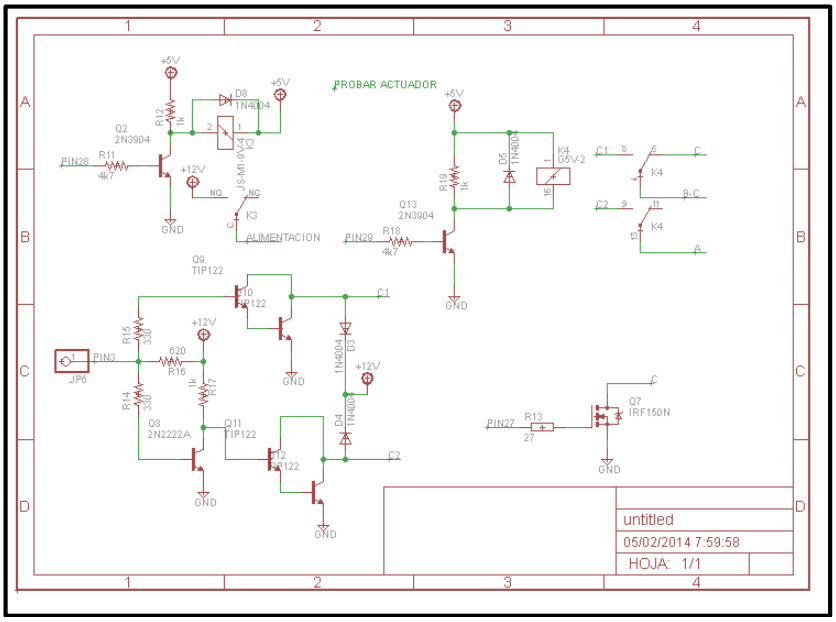


Figura 3.3. Esquemático de circuito para probar actuador

3.1.4 Ruteado de la placa:

Con el afán de que el costo del equipo sea el menor, se ha diseñado una placa de acoplamiento de señal de una sola cara, donde irán montados todos los elementos encargados el acoplamiento y conmutación de los circuitos para probar cada uno de los sensores, actuadores o la unidad electrónica de control.

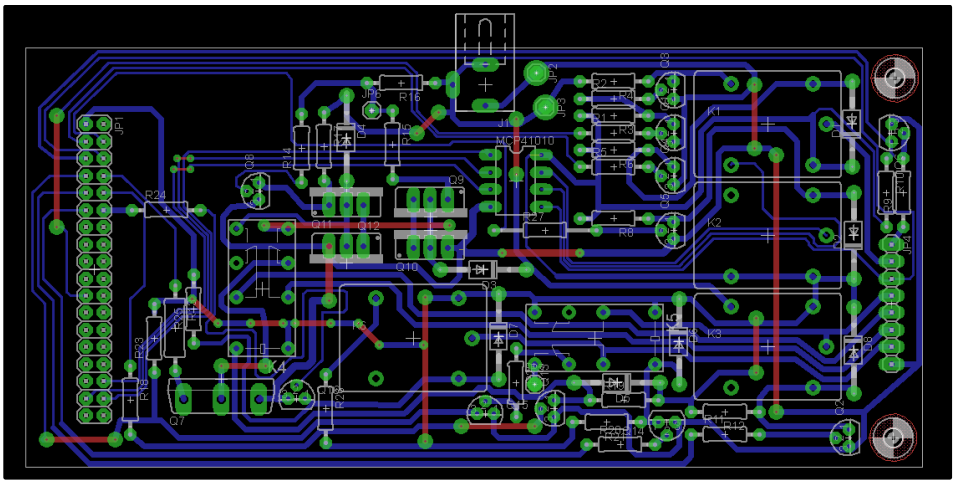


Figura 3.4. PCB placa de acoplamiento de señales

3.1.5 Construcción de la placa

Una vez diseñada, se procede con la construcción de la placa.

Dicha placa se construye sobre una placa de cobre y se obtiene el siguiente resultado:

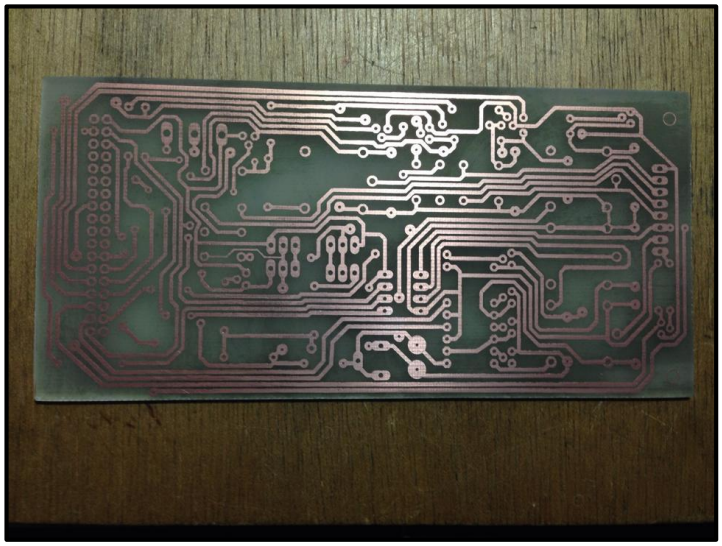


Figura 3.5. Placa de acoplamiento de señales construida (reverso)

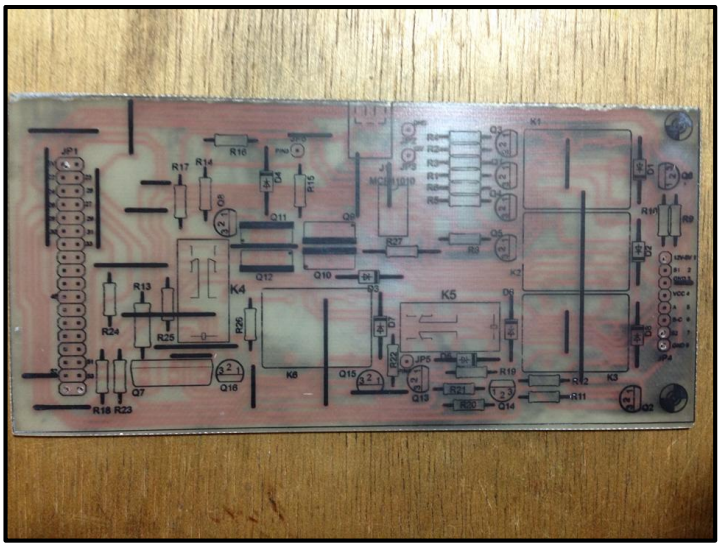


Figura 3.6. Placa de acoplamiento de señales construida (anverso)

Construida la placa, procedemos al armado de la placa con todos los componentes necesarios para acoplar las señales de sensores, actuadores y unidad electrónica de control.

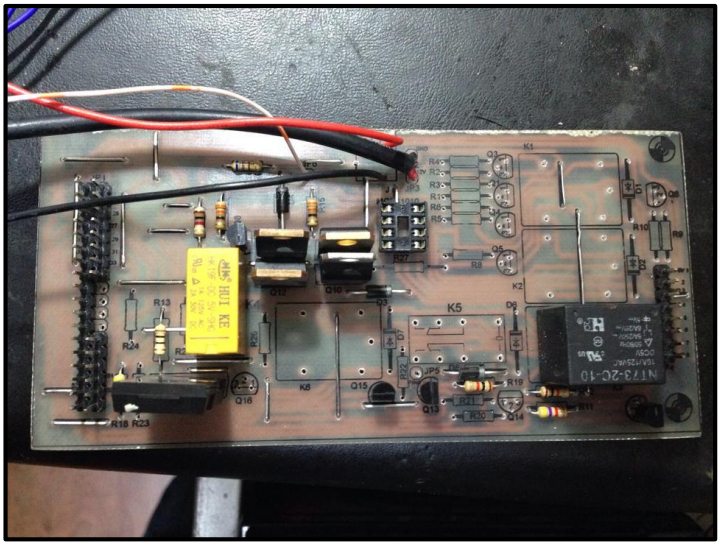


Figura 3.7. Ensamblaje de la placa de acoplamiento (simular sensor)

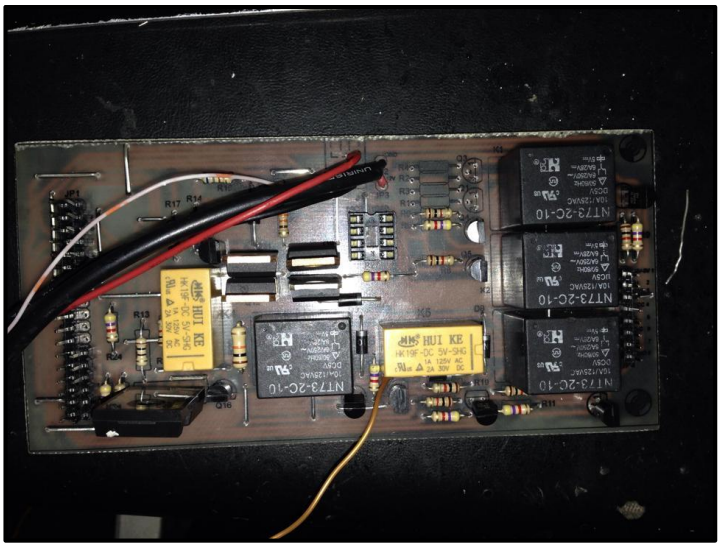


Figura 3.8. Ensamblaje de la placa de acoplamiento (probar sensor)

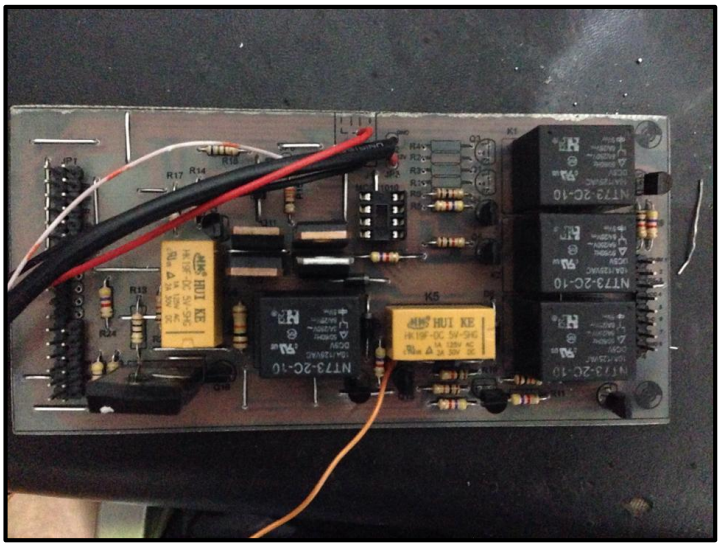


Figura 3.9. Ensamblaje de la placa de acoplamiento final (probar actuador)

CAPITULO 4

DESARROLLO DE HERRAMIENTA VIRTUAL PARA VISUALIZACION DE SEÑALES

En el desarrollo de este capítulo se indica el método de conexión de la PC con el interface ELM327 con el afán de desarrollar un *software* de visualización en tiempo real de las señales generadas por el automóvil.

Para el desarrollo del *software* de visualización se ha elegido la programación en *LabVIEW* 2009 (*Laboratory Virtual Instrumentation Engineering Workbench*), ya que posee las herramientas adecuadas para la conexión e interpretación de las señales de manera amigable con el programador, además la presentación de los resultados es amigable con el usuario al ser netamente gráfica.

A continuación se detalla el desarrollo de la herramienta virtual: conexión con la PC, solicitud, acondicionamiento, interpretación y presentación de la señal, para determinar si el elemento analizado (sensor) está funcionando correctamente.

4.1. Conexión del interface ELM327 con la PC

La conexión del interface ELM327 se realiza mediante la herramienta VISA (Conexión de puerto serial) del software *LABVIEW*. Para la utilización de esta herramienta se deben tener en cuenta los siguientes parámetros para su correcto funcionamiento:

- *Configuración del Puerto Serial VISA*

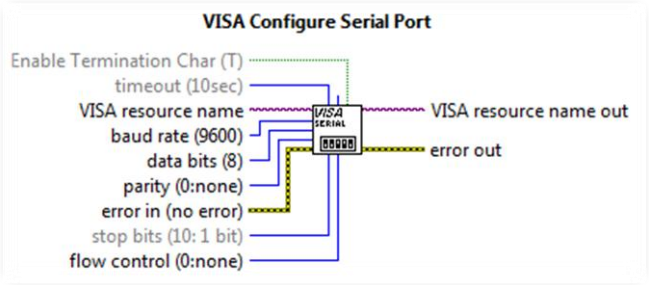


Figura 4.1. Configuración del Puerto para comunicación Serial

Cada uno de los parámetros de entrada de la configuración del puerto serial debe tener los siguientes datos

a. *VISA* resource name: Selecciona el puerto de comunicación serial COM por el cual se dará la transmisión y recepción de datos entre la PC y ELM327.

b. *Baud rate* (9600): Selecciona la velocidad de transmisión y recepción de datos entre la PC y ELM327, para el caso de la comunicación entre los dispositivos se utiliza la velocidad 115200 baudios, de acuerdo a las especificaciones de comunicación del ELM327.

c. *Data bits, Parity, Error in, Flow control*: para nuestro caso de comunicación se crea constantes por defecto.

- *Transmisión de Datos entre la PC y ELM327*

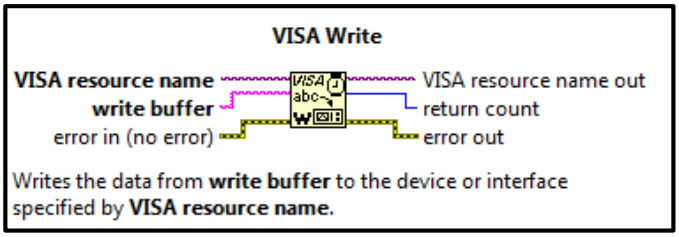


Figura 4.2. Transmisión de Datos de PC a ELM327

La misma herramienta VISA posee la característica de comunicación con la PC mediante VISA *Write*, permitiendo al usuario enviar datos hacia ELM327 desde *Write buffer* utilizando controles estáticos o dinámico dependiendo de la necesidad.

- *Recepción de Datos entre la PC y ELM327*

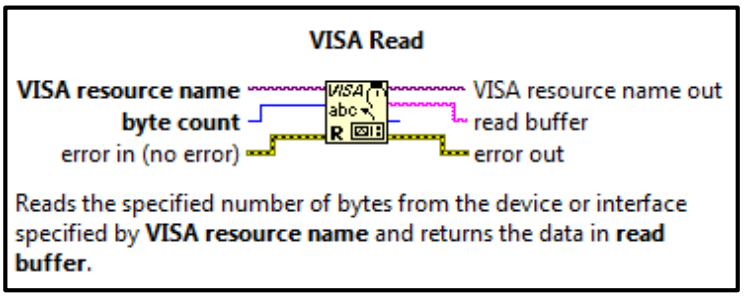


Figura 4.3. Recepción de Datos de PC desde ELM327

La utilización de la herramienta VISA permite además de escribir datos en el puerto serial, permite leer el puerto serial, con el afán de receptar los datos enviados por el ELM327 para que estos sean interpretados por el programador para la visualización de datos, en nuestro caso para visualizar las señales del automóvil.

- Cierre del Puerto Serial

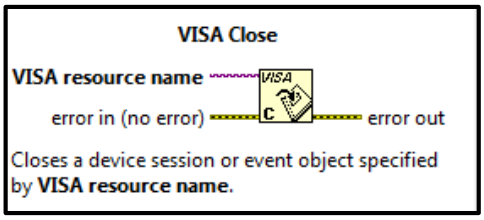


Figura 4.4. Cierre de Puerto Serial

Para completar el ciclo de trabajo del puerto serial de la PC, debe ser cerrado, para ello, la misma herramienta VISA posee la característica para hacerlo mediante VISA *Close*.

- Identificación del protocolo de comunicación entre ELM327 y el automóvil

Como ya se señaló anteriormente, existen muchos protocolos de comunicación con el automóvil. Para la identificación del protocolo mediante el ELM327, se envía la instrucción **ATSP6** mediante el puerto serial de la PC, estableciendo la comunicación con la ECU del automóvil.

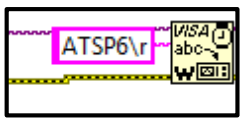


Figura 4.5. Identificación del Protocolo de Comunicación con el Automóvil

- Identificación de PID de los sensores del automóvil

Una vez establecida la comunicación entre el automóvil, ELM327 y la PC, se procede a obtener los valores de cada uno de los sensores para ser visualizados. Estos datos se obtienen enviando al ELM327 ciertos códigos por el *Write buffer* de manera

que este requiera información a la ECU del automóvil, y responda con el valor en tiempo real en el cual se encuentra un determinado sensor.

Dichos códigos PID's se encuentran en la tabla que se detalla a continuación:

SENSOR	***PID***
MAP	*010B*
ECT	*0105*
RPM	*010C*
VSS	*010D*
IAT	*010F*
TPS	*0111*
O2	*0114*
MAF	*0110*

Tabla 4.1. Instrucciones para identificación de valores de sensores

- *Interpretación de PID's de los sensores del automóvil*

Luego de enviar las instrucciones al ELM327, se obtiene una respuesta de los PID's de cada uno de los sensores del automóvil que son interpretados de la siguiente manera, según el sensor que se esté leyendo:

SENSOR	***INTERPRETACION***	***UNIDAD***
MAP	*A*	*KPa*
ECT	*A-40*	*°C*
RPM	((A*256)+B)/4	rpm
VSS	A	km/h
IAT	A	°C
TPS	A*100/255	%
O2	A/200	V
MAF	((A*256)+B) / 100	gr/s

Tabla 4.2. Interpretación de PID's de los sensores del automóvil

Cabe recalcar que los datos recibidos por la PC no son solo los valores leídos de los sensores, sino que también están involucrados datos de conformidad en la comunicación, es por ello, que en la interpretación de datos se deben elegir los valores correctos a ser leídos. Además hay que recordar que los datos están en sistema hexadecimal y tienen que ser transformados a sistema decimal.

La ubicación de los datos, dentro de la trama de comunicación, así como la conversión de sistema hexadecimal a decimal se la realiza utilizando las herramientas del software *LABVIEW*.

Únicamente lo que falta por hacer es presentar los datos mediante indicadores en software de visualización para interpretar el correcto funcionamiento del sensor analizado.

- *Identificación del código de falla*

Para la identificación del código de falla, en caso de existir, en principio se detectará, luego se visualizará y finalmente se podrá eliminar el código de falla.

Para identificar el código de falla se envía la instrucción **03** por el *Write buffer* del puerto serial, con lo que ELM327 solicita a la ECU, los códigos de falla existentes en el automóvil, en caso de existencia de códigos devuelve una codificación con un encabezado igual a 43, en caso de no existencia el encabezado cambia a 44. Según estas características el programa detecta el número de fallas existentes y se dirige a

un archivo donde busca el código perteneciente a ese número para indicar el tipo de falla que se produjo en el automóvil.

4.2. Desarrollo y funcionamiento del *Software* sobre el automóvil

- *Detalle de funcionamiento y menús de presentación de la herramienta*

Para comprobar el funcionamiento del *software* de visualización de las señales de los sensores, en principio, tenemos que conectar el dispositivo ELM327 al conector OBD2 del automóvil y a la PC.

La herramienta de visualización desarrollada en *LABVIEW 2009*, tiene la característica de poder presentar su funcionamiento en diferentes menús. A continuación se detalla el funcionamiento de cada una de ellos.

4.2.1. Menú Principal

Figura 4.6. Menú principal del Sistema de Visualización

El menú principal muestra tres botones, dos de los cuales se encuentran activos.

- El botón **ESTABLECER CONEXIÓN** garantiza que la comunicación entre la PC, ELM327 y la ECU sea la correcta.
- El botón **DIAGNOSTICAR SISTEMAS** ingresa a la siguiente pantalla para definir si se desea visualizar las señales de los diferentes sensores o ingresar a la pantalla que trata los códigos de falla del automóvil.
- El botón **SALIR** hace que el programa deje de ejecutarse y se cierre.

Una característica muy importante de este menú es que mientras no se establezca la conexión correcta entre la PC, ELM327 y la ECU, no se activa el botón para ingresar al menú **DIAGNOSTICAR SISTEMAS**

4.2.2. Menú Diagnosticar Sistemas

Si se logró establecer la conexión correcta con la ECU del automóvil y se presionó el botón de diagnóstico de sistemas en el menú principal, el *software* nos mostrará el siguiente menú:

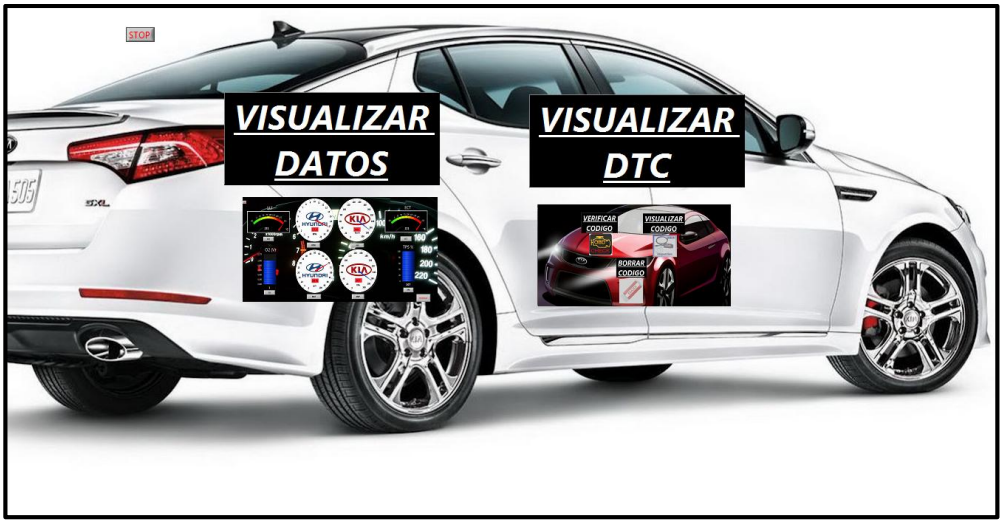


Figura 4.7. Menú Diagnosticar Sistemas

En este menú se pueden observar dos botones activos:

- Visualizar datos: si es pulsado este botón, se ingresa a un menú donde se visualizan los datos de los sensores del automóvil en tiempo real, con un desfase de 220 ms por sensor, debido a que es la velocidad de trabajo del ELM327.
- Visualizar DTC: si es pulsado este botón, se ingresa a un menú donde se determina, se visualiza y se borran los códigos de falla del automóvil (en caso de existir dichos códigos).

4.2.3. Menú Visualizar Datos

En este menú se pueden visualizar los datos en tiempo real de los siguientes sensores: MAP, IAT, VSS, O2, TPS, ECT, MAF, mediante la siguiente interface gráfica.

Figura 4.8. Menú Visualizar Datos

Como se puede ver en la **figura 4.8** debajo de cada uno de los indicadores de los valores de los sensores se tiene botones que activan los últimos menús para visualizar

los datos individualmente de cada sensor. En las siguientes figuras podemos observar cada una de las interfaces gráficas:

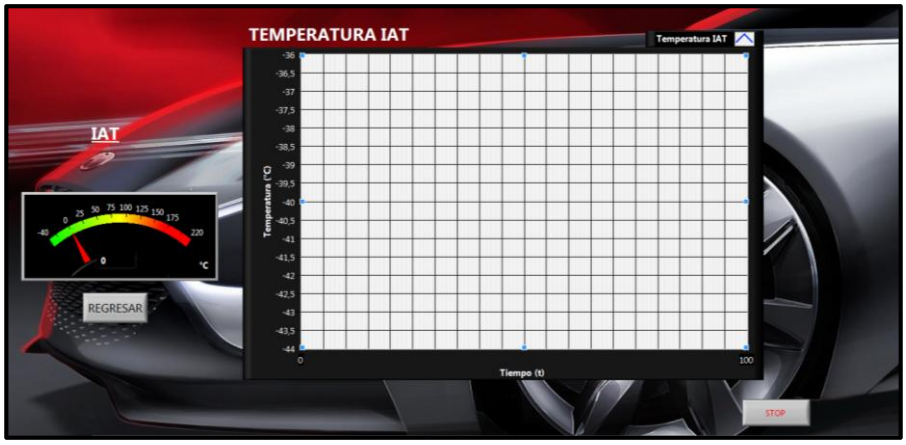


Figura 4.9. Interface IAT

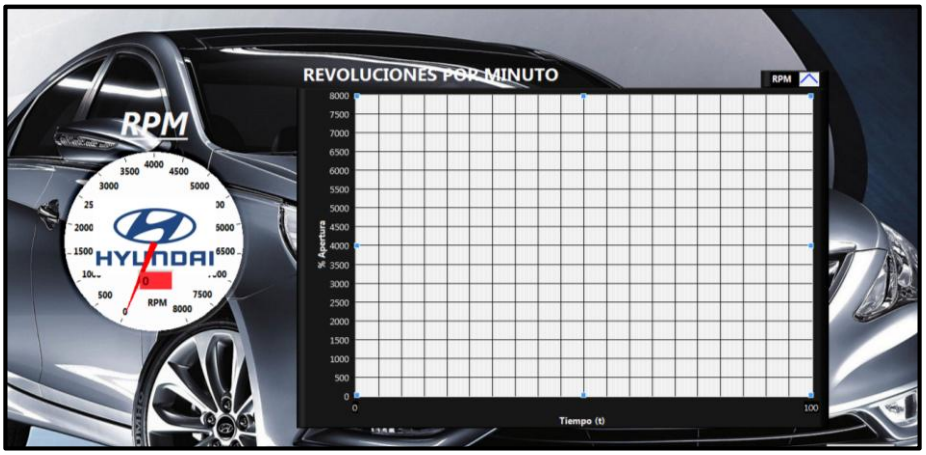


Figura 4.10. Interface RPM

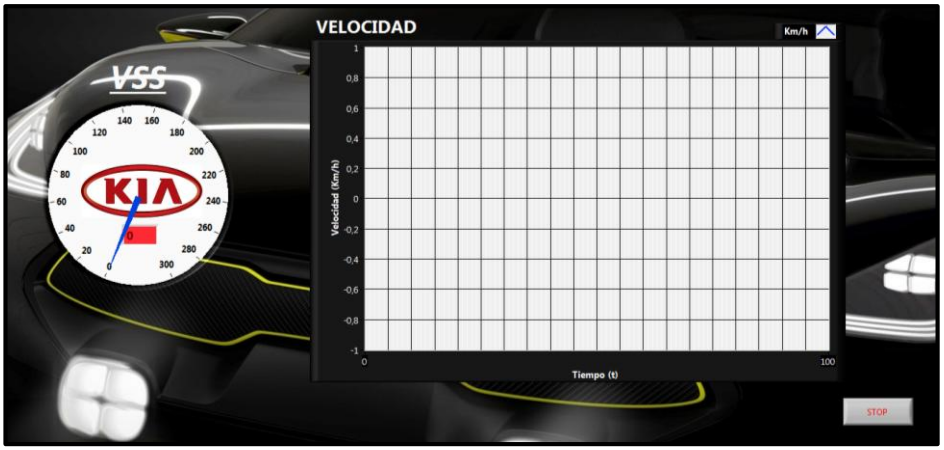


Figura 4.11. Interface VSS

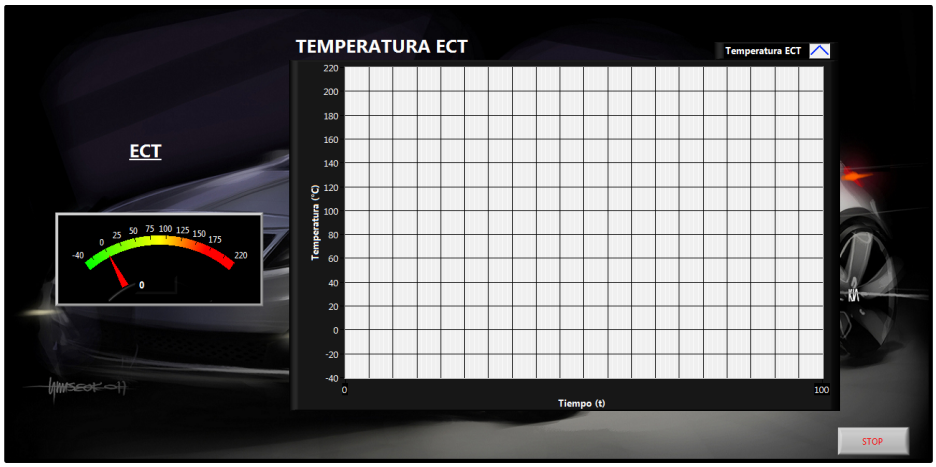


Figura 4.12. Interface ECT

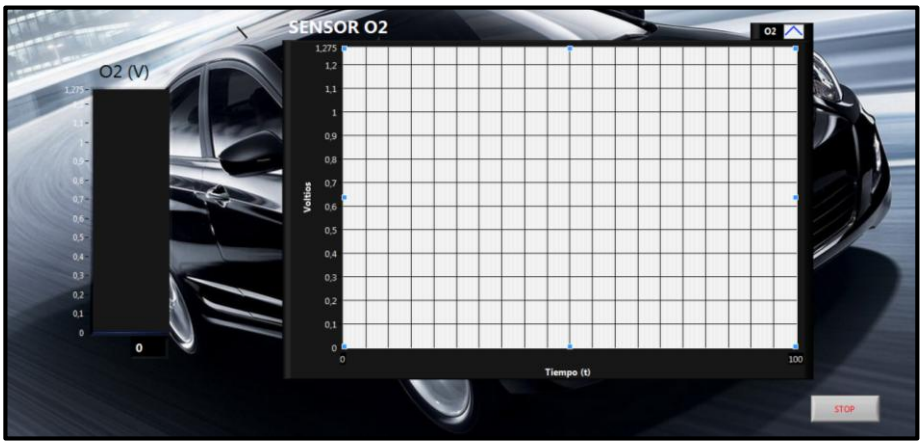


Figura 4.13. Interface O2

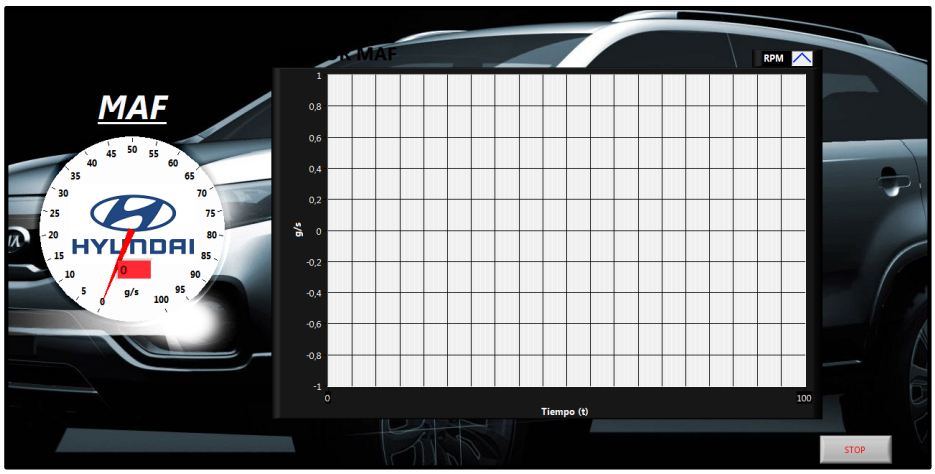


Figura 4.14. Interface MAF

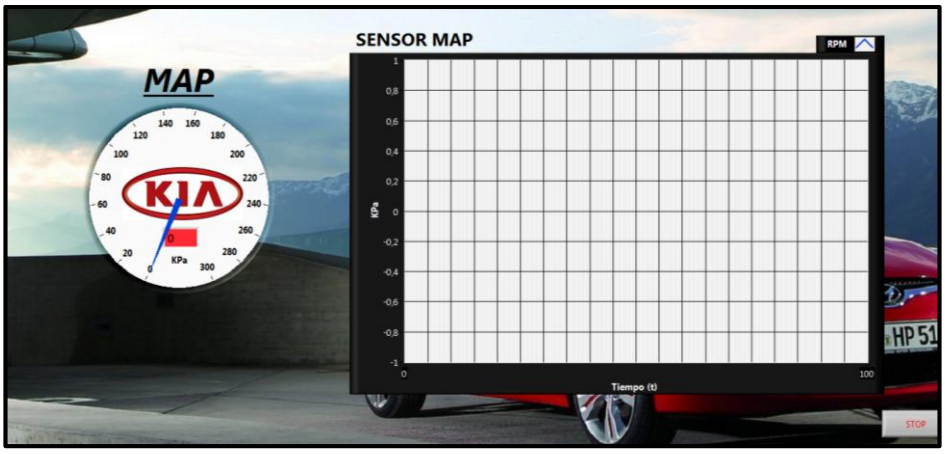


Figura 4.15. Interface MAP

Figura 4.16. Interface TPS

4.2.5. Menú Visualizar DTC

En caso de no requerirse o desearse visualizar los valores actuales de los sensores, se puede optar directamente por ingresar al menú **Visualizar DTC,** mismo que nos llevará a la siguiente interface gráfica:

Figura 4.17. Interface Visualizar DTC

Se pueden visualizar tres botones, que su accionar determinan una función del *software*, siendo:

- Verificar código: Es el botón principal de este menú, activa los siguientes botones y confirma que en el vehículo exista un código de falla registrado por la ECU.
- Visualizar código: Al presionar este botón aparecerá un mensaje indicando según la ECU, el código de falla almacenado en su memoria.
- Borrar código: solamente una vez verificado y visualizado el código de falla podrá ser borrado de la memoria de la ECU.

CAPITULO 5

ETAPA DE PRUEBAS DEL EQUIPO DE DIAGNÓSTICO DE SENSORES, ACTUADORES Y UNIDAD ELECTRONICA DE CONTROL

Antes de realizar la construcción de las interfaces para cada uno de los sensores, actuadores o la unidad electrónica de control, se fueron realizando pruebas prototipo de cada una de las interfaces montadas sobre un *protoboard*. En el siguiente capítulo se presenta en primera instancia las etapas previas sobre el *protoboard* para terminar con las pruebas sobre las interfaces construidas.

5.1. Conexiones del equipo

Vamos a empezar revisando los conectores y las conexiones que se deben realizar para utilizar el equipo de diagnóstico.

1. Bajo la columna de la dirección hacia la izquierda encontramos el conector OBD2 del automóvil, el mismo es como se muestra en la figura.

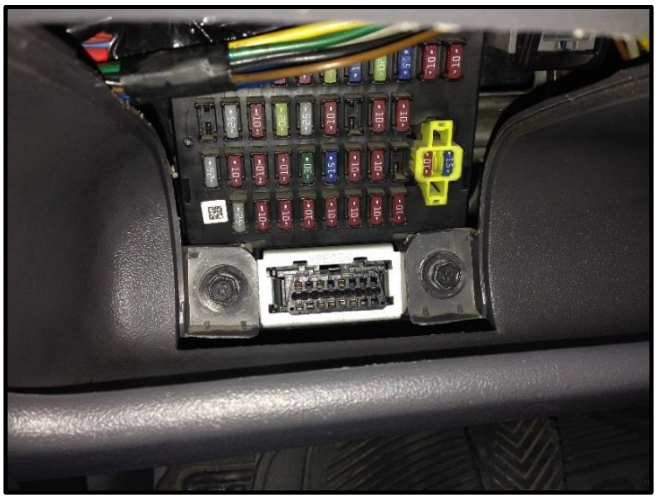


Figura 5.1. Conector OBD2 en el automóvil

2. El microcontrolador ELM327 va a ser acoplado a este conector para establecer la comunicación serial entre la ECU y la tarjeta Arduino Mega 2560. Si bien es cierto la carcasa y el cable es del dispositivo ELMSCAN 5, se utiliza la señal de salida del ELM327 como ingreso en la tarjeta Arduino. Con esto se aprovecha el acondicionamiento de señal que ya presenta el dispositivo.

Figura 5.2. Transceiver OBD2 ELMSCAN5

El ELM327 conectado se visualizaría como en la figura 5.3:

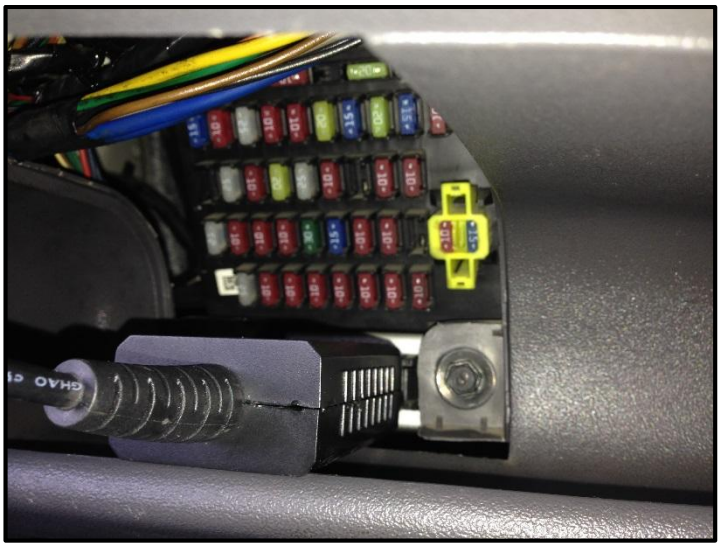


Figura 5.3. Conexión OBD2 con ELMSCAN5

El otro extremo del conector ELM327 será conectado a la tarjeta Arduino mediante un conector usb (en realidad es una transmisión serial).

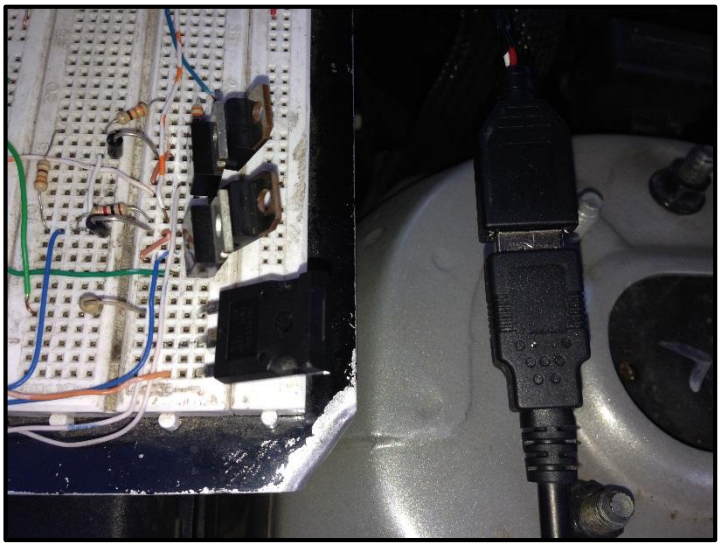


Figura 5.4. Conexión ELMSCAN5 al equipo

El equipo está listo para ser encendido y comenzar con las pruebas.

5.2. Diagnóstico de sensores

Al diagnosticar los sensores, paralelamente iremos comprobando el funcionamiento de la comunicación equipo-usuario (es decir, la visualización que tiene el usuario de los mensajes para la manipulación del equipo)

Para empezar encendemos el equipo, el mismo presenta el siguiente cuadro de diálogo:

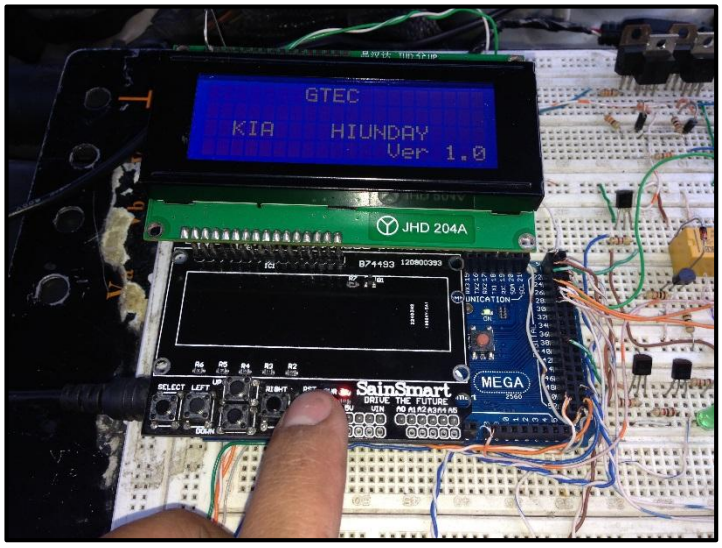


Figura 5.5. Interface de inicialización del equipo

Inmediatamente por su programación, presenta las opciones principales al usuario, en la figura se muestra su visualización:

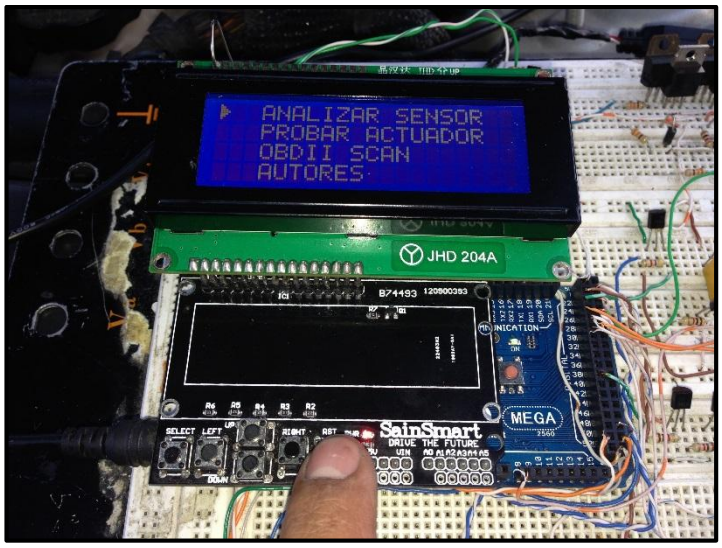


Figura 5.6. Interface de selección principal

5.2.1. Analizar sensor

Para efectos explicativos del funcionamiento del equipo, supondremos que el usuario selecciona la primera opción **ANALIZAR SENSOR,** lo que lo lleva a la siguiente pantalla:

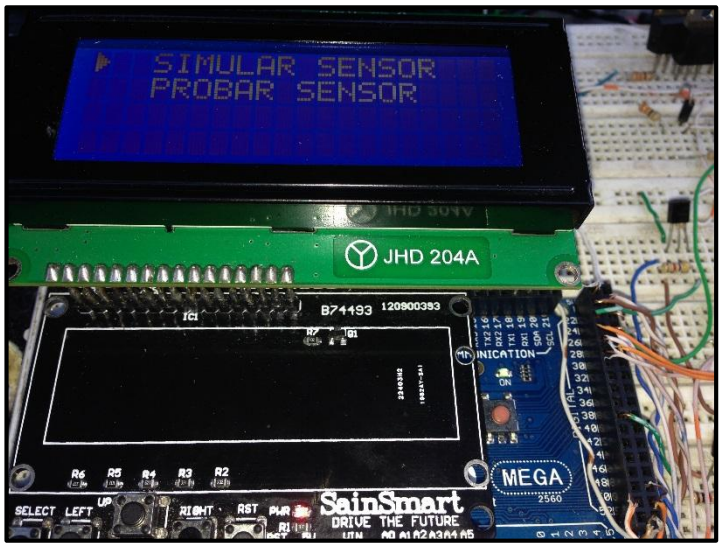


Figura 5.7. Interface para Analizar Sensor

- **Simulación de sensores**

Comenzaremos con la opción **SIMULAR SENSOR**, donde puede elegir la simulación de los sensores indicados en la pantalla:

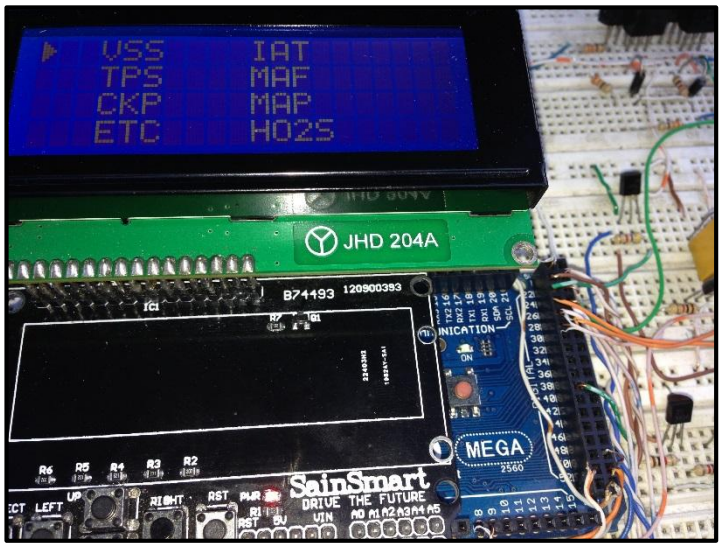


Figura 5.8. Interface de selección de sensor para simulación

Una vez se ingresa en la opción **VSS** podremos simular el funcionamiento del sensor, debiendo desconectarlo desconectar el mismo y conectar el equipo al conector que va hacia la ECU, tal como se indica en las siguientes figuras

Figura 5.9. Interface de conexión sensor VSS

Figura 5.10. Conector sensor VSS

Figura 5.11. Conexión VSS al equipo

Ya seleccionado la opción de simulación VSS, el equipo diagnostica los valores de voltaje en cada uno de los pines, y presenta la siguiente pantalla:

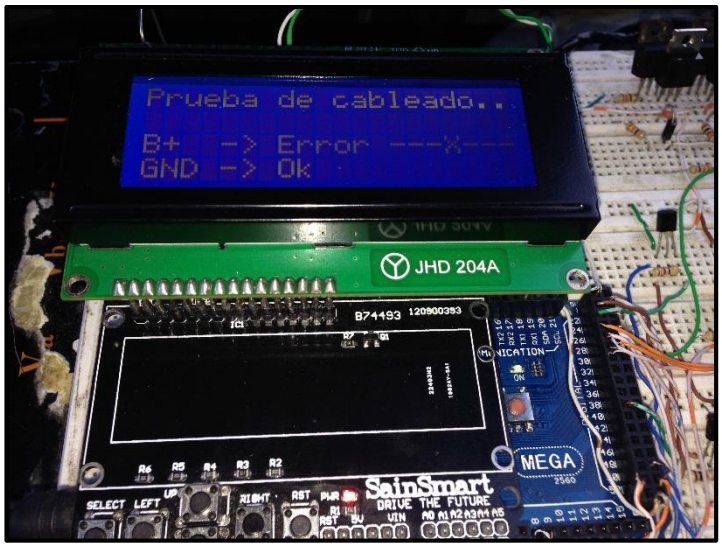


Figura 5.12. Interface de diagnóstico de cableado sensor VSS

Como se observa en la figura, B+--------> Error -------, indica al usuario que no existe voltaje en el pin. En este caso en particular, el motivo por el cual aparece este mensaje, es debido a que el switch de arranque se encuentra apagado, error forzado que lo realizamos para indicar el funcionamiento del mismo.

Vale la pena recalcar que el switch de arranque debe estar colocado en la posición ON para realizar cualquier simulación en los sensores.

Corrigiendo el error para ello ponemos el switch en ON, el equipo automáticamente pasa a la siguiente pantalla donde muestra el valor simulado y el leído por la ECU.

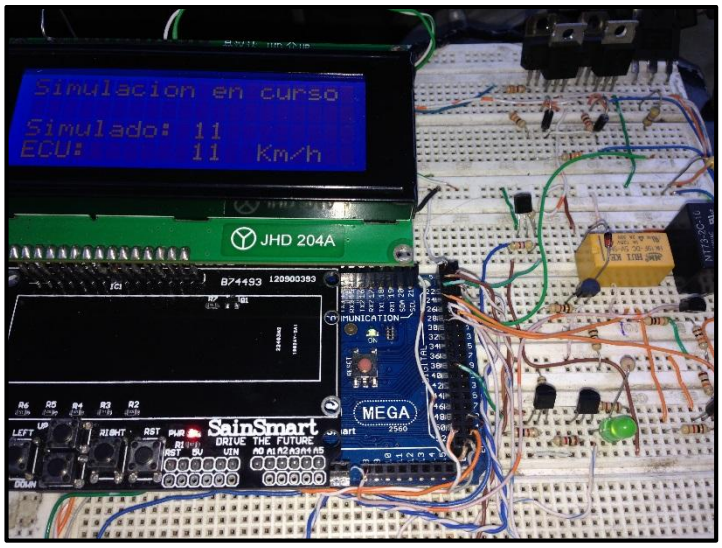


Figura 5.13. Interface de simulación del sensor VSS

Como se puede observar el valor simulado con el valor leído por la ECU es el mismo por lo tanto en el velocímetro tenemos el mismo valor de velocidad como se muestra en la figura:

Figura 5.14. Verificación de funcionamiento del equipo en el tacómetro

El siguiente sensor a simular de acuerdo al listado presentado es el TPS, su funcionamiento se presenta en la siguiente figura:

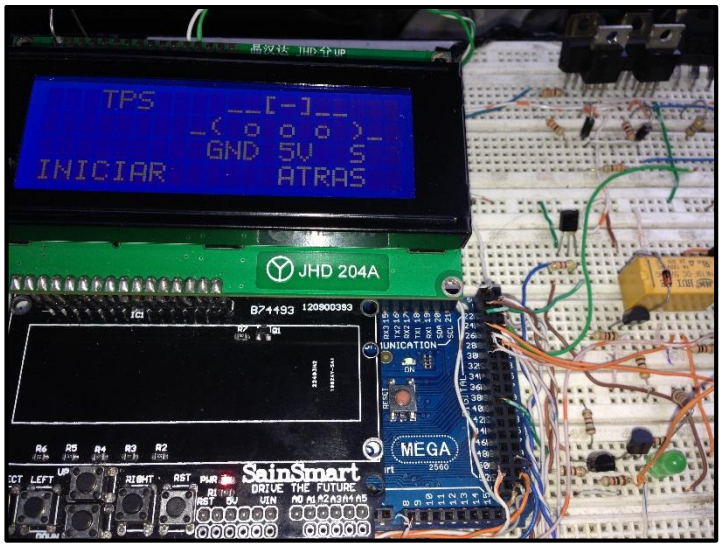


Figura 5.15. Interface de conexión de sensor TPS.

Desconectamos el sensor TPS y conectamos los cables de prueba del equipo al conector del sensor que va hacia la ECU, en la figura se indica su conexión:

Figura 5.16. Conector del sensor TPS

Como se puede ver en cada una de las pantallas presentadas en el LCD se da una guía al usuario de cómo realizar la conexión en cada uno de los sensores que se está probando.

Iniciamos la simulación e inmediatamente tenemos la visualización de la siguiente figura:

Figura 5.17. Interface de simulación de sensor TPS

Como se puede visualizar la simulación empieza en el valor 0%, presentando el valor leído por la ECU como 0%.

A medida que aumentamos el valor simulado, el valor leído por la ECU varía en función del mismo, es tan variable como el valor ingresado.

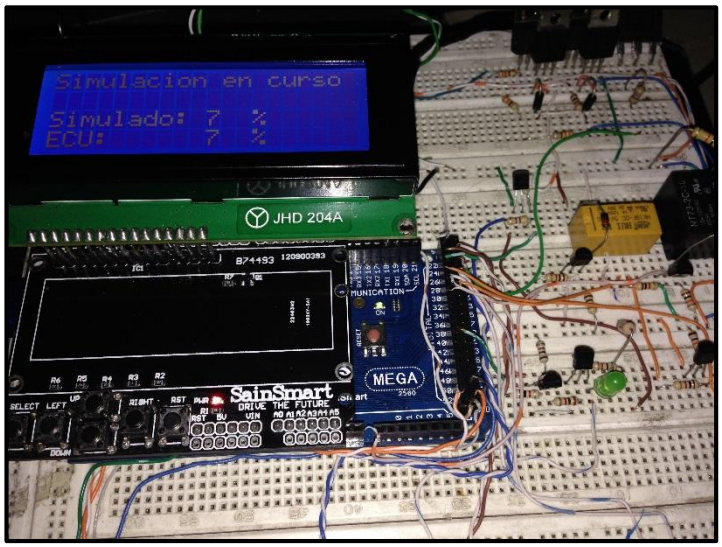


Figura 5.18. Interface de simulación sensor TPS al 7%

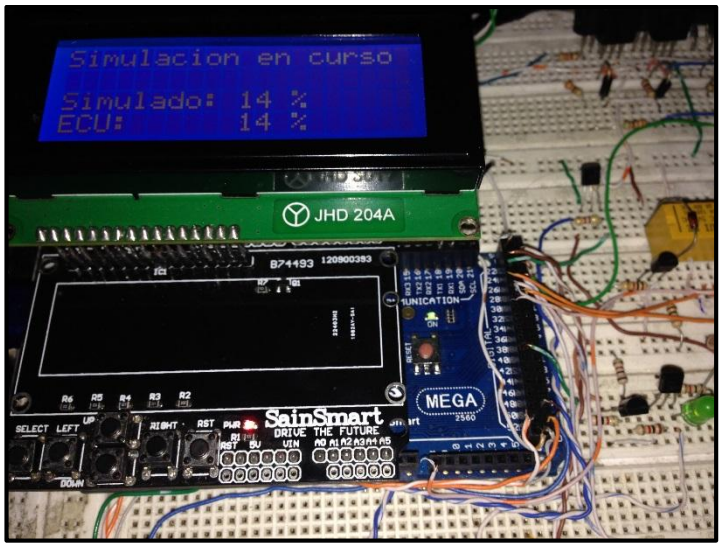


Figura 5.19. Interface de simulación sensor TPS al 14%

Para comprobar el correcto funcionamiento del automóvil, a medida que se aumenta el porcentaje de apertura del sensor TPS, las revoluciones del motor y los pulsos de inyección son más rápidos.

El motor empieza a fallar cuando la apertura del TPS alcanza valores altos, ya que la ECU detecta como una gran apertura en sus entradas, pero físicamente la mariposa en el múltiple de admisión está en la posición de ralentí.

La comprobación del sensor CKP se realiza de la siguiente manera: seleccionamos CKP para comprobarlo.

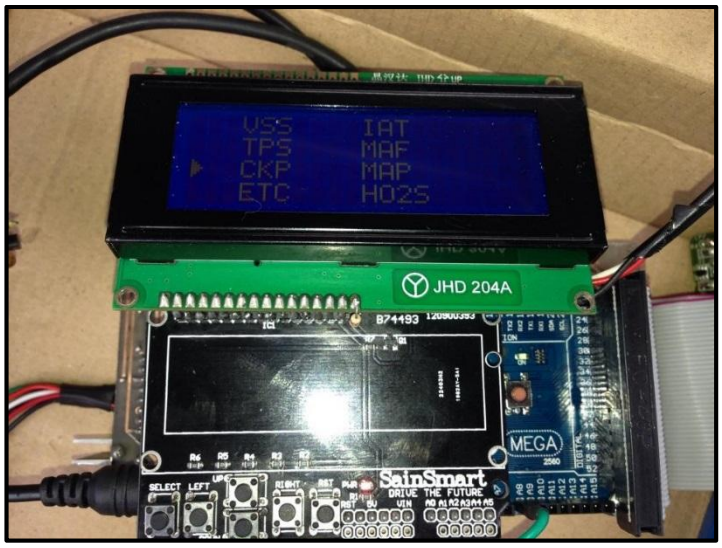


Figura 5.20. Interface de selección para simulación de sensor CKP

Ahora el equipo nos indica cómo se debe conectar el sensor al equipo en el terminar:

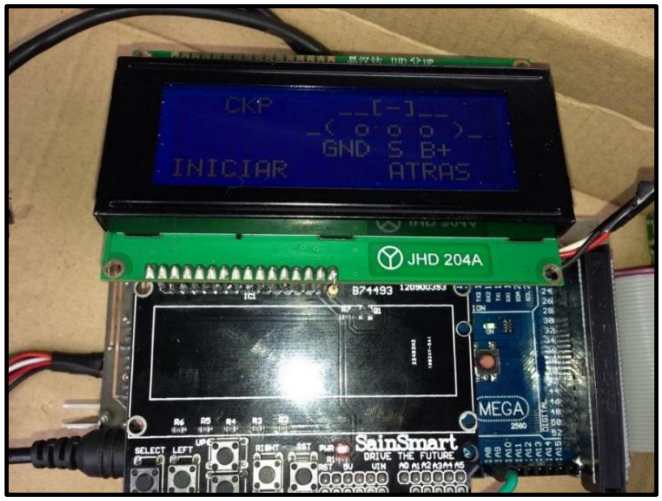


Figura 5.21. Interface de conexión del sensor CKP

El siguiente paso es conectar el terminal del señor CKP al equipo, como se indica en la figura:

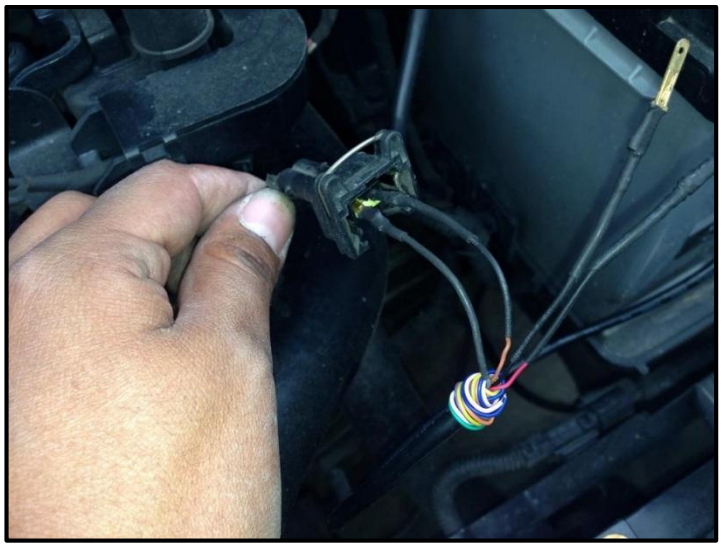


Figura 5.22. Conexión del sensor CKP al equipo

Y procedemos con la simulación del sensor CKP:

Figura 5.23. Interface de simulación sensor CKP a 497 rpm

Para la simulación del sensor CKP, se puede observar que se está simulando a 497 rpm, significa que la ECU del automóvil, reconoce la señal generada por el equipo a la razón indicada.

Continuamos con los sensores, el siguiente es el ECT, presentado de la siguiente manera:

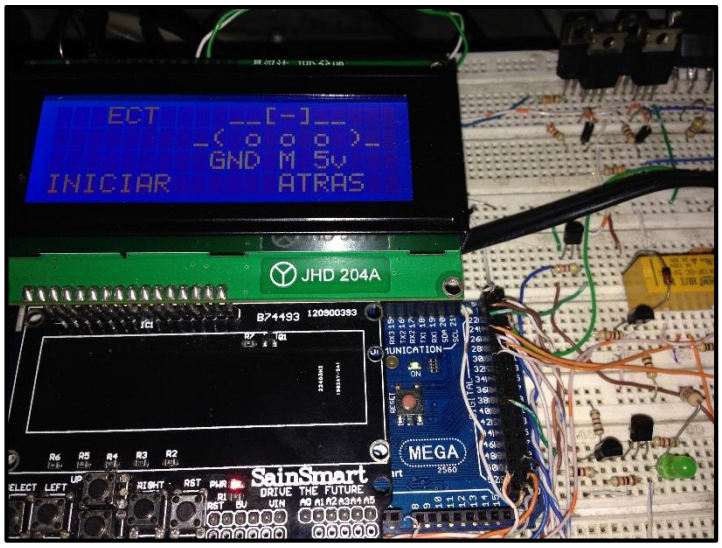


Figura 5.24. Interface de conexión del sensor ECT

Se desconecta el sensor ECT y conectamos el equipo al conector hacia la ECU, como se indica en la siguiente figura:

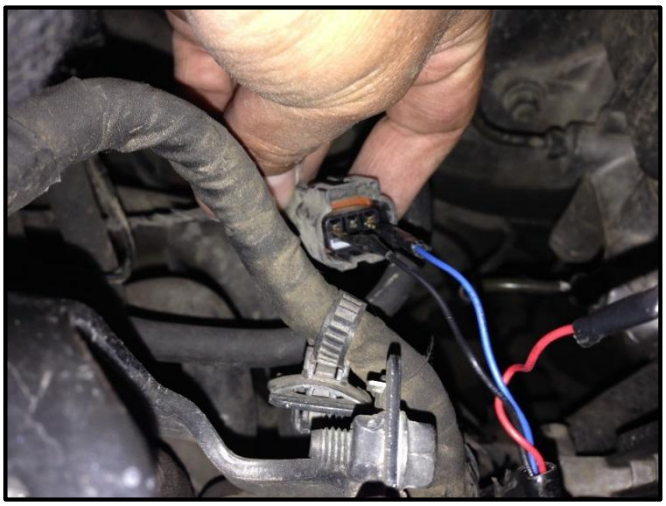


Figura 5.25. Conexión ECT al equipo

Iniciamos la simulación representada en la figura 5.26:

Figura 5.26. Interface de simulación del sensor ECT.

En este caso para comprobar el correcto funcionamiento, comenzamos a incrementar el valor simulado del ECT, y la ECU responde con el mismo valor, al llegar a superar el valor de 95 °C, se tiene que activar el electroventilador del automóvil; dicho procedimiento se presenta en las siguientes figuras:

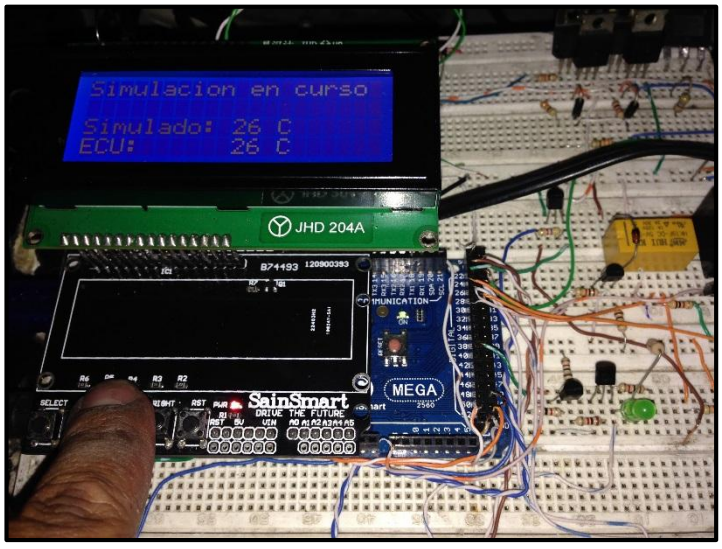


Figura 5.27. Interface de simulación sensor ECT a 26 °C.

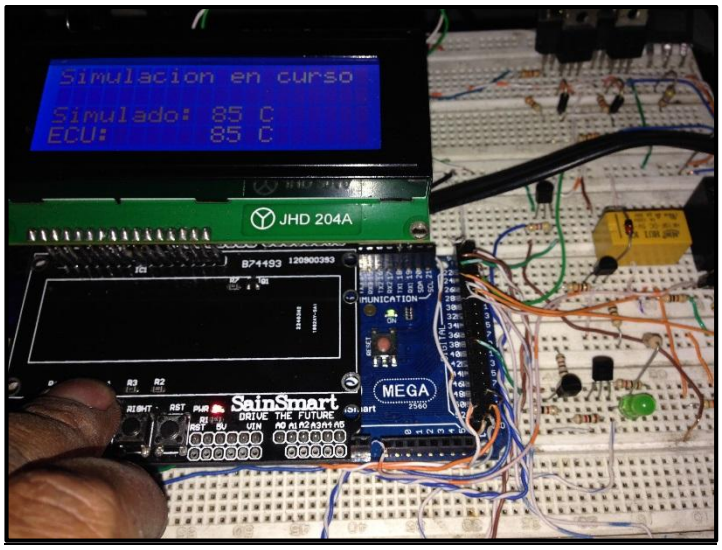


Figura 5.28. Interface de simulación sensor ECT a 85°C.

Figura 5.29. Electroventilador apagado (85°C)

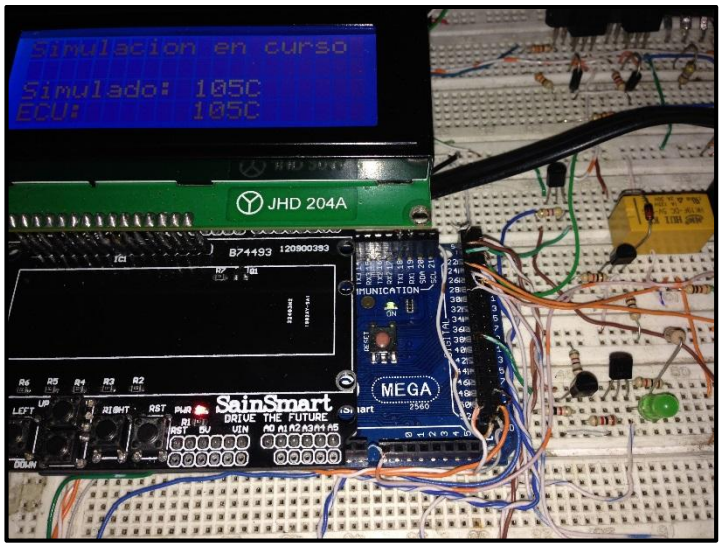


Figura 5.30. Interface de simulación sensor ECT a 105°C.

Figura 5.31. Electroventilador encendido (105°C)

Al activarse el ventilador se ha comprobado el correcto funcionamiento del equipo.

IAT es el siguiente sensor a simular, para lo cual regresamos en el menú para poder seleccionar su simulación:

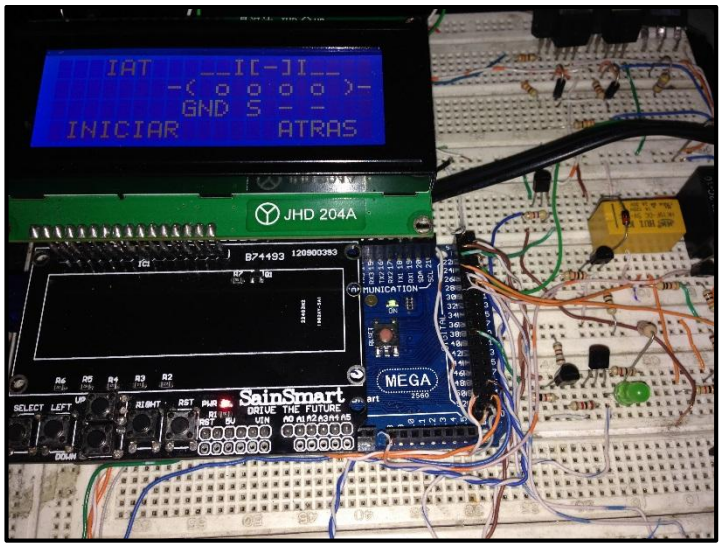


Figura 5.32 Interface de conexión del sensor IAT

Para simular el sensor IAT, lo desconectamos y conectamos el equipo de la siguiente manera:

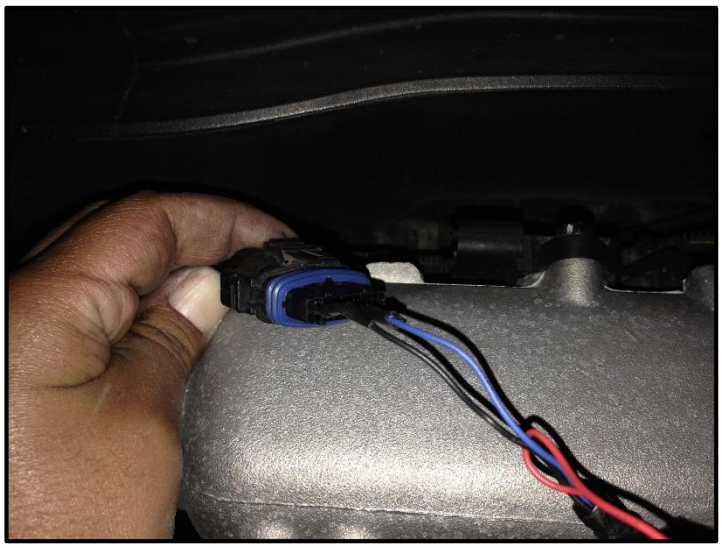


Figura 5.33 Conexión del sensor IAT al equipo

Se inicia la simulación, comprobando los voltajes de los pines del sensor IAT, comenzando inmediatamente con la simulación:

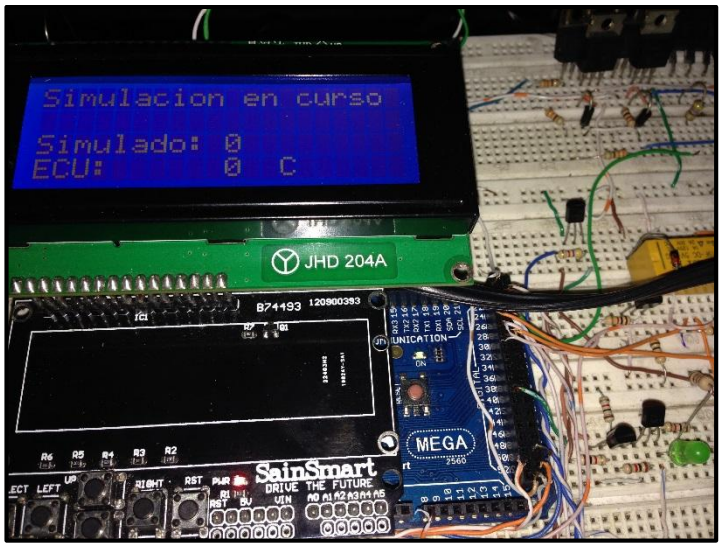


Figura 5.34. Interface de simulación sensor IAT

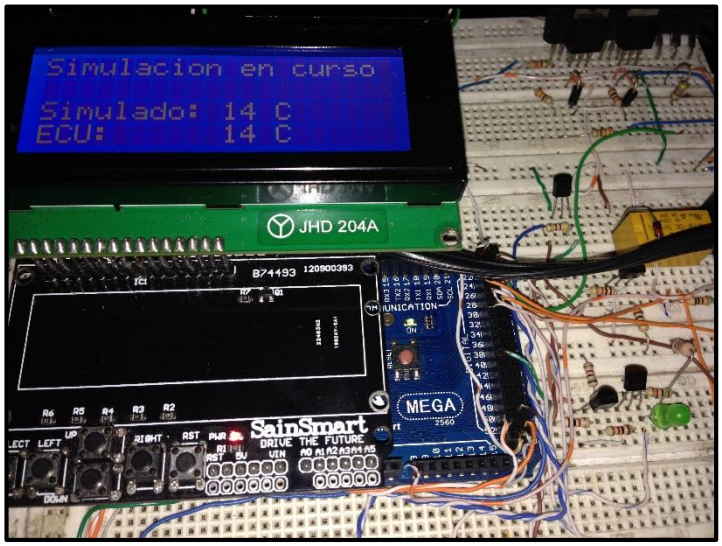


Figura 5.35 Interface de simulación sensor IAT 14°C

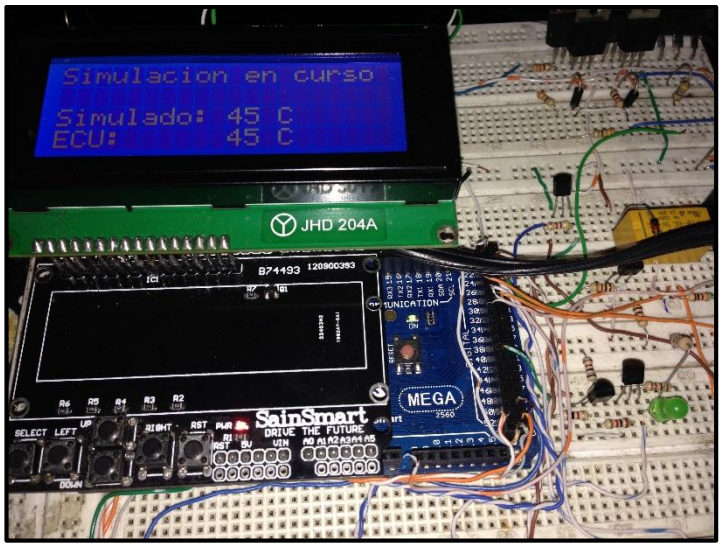


Figura 5.36 Interface de simulación sensor IAT 45°C

Funciona exactamente como el sensor ECT, solamente que en este sensor el rango de funcionamiento es mucho menor.

La comprobación del sensor MAP, está presentada en la siguiente figura:

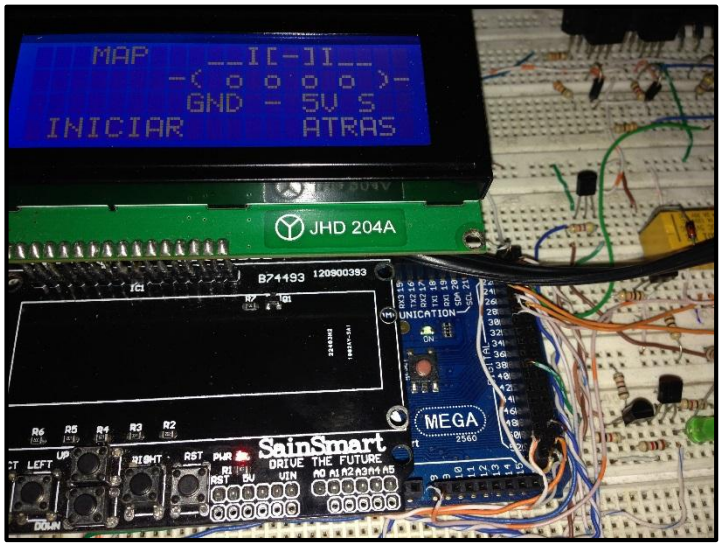


Figura 5.37 Interface de conexión de sensor MAP

Conectamos el equipo al conector del sensor hacia la ECU.

Figura 5.38 Conexión de sensor MAP al equipo.

E iniciamos la simulación:

Figura 5.39 Interface de simulación sensor IAT a 1 KPa

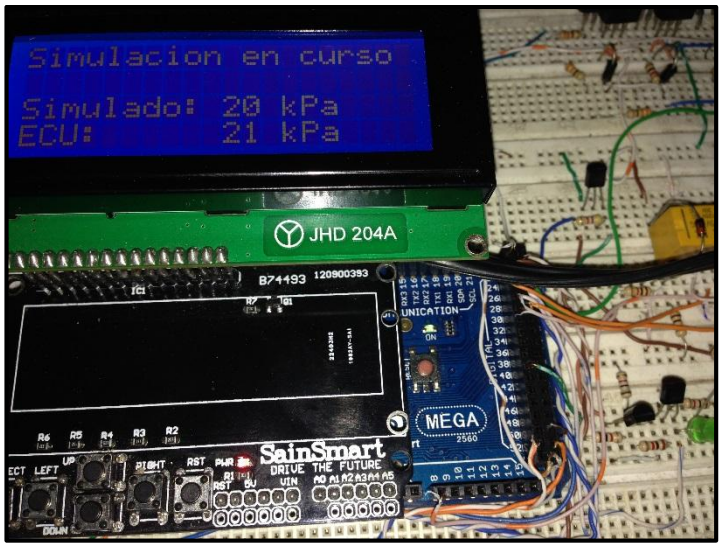


Figura 5.40 Interface de simulación sensor IAT a 21 KPa

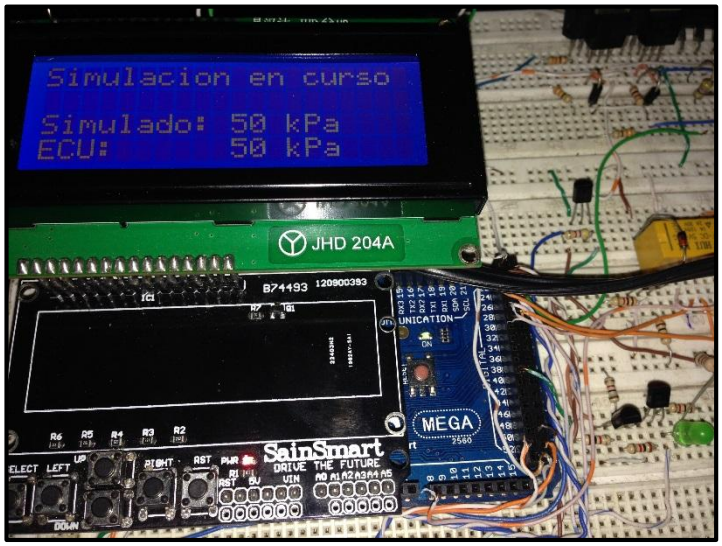


Figura 5.41 Interface de simulación sensor IAT a 50 KPa

Y por último tenemos la simulación del sensor de oxígeno HO2S, que cumple con el mismo procedimiento para su simulación, en las siguientes figuras se muestra su simulación:

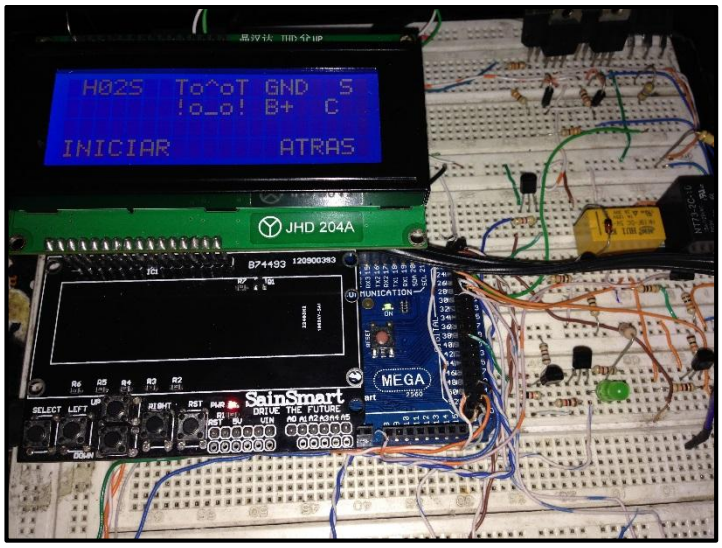


Figura 5.42 Interface de conexión sensor HO2S

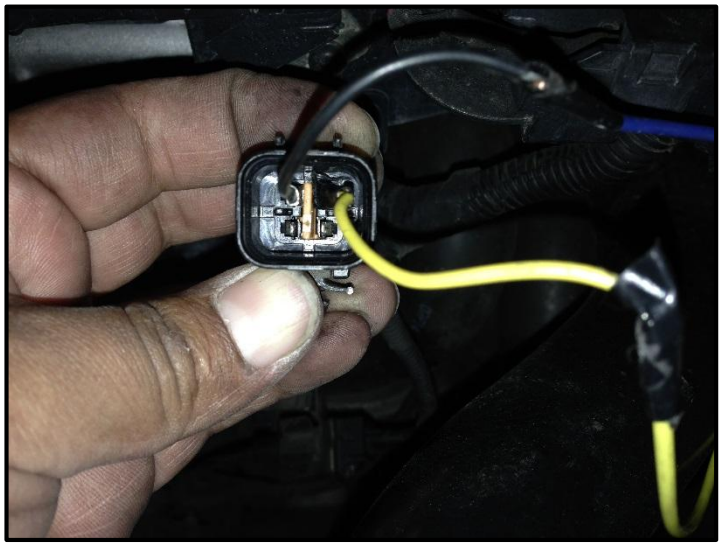


Figura 5.43. Conexión del sensor HO2S al equipo.

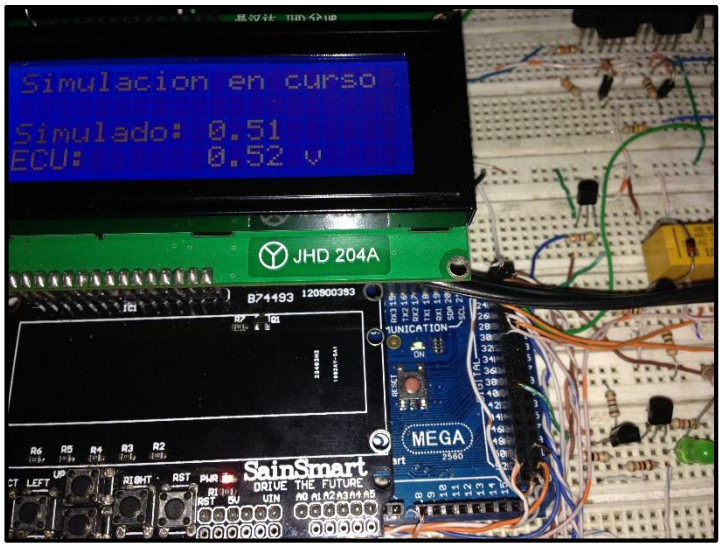


Figura 5.44. Interface de simulación de sensor HO2S

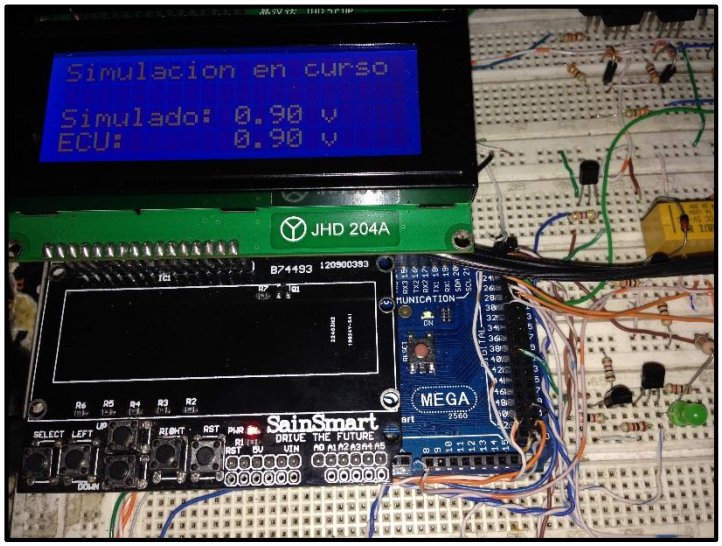


Figura 5.45. Interface de simulación de sensor HO2S a 0,9V

Figura 5.46. Interface de simulación del sensor HO2S a 0,21V.

5.2.1.2. Diagnóstico de sensores

Ahora vamos a regresar en la interface del LCD a fin de encontrar la opción probar sensor:

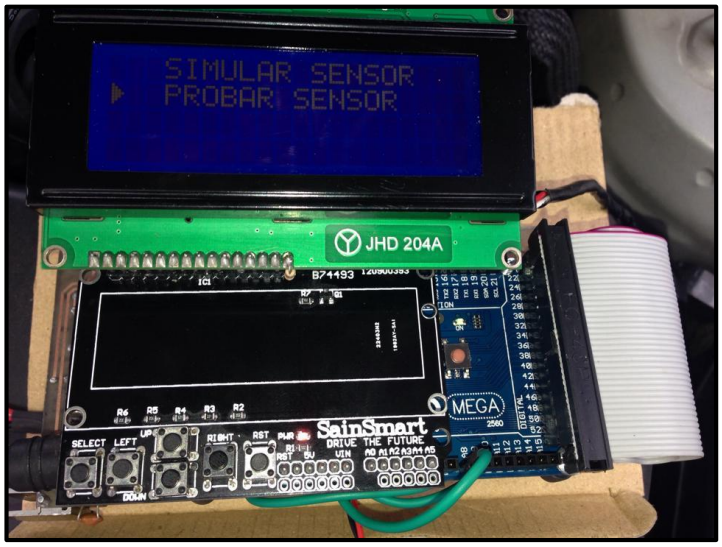


Figura 5.47. Interface para probar sensor

Obtenemos entonces el siguiente menú donde podremos seleccionar el sensor a probar:

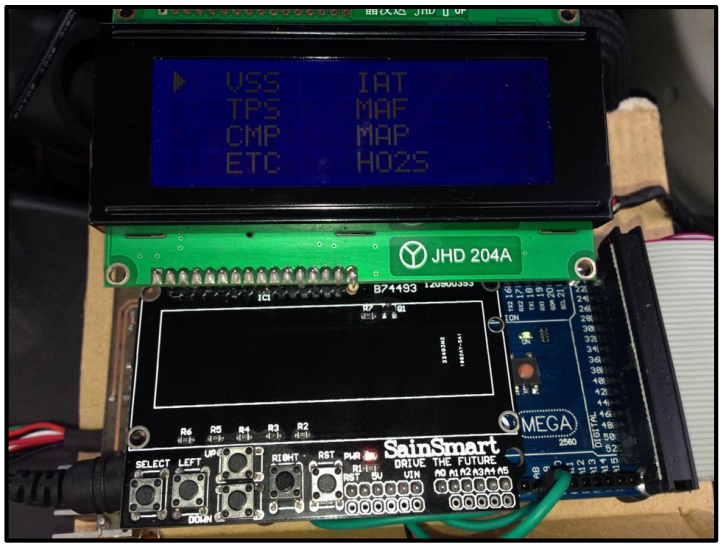


Figura 5.48. Interface se selección para probar un sensor

- **Comprobación sensor VSS**

Figura 5.49. Interface de conexión para probar sensor VSS

Una vez se conecte el sensor al dispositivo empezará a leer los pulsos generados en el mismo.

Figura 5.50. Interface de prueba de sensor VSS

Figura 5.51. Interface de prueba de sensor VSS a 5 pulsos.

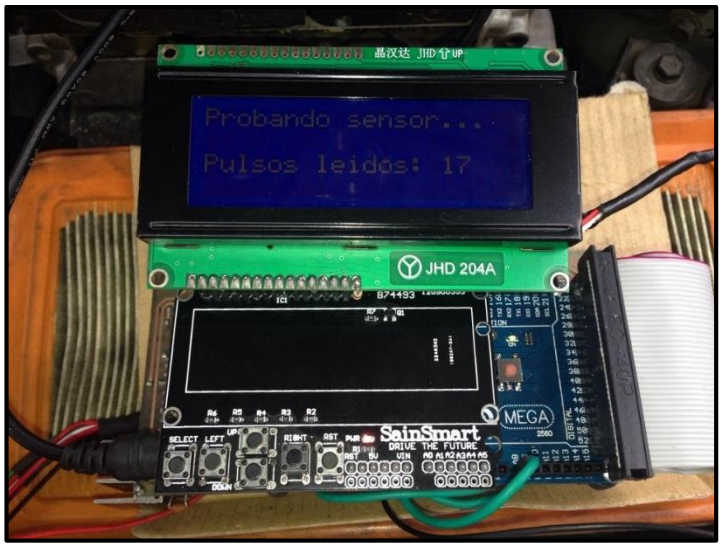


Figura 5.52. Interface de prueba de sensor VSS a 17 pulsos

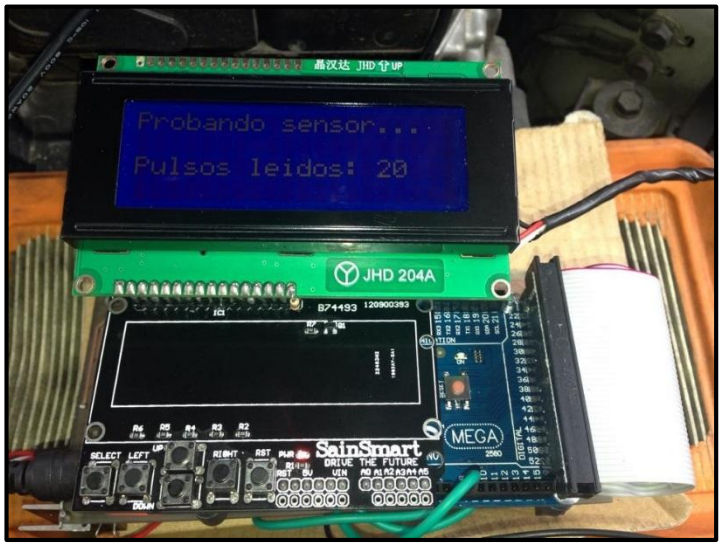


Figura 5.53. Interface de prueba de sensor VSS a 20 pulsos.

- **Comprobación sensor TPS**

Continuamos con el sensor TPS, lo conectamos, seleccionamos comprobar el sensor y ejecutamos para poder obtener las siguientes pantallas:

Figura 5.54. Interface de selección para prueba de sensor TPS.

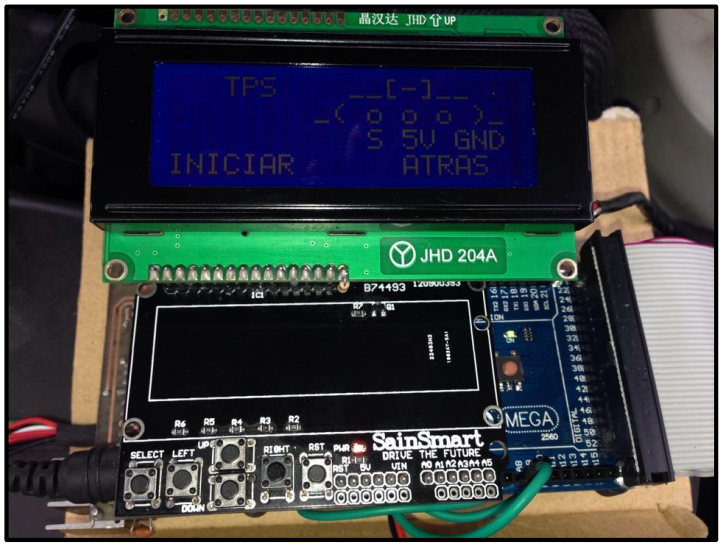


Figura 5.55. Interface de conexión para prueba de sensor TPS

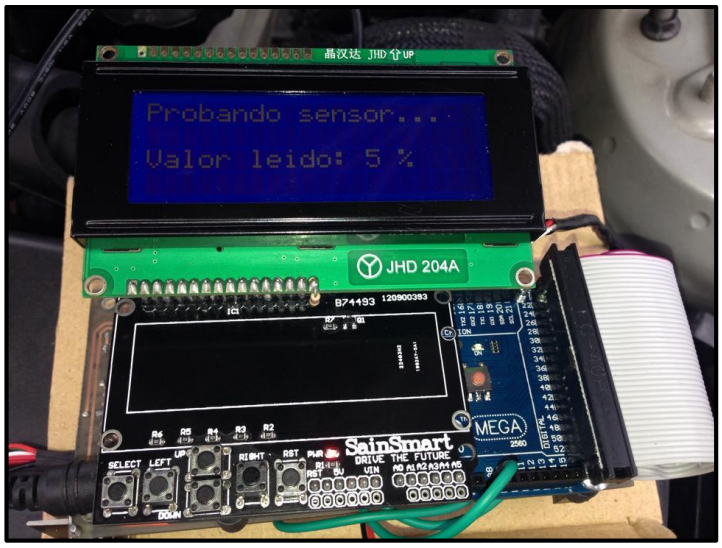


Figura 5.56. Interface de prueba sensor TPS al 5%

Figura 5.57. Interface de prueba sensor TPS a 28%

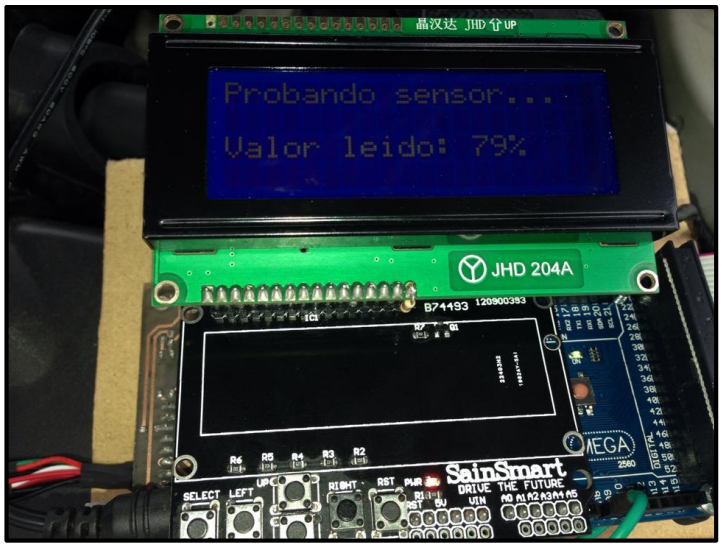


Figura 5.58. Interface de prueba sensor TPS a 79%

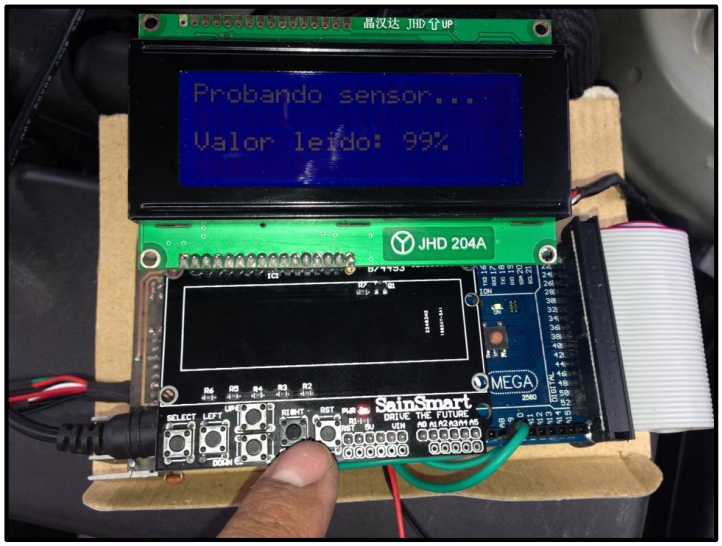


Figura 5.59. Interface de prueba sensor TPS a 99%

- **Comprobación sensor CMP**

El siguiente sensor a probar es el CMP, conectándolo, seleccionando probarlo y ejecutando se tiene los siguientes resultados:

Figura 5.60. Ubicación del sensor CMP en el automóvil

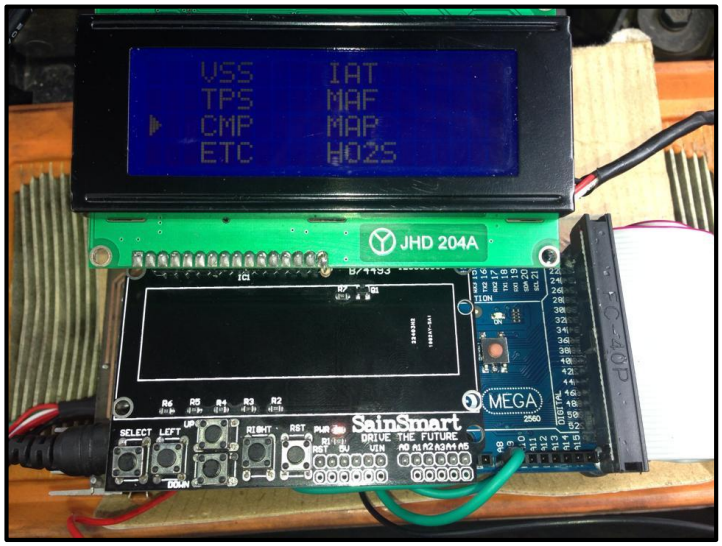


Figura 5.61. Interface de selección de prueba de sensor CMP

Figura 5.62. Interface de conexión para prueba de sensor CMP

Figura 5.63. Interface de prueba sensor CMP

Figura 5.64. Interface de prueba sensor CMP a 5 pulsos

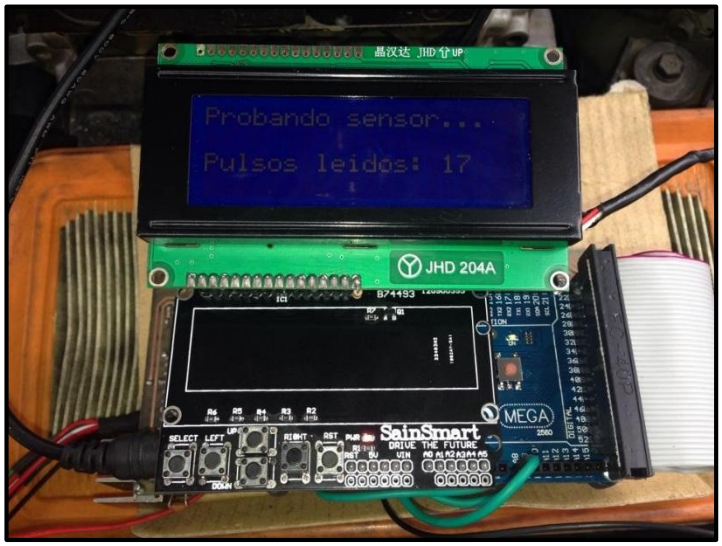


Figura 5.65. Interface de prueba sensor CMP a 17 pulsos

Figura 5.66. Interface de prueba sensor CMP a 20 pulsos

- **Comprobación sensor ECT**

El ECT es el siguiente sensor a comprobar, se conecta el mismo al equipo para comprobarlo obteniendo los siguientes resultados:

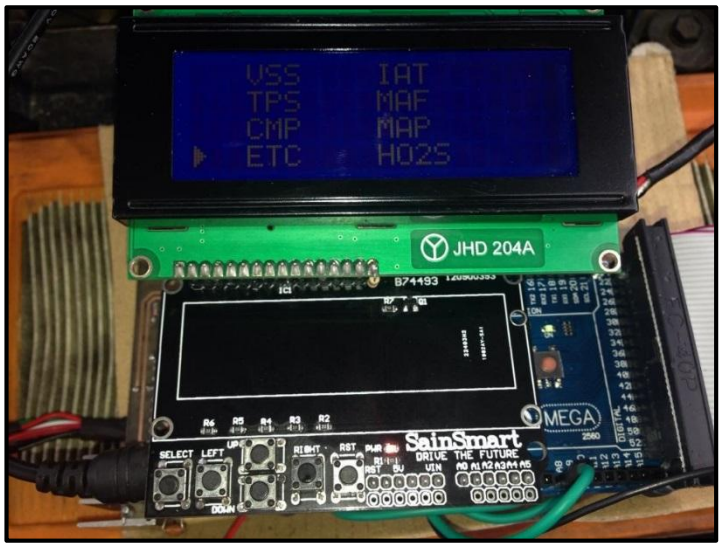


Figura 5.67. Interface de selección de sensor ECT

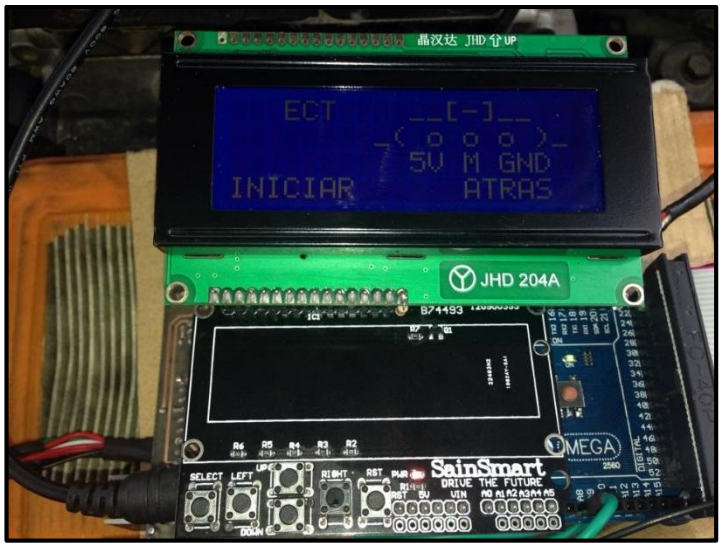


Figura 5.68. Interface de conexión para prueba sensor ECT

Figura 5.69. Interface de prueba de sensor ECT a 21°C

Figura 5.70. Interface de prueba sensor ECT a 22°C

- **Comprobación del sensor IAT**

La prueba del sensor IAT, es idéntica al sensor ECT, para prueba de ello se presentan las siguientes figuras:

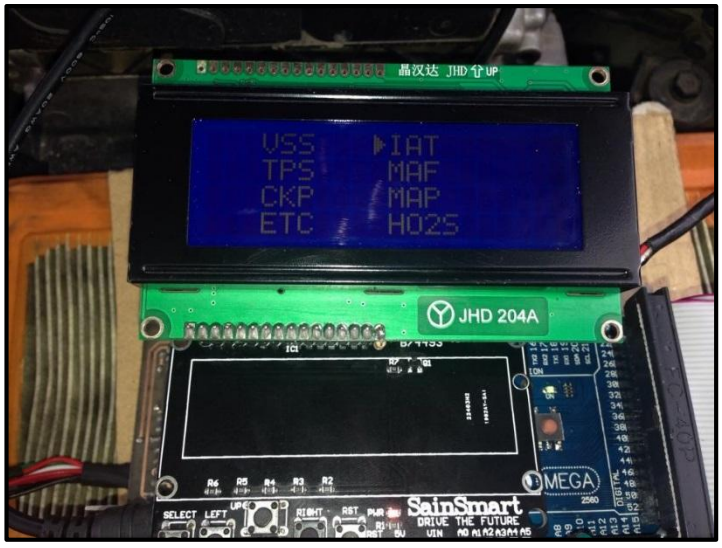


Figura 5.71. Interface de selección para prueba sensor IAT

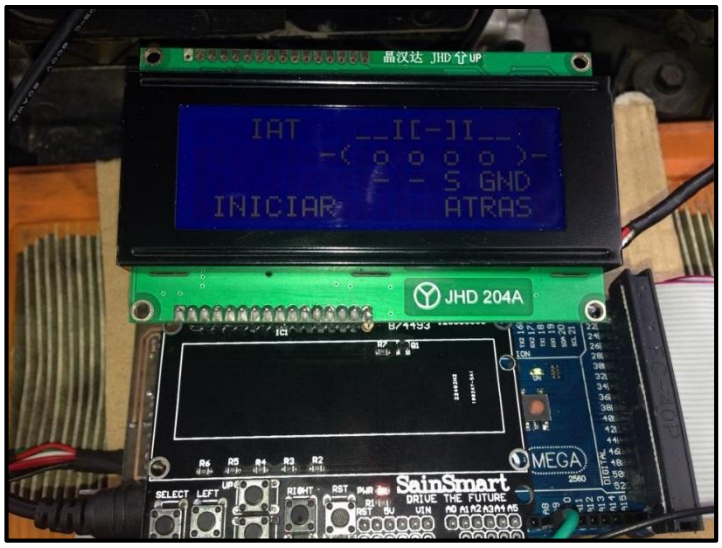


Figura 5.72. Interface de conexión de sensor IAT

Figura 5.73. Interface de prueba sensor IAT a 21°C

Figura 5.74. Interface de prueba sensor IAT a 22°C

- **Comprobación sensor MAF**

Figura 5.75. Interface de selección de sensor MAF

Figura 5.76. Interface de conexión de prueba sensor MAF

La comprobación para el sensor MAF, queda pendiente ya que al existir el sensor MAP en el automóvil que estamos utilizando como base para nuestro estudio, el sensor MAF no existe, pero el sistema del dispositivo deja abierto un espacio para la programación del mismo.

- **Comprobación del sensor MAP**

Figura 5.77. Interface de selección de prueba de sensor MAP

Figura 5.78. Interface de conexión para prueba sensor MAP

Figura 5.79. Interface de prueba sensor MAP a 21 KPa

Figura 5.80. Interface de prueba sensor MAP a 27 kPa

- **Comprobación del sensor de oxígeno**

Figura 5.81. Interface de selección de prueba sensor HO2S

Figura 5.82. Interface de conexión para prueba sensor HO2S

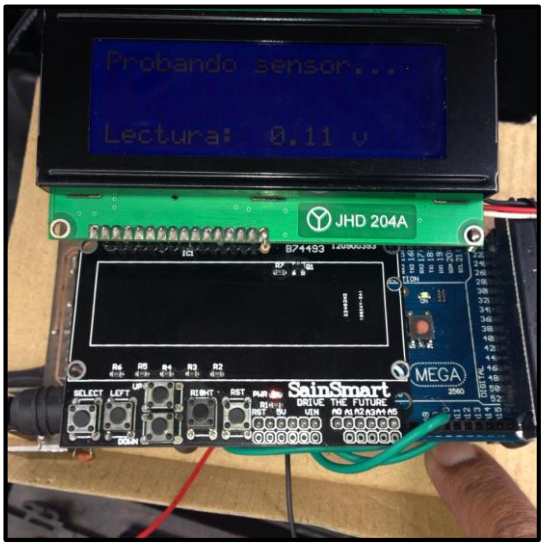


Figura 5.83. Interface de prueba sensor HO2S a 0,11V

Figura 5.84. Interface de prueba sensor HO2S con valores de referencia

5.3 Diagnóstico de actuadores

- Comprobación del inyector

Figura 5.85 Interface de selección para diagnóstico de inyector.

Se conecta el inyector al dispositivo y tras seleccionar comprobar el inyector, al presionar la tecla de iniciar se escucha la activación de la bobina del mismo. Cabe indicar que esta prueba se realiza con el inyector montado en el motor el automóvil.

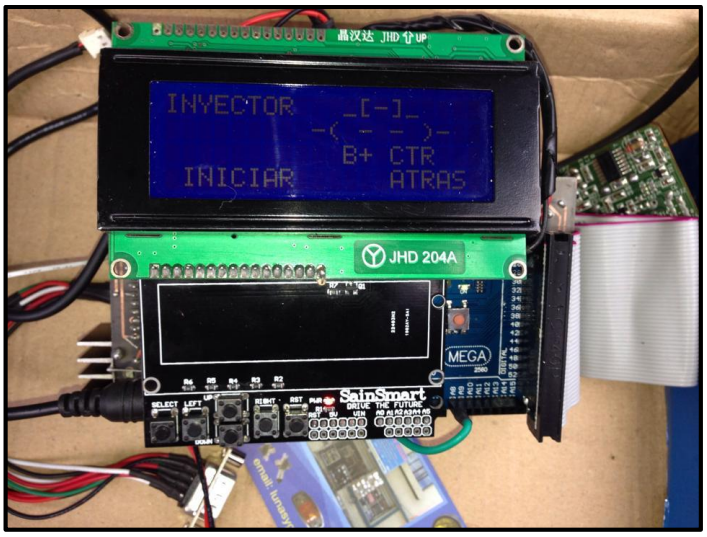


Figura 5.86 Interface de activación del inyector

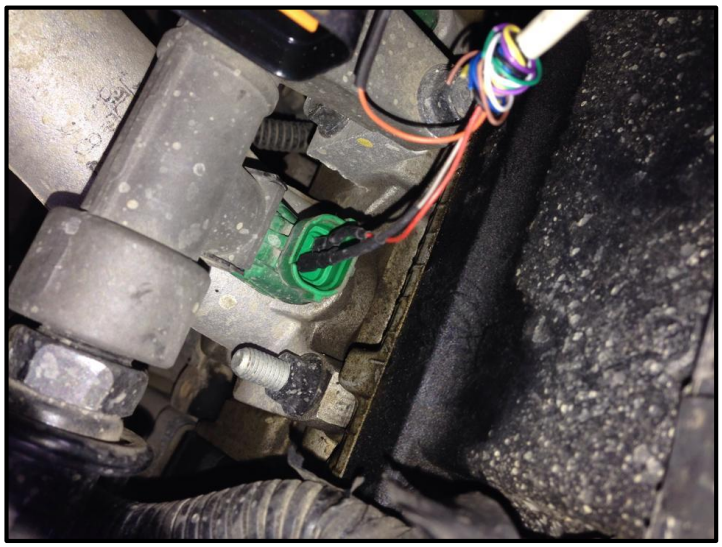


Figura 5.87. Conexión del inyector al equipo

- **Comprobación de las bobinas del automóvil**

Su comprobación es similar al de los inyectores con la diferencia que ahora el dispositivo se conectará a los pines de conexión de la bobina.

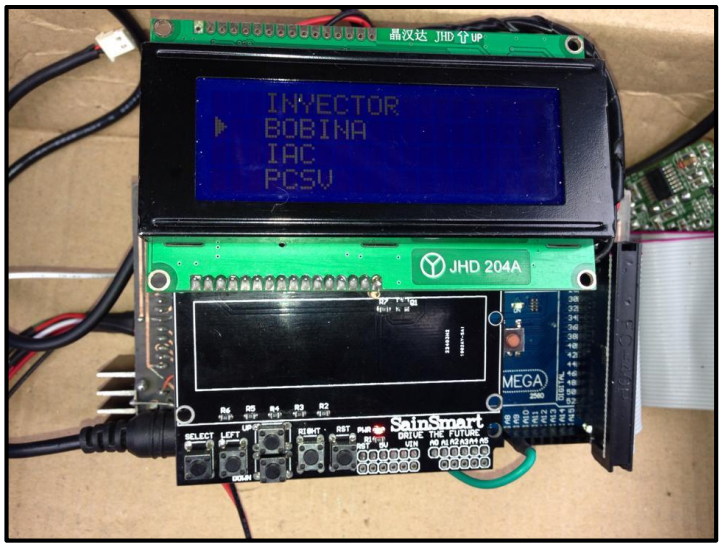


Figura 5.88. Interface de selección para diagnóstico de bobinas de encendido

Figura 5.89. Conexión de una bobina de encendido al equipo

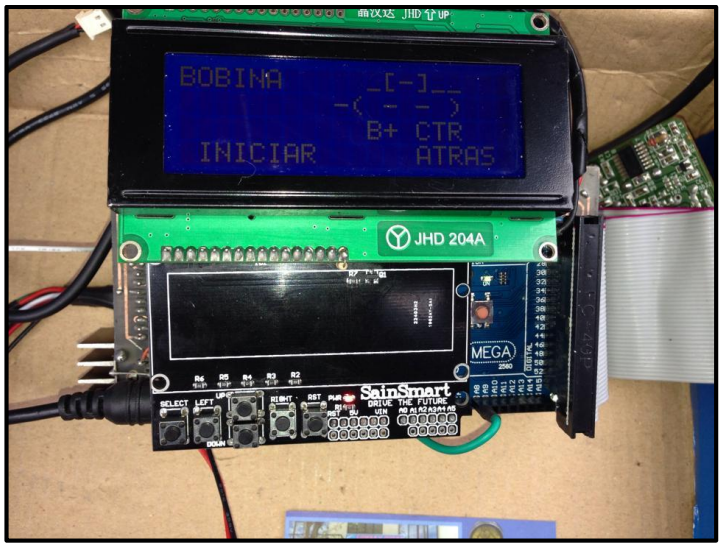


Figura 5.90. Interface de activación de la bobina de encendido

Para comprobar el funcionamiento de las bobinas, desconectamos el cable de la bujía, y con un destornillador comprobamos que se genere una chispa al derivarlo a gnd, al presionar el botón de iniciar en el dispositivo.

Comprobación de la IAC

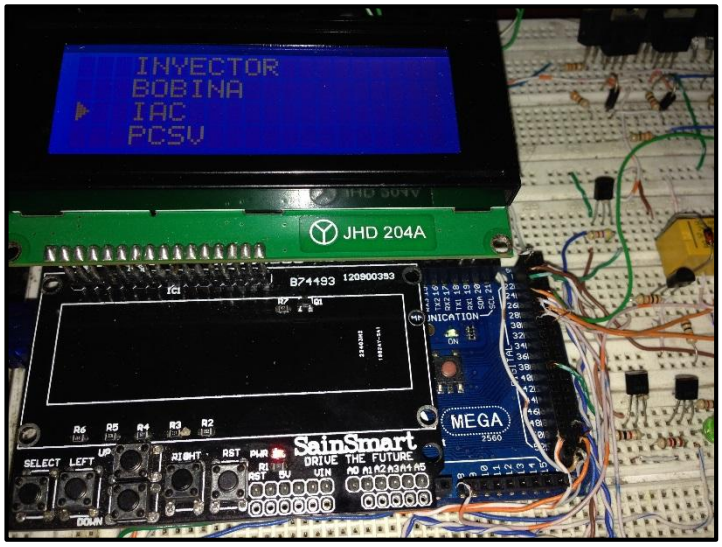


Figura 5.91. Interface de selección para prueba de IAC

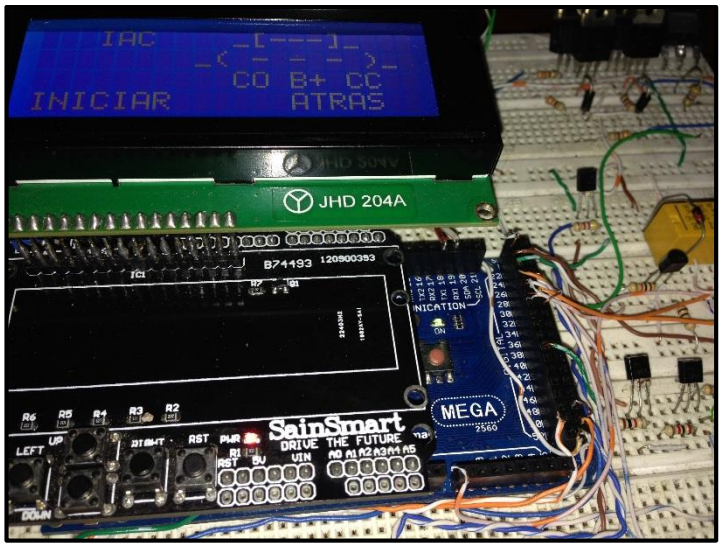


Figura 5.92. Interface de conexión de la IAC al equipo

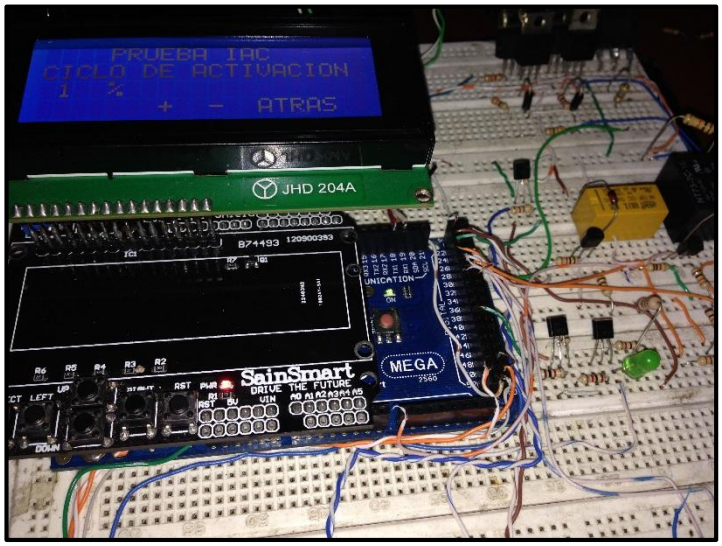


Figura 5.93. Interface de prueba de IAC al 1%

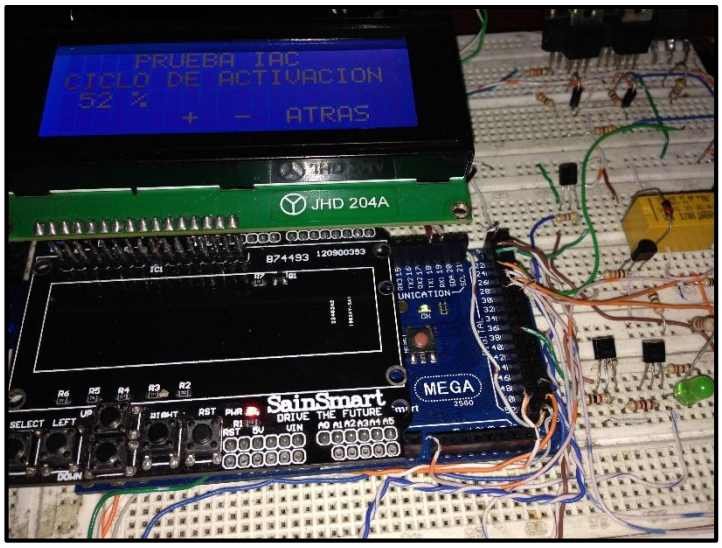


Figura 5.94. Interface de prueba de IAC al 52%

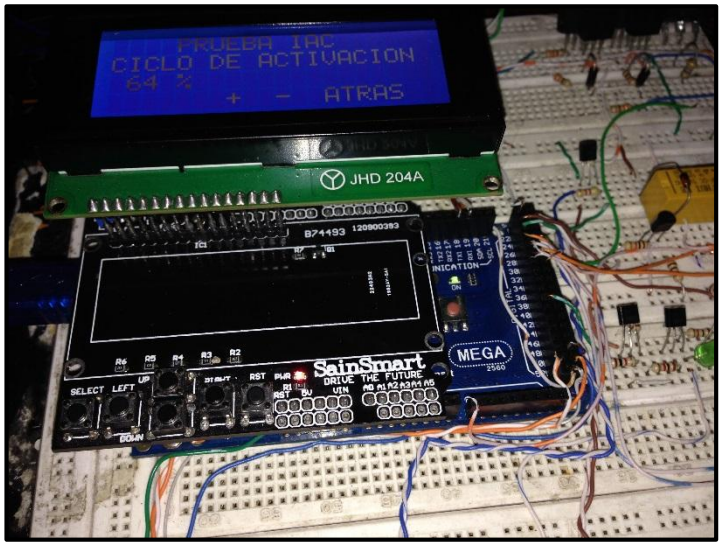


Figura 5.95. Interface de prueba de IAC al 64%

Figura 5.96. Conexión de la IAC al equipo

Para la comprobación de la IAC, se consiguió un actuador de ese tipo y se puedo comprobar el funcionamiento del dispositivo sobre el actuador. A medida que se aumenta el ciclo de activación, el actuador va cambiando su posición.

Cuando es probado sobre el automóvil, lo que se hace es escuchar el actuador cuando el ciclo de activación es cambiado, ya que este igual que el inyector emite un sonido característico de funcionamiento.

- **Comprobación de PCSV**

Una vez comprobados tanto sensores como actuadores y a su vez la unidad electrónica de control, se procede con comprobación de la lectura de códigos de error generados en la ECU, para ello procedemos como se indican en las siguientes figuras:

Figura 5.97. Interface de selección para prueba PCSV

Figura 5.98. Conexión PCSV al equipo

Figura 5.99. Interface de activación PCSV

5.4 Comprobación de Lectura de Códigos de Diagnóstico

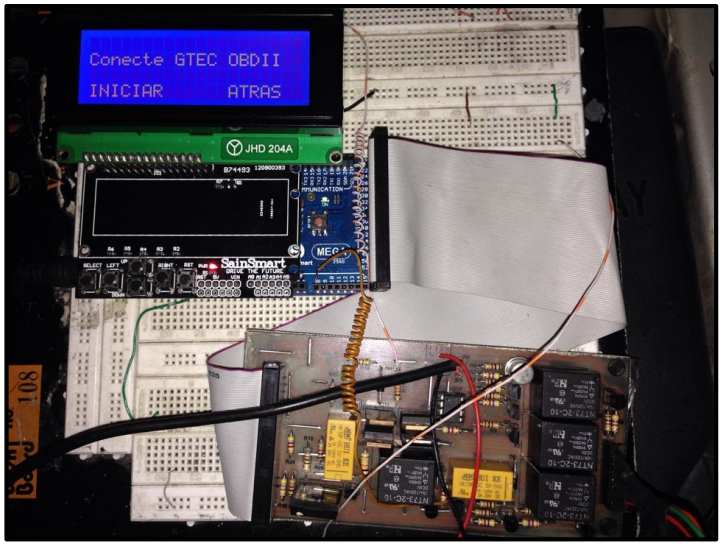


Figura 5.100. Interface de inicialización de lectura de DTC

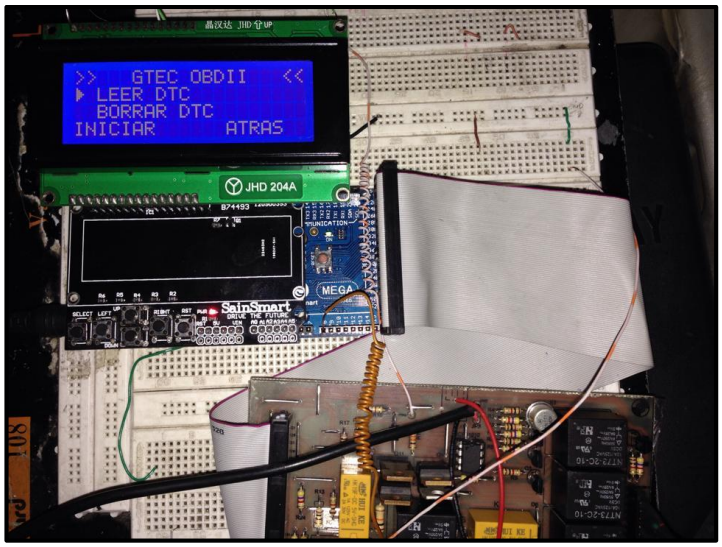


Figura 5.101. Interface selección para leer DTC

Para comprobar la lectura de los códigos de falla generados en la unidad electrónica de control tras un fallo en cualquier sensor o actuador, vamos a forzar el fallo de uno de sus sensores. Desconectamos el sensor TPS y verificamos la lectura con el dispositivo.

Figura 5.102. Desconexión de sensor TPS

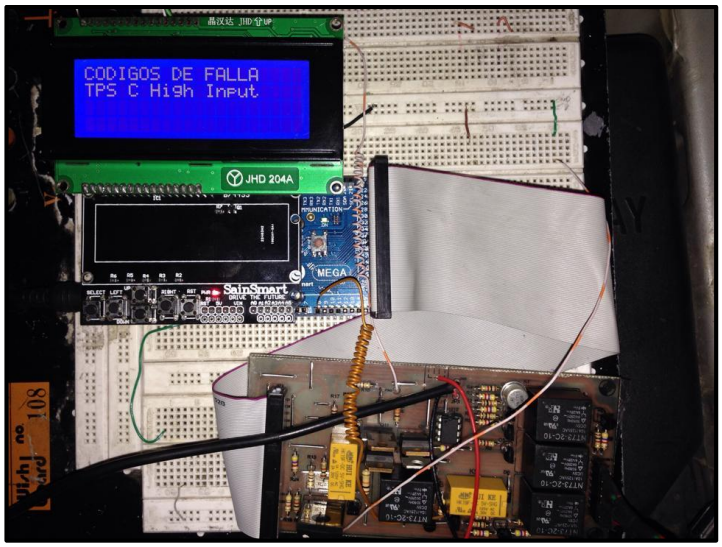


Figura 5.103 Lectura de DTC

Ahora para comprobar el funcionamiento con más de un código de error generado en la unidad electrónica de control, se desconecta el sensor MAP y se efectúa nuevamente la lectura del código de falla

Figura 5.104. Desconexión del sensor MAP

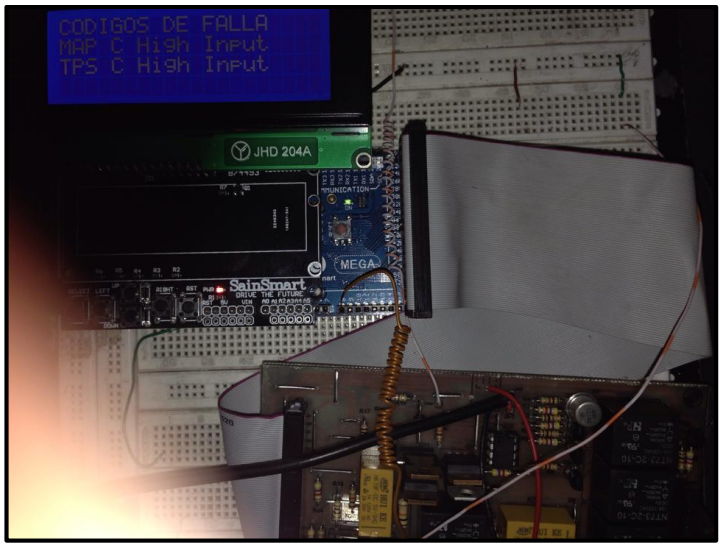


Figura 5.105. Lectura de varios DTC

Como se puede observar se verifica que existe ya no solo el código de falla anterior, sino que el nuevo código aparecen en pantalla.

La reparación de fallas en este caso consiste únicamente en la conexión de los sensores que se desconectaron para forzar los códigos.

Ahora procedemos a borrar los códigos de falla, ya que fueron reparados los orígenes de los mismos:

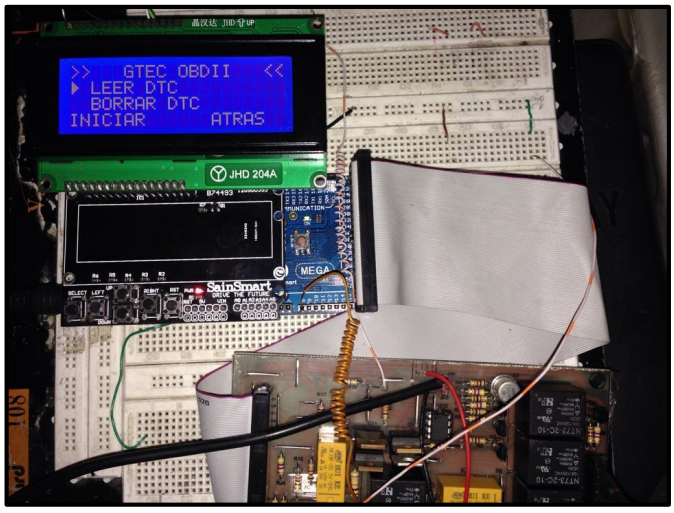


Figura 5.106. Interface de selección para borrar DTC

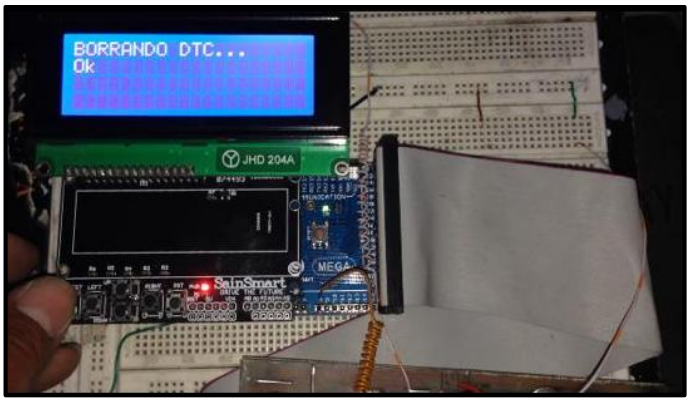


Figura 5.107. Interface de borrado de DTC

Se han borrado los códigos de falla, además de apagarse la luz indicadora **CHECK ENGINE** en el tablero del automóvil, se comprueba con el mismo dispositivo que no existe código de falla al volver a seleccionar leer código.

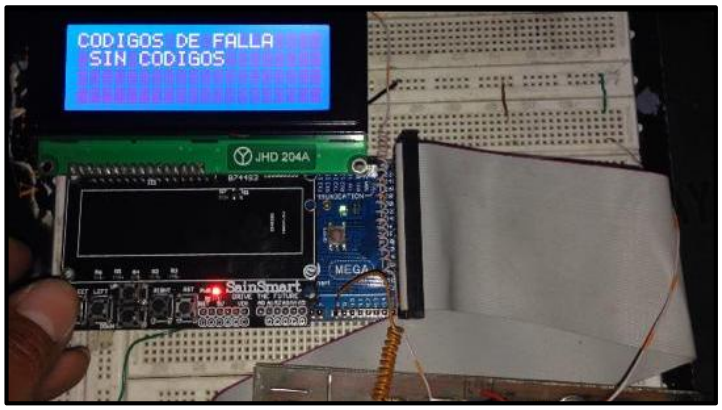


Figura 5.108. Interface de comprobación de eliminación de DTC

5.5. Resultados final del equipo de diagnóstico

Culminadas todas las pruebas sobre los sensores, actuadores y la unidad electrónica de control, procedemos a montar el equipo sobre una base fija, adecuando los controles para mayor versatilidad en su uso. En las siguientes figuras se muestra el resultado final del equipo de diagnóstico:

Figura 5.109. Equipo de diagnóstico de sensores actuadores y ECU's automotrices

Figura 5.110. Cableado del equipo

5.5. Visualización de señales mediante la herramienta virtual

Visualización de señal VSS

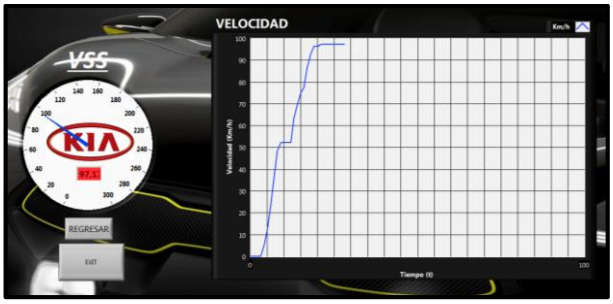


Figura 5.111. Interface de comprobación de señal VSS

Visualización de RPM

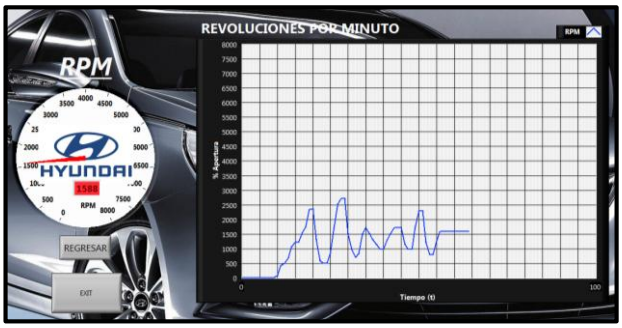


Figura 5.112. Interface de comprobación de RPM

Visualización señal de ECT

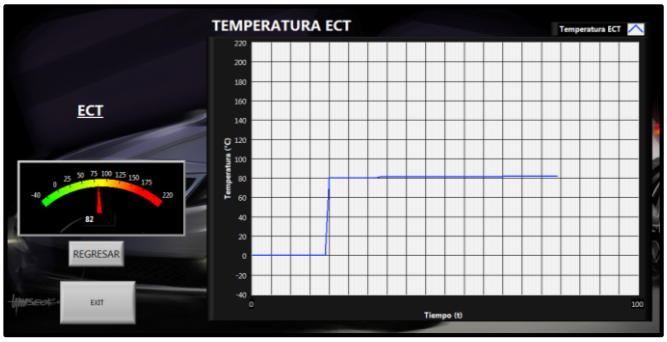


Figura 5.113. Interface de comprobación de señal ECT

Visualización de señal IAT

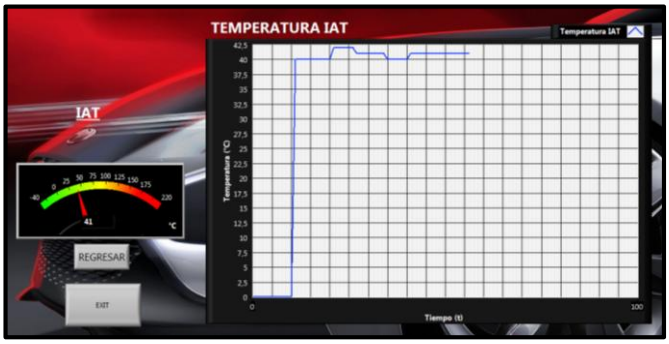


Figura 5.114. Interface de comprobación de señal IAT

Visualización de señal TPS

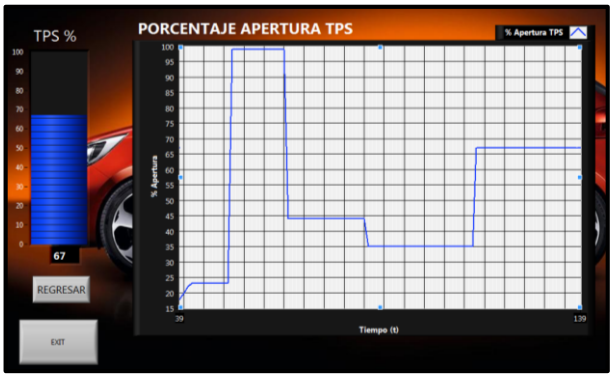


Figura 5.115. Interface de comprobación de señal TPS

Visualización de señal MAP

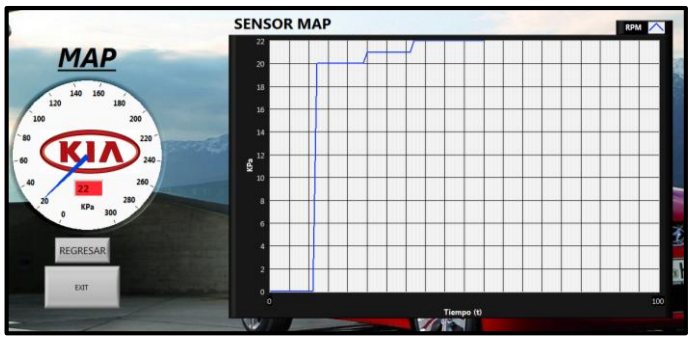


Figura 5.116. Interface de comprobación de señal MAP

Visualización de señal MAF

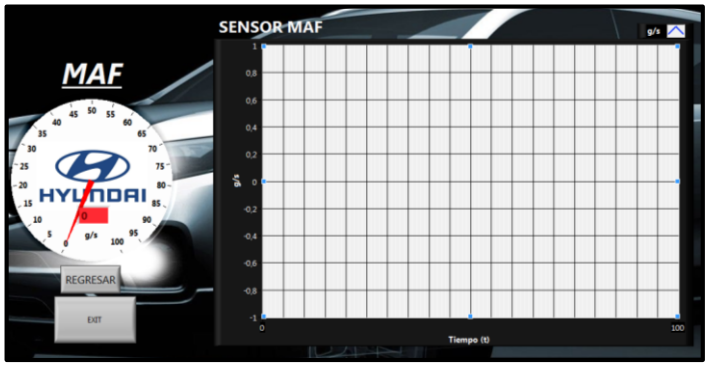


Figura 5.117. Interface de comprobación de señal MAF

Como se ha venido recalcando se deja el espacio para la mejora del equipo cuando se disponga de un automóvil con sensor MAF para las respectivas pruebas.

Visualización de señal sensor HO2S

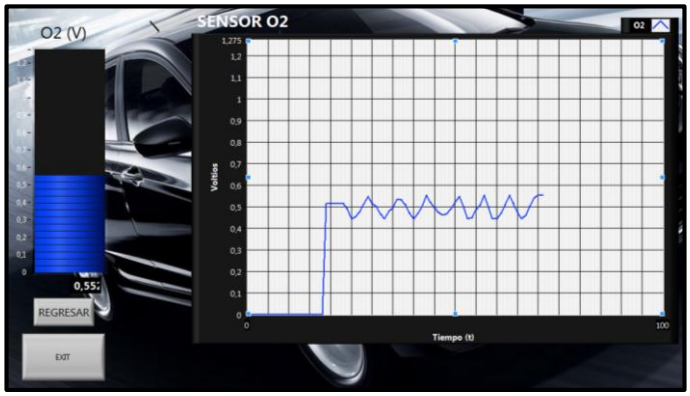


Figura 5.118. Interface de comprobación de señal HO2S

CONCLUSIONES

- El proyecto desarrollado consiste en un equipo de diagnóstico para sensores: VSS, TPS, CKP, CMP, ECT, IAT, MAP, MAF y HO2S, programado para simular o probar los mismos. La simulación diagnostica el correcto funcionamiento del cableado, el hardware y software de la unidad electrónica de control. Además obtiene el diagnóstico de cada uno de los sensores mediante la lectura de sus valores en su funcionamiento.

  El equipo diagnostica además cuatro actuadores: Inyectores, Bobinas de encendido, IAC y PCSV, generando señales específicas para cada uno de los actuadores en función de sus propias características.

- Con el diseño y construcción del equipo se cumplieron con los objetivos impuestos en el planteamiento de este trabajo. El resultado del mismo es un equipo que cumple con las expectativas de diagnóstico para cada uno de los sensores y actuadores impuestos; y la unidad electrónica de control de los automóviles KIA y HYUNDAI, con la condición de tener un sistema de diagnóstico OBD2.

- El equipo es portátil, amigable con el usuario, de operación sencilla y en pantalla presenta los diagramas de conexión a fin de que el operario no tenga complicaciones al momento de utilizar el equipo.

- La construcción de placas y componentes del equipo son de fácil adquisición en el mercado, lo que representa una ventaja en caso de tener que realizar algún tipo de reparación.

- Todas las pruebas fueron realizadas sobre el automóvil KIA RIO XCITE, siendo también probado en un automóvil HYUNDAI TUCSON, sobre el cual se pudo leer los datos tal y como se plantea en el desarrollo del trabajo, con lo que se comprueba que funciona para las dos marcas, siempre y cuando posean sistema de diagnóstico OBD2.

- En comparación con otros equipos en el mercado, nuestro equipo presenta la ventaja de poseer tres cables para el diagnóstico: uno para simular sensores y comprobar la unidad electrónica de control, otro para probar sensores y el último para probar actuadores, que comparado con equipos que se ofrecen en el mercado, estos llevan un par o grupo de cables para comprobar cada uno de los sensores, lo que hace muy versátil al equipo diseñado.

- El costo del equipo es relativamente bajo, ya que los elementos en uso son comúnmente comercializados en nuestro mercado tanto automotriz como electrónico.

- El desarrollo del proyecto alcanzó un punto adicional en el desarrollo del mismo, ya que el equipo además de realizar y cumplir con los objetivos

planteados, puede leer y borrar los códigos de falla generados en la memoria de la unidad electrónica de control.

- Además la herramienta virtual para la visualización de señales de los sensores es de gran apoyo para el operario, ya que corrobora datos obtenidos con el equipo, a fin de dar solución a un determinado fallo en los automóviles de marcas ya señaladas.

RECOMENDACIONES

- Antes de desarrollar el proyecto se debe tener un conocimiento medio-avanzado sobre las señales que generan los sensores del vehículo para poder interpretarlas correctamente, ya que durante el desarrollo del mismo se tuvieron varios inconvenientes al intentar hacer que la unidad electrónica del automóvil reconozca las señales generadas para simular los sensores.

- La visualización en paralelo de las señales generadas y la lectura de la unidad electrónica de control no es exacta en todos los casos, ya que por el ajuste de curvas realizado, el ingreso en la variación de las señales únicamente no se visualiza, pero si se genera. (Ejm ECT o IAC en lo que a simulación de sensor se refiere).

- Al conectar el equipo a cualquiera de los sensores o actuadores, se debe tener precaución y revisar el diagrama de conexión tanto del equipo como del sensor o actuador, con el afán de evita averías en las entradas de la unidad electrónica de control, en el sensor, en el actuador o en el mismo equipo de diagnóstico.

- El equipo desarrollado, deja abierta la opción de poder ser modificado para ampliar el número de sensores para ser simulados, diagnosticados o implementar mayor número de actuadores a ser comprobados.

BIBLIOGRAFIA

Referencias Electrónicas

- WIKIPEDIA. OBD2 PID's [en línea] [Consulta: 25 de Octubre de 2013] Disponible en: http://en.wikipedia.org/wiki/OBD-II_PIDs
- ARDUINO. Arduino Mega 2560 [en línea] [Consulta: 15 de septiembre de 2013] Disponible en: http://arduino.cc/en/Main/arduinoBoardMega2560
- STRAIGHT TALK AUTOMOTIVE. ELMSCAN5 [en línea] [Consulta: 8 de Noviembre de 2013] Disponible en: http://straighttalkautomotive.com/articles/elmscan-5-compact-review.html.
- TU TALLER MECANICO. Sensores y Actuadores [en línea] [Consulta: 3 de Agosto 2013] Disponible en: http://www.slideshare.net/242013/sensores-del-automovil-24436791
- OBD2 RESOURCE. OBD2 PID's [en línea] [Consulta: 28 de Octubre de 2013] Disponible en: http://obdcon.sourceforge.net/2010/06/obd-ii-pids/
- OUTILS OBD FACILE, Automotive electronic diagnostic [en línea] [Consulta: 15 de Noviembre de 2013] Disponible en: http://www.outilsobdfacile.com/obd-mode-pid.php

ANEXOS

ANEXO 1: Hoja de comando del ELM327

ANEXO 2: Firmware completo del equipo de diagnóstico

```
#include <avr/io.h>
#include <avr/interrupt.h>
#include <LiquidCrystal.h>
#include <SPI.h>
byte flecha[8] = {
  0b01000,
  0b01100,
  0b01110,
  0b01111,
  0b01110,
  0b01100,
  0b01000,
  0b00000
};
byte blanco[8] = {
  0b00000,
  0b00000,
  0b00000,
  0b00000,
  0b00000,
  0b00000,
  0b00000,
  0b00000
};

int sensorPin = A8;            // Selecciona el pin de entrada analogica para medir sensores
const int PinPWM1 = 2;
const int PinPWM2 = 3;
const int VVSS = 22;            //Pin de entrada VSS, detecta si hay Voltaje 12V
const int GVSS = 23;            //Pin de entrada VSS, detecta si hay Negativo GND
const int VTPS = 24;            //Pin de entrada TPS, detecta si hay Voltaje 5V
const int GTPS = 33;            //Pin de entrada TPS, detecta si hay Negativo Gnd
const int PSENS = 26;            //Pin de entrada, detecta pulsos provenientes del sensor(Pruebas)
const int PACTU = 27;            // Pin de salida, para prueba de actuadores.


const int SVSS = 40;            //Pin de salida para senal VSS
const int slaveSelectPin = 53;
int numRows    =0;int numCols   =0;int numRows1  =0;int numCols1  =-1;int numRows0  =0;int numCols0  =0;
long val       =0;              // variable to read the value from the analog pin
int val1       =0;             // variable to read the value from the analog pin
int val2       =0;
int val3       =0;
int cartel     =0;           // variable para el tipo de cartel
int DatoRx     =0;           // variable en donde se almacena los datos provenientes la comunicacion Serial.
int DatoUtil   =0;
int Dato       =0;           //variable de respado para el calculo de los PID
int Dato1      =0;           //variable de respado para el calculo de los PID
int Dato2      =0;           //variable de respado para el calculo de los PID
int cont       =0;
int diag       =0;          //
float LHO2S    =0;
float Data     =0;
int estadoVVSS  =0;int estadoGVSS=0;int TVSS=100;int LVSS=0;int valVss=0;
int estadoVTPS  =0;int estadoGTPS=0;int STPS=255;int LTPS=0;int valTPS=0;
int valCMP      =0;int valTEMP=0;int valMAP=0;
int TSensor     =0;       //variable que permite seleccionar que tipo de sensor elegido.
int PosDatoSerial = 0;    //Variable q permite seleccionar el dato util proveniente de la comunicion serial: > FF FF XX XX >
```

```
int ini =0;
int x   =0;

// initialize the library with the numbers of the interface pins
LiquidCrystal lcd(8, 9, 4, 5, 6, 7);
int Sen[ ] = {

127,130,133,136,139,143,146,149,152,155,158,161,164,167,170,173,176,178,181,184,187,190,192,195,198,200,203,205,208,2
10,212,215,217,219,221,223,225,227,229,231,233,234,236,238,239,240,

242,243,244,245,247,248,249,249,250,251,252,252,253,253,253,254,254,254,254,254,254,254,253,253,253,252,252,251,250,2
49,249,248,247,245,244,243,242,240,239,238,236,234,233,231,229,227,225,223,

221,219,217,215,212,210,208,205,203,200,198,195,192,190,187,184,181,178,176,173,170,167,164,161,158,155,152,149,146,1
43,139,136,133,130,127,124,121,118,115,111,108,105,102,99,96,93,90,87,84,81,78,

76,73,70,67,64,62,59,56,54,51,49,46,44,42,39,37,35,33,31,29,27,25,23,21,20,18,16,15,14,12,11,10,9,7,6,5,5,4,3,2,2,1,1,1,0,0,0,
0,0,0,0,1,1,1,2,2,3,4,5,5,6,7,9,10,11,12,14,15,16,18,20,21,23,25,27,29,31,
  33,35,37,39,42,44,46,49,51,54,56,59,62,64,67,70,73,76,78,81,84,87,90,93,96,99,102,105,108,111,115,118,121,124

};
int adc_key_val[5] = {50, 200, 400, 600, 800 };
int NUM_KEYS = 5;
int adc_key_in;
int key=-1;
int oldkey=-1;

void setup()
{
  pinMode(22, OUTPUT);          // Control rele #1
  pinMode(23, OUTPUT);          // Control rele #2
  pinMode(24, INPUT_PULLUP);
  pinMode(33, INPUT_PULLUP);

  pinMode(26, INPUT_PULLUP);
  pinMode(27, OUTPUT);          // Control C, para probar, Inyector, Bobina, PCV.
  pinMode(28, OUTPUT);          // Control rele #3, Voltaje de salidan 12V
  pinMode(29, OUTPUT);          // Control rele doble #4, Selecciona para probar ISA o Inyec-Bobina-PCV
  pinMode(30, OUTPUT);          // Control rele doble #5, Voltaje de salida 5V,
  pinMode(31, OUTPUT);          // Control rele #6
  pinMode(35, OUTPUT);          // Control rele #7
  pinMode(PinPWM1, OUTPUT);     // Control PWM Simular sensor CKP
  pinMode(PinPWM2, OUTPUT);     // Control PWM Porbar ISA
  pinMode(40, OUTPUT);

  digitalWrite(22, LOW);
  digitalWrite(23, LOW);
  digitalWrite(27, LOW);
  digitalWrite(28, LOW);
  digitalWrite(29, LOW);
  digitalWrite(30, LOW);
  digitalWrite(31, LOW);
  digitalWrite(35, LOW);
  digitalWrite(40, HIGH);

  // set the slaveSelectPin as an output:
  pinMode (slaveSelectPin, OUTPUT);
  // initialize SPI:

  SPI.begin();
  lcd.createChar(1, flecha);
  lcd.createChar(2, blanco);
  //pinMode(PinPWM1, OUTPUT);   // sets the pin as output
  lcd.begin(20, 4);
  lcd.setCursor(7, 0);lcd.write("GTEC");
  lcd.setCursor(0, 2);lcd.write("  KIA    HIUNDAY");
```

```
 lcd.setCursor(13, 3);lcd.write("Ver 1.0");
 // initialize the serial communications:
 Serial1.begin(115200);
 delay(1000);
 lcd.clear();
 CartelLCD();cursorLCD();
}

int get_key(unsigned int input)
{
   int k;
   for (k = 0; k < NUM_KEYS; k++)
   {
     if (input < adc_key_val[k])
     {
       return k;
     }
   }
   if (k >= NUM_KEYS)k = -1;  // Error en la lectura
   return k;
 }

void loop()
{

 selecCursor0();
 ////////////////////////////////////////////////////////////////////////////
 //Pulsantes leidos

 adc_key_in = analogRead(0);          // Leemos el valor de la pulsacion
 key = get_key(adc_key_in);           // Obtenemos el boton pulsado

 if (key != oldkey)                   // if keypress is detected
 {
   delay(100);                        // Espera para evitar los rebotes de las pulsaciones
   adc_key_in = analogRead(0);        // Leemos el valor de la pulsacion
   key = get_key(adc_key_in);         // Obtenemos el boton pulsado

 if (key==4)
 {
   delay(200);
   if (numCols==0)
   {
     cartel=1;
     CartelLCD();
     encero();
     cursorLCD();
     while (cartel==1)//////////////////////Cartel: > Simular sensor
     {              ////////////////////////          Probar sensor
        adc_key_in = analogRead(0);
        key = get_key(adc_key_in);
        if ( key == 0)
        {
          cartel=0;
          CartelLCD();
          encero();
          cursorLCD();
          delay(200);
        }
        selecCursor1();
        if (key==4)
        {
          delay(200);
          if (numCols==0)
          {
            cartel=2;
```

```
CartelLCD();
cursorLCD();
while (cartel==2)//////////////////Cartel 2:  >Vss   IAT
{              ////////////////////          TPS   KS
  adc_key_in = analogRead(0); //////           CKP   MAP
  key = get_key(adc_key_in);
  if ( key == 0)
  {
    cartel=1;
    CartelLCD();
    encero();
    cursorLCD();
    delay(200);
  }
  selecCursor();
  if (key == 4)
  {
    delay(200);
    if (numCols==0)
    {
      cartel = 3;
      CartelLCD();
      while (cartel==3)//////////////////////////Cartel 3: Vss    12v  S  GND
      {              ////////////////////////          INICIAR    ATRAS
        adc_key_in = analogRead(0);
        key = get_key(adc_key_in);
        if ( key == 0)
        {
          cartel=2;
          CartelLCD();
          cursorLCD();
          delay(200);
        }

        if (key==4)
        {
          cartel=4;
          CartelLCD();
          while (cartel==4)//////////////////Cartel 4:  Simulando.....
          {
            adc_key_in = analogRead(0);
            key = get_key(adc_key_in);
            if ( key == 0)
            {
              cartel=3;
              CartelLCD();
              delay(200);
            }

            ////////////////////////////////////////////
            /////////programa para simular vss/////////////
            if (cartel==4)
               {
                 delay(200);
                 estadoVTPS = digitalRead(VTPS);
                 if (estadoVTPS == HIGH)
                 {
                   lcd.setCursor(0, 2);lcd.print("B+  -> Error ---X---");
                   delay(500);
                   lcd.setCursor(0, 2);lcd.print("                    ");
                 }
                 else
                 {
                   delay(700);
                   lcd.setCursor(0, 2);lcd.print("B+  -> Ok           ");
                 }
```

```
                    estadoGTPS = digitalRead(GTPS);
                    if (estadoGTPS == HIGH)
                    {
                     lcd.setCursor(0, 3);lcd.print("GND -> Error ---X---");
                     delay(500);
                     lcd.setCursor(0, 3);lcd.print("                    ");
                    }
                    else
                    {
                     delay(700);
                     lcd.setCursor(0, 3);lcd.print("GND -> Ok          ");
                     delay(700);
                    }
                    if (estadoVTPS==LOW )
                    {
                     if (estadoGTPS==LOW)
                     {
                       digitalWrite(22, HIGH);
                       cartel=12;CartelLCD();
                       lcd.setCursor(0, 2);lcd.print("Simulado:");
                       lcd.setCursor(0, 3);lcd.print("ECU:");
                       TSensor=0;LVSS=11;
                       ////////////////////////////////////////////////////
                       /////////Configuracion de interrupciones Timer5//////
                       cli();
                       TCCR5A=0;
                       TCCR5B=0;
                       OCR5A=1000;
                       TCCR5B |= (1<<WGM12);
                       TCCR5B |= (1<<CS10);
                       TCCR5B |= (1<<CS12);
                       TIMSK5=(1<<OCIE1A);
                       sei();
                       ////////////////////////////////////////////////////
                       ////////////////////////////////////////////////////
                       Serial1.write("ATSP6\r"); ////ISO 15765-4 CAN 11/500
                       while(cartel==12)
                       {
                          adc_key_in = analogRead(0);
                          key = get_key(adc_key_in);
                          if(key==1)
                          {
                           if (OCR5A==100)
                           {
                           }
                           else
                           {
                            OCR5A=OCR5A-10;
                            LVSS=((0.000000001393*OCR5A*OCR5A*OCR5A*OCR5A)-
(0.000003561*OCR5A*OCR5A*OCR5A)+(0.003258*OCR5A*OCR5A)-(1.288*OCR5A)+210.3);
                            lcd.setCursor(10, 2);lcd.print("    Km/h");
                           }
                          }
                          if(key==2)
                          {
                           if (OCR5A==1000)
                           {
                           }
                           else
                           {
                            OCR5A=OCR5A+10;
                            LVSS=((0.000000001393*OCR5A*OCR5A*OCR5A*OCR5A)-
(0.000003561*OCR5A*OCR5A*OCR5A)+(0.003258*OCR5A*OCR5A)-(1.288*OCR5A)+210.3);
                            lcd.setCursor(10, 2);lcd.print("    Km/h");
                           }
                          }
```

```
                    if (cartel==3)
                    {
                     digitalWrite(SVSS, HIGH);
                    }
                    else
                    {
                     lcd.setCursor(10, 2);lcd.print(LVSS);
                    }
                    //////////////VSS OBDII/////////////////
                    delay(250);
                    Serial1.write("010D\r");
                    SerialOBDII();
                    ////////////////////////////////////
                    if ( key == 0)
                    {
                     digitalWrite(22, LOW);
                     cartel=3;CartelLCD();TVSS=1000;
                     TSensor=0;Dato=0;PosDatoSerial=0;valVss=0;Dato1;Dato2=0;
                     ////////Fin de interrupcion///////
                     cli();
                     TCCR5A=0;
                     TCCR5B=0;
                     sei();
                     ////////////////////////////////
                    }
                  }
                 }
                }
               }
              }
             }
            }
           }
          }
         }
        }
       }
      }
     }
    }
   }
  }
 }
}
if (numCols==1)
{
  cartel = 5;CartelLCD();
  while (cartel==5)//////////////////////////Cartel 5: TPS    12v  S  GND
  {              ////////////////////////          INICIAR  ATRAS
    adc_key_in = analogRead(0);
    key = get_key(adc_key_in);
    if ( key == 0)
    {
      cartel=2;CartelLCD();cursorLCD();
      delay(200);
    }
    if (key==4)
    {
     cartel=4;CartelLCD();
     while (cartel==4)//////////////////Cartel 4:  Simulando.....
     {
        adc_key_in = analogRead(0);
        key = get_key(adc_key_in);
        if ( key == 0)
        {
          cartel=5;CartelLCD();
          delay(200);
        }
        ////////////////////////////////////////////////////////
        ///////////////PROGRAMA PARA SIMULAR SENSOR TPS//////////////
        if (cartel==4)
        {
          delay(200);
          estadoVTPS = digitalRead(VTPS);
          if (estadoVTPS == HIGH)
          {
            lcd.setCursor(0, 2);lcd.print("5v  -> Error ---X---");
```

```
  delay(500);
  lcd.setCursor(0, 2); lcd.print("                    ");
}
else
{
  delay(700);
  lcd.setCursor(0, 2);lcd.print("5v  -> Ok           ");
}

estadoGTPS = digitalRead(GTPS);
if (estadoGTPS == HIGH)
{
  lcd.setCursor(0, 3);lcd.print("GND -> Error ---X---");
  delay(500);
  lcd.setCursor(0, 3);lcd.print("                    ");
}
else
{
  delay(700);
  lcd.setCursor(0, 3);lcd.print("GND -> Ok           ");
  delay(700);
}
if (estadoVTPS==LOW )
{
  if (estadoGTPS==LOW)
  {
    cartel=12;CartelLCD();
    lcd.setCursor(0, 2);lcd.print("Simulado:");
    lcd.setCursor(0, 3);lcd.print("ECU:");
    TSensor=1;                          ///Selecciona tipo de sensor
    Serial1.write("ATSP6\r");                   ////ISO 15765-4 CAN 11/500

    while(cartel==12)
    {
       adc_key_in = analogRead(0);
       key = get_key(adc_key_in);
       if(key==1)
       {
         if ( STPS == 0)
         {
         }
         else
         {
           delay(10);
           STPS= STPS-1;
           LTPS=(((255-STPS+1)*100)/255);
           //LTPS=LTPS+1;
           lcd.setCursor(10, 2);lcd.print("   %");
         }
       }
       if(key==2)
       {
         if (STPS==255)
         {
         }
         else
         {
           delay(10);
           STPS= STPS+1;
           LTPS=(((255-STPS-1)*100)/255);
           //STPS=STPS+3;
           //LTPS=LTPS-1;
           lcd.setCursor(10, 2);lcd.print("   %");
         }
       }
       if (cartel==5)
```

```
                    {
                      digitalPotWrite();
                    }
                    else
                    {
                      digitalPotWrite();
                      lcd.setCursor(10, 2);lcd.print(LTPS);
                    }

                    ////////////////////TPS OBDII////////////////////
                    delay(250);
                    Serial1.write("0111\r");///PID 0111 -->PETICION TPS
                    SerialOBDII();
                    ////////////////////////////////////////////////
                    if ( key == 0)
                    {
                      cartel=5;CartelLCD();STPS=255;
                      cont=0;ini=0;TSensor=0;Dato=0;PosDatoSerial=0;valTPS=0;LTPS=0;
                      delay(200);
                    }
                  }
                 }
                }
               }
               ////////////FIN PROGRAMA PARA SIMULAR SENSOR TPS///////////////
              ////////////////////////////////////////////////////////////
             }
            }
           }
}

if (numCols==2)
{
  cartel = 6;CartelLCD();
  while (cartel==6)//////////////////////////Cartel 6: CKP    GND  S  B+
  {              //////////////////////////          INICIO      ATRAS
    adc_key_in = analogRead(0);
    key = get_key(adc_key_in);
    if ( key == 0)
    {
      cartel=2;CartelLCD();cursorLCD();
      delay(200);
    }
    if (key==4)
    {
     cartel=4;CartelLCD();
     while (cartel==4)//////////////////Cartel 4:  Simulando.....
     {
        adc_key_in = analogRead(0);
        key = get_key(adc_key_in);
        if ( key == 0)
        {
          cartel=6;CartelLCD();
          delay(200);
        }
        ////////////////////////////////////////////////////////
        ///////////////PROGRAMA PARA SIMULAR SENSOR CKP///////////////
        if (cartel==4)
        {

            digitalWrite(22, HIGH);
            digitalWrite(35, HIGH);
            cartel=12;CartelLCD();
            lcd.setCursor(0, 2);lcd.print("Simulado:");
            lcd.setCursor(0, 3);lcd.print("ECU:");
            TSensor=2;                                ////Selecciona tipo de sensor
```

```
                ////////////////////////////////////////////////
                /////////Configuracion de interrupciones Timer5//////
                cli();
                TCCR5A=0;
                TCCR5B=0;
                OCR5A=1000;
                TCCR5B |= (1<<WGM12);
                TCCR5B |= (1<<CS10);
                TCCR5B |= (1<<CS12);
                TIMSK5=(1<<OCIE1A);
                sei();
                ////////////////////////////////////////////////
                ////////////////////////////////////////////////
                Serial1.write("ATSP6\r");                       ////ISO 15765-4 CAN 11/500

                while(cartel==12)
                {
                   adc_key_in = analogRead(0);
                   key = get_key(adc_key_in);


                   ////////////////CMP OBDII////////////////////////////
                   delay(250);
                   Serial1.write("010C\r");///PID Peticion CMP
                   SerialOBDII();
                   ////////////////////////////////////////////////////
                   if ( key == 0)
                   {
                    digitalWrite(22, LOW);
                    digitalWrite(35, LOW);
                    cartel=6;CartelLCD();TVSS=1000;
                    TSensor=0;Dato=0;Dato2=0;Dato1=0;PosDatoSerial=0;
                    analogWrite(PinPWM1, 0);

                    ///////////Fin de interrupcion Timer 5=OFF//////////
                    cli();
                    TCCR5A=0;
                    TCCR5B=0;
                    sei();
                    ////////////////////////////////////////////////
                   }




                }
            }
            ///////////////FIN DE PROGRAMA PARA SIMULAR CMP//////////////////
            ////////////////////////////////////////////////////////////////
        }
      }
    }
}

if (numCols==3)
{
  cartel = 7;CartelLCD();
  while (cartel==7)//////////////////////////Cartel 7: ECT    GND  MEDIDOR  S
  {             ////////////////////////          INICIO       ATRAS
    adc_key_in = analogRead(0);
    key = get_key(adc_key_in);
    if ( key == 0)
    {
      cartel=2;CartelLCD();cursorLCD();
      delay(200);
     }
```

```
if (key==4)
{
 cartel=4;CartelLCD();
 while (cartel==4)//////////////////Cartel 4:  Simulando.....
 {
    adc_key_in = analogRead(0);
    key = get_key(adc_key_in);
    if ( key == 0)
    {
     cartel=7;CartelLCD();
     delay(200);
    }
    ////////////////////////////////////////////////////////////////////
    ////////////////PROGRAMA PARA SIMULACION DEL SENSOR ECT//////////////////
    if (cartel==4)
    {
     delay(100);
     estadoGTPS = digitalRead(GTPS);
     if (estadoGTPS == HIGH)
     {
      lcd.setCursor(0, 3);lcd.print("GND -> Error ---X---");
      delay(500);
      lcd.setCursor(0, 3);lcd.print("                    ");
     }
     else
     {
      delay(700);
      lcd.setCursor(0, 3);lcd.print("GND -> Ok           ");
      delay(800);
     }

      if (estadoGTPS==LOW)
      {
       digitalWrite(23, HIGH);
       cartel=12;CartelLCD();
       lcd.setCursor(0, 2);lcd.print("Simulado:");
       lcd.setCursor(0, 3);lcd.print("ECU:");

       TSensor=3;STPS=96;
       Serial1.write("ATSP6\r");                    ////ISO 15765-4 CAN 11/500

       while(cartel==12)
       {
          adc_key_in = analogRead(0);
          key = get_key(adc_key_in);
          if(key==1)
          {
           if ( STPS == 255)
           {
           }
           else
           {
            delay(20);
            LTPS=LTPS+1;
            STPS= ((0.000000024067*LTPS*LTPS*LTPS*LTPS*LTPS) -
(0.000010094*LTPS*LTPS*LTPS*LTPS) + (0.0016854*LTPS*LTPS*LTPS) - (0.14494*LTPS*LTPS) +6.8711*LTPS +
98.256);
            lcd.setCursor(10, 2);lcd.print("   C");
           }
          }
          if(key==2)
          {
           if (STPS==96)
           {
           }
           else
```

```
                                {
                                 delay(30);
                                 LTPS=LTPS-1;
                                 STPS= ((0.000000024067*LTPS*LTPS*LTPS*LTPS*LTPS) -
(0.000010094*LTPS*LTPS*LTPS*LTPS) + (0.0016854*LTPS*LTPS*LTPS) - (0.14494*LTPS*LTPS) +6.8711*LTPS +
98.256);
                                 lcd.setCursor(10, 2);lcd.print("   C");
                                }
                               }
                               if (cartel==7)
                               {
                                digitalPotWrite();
                               }
                               else
                               {
                                digitalPotWrite();
                                //LTPS= (((STPS*80)/249)-30);
                                lcd.setCursor(10, 2);lcd.print(LTPS);
                               }

                               ////////////////////ECT OBDII////////////////////////////
                               delay(250);
                               Serial1.write("0105\r");  ///PID peticion ECT
                               SerialOBDII();
                               ////////////////////////////////////////////////////////
                               if ( key == 0)
                               {
                                cartel=7;CartelLCD();STPS=96;
                                cont=0;ini=0;TSensor=0;Dato=0;Dato1=0;Dato2=0;PosDatoSerial=0;valTEMP=0;LTPS=0;
                                digitalWrite(23, LOW);
                                delay(200);
                               }
                              }
                             }
                            }
                           ////////////////////FIN DE PROGRAMA PARA SIMULAR SENSOR ECT//////////////////
                            ////////////////////////////////////////////////////////////////////////
                          }
                         }
                        }
                       }

                       if (numCols==6)
                       {
                         cartel = 8;CartelLCD();
                         while (cartel==8)/////////////////////////Cartel 8:  IAT
                         {              ////////////////////////              INICIO    ATRAS
                           adc_key_in = analogRead(0);
                           key = get_key(adc_key_in);
                           if ( key == 0)
                           {
                             cartel=2;CartelLCD();cursorLCD();
                             delay(200);
                           }
                           if (key==4)
                           {
                            cartel=4;CartelLCD();
                            while (cartel==4)//////////////////Cartel 4:  Simulando.....
                            {
                               adc_key_in = analogRead(0);
                               key = get_key(adc_key_in);
                               if ( key == 0)
                               {
                                 cartel=8;CartelLCD();
                                 delay(200);
                               }
```

```
                    ////////////////////////////////////////////////////////////////
                    ////////////////////PROGRAMA PARA SIMULAR SENSOR IAT///////////////
                    if (cartel==4)
                    {
                     delay(200);
                     estadoGTPS = digitalRead(GTPS);
                     if (estadoGTPS == HIGH)
                     {
                      lcd.setCursor(0, 3);lcd.print("GND -> Error ---X---");
                      delay(500);
                      lcd.setCursor(0, 3);lcd.print("                    ");
                     }
                     else
                     {
                      delay(700);
                      lcd.setCursor(0, 3);lcd.print("GND -> Ok           ");
                      delay(800);
                     }

                      if (estadoGTPS==LOW)
                       {
                        digitalWrite(23, HIGH);
                        cartel=12;
                        CartelLCD();
                        lcd.setCursor(0, 2);lcd.print("Simulado:");
                        lcd.setCursor(0, 3);lcd.print("ECU:");

                        TSensor=3;STPS=96;
                        Serial1.write("ATSP6\r"); ////ISO 15765-4 CAN 11/500

                        while(cartel==12)
                        {
                           adc_key_in = analogRead(0);
                           key = get_key(adc_key_in);
                           if(key==1)
                           {
                            if ( STPS == 255)
                            {
                            }
                            else
                            {
                             delay(20);
                             LTPS=LTPS+1;
                             STPS= ((0.000000024067*LTPS*LTPS*LTPS*LTPS*LTPS) -
(0.000010094*LTPS*LTPS*LTPS*LTPS) + (0.0016854*LTPS*LTPS*LTPS) - (0.14494*LTPS*LTPS) +6.8711*LTPS +
98.256);
                              lcd.setCursor(10, 2);lcd.print("   C");
                            }
                           }
                           if(key==2)
                           {
                            if (STPS==96)
                            {
                            }
                            else
                            {
                             delay(30);
                             LTPS=LTPS-1;
                             STPS= ((0.000000024067*LTPS*LTPS*LTPS*LTPS*LTPS) -
(0.000010094*LTPS*LTPS*LTPS*LTPS) + (0.0016854*LTPS*LTPS*LTPS) - (0.14494*LTPS*LTPS) +6.8711*LTPS +
98.256);
                              lcd.setCursor(10, 2);lcd.print("   C");
                            }
                           }
                           if (cartel==8)
                           {
```

```
                digitalPotWrite();
                }
                else
                {
                 digitalPotWrite();
                 lcd.setCursor(10, 2);lcd.print(LTPS);
                }

                ////////////////////////IAT OBDII////////////////////////////////
                delay(250);
                Serial1.write("010F\r");///PID peticion IAT
                SerialOBDII();
                ////////////////////////////////////////////////////////////////
                if ( key == 0)
                {
                 cartel=8;CartelLCD();STPS=96;
                 cont=0;ini=0;TSensor=0;Dato=0;Dato1=0;Dato2=0;PosDatoSerial=0;LTPS=0;
                 digitalWrite(23, LOW);
                 delay(200);
                }
              }
             }
           }
           ////////////////////FIN DE PROGRAMA PARA SIMULAR IAT//////////////////////
           ////////////////////////////////////////////////////////////////////////
       }
      }
    }
}

if (numCols==7)
{
  cartel = 9;CartelLCD();
  while (cartel==9)//////////////////////////Cartel 9:   MAF  IAT G MAF B+ G
  {              ////////////////////////          INICIO    ATRAS
    adc_key_in = analogRead(0);
    key = get_key(adc_key_in);
    if ( key == 0)
    {
      cartel=2;CartelLCD();cursorLCD();
      delay(200);
    }
    if (key==4)
    {
     cartel=4;CartelLCD();
     while (cartel==4)//////////////////Cartel 4:  Simulando.....
     {
        adc_key_in = analogRead(0);
        key = get_key(adc_key_in);
        if ( key == 0)
        {
         cartel=9;CartelLCD();
         delay(200);
        }
        ////////////////////////////////////////////////////////////////
        ////////////////////PROGRAMA PARA SIMULAR SENSOR MAF/////////////////
         if (cartel==4)
        {
         delay(200);
         estadoVVSS = digitalRead(VVSS);
         if (estadoVVSS == HIGH)
         {
           lcd.setCursor(0, 2); lcd.print("B+  -> Error ---X---");
           delay(500);
           lcd.setCursor(0, 2); lcd.print("                    ");
         }
```

```
else
{
 delay(700);
 lcd.setCursor(0, 2);  lcd.print("B+  -> Ok           ");
}

estadoGVSS = digitalRead(GVSS);
if (estadoGVSS == HIGH)
{
 lcd.setCursor(0, 3); lcd.print("GND -> Error ---X---");
 delay(500);
 lcd.setCursor(0, 3); lcd.print("                    ");
}
else
{
 delay(700);
 lcd.setCursor(0, 3); lcd.print("GND -> Ok          ");
 delay(700);
}

if (estadoVVSS==LOW )
{
 if (estadoGVSS==LOW)
 {
  cartel=12;CartelLCD();
  lcd.setCursor(0, 2);lcd.print("Simulado:");
  lcd.setCursor(0, 3);lcd.print("ECU:");
  TSensor=4;                          ///Selecciona tipo de sensor
  Serial1.write("ATSP6\r");                ////ISO 15765-4 CAN 11/500

  while(cartel==12)
  {
    adc_key_in = analogRead(0);
    key = get_key(adc_key_in);
    if(key==1)
    {
     if ( STPS == 0)
     {
     }
     else
     {
      delay(20);
      STPS=STPS-2;
      LTPS=LTPS+1;
      lcd.setCursor(10, 2);lcd.print("   grams/s");
     }
    }
    if(key==2)
    {
     if (STPS==255)
     {
     }
     else
     {
      delay(30);
      STPS=STPS+2;
      LTPS=LTPS-1;
      lcd.setCursor(10, 2);lcd.print("   grams/s");
     }
    }
    if (cartel==5)
    {
     digitalPotWrite();
    }
    else
    {
```

```
                digitalPotWrite();
                lcd.setCursor(10, 2);lcd.print(LTPS);
               }

               ////////////////////MAF OBDII////////////////////
               delay(250);
               Serial1.write("0110\r");///PID 011O -->PETICION MAF
               SerialOBDII();
               ////////////////////////////////////////////////
               if ( key == 0)
               {
                cartel=9;CartelLCD();STPS=255;
                cont=0;ini=0;TSensor=0;Dato=0;PosDatoSerial=0;valTPS=0;LTPS=0;
                delay(200);
               }
              }
             }
            }
           }

          ////////////////////FIN DE PROGRAMA SIMULAR SENSOR MAF////////////////
          ////////////////////////////////////////////////////////////////////
      }
     }
   }
}

if (numCols==8)
{
  cartel = 10;CartelLCD();
  while (cartel==10)//////////////////////////Cartel 10:  MAP  GND -- 5V   S
  {             ////////////////////////        INICIO    ATRAS
    adc_key_in = analogRead(0);
    key = get_key(adc_key_in);
    if ( key == 0)
    {
      cartel=2;CartelLCD();cursorLCD();
      delay(200);
    }
    if (key==4)
    {
     cartel=4;CartelLCD();
     while (cartel==4)//////////////////Cartel 4:  Simulando.....
     {
        adc_key_in = analogRead(0);
        key = get_key(adc_key_in);
        if ( key == 0)
        {
          cartel=10;CartelLCD();
          delay(200);
        }
        ////////////////////////////////////////////////////////////////////
        ////////////////////PROGRAMA PARA SIMULAR SENSOR MAP/////////////////
        if (cartel==4)
        {
          delay(200);
          estadoVTPS = digitalRead(VTPS);
          if (estadoVTPS == HIGH)
          {
            lcd.setCursor(0, 2);lcd.print("5v  -> Error ---X---");
            delay(500);
            lcd.setCursor(0, 2); lcd.print("                    ");
          }
          else
          {
            delay(700);
```

```
 lcd.setCursor(0, 2);lcd.print("5v  -> Ok           ");
}

estadoGTPS = digitalRead(GTPS);
if (estadoGTPS == HIGH)
{
 lcd.setCursor(0, 3);lcd.print("GND -> Error ---X---");
 delay(500);
 lcd.setCursor(0, 3);lcd.print("                    ");
}
else
{
 delay(700);
 lcd.setCursor(0, 3);lcd.print("GND -> Ok           ");
 delay(700);
}

if (estadoVTPS==LOW )
{
 if (estadoGTPS==LOW)
 {
  cartel=12;
  CartelLCD();
  lcd.setCursor(0, 2);lcd.print("Simulado:");
  lcd.setCursor(0, 3);lcd.print("ECU:");

  TSensor=5;STPS=253;LTPS=1;
  Serial1.write("ATSP6\r"); ////ISO 15765-4 CAN 11/500

  while(cartel==12)
  {
    adc_key_in = analogRead(0);
    key = get_key(adc_key_in);
    if(key==1)
    {
     if ( STPS == 0)
     {
     }
     else
     {
      delay(20);
      STPS=STPS-2;
      LTPS=LTPS+1;
      lcd.setCursor(10, 2);lcd.print("   kPa");
     }
    }
    if(key==2)
    {
     if (STPS==253)
     {
     }
     else
     {
      delay(30);
      STPS=STPS+2;
      LTPS=LTPS-1;
      lcd.setCursor(10, 2);lcd.print("   kPa");
     }
    }
    if (cartel==5)
    {
     digitalPotWrite();
    }
    else
    {
     digitalPotWrite();
```

```
                    lcd.setCursor(10, 2);lcd.print(LTPS);
                   }

                   ////////////////////MAP OBDII////////////////////
                   delay(250);
                   Serial1.write("010B\r");///PID 0111 -->PETICION MAP
                   SerialOBDII();
                   ////////////////////////////////////////////////
                   if ( key == 0)
                   {
                    cartel=10;CartelLCD();
                    cont=0;ini=0;TSensor=0;Dato=0;Dato1=0;Dato2=0;PosDatoSerial=0;LTPS=0;
                    delay(200);
                   }
                 }
               }
             }
           }
           ////////////////////FIN DE PROGRAMA SIMULAR SENSOR MAP////////////////
           ////////////////////////////////////////////////////////////////////
       }
      }
    }
}

if (numCols==9)
{
  cartel = 11;CartelLCD();
  while (cartel==11)//////////////////////////Cartel 11:  HO2S  GND -- 5V   S
  {              ////////////////////////              INICIO    ATRAS
    adc_key_in = analogRead(0);
    key = get_key(adc_key_in);
    if ( key == 0)
    {
      cartel=2;CartelLCD();cursorLCD();
      delay(200);
    }
    if (key==4)
    {
     cartel=4;CartelLCD();
     while (cartel==4)//////////////////Cartel 4:  Simulando.....
     {
        adc_key_in = analogRead(0);
        key = get_key(adc_key_in);
        if ( key == 0)
        {
          cartel=11;CartelLCD();
          delay(200);
        }
        ////////////////////////////////////////////////////////////////////
        ////////////////////PROGRAMA PARA SIMULAR SENSOR HO2S/////////////////
        if (cartel==4)
        {
          delay(200);
          estadoGTPS = digitalRead(GTPS);
          if (estadoGTPS == HIGH)
          {
            lcd.setCursor(0, 3);lcd.print("GND -> Error ---X---");
            delay(500);
            lcd.setCursor(0, 3);lcd.print("                    ");
          }
          else
          {
            delay(700);
            lcd.setCursor(0, 3);lcd.print("GND -> Ok           ");
            delay(800);
```

```
}

 if (estadoGTPS==LOW)
 {
  cartel=12;CartelLCD();
  lcd.setCursor(0, 2);lcd.print("Simulado:");
  lcd.setCursor(0, 3);lcd.print("ECU:");

  TSensor=6;STPS=95;LHO2S= 0.51;
  Serial1.write("ATSP6\r"); ////ISO 15765-4 CAN 11/500

  while(cartel==12)
  {
     adc_key_in = analogRead(0);
     key = get_key(adc_key_in);
     if(key==1)
     {
      if ( STPS == 62)
      {
      }
      else
      {
       delay(20);

       STPS=STPS-1;
       LHO2S=(-0.019116*STPS+2.3349);
       lcd.setCursor(10, 2);lcd.print("     v");
      }
     }
     if(key==2)
     {
      if (STPS==114)
      {
      }
      else
      {
       delay(30);
       STPS=STPS+1;
       LHO2S=(-0.019116*STPS+2.3349);
       lcd.setCursor(10, 2);lcd.print("     v");
      }
     }
     if (cartel==11)
     {
      digitalPotWrite();
     }
     else
     {
      digitalPotWrite();
      lcd.setCursor(10, 2);lcd.print(LHO2S);
     }

     ////////////////////////HO2S OBDII//////////////////
     delay(250);
     Serial1.write("0114\r");///PID peticion HO2S
     SerialOBDII();
     ////////////////////////////////////////////////

     if ( key == 0)
     {
      cartel=11;CartelLCD();
      cont=0;ini=0;TSensor=0;Dato=0;Dato1=0;Dato2=0;PosDatoSerial=0;LTPS=0;
      delay(200);
     }
  }
 }
```

```
                    }

                    ////////////////////FIN DE PROGRAMA SIMULAR SENSOR HO2S////////////////
                    ////////////////////////////////////////////////////////////////////
                  }
                 }
                }
               }
              }
             }
            }
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////PROBAR SENSOR//////////////////////////////////////////////////////////////////
    if (numCols==1)
      {
       cartel=2;CartelLCD();encero();cursorLCD();
        lcd.setCursor(3, 2);lcd.write("CMP");
       while (cartel==2)                        //Cartel 2:  >Vss   IAT
       {                                  //          TPS   KS
         adc_key_in = analogRead(0);              //          CKP   MAP
         key = get_key(adc_key_in);
         if ( key == 0)
         {
          cartel=1;CartelLCD();encero();cursorLCD();
          delay(200);
         }
         selecCursor();
         if (key == 4)
         {
          delay(200);
          if (numCols==0)
          {
           cartel = 20;CartelLCD();
           while (cartel==20)                     //Cartel 3: Vss    12v  S  GND
           {                                  //          INICIAR    ATRAS
            adc_key_in = analogRead(0);
            key = get_key(adc_key_in);
            if ( key == 0)
            {
             cartel=2;CartelLCD();cursorLCD();
             delay(200);
            }

            if (key==4)
            {
             digitalWrite(28, HIGH);                  //Activa rele 3 12V
             cartel=13;CartelLCD();
             lcd.setCursor(0, 2);lcd.print("Pulsos leidos:");
             while (cartel==13)                   //Cartel 4:  Probando.....
             {

              ////////////////////////////////////////////////////////////////////
              ////////////////////////////PROGRAMA PARA PROBAR VSS/////////////////////


              lcd.setCursor(15, 2);lcd.print(val);
              diag = digitalRead(PSENS);
              if (diag == LOW)
              {
               val=val+1;
               while (diag == LOW)
               {
                diag = digitalRead(PSENS);
                adc_key_in = analogRead(0);
                key = get_key(adc_key_in);
                if ( key == 0)
```

```
            {
              if (val==1)
                {
                  lcd.setCursor(0, 2);lcd.print("SENSOR DEFECTUOSO");
                  delay(1500);
                }
               else
               {
                 lcd.setCursor(0, 2);lcd.print("SENSOR OK        ");
                 delay(1500);
               }
              cartel=20;
              diag=HIGH;
              val=0;
              CartelLCD();
              delay(200);

            }
          }
        }
        adc_key_in = analogRead(0);
        key = get_key(adc_key_in);
        if ( key == 0)
        {
          if (val==0)
            {
              lcd.setCursor(0, 2);lcd.print("SENSOR DEFECTUOSO");
              delay(1000);
            }
          else
            {
              lcd.setCursor(0, 2);lcd.print("SENSOR OK        ");
              delay(1500);
            }
          digitalWrite(28, LOW);                    //Desactiva rele 3 12V
          cartel=20;CartelLCD();
          val=0;
          delay(200);
        }
        ////////////////////////////FIN DE PROGRAMA PARA PROBAR VSS//////////////////////////
        ////////////////////////////////////////////////////////////////////////////////
      }
    }
  }
}
if (numCols==1)
{
  cartel = 21;CartelLCD();
  while (cartel==21)                        //Cartel 5: TPS    12v  S  GND
  {                                  //           INICIAR  ATRAS
    adc_key_in = analogRead(0);
    key = get_key(adc_key_in);
    if ( key == 0)
    {
      cartel=2;CartelLCD();cursorLCD();
      delay(200);
    }

    if (key==4)
    {
     digitalWrite(30, HIGH);                    //Activa rele #5 5V
     cartel=13;CartelLCD();
     lcd.setCursor(0, 2);lcd.print("Valor leido:");
     while (cartel==13)                         //Cartel 4:  Probando.....
     {
       ////////////////////////////////////////////////////////////////////////////////
```

```
          ////////////////////////PROGRAMA PARA PROBAR TPS////////////////////////////////
            lcd.setCursor(13, 2);lcd.print("  % ");
            lcd.setCursor(13, 2);lcd.print(val);
            val = analogRead(sensorPin);
            val=((val*100)/1024);
            delay(200);

            adc_key_in = analogRead(0);
            key = get_key(adc_key_in);
            if ( key == 0)
            {
              if (val>90)
              {
                lcd.setCursor(0, 2);lcd.print("SENSOR DEFECTUOSO");
                val=0;
                delay(2000);
              }
              if (val<3)
              {
                lcd.setCursor(0, 2);lcd.print("SENSOR DEFECTUOSO");
                delay(2000);
              }
              if (val>3)
              {
                lcd.setCursor(0, 2);lcd.print("SENSOR OK         ");
                delay(2000);
              }
              digitalWrite(30, LOW);                    //Desactiva rele #5 5V
              cartel=21;
              CartelLCD();
              val=0;
              delay(200);
            }

            /////////////////FIN DE PROGRAMA PARA PROBAR TPS////////////////////////////////
            ////////////////////////////////////////////////////////////////////////////////

        }
      }
    }
}

if (numCols==2)
{
  cartel = 22;CartelLCD();

  while (cartel==22)                          //Cartel 6: CMP    GND  S  B+
  {                                       //         INICIO      ATRAS
    adc_key_in = analogRead(0);
    key = get_key(adc_key_in);
    if ( key == 0)
    {
      cartel=2;CartelLCD();cursorLCD();
      delay(200);
    }
    if (key==4)
    {
     digitalWrite(28, HIGH);                    //Activa rele 3 12V
     cartel=13;CartelLCD();
     lcd.setCursor(0, 2);lcd.print("Pulsos leidos:");
     while (cartel==13)                         //Cartel 4:  Probando.....
     {
     ////////////////////////////////////////////////////////////////////
     ////////////////////////PROGRAMA PARA PROBAR CMP/////////////////////
```

```
      lcd.setCursor(15, 2);lcd.print(val);
      diag = digitalRead(PSENS);
      if (diag == LOW)
      {
       val=val+1;
       while (diag == LOW)
       {
        diag = digitalRead(PSENS);
        adc_key_in = analogRead(0);
        key = get_key(adc_key_in);
        if ( key == 0)
        {
         if (val==1)
           {
            lcd.setCursor(0, 2);lcd.print("SENSOR DEFECTUOSO   ");
            delay(1500);
           }
          else
          {
            lcd.setCursor(0, 2);lcd.print("SENSOR OK           ");
            delay(1500);
          }
         cartel=22;
         diag=HIGH;
         val=0;
         CartelLCD();
         delay(200);

        }
       }
      }
      adc_key_in = analogRead(0);
      key = get_key(adc_key_in);
      if ( key == 0)
      {
       if (val==0)
         {
          lcd.setCursor(0, 2);lcd.print("SENSOR DEFECTUOSO   ");
          delay(1000);
         }
       else
         {
          lcd.setCursor(0, 2);lcd.print("SENSOR OK          ");
          delay(1500);
         }
       digitalWrite(28, LOW);                    //Desactiva rele 3 12V
       cartel=22;
       val=0;
       CartelLCD();
       delay(200);
      }
      ////////////////////////////FIN DE PROGRAMA PARA PROBAR CMP//////////////////////////
      ////////////////////////////////////////////////////////////////////////////////////
      }
     }
   }
}

if (numCols==3)
{
  cartel = 23;CartelLCD();
  while (cartel==23)//////////////////////////Cartel 7: ECT    GND  MEDIDOR  S
  {             ////////////////////////        INICIO       ATRAS
    adc_key_in = analogRead(0);
    key = get_key(adc_key_in);
    if ( key == 0)
```

```
    {
      cartel=2;CartelLCD();cursorLCD();
      delay(200);
     }
     if (key==4)
     {
      digitalWrite(30, HIGH);                    //Activa rele #5 5V
      digitalWrite(31, HIGH);                    //Activa rele #6
      cartel=4;CartelLCD();
      while (cartel==4)//////////////////Cartel 4:  Simulando.....
      {
         adc_key_in = analogRead(0);
         key = get_key(adc_key_in);
         if ( key == 0)
         {
           cartel=23;CartelLCD();
           delay(200);
         }
         ////////////////////////////////////////////////////////////////////////////
        ///////////////////////PROGRAMA PARA PROBAR ECT/////////////////////////////////

         lcd.setCursor(0, 2);lcd.print("Tem. Actual:");
         lcd.setCursor(13, 2);lcd.print("    C");
         val = analogRead(sensorPin);
         val = (((-0.0000211)* val* val*val)+(0.00899*val*val) - (1.4384 * val) +116.1);
         lcd.setCursor(13, 2);lcd.print(val);
         delay(200);

         adc_key_in = analogRead(0);
         key = get_key(adc_key_in);
         if ( key == 0)
         {
           if (val>100)
           {
             lcd.setCursor(0, 2);lcd.print("SENSOR DEFECTUOSO   ");
             delay(2000);
           }
           if (val<3)
           {
             lcd.setCursor(0, 2);lcd.print("SENSOR DEFECTUOSO   ");
             delay(2000);
           }
           if (val>3)
           {
            if( val<100)
            {
              lcd.setCursor(0, 2);lcd.print("SENSOR OK           ");
              delay(2000);
            }
           }
           digitalWrite(30, LOW);                    //Desactiva rele #5 5V
           digitalWrite(31, LOW);                    //Desactiva rele #6
           cartel=23;CartelLCD();
           val=0;
           delay(200);
         }
        ///////////////////////FIN PROGRAMA PARA PROBAR ECT/////////////////////////////////
        ////////////////////////////////////////////////////////////////////////////
       }
     }
   }
}

if (numCols==6)
{
  cartel = 24;CartelLCD();
```

```
  while (cartel==24)//////////////////////////Cartel 8:  IAT
  {              ////////////////////////              INICIO    ATRAS
    adc_key_in = analogRead(0);
    key = get_key(adc_key_in);
    if ( key == 0)
    {
      cartel=2;CartelLCD();cursorLCD();
      delay(200);
     }
     if (key==4)
     {
      digitalWrite(30, HIGH);                    //Activa rele #5 5V
      digitalWrite(31, HIGH);                    //Activa rele #6
      cartel=4;CartelLCD();
      while (cartel==4)//////////////////Cartel 4:  Simulando.....
      {
         adc_key_in = analogRead(0);
         key = get_key(adc_key_in);
         if ( key == 0)
         {
           cartel=24;CartelLCD();
           delay(200);
         }
         ////////////////////////////////////////////////////////////////////////////
         ///////////////////////PROGRAMA PARA PROBAR IAT//////////////////////////////
         lcd.setCursor(0, 2);lcd.print("Tem. Actual:");
         lcd.setCursor(13, 2);lcd.print("    C");
         val = analogRead(sensorPin);
         val = (((-0.0000211)* val* val*val)+(0.00899*val*val) - (1.4384 * val) +116.1);
         lcd.setCursor(13, 2);lcd.print(val);
         delay(200);

         adc_key_in = analogRead(0);
         key = get_key(adc_key_in);
         if ( key == 0)
         {
           if (val>100)
           {
             lcd.setCursor(0, 2);lcd.print("SENSOR DEFECTUOSO  ");
             delay(2000);
           }
           if (val<3)
           {
             lcd.setCursor(0, 2);lcd.print("SENSOR DEFECTUOSO   ");
             delay(2000);
           }
           if (val>3)
           {
             if( val<100)
             {
               lcd.setCursor(0, 2);lcd.print("SENSOR OK           ");
               delay(2000);
             }
           }
           digitalWrite(30, LOW);                    //Desactiva rele #5 5V
           digitalWrite(31, LOW);                    //Desactiva rele #6
           cartel=24;CartelLCD();
           val=0;
           delay(200);
         }
         ///////////////////////FIN PROGRAMA PARA PROBAR IAT//////////////////////////////
        ////////////////////////////////////////////////////////////////////////////
      }
     }
   }
}
```

```
if (numCols==7)
{
  cartel = 25;CartelLCD();
  while (cartel==25)//////////////////////////Cartel 25:  MAF  G B+ MAF G IAT
  {              ////////////////////////              INICIO    ATRAS
    adc_key_in = analogRead(0);
    key = get_key(adc_key_in);
    if ( key == 0)
    {
      cartel=2;CartelLCD();cursorLCD();
      delay(200);
    }
    if (key==4)
    {
     digitalWrite(30, HIGH);                    //Activa rele #5 5V
     cartel=13;
     CartelLCD();
     lcd.setCursor(0, 2);lcd.print("Valor leido:");
     while (cartel==13)//////////////////Cartel 4:  Probando.....
     {
        adc_key_in = analogRead(0);
        key = get_key(adc_key_in);
        if ( key == 0)
        {
          cartel=25;CartelLCD();
          delay(200);
        }
       ////////////////////////////////////////////////////////////////////////////
       ///////////////////////PROGRAMA PARA PROBAR MAF/////////////////////////////////
        lcd.setCursor(13, 2);lcd.print("   gr/s ");
        lcd.setCursor(13, 2);lcd.print(val);
        val = analogRead(sensorPin);
        val=((val*127)/1024);
        delay(200);

        adc_key_in = analogRead(0);
        key = get_key(adc_key_in);
        if ( key == 0)
        {
          if (val>75)
          {
            lcd.setCursor(0, 2);lcd.print("SENSOR DEFECTUOSO   ");
            val=0;
            delay(2000);
          }
          if (val<20)
          {
            lcd.setCursor(0, 2);lcd.print("SENSOR DEFECTUOSO   ");
            delay(2000);
          }
          if (val>20)
          {
            lcd.setCursor(0, 2);lcd.print("SENSOR OK           ");
            delay(2000);
          }
          digitalWrite(30, LOW);                    //Desactiva rele #5 5V
          cartel=25;CartelLCD();
          val=0;
          delay(200);
        }
        /////////////////FIN PROGRAMA PROBAR MAF////////////////////////
        ////////////////////////////////////////////////////////////
     }
    }
  }
```

```
}

if (numCols==8)
{
  cartel = 26;CartelLCD();
  while (cartel==26)//////////////////////////Cartel 26:  MAP  GND -- 5V   S
  {              ////////////////////////              INICIO    ATRAS
    adc_key_in = analogRead(0);
    key = get_key(adc_key_in);
    if ( key == 0)
    {
      cartel=2;CartelLCD();cursorLCD();
      delay(200);
    }
     if (key==4)
     {
      digitalWrite(30, HIGH);                    //Activa rele #5 5V
      cartel=13;CartelLCD();
      lcd.setCursor(0, 2);lcd.print("Valor leido:");
      while (cartel==13)//////////////////Cartel 4:  Probando.....
      {
         adc_key_in = analogRead(0);
         key = get_key(adc_key_in);
         if ( key == 0)
         {
           cartel=26;CartelLCD();
           delay(200);
         }
         ////////////////////////////////////////////////////////////////////////////
        ///////////////////////PROGRAMA PARA PROBAR MAP/////////////////////////////////
         lcd.setCursor(13, 2);lcd.print("   kPa ");
         lcd.setCursor(13, 2);lcd.print(val);
         val = analogRead(sensorPin);
         val=((val*127)/1024);
         delay(200);

         adc_key_in = analogRead(0);
         key = get_key(adc_key_in);
         if ( key == 0)
         {
           if (val>75)
           {
             lcd.setCursor(0, 2);lcd.print("SENSOR DEFECTUOSO ");
             val=0;
             delay(2000);
           }
           if (val<20)
           {
             lcd.setCursor(0, 2);lcd.print("SENSOR DEFECTUOSO ");
             delay(2000);
           }
           if (val>20)
           {
             lcd.setCursor(0, 2);lcd.print("SENSOR OK         ");
             delay(2000);
           }
           digitalWrite(30, LOW);                    //Desactiva rele #5 5V
           cartel=26;CartelLCD();
           val=0;
           delay(200);
         }
         /////////////////FIN PROGRAMA PROBAR MAP////////////////////////
         ////////////////////////////////////////////////////////////////
      }
    }
  }
```

```
            }

            if (numCols==9)
            {
              cartel = 27;CartelLCD();
              while (cartel==27)//////////////////////////Cartel 11:  HO2S  GND -- 5V   S
              {              ////////////////////////              INICIO    ATRAS
                adc_key_in = analogRead(0);
                key = get_key(adc_key_in);
                if ( key == 0)
                {
                  cartel=2;CartelLCD();cursorLCD();
                  delay(200);
                 }
                 if (key==4)
                 {
                  TSensor=6;
                  Serial1.write("ATSP6\r"); ////ISO 15765-4 CAN 11/500
                 cartel=13;CartelLCD();
                 while (cartel==13)//////////////////Cartel 4:  Simulando.....
                 {
                    adc_key_in = analogRead(0);
                    key = get_key(adc_key_in);
                    if ( key == 0)
                    {
                      cartel=27;CartelLCD();
                      delay(200);
                    }
                    ////////////////////////////////////////////////////////////////////////
                   ///////////////////////PROGRAMA PARA PROBAR HO2S//////////////////////////////////
                    lcd.setCursor(0, 3);lcd.print("Lectura:");
                    //lcd.setCursor(13, 2);lcd.print(val);
                    //val = analogRead(sensorPin);
                    //delay(200);
                    ////////////////////////HO2S OBDII//////////////////
                    delay(250);
                    Serial1.write("0114\r");///PID peticion HO2S
                    SerialOBDII();
                     ////////////////////////////////////////////////
                    adc_key_in = analogRead(0);
                    key = get_key(adc_key_in);
                    if ( key == 0)
                    {
                      lcd.setCursor(0, 1);lcd.print("Sensor OK, si:");
                      lcd.setCursor(0, 2);lcd.print("Lectura entre:");
                      lcd.setCursor(0, 3);lcd.print("0.450v - 0.550v");
                      delay(8000);
                      cartel=27;CartelLCD();TSensor=0;
                      val=0;
                      delay(200);
                    }
                  }
                 }
                }
              }
             }
            }
           }
          }
        }
      }
    }
  }
}
////////////////////////////////////////////////////////////////////////////////
////////////////////////////ANALISIS DE ACTUADORES//////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
if(numCols==1)
{
```

```
cartel=15;CartelLCD();encero();cursorLCD();
while (cartel==15)
{
   adc_key_in = analogRead(0);    // Leemos el valor de la pulsacion
   key = get_key(adc_key_in);    // Obtenemos el boton pulsado
   if ( key == 0)
   {
    cartel=0;CartelLCD();encero();cursorLCD();
    delay(200);
   }
   selecCursor0();
   if (key == 4)
   {
    delay(200);
    if (numCols==0)
    {
     cartel=18;CartelLCD();
     digitalWrite(28, HIGH);                    //Activa rele 3 12V
     while (cartel==18)//////////////////////////Cartel 18: INYECTOR  CTR  B+
     {              ////////////////////////      INICIO    ATRAS
       adc_key_in = analogRead(0);
       key = get_key(adc_key_in);
       if ( key == 0)
       {
         cartel=15;CartelLCD();cursorLCD();
         digitalWrite(28, LOW);                 //Desactiva rele 3 12V
         delay(200);
        }
        if ( key == 4)
        {
         while (key == 4)
         {
          digitalWrite(27, HIGH);
          delay(20);
          digitalWrite(27, LOW);
          delay(300);
          adc_key_in = analogRead(0);
          key = get_key(adc_key_in);
         }
        }
     }

    }
    if (numCols==1)
    {
     cartel=19;CartelLCD();
     digitalWrite(28, HIGH);                    //Activa rele 3 12V
     while (cartel==19)//////////////////////////Cartel 18: BOBINA  CTR  B+
     {              ////////////////////////      INICIO    ATRAS
       adc_key_in = analogRead(0);
       key = get_key(adc_key_in);
       if ( key == 0)
       {
         cartel=15;CartelLCD();cursorLCD();
         digitalWrite(28, LOW);                 //Desactiva rele 3 12V
         delay(200);
        }
        if ( key == 4)
        {
         while (key == 4)
         {

          digitalWrite(27, HIGH);
          delay(10);
          digitalWrite(27, LOW);
          delay(300);
```

```
        adc_key_in = analogRead(0);
        key = get_key(adc_key_in);
       }

     }
   }

 }
 if (numCols==2)
 {
  cartel=28;CartelLCD();
  while (cartel==28)//////////////////////////Cartel 18: IAC  CO  B+  CC
  {              ////////////////////////             INICIO    ATRAS
    adc_key_in = analogRead(0);
    key = get_key(adc_key_in);
    if ( key == 0)
    {
      cartel=15;CartelLCD();cursorLCD();
      delay(200);
     }
     if ( key == 4)
    {
      cartel=33;
      CartelLCD();
      digitalWrite(28, HIGH);                  //Activa rele 3 12V
      digitalWrite(29, HIGH);                  //Activa rele 4 Control IAC
      val1=1;
      ////////////////////////////////////////////////////////////////////
      ////////////////CAMBIA LA FRECUECIA PWM//////////////////////////////
      int myEraser = 7;               // this is 111 in binary and is used as an eraser
      TCCR3B &= ~myEraser;            // this operation (AND plus NOT),  set the three bits in TCCR2B to 0
      int myPrescaler = 4;            // this could be a number in [1 , 6]. In this case, 3 corresponds in binary to 011.
      TCCR3B |= myPrescaler;          //this operation (OR), replaces the last three bits in TCCR2B with our new value
011
      ////////////////////////////////////////////////////////////////////
      ////////////////////////////////////////////////////////////////////

     while (cartel==33)//////////////////////////Cartel 18: IAC  CO  B+  CC
     {              ////////////////////////             INICIO    ATRAS
       adc_key_in = analogRead(0);
       key = get_key(adc_key_in);

       if (key == 1)
       {
        if (val1==1)
        {
        }
        else
        {
         val1 = val1 - 1;
         val2 = ((val1*100)/255);
         lcd.setCursor(1, 2);lcd.write("   %");
         lcd.setCursor(1, 2);
         lcd.print(val2);
         delay(50);
        }
       }
       if (key == 2)
       {
        if (val1==255)
        {
        }
        else
        {
         val1 = val1 + 1;
```

```
                val2 = ((val1*100)/255);
                lcd.setCursor(1, 2);lcd.write("   %");
                lcd.setCursor(1, 2);
                lcd.print(val2);
                delay(50);
               }
              }

              analogWrite(PinPWM2, val1);
              if ( key == 0)
              {
                lcd.clear();
                cartel=28;CartelLCD();
                val1=1;
                val2=1;
                analogWrite(PinPWM2, val1);
                digitalWrite(28, LOW);              //Desactiva rele 3 12V
                digitalWrite(29, LOW);              //Desactiva rele 4 Control IAC
                delay(200);
              }
             }
            }
           }
          }
          if (numCols==3)
          {
           cartel=29;CartelLCD();
           digitalWrite(28, HIGH);                  //Activa rele 3 12V
           while (cartel==29)//////////////////////////Cartel 18: PCSV  B+  CTR
           {            ////////////////////////      INICIO    ATRAS
             adc_key_in = analogRead(0);
             key = get_key(adc_key_in);
             if ( key == 0)
             {
               cartel=15;CartelLCD();cursorLCD();
               digitalWrite(28, LOW);               //Desactiva rele 3 12V
               delay(200);
             }
              if ( key == 4)
             {
               while (key == 4)
               {
                digitalWrite(27, HIGH);
                delay(150);
                digitalWrite(27, LOW);
                delay(300);
                adc_key_in = analogRead(0);
                key = get_key(adc_key_in);
               }

             }
           }
          }
    }

  }
}
 ////////////////////////////////////////////////////////////////////////////////
 ////////////////////////////////OBDII SCAN//////////////////////////////////////////////
 ////////////////////////////////////////////////////////////////////////////////
 if (numCols==2)
 {
   cartel=17;CartelLCD();
   while (cartel==17)
   {
      adc_key_in = analogRead(0);            // Leemos el valor de la pulsacion
```

```
    key = get_key(adc_key_in);                // Obtenemos el boton pulsado
    if ( key == 0)
    {
     cartel=0;CartelLCD();cursorLCD();
     delay(200);
    }
     if ( key == 4)
    {
     cartel=30;CartelLCD();encero();selecCursor2();cursorLCD();
     delay(200);
     //Serial1.write("0101\r"); ////ISO 15765-4 CAN 11/500
     while (cartel==30)
     {
       adc_key_in = analogRead(0);            // Leemos el valor de la pulsacion
       key = get_key(adc_key_in);             // Obtenemos el boton pulsado
       if ( key == 0)
       {
         cartel=17;CartelLCD();
         delay(200);
       }
       selecCursor2();
       if ( key == 4)
       {
         delay(200);
         if (numCols==1)
         {
            cartel=31;CartelLCD();
            while (cartel==31)
            {
              delay(300);
              Serial1.write("03\r"); ////ISO 15765-4 CAN 11/500
              diag=4;
              SerialOBDII2();

              adc_key_in = analogRead(0);    // Leemos el valor de la pulsacion
              key = get_key(adc_key_in);    // Obtenemos el boton pulsado
              if ( key == 0)
              {
                cartel=30;CartelLCD();cursorLCD();
                delay(200);
              }
            }

         }
         if (numCols==2)
         {
            cartel=32;CartelLCD();
            while (cartel==32)
            {
              delay(250);
              Serial1.write("04\r"); ////ISO 15765-4 CAN 11/500
              SerialOBDII2();
              cartel=30;CartelLCD();cursorLCD();
              delay(200);
            }
         }
       }
     }
   }
  }
 }

////////////////////////////////////////////////////////////////////////////////
//////////////////////////////AUTORES///////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
```

```
   if (numCols==3)
   {
     cartel=16;CartelLCD();
     while (cartel==16)
     {
       adc_key_in = analogRead(0);    // Leemos el valor de la pulsacion
       key = get_key(adc_key_in);    // Obtenemos el boton pulsado
       if ( key == 0)
       {
         cartel=0;CartelLCD();encero();cursorLCD();
         delay(200);
       }
     }
   }
  }
}
}

//////////////////////////////////////////////////////////////////////
/////////Servicio de Interrupcion//////////////////////////////////////
//////////////////////////////////////////////////////////////////////
ISR(TIMER5_COMPA_vect)
{
  digitalWrite(SVSS, !digitalRead(SVSS));
  analogWrite(PinPWM1, 252);
  if(x++ == 255)
  {
    x=0;
  }
}

//////////////////////////////////////////////////////////////////////
////////////////////SUBRUTINAS//////////////////////////////////////
//////////////////////////////////////////////////////////////////////
void digitalPotWrite()
{
  // take the SS pin low to select the chip:
  digitalWrite(slaveSelectPin,LOW);
  //  send in the address and value via SPI:
  SPI.transfer(0x11);SPI.transfer(STPS);
  // take the SS pin high to de-select the chip:
  digitalWrite(slaveSelectPin,HIGH);
}
//////////////////////////////////////////////////////////////////
void SerialOBDII()
{
 if ( Serial1.available())
  {
    lcd.setCursor(10, 4);
    // Lee todos los caracteres diponibles
    while ( Serial1.available() > 0)
    {
      // Guarda cada caracter
      DatoRx = Serial1.read();

        //lcd.write(DatoRx);
        if (DatoRx==13)
        {
          val=val+1;
        }
        if (val == 1)
        {
           if (DatoRx != 13)
           {
             if (DatoRx != 32)
             {
```

```
              PosDatoSerial= PosDatoSerial + 1;
              if(PosDatoSerial >= 5)
              {
                val3=val3+1;
                Conversion();TipoSensor();
              }
             }
            }
        }
        if (val==3)
        {
          val=0;lcd.setCursor(10, 4);PosDatoSerial=0;
        }

    }
    Dato1=0;Dato2=0;Dato=0;PosDatoSerial=0;DatoRx=0;val3=0;
  }
}
////////////////////////////////////////////////////////////////
/////////////////Atiende conexion OBDII////////////////////////////
////////////////////////////////////////////////////////////////
void SerialOBDII2()
{
 if ( Serial1.available())
  {
     delay(100);
     lcd.setCursor(1, 1);
    // Lee todos los caracteres diponibles
    while ( Serial1.available() > 0)
    {
       // Guarda cada caracter
       DatoRx =(Serial1.read());
       //lcd.write(DatoRx);
       if (DatoRx==13)
       {
       val=val+1;
       }
       if (val == 1)
       {
         if (DatoRx != 13)
         {
           if (DatoRx != 32)
           {
             PosDatoSerial= PosDatoSerial + 1;
             if(PosDatoSerial >= diag)
             {
               if (DatoRx==48)
               {
                  if(PosDatoSerial == 4)
                   {
                   lcd.setCursor(1, 1);
                   lcd.write("SIN CODIGOS");
                   PosDatoSerial=0;
                   }
               }
               if (PosDatoSerial == 4)
               {
                   lcd.write("           ");
                   lcd.setCursor(1, 1);
                   diag=5;
               }
               if (DatoRx==69)
               {
                 lcd.setCursor(1, 1);
                 lcd.write("ERROR      ");
                 PosDatoSerial=5;
```

```
            }

            if (PosDatoSerial >= 5)
            {
              Conversion();
              Codigo();
              IdCodigo();
            }
           }
          }
        }
       }
       if (val==3)
       {
         val=0;PosDatoSerial=0;val3=0;
       }
     }
     Dato1=0;Dato2=0;Dato=0;PosDatoSerial=0;DatoRx=0;
   }
}

////////////////////////////////////////////////////////////////////
//////////////Conversion de ASCII a valor decimal.//////////////////////
////////////////////////////////////////////////////////////////////
 void Conversion()
 {
   if(DatoRx==48){ Dato=0;return;}
   if(DatoRx==49){ Dato=1;return;}
   if(DatoRx==50){ Dato=2;return;}
   if(DatoRx==51){ Dato=3;return;}
   if(DatoRx==52){ Dato=4;return;}
   if(DatoRx==53){ Dato=5;return;}
   if(DatoRx==54){ Dato=6;return;}
   if(DatoRx==55){ Dato=7;return;}
   if(DatoRx==56){ Dato=8;return;}
   if(DatoRx==57){ Dato=9;return;}
   if(DatoRx==65){ Dato=10;return;}
   if(DatoRx==66){ Dato=11;return;}
   if(DatoRx==67){ Dato=12;return;}
   if(DatoRx==68){ Dato=13;return;}
   if(DatoRx==69){ Dato=14;return;}
   if(DatoRx==70){ Dato=15;return;}

 }
////////////////////////////////////////////////////////////////////
////////Conversion de codigo de falla ////////////////////////////////////
////////////////////////////////////////////////////////////////////
void Codigo()
 {
   if (val2==0){Dato=Dato*1000;}
   if (val2==1){Dato1=Dato*100;}
   if (val2==2){Dato2=Dato*10;}
   if (val2==3){Dato=Dato + Dato1 + Dato2;val2=-1;val3=val3+1;IdCodigo();Dato=0;}
   //lcd.print(Dato);
   val2=val2+1;
 }
////////////////////////////////////////////////////////////////
///////////////Tabla de codigos de falla/////////////////////////
////////////////////////////////////////////////////////////////
 void IdCodigo()
 {
   if (Dato==100){lcd.setCursor(0, val3);lcd.write("MAF C Malfunction");}
   if (Dato==101){lcd.setCursor(0, val3);lcd.write("MAF C R/P Problem");}
   if (Dato==102){lcd.setCursor(0, val3);lcd.write("MAF C Low Input");}
   if (Dato==103){lcd.setCursor(0, val3);lcd.write("MAF C High Input");}
   if (Dato==104){lcd.setCursor(0, val3);lcd.write("MAF C Intermittent");}
```

```
  if (Dato==105){lcd.setCursor(0, val3);lcd.write("MAP C Malfunction");}
  if (Dato==106){lcd.setCursor(0, val3);lcd.write("MAP C R/P Problem");}
  if (Dato==107){lcd.setCursor(0, val3);lcd.write("MAP C Low Input");}
  if (Dato==108){lcd.setCursor(0, val3);lcd.write("MAP C High Input");}
  if (Dato==109){lcd.setCursor(0, val3);lcd.write("MAP C Intermittent");}

  if (Dato==110){lcd.setCursor(0, val3);lcd.write("IAT C Malfunction");}
  if (Dato==111){lcd.setCursor(0, val3);lcd.write("IAT C R/P Problem");}
  if (Dato==112){lcd.setCursor(0, val3);lcd.write("IAT C Low Input");}
  if (Dato==113){lcd.setCursor(0, val3);lcd.write("IAT C High Input");}
  if (Dato==114){lcd.setCursor(0, val3);lcd.write("IAT C Intermittent");}

  if (Dato==115){lcd.setCursor(0, val3);lcd.write("ECT C Malfunction");}
  if (Dato==116){lcd.setCursor(0, val3);lcd.write("ECT C R/P Problem");}
  if (Dato==117){lcd.setCursor(0, val3);lcd.write("ECT C Low Input");}
  if (Dato==118){lcd.setCursor(0, val3);lcd.write("ECT C High Input");}
  if (Dato==119){lcd.setCursor(0, val3);lcd.write("ECT C Intermittent");}

  if (Dato==120){lcd.setCursor(0, val3);lcd.write("TPS C Malfunction");}
  if (Dato==121){lcd.setCursor(0, val3);lcd.write("TPS C R/P Problem");}
  if (Dato==122){lcd.setCursor(0, val3);lcd.write("TPS C Low Input");}
  if (Dato==123){lcd.setCursor(0, val3);lcd.write("TPS C High Input");}
  if (Dato==124){lcd.setCursor(0, val3);lcd.write("TPS C Intermittent");}

  if (Dato==125){lcd.setCursor(0, val3);lcd.write("Low Coolant Temp ");}
  if (Dato==126){lcd.setCursor(0, val3);lcd.write("Low Coolant Temp");}
  if (Dato==127){lcd.setCursor(0, val3);lcd.write("IAT Too High");}
  if (Dato==128){lcd.setCursor(0, val3);lcd.write("Coolant Thermostat");}
  if (Dato==129){lcd.setCursor(0, val3);lcd.write("Barome Press Too Low");}

  if (Dato==505){lcd.setCursor(0, val3);lcd.write("ISA C Malfunction");}
  if (Dato==506){lcd.setCursor(0, val3);lcd.write("ISA RPM Lower");}
  if (Dato==507){lcd.setCursor(0, val3);lcd.write("ISA RPM Lower");}
  if (Dato==508){lcd.setCursor(0, val3);lcd.write("ISA System C Low");}
  if (Dato==509){lcd.setCursor(0, val3);lcd.write("ISA System C High");}



 }
//////////////////////////////////////////////////////////////////
void encero()
{
  numRows = 0;numCols = 0;numRows1 = 0;numCols1 = -1;numRows0 = 0;numCols0 = 0;
}
//////////////////////////////////////////////////////////////////
///////////////Posicion del cursor Cartel 2 ////////////////////////
//////////////////////////////////////////////////////////////////
void selecCursor()
{
      if (key == 1)
      {  // Se ha pulsado la tecla arriba
          delay(100);
          if (numCols >0)
          {
             if(numCols < 4)
             {
               numRows0=0;numRows1=0;numCols1=numCols0;numCols=numCols-1;numCols0=numCols0-1;//////cursor
             }
            if (numCols == 6)
             {
               numRows0=0;numRows1=0;numCols=3;numCols0=3;numCols1=2;
               lcd.setCursor(9,0);lcd.write(2);
             }
             if (numCols > 6)
             {
```

```
              numRows0=9;numRows1=9;numCols1=numCols0;numCols=numCols-1;numCols0=numCols0-1;//////cursor
             }
             cursorLCD();
             delay(200);
           }
      }

      if (key == 2)
      {  // Se ha pulsado la tecla abajo
         delay(100);
         if (numCols < 9)
         {
          if (numCols < 4)
          {
           numRows0=0;numRows1=0;
           numCols=numCols+1;////////selec opcion
           numCols0=numCols0+1;//////cursor
           numCols1=numCols0-1;///////borra cursor
          }
          if (numCols == 4)
          {
           numCols0=-1;numCols1=-2;
           numCols=numCols+1;////////selec opcion
           lcd.setCursor(0,3);lcd.write(2);
          }
          if (numCols > 4)
          {
           numRows0=9;numRows1=9;numCols=numCols+1;numCols0=numCols0+1;numCols1=numCols0-1;
          }
          cursorLCD();
          delay(200);
         }
      }
}
//////////////////////////////////////////////////////////////////
//////////////////////////Posicion del cursor Cartel 0////////////////
//////////////////////////////////////////////////////////////////

void selecCursor0()
{
      if (key == 1)
      {  // Se ha pulsado la tecla arriba
         delay(100);
         if (numCols >0)
         {
          numRows0=0;numRows1=0;numCols1=numCols0;numCols=numCols-1;numCols0=numCols0-1;
          cursorLCD();
          delay(100);
         }
      }

      if (key == 2)
      {  // Se ha pulsado la tecla abajo
         delay(100);
         if (numCols < 3)
         {
          numRows0=0;numRows1=0;numCols=numCols+1;numCols0=numCols0+1;numCols1=numCols0-1;
          cursorLCD();
          delay(100);
         }
      }
}
//////////////////////////////////////////////////////////////////
//////////////////////////Posicion del cursor Cartel 1////////////////
//////////////////////////////////////////////////////////////////
```

```
void selecCursor1()
{
      if (key == 1)
      {  // Se ha pulsado la tecla arriba
          delay(100);
          if (numCols >0)
          {
           numRows0=0;numRows1=0;numCols1=numCols0;numCols=numCols-1;numCols0=numCols0-1;
           cursorLCD();
           delay(100);
          }
      }

      if (key == 2)
      {  // Se ha pulsado la tecla abajo
          delay(100);
          if (numCols < 1)
          {
           numRows0=0;numRows1=0;numCols=numCols+1;numCols0=numCols0+1;numCols1=numCols0-1;
           cursorLCD();
           delay(100);
          }
      }
}
////////////////////////////////////////////////////////////////////////////
//////////////////////Posicion del cursor Cartel 30////////////////////////////
////////////////////////////////////////////////////////////////////////////
void selecCursor2()
{
      if (key == 1)
      {  // Se ha pulsado la tecla arriba
          delay(100);
          if (numCols >1)
          {
           numRows0=0;numRows1=0;numCols1=numCols0;numCols=numCols-1;numCols0=numCols0-1;
           cursorLCD();
           delay(100);
          }
      }

      if (key == 2)
      {  // Se ha pulsado la tecla abajo
          delay(100);
          if (numCols < 2)
          {
           numRows0=0;numRows1=0;numCols=numCols+1;numCols0=numCols0+1;numCols1=numCols0-1;
           cursorLCD();
           delay(100);
          }
      }
}
//////////////////////////////////////////////////////////
///////////////Escribe el cursor en el LCD//////////////////
//////////////////////////////////////////////////////////
void cursorLCD()
{
   lcd.setCursor(numRows0, numCols0);
   lcd.write(1);
   lcd.setCursor(numRows1, numCols1);
   lcd.write(2);
}

 ////////////////////////////////////////////////////
 ////////Selecciona tipo de calculo  para cada PID//////////
 ////////////////////////////////////////////////////
 void TipoSensor()
```

```
{
 switch (TSensor)
 {
  case 0:  //VSS

    if (val3==1)
    {
     Dato1=Dato*16;
    }
    if (val3==2)
    {
     Dato=Dato1+Dato;
     lcd.print("    Km/h");lcd.setCursor(10, 4);lcd.print(Dato);
    }
    break;


 case 1:    // TPS
    if (val3==1)
    {
     Dato1=Dato*16;
    }
    if (val3==2)
    {
     Dato=(((Dato1+Dato)*100)/255);
     lcd.print("   %");lcd.setCursor(10, 4);lcd.print(Dato);
    }
    break;


 case 2:    // CMP
    if (val3==1)
    {
     Dato1=Dato*16;
    }
    if (val3==2)
    {
     Dato2=Dato+Dato1;
    }

    if (val3==3)
    {
     Dato2=Dato2*256;
     Dato1=Dato*16;
    }
    if (val3==4)
    {
     Dato=((Dato1+Dato+Dato2)/4);
     lcd.print("     RPM");lcd.setCursor(10, 4);lcd.print(Dato);
    }
    break;

  case 3:    // ECT - IAT
    if (val3==1)
    {
     Dato1=Dato*16;
    }
    if (val3==2)
    {
     Dato=Dato1+Dato-40;
     lcd.print("   C");lcd.setCursor(10, 4);lcd.print(Dato);

    }
    break;

  case 4:    // MAF
```

```
    if (val3==1)
    {
     Dato1=Dato*16;
    }
    if (val3==2)
    {
     Dato2=Dato+Dato1;
    }

    if (val3==3)
    {
     Dato2=Dato2*256;
     Dato1=Dato*16;
    }
    if (val3==4)
    {
     Dato=((Dato1+Dato+Dato2)/100);
     lcd.print("   grams/s");lcd.setCursor(10, 4);lcd.print(Dato);
    }
    break;


  case 5:    // MAP
    if (val3==1)
    {
     Dato1=Dato*16;
    }
    if (val3==2)
    {
     Dato=(Dato1+Dato);
     lcd.print("   kPa");lcd.setCursor(10, 4);lcd.print(Dato);
    }
    break;

    case 6:    // HO2S
    if (val3==1)
    {
     Dato1=Dato*16;
    }
    if (val3==2)
    {
     Dato1=Dato+Dato1;
    }

    if (val3==3)
    {
     //Dato1=Dato*16;
    }
    if (val3==4)
    {
     //Dato1=Dato+Dato1;
     Data=(float)Dato1/200;
     lcd.print("     v");lcd.setCursor(10, 4);lcd.print(Data);
    }
    break;

  }
}

////////////////////////////////////////////////////////////////////////////
//////////////////Carteles imprimibles en el LCD////////////////////////////
////////////////////////////////////////////////////////////////////////////
void CartelLCD()
{
  lcd.clear();
  switch (cartel)
```

```
{
 case 0:    // your hand is on the sensor

   lcd.setCursor(3, 0);lcd.write("ANALIZAR SENSOR");
   lcd.setCursor(3, 1);lcd.write("PROBAR ACTUADOR");
   lcd.setCursor(3, 2);lcd.write("OBDII SCAN");
   lcd.setCursor(3, 3);lcd.write("AUTORES");
   break;

 case 1:    // your hand is on the sensor

   lcd.setCursor(3, 0);lcd.write("SIMULAR SENSOR");
   lcd.setCursor(3, 1);lcd.write("PROBAR SENSOR");
   break;

 case 2:    // SIMULACION

   lcd.setCursor(3, 0);lcd.write("VSS");
   lcd.setCursor(3, 1);lcd.write("TPS");
   lcd.setCursor(3, 2);lcd.write("CKP");
   lcd.setCursor(3, 3);lcd.write("ETC");
   lcd.setCursor(10, 0);lcd.write("IAT");
   lcd.setCursor(10, 1);lcd.write("MAF");
   lcd.setCursor(10, 2);lcd.write("MAP");
   lcd.setCursor(10, 3);lcd.write("HO2S");
   break;

 case 3:

   lcd.setCursor(3, 0);lcd.write("VSS");
   lcd.setCursor(8, 0);lcd.write("  __[-]__ ");
   lcd.setCursor(8, 1);lcd.write("_( o o o )_ ");
   lcd.setCursor(9, 2);lcd.write(" S B+ GND ");
   lcd.setCursor(0, 3);lcd.write("INICIAR      ATRAS");
   break;

case 4:
  lcd.clear();
  lcd.setCursor(0, 0);lcd.write("Prueba de cableado..");
  break;

 case 5:

   lcd.setCursor(3, 0);lcd.write("TPS");
   lcd.setCursor(8, 0);lcd.write("  __[-]__ ");
   lcd.setCursor(8, 1);lcd.write("_( o o o )_ ");
   lcd.setCursor(9, 2);lcd.write("GND 5V  S ");
   lcd.setCursor(0, 3);lcd.write("INICIAR      ATRAS");
   break;

  case 6:

   lcd.setCursor(3, 0);lcd.write("CKP");
   lcd.setCursor(8, 0);lcd.write("  __[-]__ ");
   lcd.setCursor(8, 1);lcd.write("_( o o o )_ ");
   lcd.setCursor(9, 2);lcd.write("GND S B+ ");
   lcd.setCursor(0, 3);lcd.write("INICIAR      ATRAS");
   break;

    case 7:

   lcd.setCursor(3, 0);lcd.write("ECT");
   lcd.setCursor(8, 0);lcd.write("  __[-]__ ");
   lcd.setCursor(8, 1);lcd.write("_( o o o )_ ");
   lcd.setCursor(9, 2);lcd.write("GND M 5v ");
   lcd.setCursor(0, 3);lcd.write("INICIAR      ATRAS");
```

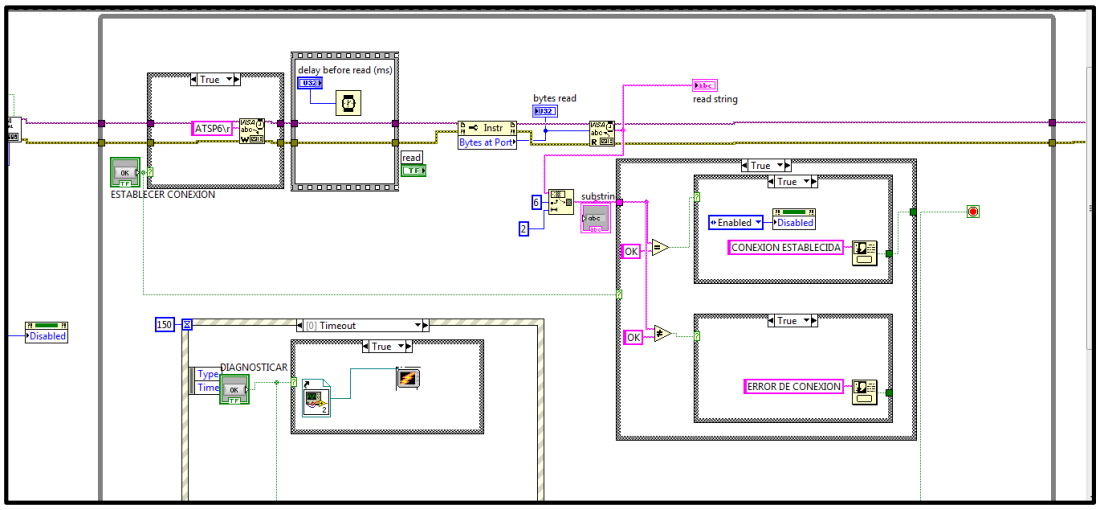
```
 break;

case 8:

 lcd.setCursor(3, 0);lcd.write("IAT");
 lcd.setCursor(7, 0);lcd.write("  __I[-]I__ ");
 lcd.setCursor(7, 1);lcd.write("-( o o o o )- ");
 lcd.setCursor(8, 2);lcd.write("GND S - - ");
 lcd.setCursor(0, 3);lcd.write(" INICIAR      ATRAS");
 break;

 case 9:

 lcd.setCursor(2, 0);lcd.write("MAF");
 lcd.setCursor(5, 0);lcd.write("  __I.---.I__ ");
 lcd.setCursor(5, 1);lcd.write("-( o o o o o )- ");
 lcd.setCursor(5, 2);lcd.write("IAT G MAF B+ G");
 lcd.setCursor(0, 3);lcd.write(" INICIAR      ATRAS");
 break;

 case 10:

 lcd.setCursor(3, 0);lcd.write("MAP");
 lcd.setCursor(7, 0);lcd.write("  __I[-]I__ ");
 lcd.setCursor(7, 1);lcd.write("-( o o o o )- ");
 lcd.setCursor(8, 2);lcd.write("GND - 5V S ");
 lcd.setCursor(0, 3);lcd.write(" INICIAR      ATRAS");
 break;

 case 11:

 lcd.setCursor(1, 0);lcd.write("H02S");
 lcd.setCursor(7, 0);lcd.write("To^oT ");
 lcd.setCursor(7, 1);lcd.write("!o_o! ");
 lcd.setCursor(13, 0);lcd.write("GND  S");
 lcd.setCursor(13, 1);lcd.write("B+  C");
 lcd.setCursor(0, 3);lcd.write("INICIAR      ATRAS");
 break;

 case 12:
lcd.clear();
lcd.setCursor(0, 0);lcd.write("Simulacion en curso");
break;

case 13:
lcd.clear();
lcd.setCursor(0, 0); lcd.write("Probando sensor...");
break;

case 15:

 lcd.setCursor(3, 0);lcd.write("INYECTOR");
 lcd.setCursor(3, 1);lcd.write("BOBINA");
 lcd.setCursor(3, 2);lcd.write("IAC");
 lcd.setCursor(3, 3);lcd.write("PCSV");
 break;

 case 16:
 lcd.setCursor(5, 0);lcd.write("UDA 2013");
 lcd.setCursor(2, 1);lcd.write("Geovanny Lupercio");
 lcd.setCursor(2, 2);lcd.write("Jhon Llivicura");
 break;

 case 17:
 //lcd.setCursor(0, 0);lcd.write("ESCANEAR VEHICULO??");
 lcd.setCursor(0, 1);lcd.write("Conecte GTEC OBDII");
```

```
 lcd.setCursor(0, 3);lcd.write("INICIAR      ATRAS");
 break;

case 18:
 lcd.setCursor(0, 0);lcd.write("INYECTOR");
 lcd.setCursor(9, 0);lcd.write("  _[-]_ ");
 lcd.setCursor(9, 1);lcd.write("-( - - )- ");
 lcd.setCursor(10, 2);lcd.write(" B+ CTR ");
 lcd.setCursor(0, 3);lcd.write(" INICIAR      ATRAS");
 break;

case 19:
 lcd.setCursor(0, 0);lcd.write("BOBINA");
 lcd.setCursor(9, 0);lcd.write("  _[-]__ ");
 lcd.setCursor(9, 1);lcd.write("-( - - ) ");
 lcd.setCursor(10, 2);lcd.write(" B+ CTR ");
 lcd.setCursor(0, 3);lcd.write(" INICIAR      ATRAS");
 break;

case 20:

 lcd.setCursor(3, 0);lcd.write("VSS");
 lcd.setCursor(8, 0);lcd.write("  __[-]__ ");
 lcd.setCursor(8, 1);lcd.write("_( o o o )_ ");
 lcd.setCursor(9, 2);lcd.write("GND B+ S ");
 lcd.setCursor(0, 3);lcd.write("INICIAR      ATRAS");
 break;

case 21:

 lcd.setCursor(3, 0);lcd.write("TPS");
 lcd.setCursor(8, 0);lcd.write("  __[-]__ ");
 lcd.setCursor(8, 1);lcd.write("_( o o o )_ ");
 lcd.setCursor(9, 2);lcd.write("  S 5V GND ");
 lcd.setCursor(0, 3);lcd.write("INICIAR      ATRAS");
 break;

case 22:

 lcd.setCursor(3, 0);lcd.write("CMP");
 lcd.setCursor(8, 0);lcd.write("  __[-]__ ");
 lcd.setCursor(8, 1);lcd.write("_( o o o )_ ");
 lcd.setCursor(9, 2);lcd.write(" B+ S GND ");
 lcd.setCursor(0, 3);lcd.write("INICIAR      ATRAS");
 break;

case 23:

 lcd.setCursor(3, 0);lcd.write("ECT");
 lcd.setCursor(8, 0);lcd.write("  __[-]__ ");
 lcd.setCursor(8, 1);lcd.write("_( o o o )_ ");
 lcd.setCursor(9, 2);lcd.write(" 5V M GND ");
 lcd.setCursor(0, 3);lcd.write("INICIAR      ATRAS");
 break;

 case 24:

 lcd.setCursor(3, 0);lcd.write("IAT");
 lcd.setCursor(7, 0);lcd.write("  __I[-]I__ ");
 lcd.setCursor(7, 1);lcd.write("-( o o o o )- ");
 lcd.setCursor(8, 2);lcd.write("  - - S GND ");
 lcd.setCursor(0, 3);lcd.write(" INICIAR      ATRAS");
 break;

 case 25:
```

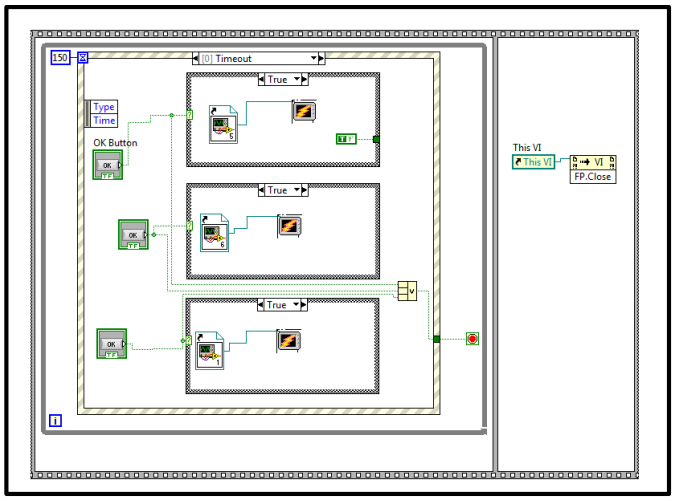
```
lcd.setCursor(2, 0);lcd.write("MAF");
lcd.setCursor(5, 0);lcd.write("  __I.---.I__ ");
lcd.setCursor(5, 1);lcd.write("-( o o o o o )- ");
lcd.setCursor(5, 2);lcd.write(" G B+ MAF G IAT");
lcd.setCursor(0, 3);lcd.write(" INICIAR      ATRAS");
break;

case 26:

lcd.setCursor(3, 0);lcd.write("MAP");
lcd.setCursor(7, 0);lcd.write("  __I[-]I__ ");
lcd.setCursor(7, 1);lcd.write("-( o o o o )- ");
lcd.setCursor(8, 2);lcd.write("  S 5V - GND ");
lcd.setCursor(0, 3);lcd.write(" INICIAR      ATRAS");
break;

case 27:

lcd.setCursor(1, 0);lcd.write("H02S");
lcd.setCursor(7, 0);lcd.write("To^oT ");
lcd.setCursor(7, 1);lcd.write("!o_o! ");
lcd.setCursor(13, 0);lcd.write("S  GND");
lcd.setCursor(13, 1);lcd.write("C  B+");
lcd.setCursor(0, 3);lcd.write("INICIAR       ATRAS");
break;


case 28:

lcd.setCursor(3, 0);lcd.write("IAC");
lcd.setCursor(8, 0);lcd.write("  _[---]_ ");
lcd.setCursor(8, 1);lcd.write("_( - - - )_ ");
lcd.setCursor(9, 2);lcd.write(" CO B+ CC ");
lcd.setCursor(0, 3);lcd.write("INICIAR      ATRAS");
break;

case 29:
lcd.setCursor(0, 0);lcd.write("PCSV");
lcd.setCursor(9, 0);lcd.write("  _[-]_ ");
lcd.setCursor(9, 1);lcd.write("-( - - )- ");
lcd.setCursor(10, 2);lcd.write(" B+ CTR ");
lcd.setCursor(0, 3);lcd.write(" INICIAR      ATRAS");
break;

case 30:
lcd.setCursor(0, 0);lcd.write(">>   GTEC OBDII   <<");
lcd.setCursor(2, 1);lcd.write("LEER DTC");
lcd.setCursor(2, 2);lcd.write("BORRAR DTC");
lcd.setCursor(0, 3);lcd.write("INICIAR      ATRAS");

break;

case 31:
lcd.setCursor(0, 0);lcd.write("CODIGOS DE FALLA");

break;

case 32:
lcd.setCursor(0, 0);lcd.write("BORRANDO DTC...");
delay(1000);
lcd.setCursor(0, 1);lcd.write("Ok");
delay(1000);

break;

case 33:
```

```
    lcd.setCursor(4, 0);lcd.write("PRUEBA IAC");
    lcd.setCursor(0, 1);lcd.write("CICLO DE ACTIVACION");
    lcd.setCursor(0, 3);lcd.write("      + - ATRAS");

    break;
  }
```

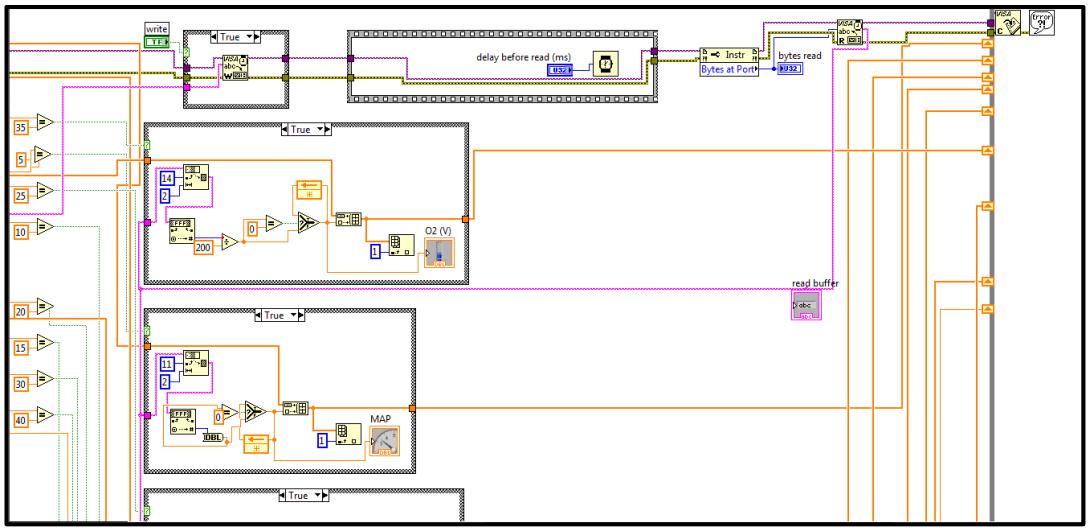
ANEXO 3: Programación gráfica de la Herramienta Virtual



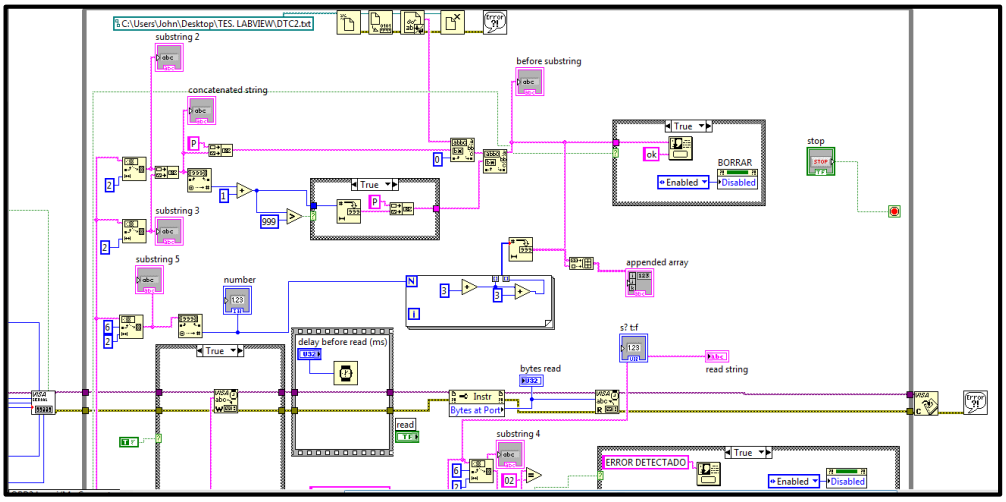
Programación gráfica Menú Principal



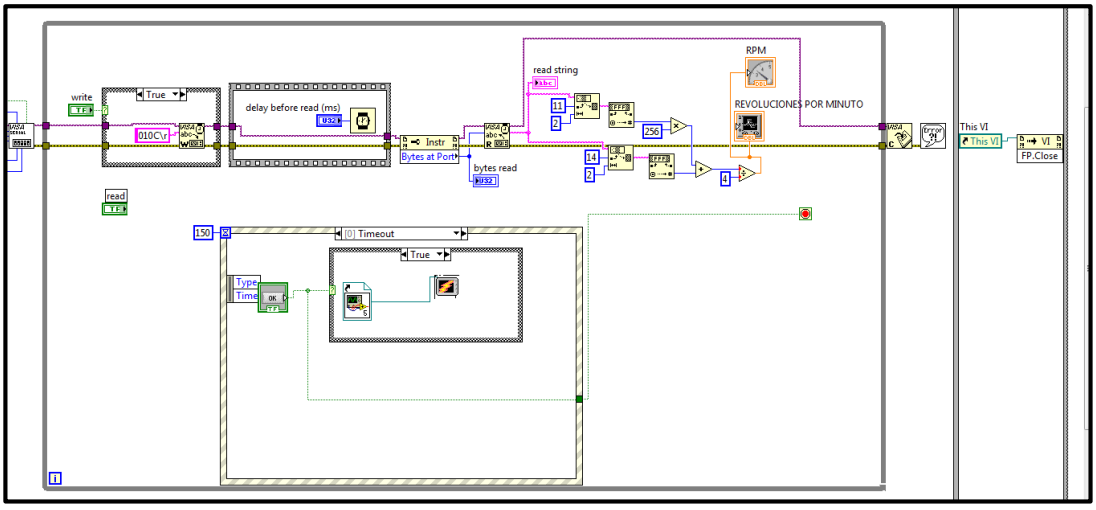
Programación gráfica Menú Diagnosticar Sistemas



Programación gráfica Visualizar Datos



Programación gráfica Análisis DTC



Programación gráfica un sensor (RPM)