UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIA APLICADAS

CARRERA DE INGENIERÍA EN MANTENIMIENTO AUTOMOTRIZ

TRABAJO DE GRADO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN MANTENIMIENTO AUTOMOTRIZ

TEMA:

"IMPLEMENTACIÓN DE UN MÓDULO DE CONTROL Y ALMACENAMIENTO DE DATOS ON-BOARD PARA TAXIS."

AUTORES:

ENRIQUEZ FOLLARAN JEFFERSON SAÚL

ORMAZA LÓPEZ FRANCISCO DAVID

DIRECTOR:

ING. FREDY ROSERO MSC.

IBARRA, 2017

UNIVERSIDAD TÉCNICA DEL NORTE

BIBLIOTECA UNIVERSITARIA

AUTORIZACIÓN DE USO Y PUBLICACIÓN
A FAVOR DE LA UNIVERSIDAD TÉCNICA DEL NORTE

1. Identificación de la obra

La Universidad Técnica del Norte dentro del proyecto Repositorio Digital Institucional, determinó la necesidad de disponer de texto completos en forma digital con la finalidad de apoyar los procesos de investigación, docencia y extensión de la Universidad.

Por medio del presente documento dejo sentada mi voluntad de participar en este proyecto, para lo cual pongo a disposición la siguiente información.

DATOS DE CONTACTO			
CÉDULA DE IDENTIDAD:	100365396-9		
APELLIDOS Y NOMBRES:	Enriquez Follaran Jefferson Saúl		
DIRECCIÓN:	Ibarra, Barrio Colinas del Sur, calle: 10 de diciembre y José Peralta		
EMAIL:	jeff_afrod44@yahoo.com		
TELEFONO FIJO:		**TELÉFONO MÓVIL:**	0992360167

DATOS DE CONTACTO	
CÉDULA DE IDENTIDAD:	100348389-6
APELLIDOS Y NOMBRES:	Ormaza López Francisco David
DIRECCIÓN:	Ibarra, Antonio José de Sucre y Tobías Mena

EMAIL:	franciscoormazal@gmail.com		
TELEFONO FIJO:	062- 600- 811	**TELÉFONO MÓVIL:**	0985270122

DATOS DE LA OBRA	
TÍTULO:	"IMPLEMENTACIÓN DE UN MÓDULO DE CONTROL Y ALMACENAMIENTO DE DATOS ON-BOARD PARA TAXIS."
AUTOR (ES):	Ormaza López Francisco David Enriquez Follaran Jefferson Saúl
FECHA: AAAMMDD	
SOLO PARA TRABAJOS DE GRADO	
PROGRAMA:	[X] **PREGRADO** [ ] **POSTGRADO**
TÍTULO POR EL QUE OPTA:	INGENIERÍA EN MANTENIMIENTO AUTOMOTRIZ
ASESOR/DIRECTOR:	ING. FREDY ROSERO MSC.

2. Autorización a favor de la universidad

Yo **Ormaza López Francisco David** con cédula de identidad Nro. **100348389-6** y **Enriquez Follaran Jefferson Saúl** con cédula de identidad Nro. **100365396-9** en calidad de autores y titulares de los derechos patrimoniales de la obra o trabajo de grado descripto anteriormente, hago entrega del ejemplar respectivo en formato digital y autorizo a la Universidad Técnica del Norte, la publicación de la obra en el Repositorio Digital Institucional y uso del archivo digital en la Biblioteca de la Universidad con fines académicos, para ampliar la disponibilidad del material y como apoyo a la educación, investigación y extensión; en concordancia con la Ley de Educación Superior Artículo 144.

3. Constancia

Los autores manifiestan que la obra objeto de la presente autorización es original y se la desarrollo, sin violar derechos del autor de terceros, por lo tanto, la obra es original y que son los titulares de los derechos patrimoniales, por lo que asumen la responsabilidad sobre el contenido de la misma y saldrán en defensa de la Universidad en caso de reclamación por parte de terceros.

Ibarra, a los 17 días del mes de Febrero de 2017.

Autores:

.....

Firma

Nombre: Ormaza López Francisco David

Cédula: 1003483896

.....

Firma

Nombre: Enriquez Follaran Jefferson Saúl

Cédula: 1003653969

UNIVERSIDAD TÉCNICA DEL NORTE

CESIÓN DE DERECHOS DE AUTOR DEL TRABAJO DE GRADO A FAVOR DE LA UNIVERSIDAD TÉCNICA DEL NORTE

Yo **Ormaza López Francisco David** con cédula de identidad Nro. **100348389-6** y **Enriquez Follaran Jefferson Saúl** con cédula de identidad Nro. **100365396-9**, manifestamos nuestra voluntad de ceder a la Universidad Técnica del Norte los derechos patrimoniales consagrados en la Ley de Propiedad Intelectual del Ecuador artículos 4, 5 y 6, en calidad de autor de la obra o trabajo de grado denominado **"IMPLEMENTACIÓN DE UN MÓDULO DE CONTROL Y ALMACENAMIENTO DE DATOS ON-BOARD PARA TAXIS."** que ha sido desarrollado para optar por el título de: **INGENIERÍA EN MANTENIMIENTO AUTOMOTRIZ** en la Universidad Técnica del Norte quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente. En nuestra Condición de autores nos reservamos los derechos morales de la obra antes citada. En concordancia suscribimos este documento en el momento que hacemos entrega del trabajo final en formato impreso y digital a la Biblioteca de la Universidad Técnica del Norte.

.....
Firma

Nombre: Ormaza López Francisco David

Cédula: 100348389-6

.....
Firma

Nombre: Enriquez Follaran Jefferson Saúl

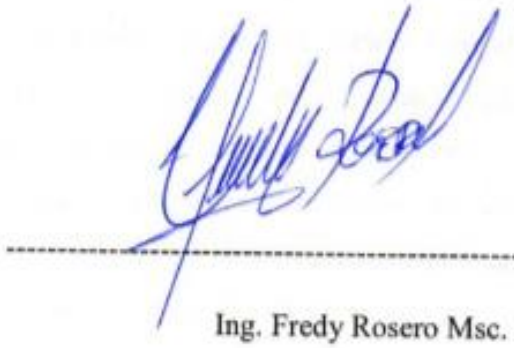
Cédula: 1003653969

Ibarra, a los 17 días del mes de Febrero de 2017

CERTIFICACIÓN

Certifico que el presente proyecto fue realizado en su totalidad por los señores: Enríquez Follaran Jefferson Saúl y Ormaza López Francisco David, como requerimiento para la obtención del título de Ingeniería en Mantenimiento Automotriz. Doy fe que dicho trabajo reúne los requisitos para ser sometido a presentación pública y evaluación por parte de los señores integrantes del jurado examinador designado.

Atentamente,



Ing. Fredy Rosero Msc.

DIRECTOR DEL TRABAJO DE GRADO

AGRADECIMIENTO

Dirijo mi más sincero agradecimiento a mis abuelos por ser mis guías, y mi apoyo incondicional en toda mi trayectoria estudiantil, a mi tutor por ser parte de este proyecto de investigación y brindarme su tiempo para desarrollarlo, por impartir sus conocimientos, enseñanzas y ser un gran guía durante todo el proceso y así permitirme llegar a ser un profesional de gran moral y ética.

Francisco Ormaza L.

Agradezco a Dios por haberme dado la salud para afrontar este camino en busca de cumplir mis metas, a mi Madre y Padrastro que siempre estuvieron a mi lado para bridarme su cariño y apoyo incondicional. A la Universidad Técnica del Norte que me abrió las puertas para estudiar esta linda carrera que elegí seguir, a todo el personal docente de la Carrera de Ingeniería en Mantenimiento Automotriz quienes con sus conocimientos me ayudaron a formarme como un buen profesional, en especial a mi tutor quien además de haberme impartido sus conocimientos como docente también confió en mi para realizar este proyecto ya que sin su paciencia y dedicación no hubiese podido culminar este proyecto.

Jefferson Enriquez F.

DEDICATORIA

Dedico este proyecto a mis abuelos José Miguel López y Teresa Margarita Quelal, porque el logro es gracias a ellos, por dedicarme su tiempo, cuidados y apoyo en esos días de mayor dificultad, permitiéndome no desfallecer ante las circunstancias, y escalar peldaño a peldaño logrando así alcanzar mis ideales y mi gran anhelada meta profesional.

Francisco Ormaza L.

Este proyecto va dedicado a Dios por haber mantenido a mis seres queridos a mi lado y por haberme rodeado de buenas amistades que me supieron apoyar durante todo este tiempo y fueron partícipes para la consecución de esta meta. Un reconocimiento especial a mi Madre por ser el pilar fundamental de mi vida a quien le debo todo, a mi hermana por su apoyo moral, a todos aquellos que se fueron de este mundo pero que dejaron en mí enseñanzas y recuerdo que nunca serán borrados y a todos aquellos que quizás no nombre, pero también formaron parte de esto a todos ellos gracias.

Jefferson Enriquez F.

RESUMEN

El presente trabajo de grado tiene como objetivo el implementar un módulo de almacenamiento de datos on-board para su aplicación en taxis que permita recolectar datos de gestión electrónica del motor, posición de marcha, estado de freno, GPS, aceleración X Y Z. Este módulo obtiene datos de gestión del motor a través de un cable OBDII que hace a este módulo poco invasivo y adaptable a cualquier vehículo que disponga de un conector OBDII, cuenta con un receptor GPS que le envía datos de fecha, hora y ubicación, también consta de sistema de estado marcha/freno que indicara al módulo en que marcha se encuentra y cuando se accione el pedal de freno. Todas estas señales son procesadas por la placa que previamente fue programada en su lenguaje nativo para que almacene estos datos en una tarjeta SD integrada al módulo, esta información se almacena en forma de un archivo de texto (txt). Estos archivos pueden ser exportados a la interface del módulo creada en Labview en la cual se puede ver y analizar todos los datos recolectados en el archivo mediante gráficas, simulación y una tabla de datos. Este módulo posee una pantalla en la que el conductor puede observar la hora y la cantidad de datos obtenidos en ese instante y los mensajes respectivos de grabación, fin y reinicio. Cuenta con una cámara de tráfico situada estratégicamente que registra fotográficamente a cada instante todo lo que sucede esto sirve como una información adicional a todos los datos proporcionados por el módulo y estos al igual que los datos pueden ser visualizados en la interface de Labview. Estos datos pueden servir para evaluar condiciones de manejo que puedan incidir en un accidente o en el deterioro del vehículo.

ABSTRACT

This degree work has like an objective to implement an on-board data storage module for its application in taxis that allows to collect data of electronic management of the engine, running position, state of brake, GPS, acceleration XY Z. This module obtains data of management of the motor through an OBDII cable that makes this module minimally invasive and adaptable to any vehicle that has an OBDII connector, it counts on a GPS receiver that sends data to it of date, time and location, also consists of a state of gear / brake state system that will indicate the module in which gear is located and when the brake pedal is operated. All these signals are processed by the board that was previously programmed in its native language to save it this data on an SD card integrated in this module, this information is stored as a text file (txt). These files can be exported to the module interface created in Labview in which you can view and analyze all the data collected in the file using graphs, simulation and a data table. This module has a screen in which the driver can observe the time and amount of data obtained at that time and the respective messages of recording, end and restart. It has a strategically located traffic camera that records photographically at every instant everything that happens this serves as an additional information to all the data provided by the module and these as well as the data can be visualized in the Labview interface. These data can serve to evaluate conditions of handling that may affect an accident or deterioration of the vehicle.

INTRODUCCIÓN

El presente proyecto tiene como meta principal diseñar e implementar un módulo de almacenamiento de datos On-board para su aplicación en vehículos tipo taxis. Cuyo objetivo es recolectar datos de gestión del motor, estado de freno, posición de marchas, GPS, registro fotográfico del trayecto mediante una cámara integrada al módulo.

De esta manera es posible almacenar estos datos en una memoria que luego de 1 o 2 días de trabajo puedan ser descargados del módulo, para analizarlos mediante la interfaz diseñada en Labview. Los usos que se le puede dar a esta información son diversos entre ellos destacamos: evaluar condiciones de manejo, determinar causas en accidentes de tránsito y determinación de los períodos para efectuar el mantenimiento vehicular.

El presente proyecto se desarrolla en la ciudad de Ibarra, tomando como objeto de estudio a los vehículos livianos de tipo taxi, los cuales cuentan con conectores OBDII lo que permite que este módulo pueda ser instalado en cualquier vehículo con modificaciones mínimas para cada modelo. Esta investigación está compuesta de seis capítulos:

El capítulo I corresponde a la contextualización del problema, donde se indican los antecedentes, planteamiento y formulación del problema que dan origen a esta investigación, luego se fijan los objetivos tanto general como específicos que se deben cumplir para dar solución a este problema. Además, se realiza una delimitación temporal y espacial la cual indica el lugar donde se lleva a cabo la investigación y el tiempo empleado en el proyecto. Se justifica el proyecto indicando las razones y los resultados que se pretende con este proyecto. Finalmente contiene toda la metodología empleada en la investigación y desarrollo del proyecto, cuenta con aspectos como el tipo de investigación, métodos y técnicas empleadas para cumplir con los objetivos planteados.

En el capítulo II se incluye todo el marco teórico que fundamenta esta investigación y sirve de guía para desarrollar el presente proyecto. Aborda desde los conceptos básicos como el trasporte, movilidad, taxis, módulos de almacenamiento, hasta conceptos más específicos como GPS como funciona y sus aplicaciones, sensores que intervienen en la gestión del motor, cajas de cambio, freno, manejo de señales digitales y analógicas. También contiene información detallada de algunos componentes utilizados como lo es la placa arduino, la cámara OV2640, el conector OBDII donde se indican sus características principales usos y aplicaciones. Finalmente, se

explican los conceptos de programación que son una parte importante de este capítulo puesto que desde la placa Arduino requiere una programación específica la cual es fundamental para poder recolectar los datos necesarios, así mismo la programación de la interfaz en Labview donde se procesan los datos.

En el capítulo III se explica el procedimiento seguido para el diseño, construcción e implementación del módulo de almacenamiento en un vehículo Chevrolet Aveo. Partimos desde la selección de los diferentes componentes como lo son las placas Arduino Mega 2560, el receptor GPS, cámara, conector OBDII con acelerómetro integrado, módulo SD, pantalla TFT 3.2" touchscreen. La siguiente parte explica el proceso para obtener la información del estado de marcha y freno, así como los diagramas eléctricos y de conexiones. Posteriormente sigue el desarrollo del código de programación de las placas Arduino para que estas recolecten, procesen y almacenen todos los datos requeridos por el proyecto. Una vez obtenidos los primeros datos en base a estos se realiza la programación de la interfaz en Labview.

En el capítulo IV se muestran las diferentes pruebas de funcionamiento realizadas con el módulo de almacenamiento de datos ya instalado en el vehículo. Aquí se indica cómo funciona el módulo que datos se puede obtener, almacenar y mostrar en la pantalla, la rutina que se produce al accionar los botones de inicio, detención y reseteo. Posteriormente se importan los datos a la interfaz de Labview para decodificar esa información y poner a prueba una a una las funciones programadas con la que dispone esta interfaz, como lo son las gráficas comparativas, simulador, visualizador de fotos, tabla de datos y finalmente se genera el reporte en Excel que contiene todos los datos ordenados obtenidos y una herramienta de filtrado donde se puede seleccionar el intervalo de tiempo en el que se quiere analizar los datos.

Finalmente, el capítulo V corresponde a las conclusiones y recomendaciones que deja esta investigación, seguido de la bibliografía empleada para sustentar esta investigación.

INDÍCE

CARÁTULA i

AUTORIZACIÓN DE USO Y PUBLICACIÓN ii

CESIÓN DE DERECHOS DE AUTOR DEL TRABAJO DE GRADO ..... **¡Error! Marcador no definido.**

CERTIFICACIÓN ..... **¡Error! Marcador no definido.**

AGRADECIMIENTO vii

DEDICATORIA viii

RESUMEN ix

ABSTRACT x

INTRODUCCIÓN xi

INDÍCE xiii

ÍNDICE DE FIGURAS xix

ÍNDICE DE TABLAS xxiii

ÍNDICE DE ABREVIATURAS xxiv

CAPÍTULO I 1

1. Contextualización del problema. 1

1.1 Antecedentes. 1

1.2 Planteamiento del problema. 1

1.3 Formulación del problema. 2

1.4 Delimitación. 2

1.4.1 Temporal. 2

1.4.2 Espacial. 2

1.5 Objetivos. 2

1.5.1 Objetivo General. 2
1.5.2 Objetivo específico. 2
1.6 Justificación. 3
1.7 Tipo de Investigación. 3
1.7.1 Investigación Tecnológica. 3
1.7.2 Investigación Bibliográfica. 3
1.8 Métodos. 3
1.9 Técnicas e instrumentos. 4
CAPÍTULO II 5
2. Marco Teórico. 5
2.1 Transporte y movilidad. 5
2.1.1 Servicio de taxi. 5
2.2 Análisis y determinación de causas de accidentes de tránsito. 6
2.2.1 Causas de accidentes de tránsito. 7
2.3 Módulos de almacenamiento. 8
2.3.1 Dispositivos pasivos sin comunicación. 8
2.3.2 Módulos de almacenamiento con sistemas de control y monitoreo. 8
2.3.3 Módulos de almacenamiento en el mercado. 9
2.4 Sistema OBDII 9
2.4.1 Conector OBDII. 10
2.4.2 Tipos de conectores y distribución de pines. 10
2.4.3 Protocolos de comunicación. 12
2.4.4 Protocolos de vehículos tipo taxi de la ciudad de Ibarra. 12
2.4.5 Datos de diagnóstico disponibles OBDII. 13
2.4.6 PIDs del sistema OBDII. 14

2.5 Sistema GPS. 16
2.5.1 Modo de operación de un sistema GPS. 16
2.5.2 Receptores GPS. 17
2.5.3 Tipos de datos que puede proporcionar un sistema GPS 18
2.6 Arduino 19
2.6.1 Hardware. 20
2.6.1.1 Entradas y salidas. 20
2.6.1.2 Alimentación. 20
2.6.1.3 Memoria. 21
2.6.1.4 Puerto USB y protección de sobrecarga. 22
2.6.2 Software. 22
2.6.3 Lenguaje de programación Arduino. 23
2.7 Cámara. 26
2.8 Labview 27
2.8.1 Panel de frontal. 27
2.8.2 Diagrama de bloques. 28
2.8.3 Herramientas de Labview. 29
2.8.4 Estructuras en Labview. 30
2.8.5 Arreglos. 32
2.8.6 Tipos de datos en Labview. 33
2.9 Señales analógicas y digitales. 34
2.9.1 Señal analógica. 34
2.9.2 Digitalización de la señal analógica. 35
2.10 Sensores de gestión del motor. 36
2.10.1 Sensores de posición/velocidad. 36

2.10.1.1 Sensor inductivo. 36
2.10.1.2 Sensores de efecto hall. 38
2.10.2 Sensor de temperatura. 39
2.10.2.1 Sensor ECT (Sensor de temperatura del refrigerante del motor). 40
2.10.2.2 Sensor IAT (Sensor de temperatura del aire de admisión) 40
2.10.2.3 Sensor EOT (Sensor de temperatura del aceite del motor). 40
2.10.3 Sensor de flujo de aire (MAF). 41
2.10.4 Sensor de presión de aire (MAP). 41
2.11 Sistema de Freno. 41
2.11.1 Tipos de frenos. 42
2.12 Caja de cambios. 43
CAPÍTULO III 44
3. Diseño y construcción del módulo de almacenamiento de datos on-board. 44
3.1 Diseño del Hardware para el módulo de almacenamiento de datos. 44
3.1.1 Selección de la placa arduino. 46
3.1.2 Selección de Pantalla LCD. 47
3.1.3 Selección modulo GPS. 50
3.1.4 Selección de la cámara. 52
3.1.5 Sistema de adquisición de estado de marcha. 55
3.1.6 Sistema de adquisición de estado del freno. 58
3.1.7 Botones de operación. 60
3.2 Diseño de software para el módulo de almacenamiento de datos. 61
3.2.1 Fases de diseño 61
3.2.2 Programación para el módulo de almacenamiento de datos. 64
3.3 Diseño de la interfaz en Labview. 74

3.3.1 Panel frontal de la interfaz en Labview. 75
3.3.2 Configuración y filtrado de datos por hora. 77
3.3.3 Importado y almacenamiento de datos en la interfaz de Labview. 79
3.3.4 Graficas. 81
3.3.5 Simulador 83
3.3.7 Tabla de datos. 88
3.4 Adaptabilidad a sistemas instalados en taxis 91
3.5 Mejoras para el módulo de almacenamiento de datos. 92
CAPITULO IV 93
4. Pruebas de Funcionamiento y análisis de datos obtenidos. 93
4.1 Pruebas de funcionamiento de la pantalla. 93
4.2 Prueba de comunicación del OBDII. 94
4.3 Prueba de comunicación del sistema de estado de marcha y freno. 95
4.4. Prueba de comunicación del módulo GPS. 95
4.5 Prueba de comparación de valores. 96
4.7 Prueba de funcionamiento módulo de almacenamiento de datos 102
4.8 Prueba de funcionamiento de la cámara. 103
4.9 Uso de datos y dinámica del equipo para futuras aplicaciones. 105
4.9.1 Desempeño del vehículo 105
4.9.2 Hábitos de conducción 106
4.9.3 Ciclos de conducción. 107
4.10 Guía de Uso Módulo de Almacenamiento de datos On-board para taxis. 108
4.10.1 Introducción. 109
4.10.2 Objetivo del manual 109
4.10.3 Especificaciones técnicas. 109

4.10.4 Modo de instalación y conexión en el vehículo. 110
4.10.5 Modo de operación del módulo. 111
4.10.6 Importación de datos a la interfaz de Labview. 113
CAPITULO VI 119
5. Conclusiones y Recomendaciones. 119
5.1 Conclusiones. 119
5.2 Recomendaciones. 120
BIBLIOGRAFÍA 121
ANEXOS. 123

ÍNDICE DE FIGURAS

pág.

Figura 1. Vehículo tipo taxi. 6
Figura 2. Vehículo accidentado. 7
Figura 3. Ubicación del conector obdii de 16 pines..... 10
Figura 4. Tipos de conectores obdii según sae. 11
Figura 5. Receptor gps. 18
Figura 6. Placa arduino mega..... 19
Figura 7. Software ide arduino..... 23
Figura 8. Programa labview 2015. 27
Figura 9. Vista del panel frontal de labview. 28
Figura 10. Diagrama de bloques de labview..... 29
Figura 11. Estructura for loop en labview. 30
Figura 12. Estructura while loop de labview. 31
Figura 13. Estructura case en labview. 31
Figura 14. Estructura flat sequence de labview. 32
Figura 15. Nodo de fórmula en labview. 32
Figura 16. Tipos de datos usados en labview. 34
Figura 17. Forma de onda de una señal análoga simulada 35
Figura 18. Conversión analógico-digital..... 36
Figura 19. Señal de un sensor inductivo simulada en livewire. 37
Figura 20. Sensor inductivo de velocidad. 38
Figura 21. Esquema efecto hall. 39
Figura 22. Sensor ect..... 40
Figura 23. Diagrama de bloques (módulo de almacenamiento de datos). 44
Figura 24. Elementos que componen el módulo de almacenamiento. 45
Figura 25. Esquema de pines arduino mega 2560. 46
Figura 26. Pantalla tft 3.2”..... 48
Figura 27. Diagrama de la pantalla tft 3.2”..... 49
Figura 28. Receptor gps ubx-m8030ka. 51

Figura 29. Cámara. 52
Figura 30. Diagrama de conexión. 54
Figura 31. Descripción de un contacto normalmente abierto. 55
Figura 32. Diagrama de conexión del sistema del estado de marcha. 56
Figura 33. Resistencia pull down. 57
Figura 34. Resistencia pull up. 57
Figura 35. Esquema de conexión del sistema de estado de marcha. 58
Figura 36. Diagrama de conexión del sistema de estado del freno. 59
Figura 37. Cuadro de descripción de un contacto normalmente cerrado. 59
Figura 38. Esquema de conexión del sistema de estado del freno. 60
Figura 39. Esquema de conexión de los botones de operación. 60
Figura 40. Diagrama de proceso módulo/interfaz labview. 62
Figura 41. Flujograma de funcionamiento para módulo de almacenamiento. 63
Figura 42. Flujograma de funcionamiento para la interfaz en labview. 64
Figura 43. Flujograma de proceso para el módulo de almacenamiento de datos. 65
Figura 44. Librerías y variables incluidas en el código arduino. 66
Figura 45. Llamado de funciones mediante pids. 67
Figura 46. Flujograma de lectura de pids y adquisición de datos. 68
Figura 47. Código de llamado para la función del receptor gps. 69
Figura 48. Flujograma de adquisión del gps. 69
Figura 49. Código de llamado para la función del acelerómetro. 70
Figura 50. Flujograma de adquisición de datos del acelerómetro. 71
Figura 51. Código para la función de marchas y freno. 72
Figura 52. Flujograma de ejecución para el módulo de almacenamiento de datos 73
Figura 53. Flujograma de funciones para la interfaz de labview. 75
Figura 54. Panel frontal de la interfaz en labview. 76
Figura 55. Flujograma de configuración y filtrado de datos. 77
Figura 56. Configuración y filtrado por hora (vista panel frontal). 78
Figura 57. Código de filtrado de datos por hora. 79
Figura 58. Código para importar y almacenar los datos en labview. 80
Figura 59. Flujograma de importado y almacenamiento de datos. 81

Figura 60. Código en labview para graficar los datos. 82
Figura 61. Grafica de datos en el panel frontal. 82
Figura 62. Flujograma para graficar los datos en labview. 83
Figura 63. Flujograma de ejecución para el simulador. 84
Figura 64. Apariencia del simulador en el panel frontal. 85
Figura 65. Código principal del simulador. 85
Figura 66. Código del acelerómetro. 86
Figura 67. Código del indicador de marchas y freno para el simulador. 86
Figura 68. Visualizador de fotos (vista del panel frontal). 87
Figura 69. Código en labview para el visualizador de fotos. 87
Figura 70. Flujograma para el visualizador de fotos 88
Figura 71. Tabla de datos en labview (panel frontal). 89
Figura 72. Código de la tabla de datos en labview. 89
Figura 73. Código en labview para generar el reporte en excel. 90
Figura 74. Flujograma para generar la tabla de datos. 90
Figura 75. Grabador de audio hiletgo icsh041a 4 channel. 91
Figura 76. Portada del módulo de almacenamiento de datos. 93
Figura 77. Aspecto de la segunda pantalla. 94
Figura 78. Datos obdii a través de monitor serial 94
Figura 79. Datos sistema marcha-freno a través de monitor serial. 95
Figura 80. Datos gps a través de monitor serial 96
Figura 81. Mensaje de sincronizando en la pantalla. 99
Figura 82. Botón de inicio y test de mensaje en la pantalla. 99
Figura 83. Flujograma de ejecución del botón de inicio. 100
Figura 84. Botón de detención y test de mensaje en la pantalla. 100
Figura 85. Flujograma de ejecución del botón fin de grabación. 101
Figura 86. Botón de reinicio y test de mensaje en la pantalla. 101
Figura 87. Flujograma de ejecución del botón de reinicio 102
Figura 88. Funcionamiento módulo de almacenamiento 102
Figura 89. Archivo de texto generado por el módulo de almacenamiento. 103
Figura 90. Calidad de imagen (2mp). 104

Figura 91. Etapas de funcionamiento de la cámara. 104
Figura 92. Modelo de ciclo de conducción urbano. 107
Figura 93. Modelo de ciclo de conducción interurbano. 108
Figura 94. Módulo de almacenamiento de datos on-board. 108
Figura 95. Socket parte inferior izquierda 111
Figura 96. Pantalla de inicio. 112
Figura 97. Mensaje de sincronizando en la pantalla. 112
Figura 98. mensajes en la pantalla al presionar los botones de acción. 113
Figura 99. Pestaña de guía y configuración de la interfaz. 114
Figura 100. Path de selección de la carpeta de fotos. 114
Figura 101. Path de selección del archivo txt. 115
Figura 102. Herramienta de filtrado por hora. 115
Figura 103. Gráfica mixta (ventana gestión del motor). 116
Figura 104. Ventana gráficas comparativas. 116
Figura 105. Ventana del simulador. 117
Figura 106. Ventana del visualizador de imágenes. 117
Figura 107. Ventana de la tabla de datos en labview. 118

ÍNDICE DE TABLAS

pág.

Tabla 1. Causas de accidentes de tránsito. 7
Tabla 2. Distribución de pines conector obdii estandarizado. 11
Tabla 3. Protocolos de vehículos utilizados como taxis en ibarra. 13
Tabla 4. Lista de pids obdii. 14
Tabla 5. Funciones programa arduino. 24
Tabla 6. Tipos de datos en arduino. 25
Tabla 7. Comandos programables en arduino. 25
Tabla 8. Ejemplos de librerías en arduino 26
Tabla 9. Datos técnicos de la placa arduino mega 2560. 47
Tabla 10. Lista de pines usados por la pantalla tft 3.2” 49
Tabla 11. Conexiones del receptor gps ubx-m8030ka. 51
Tabla 12. Descripción de pines de la cámara ov2640. 53
Tabla 13. Pines en arduino usados para detectar cada marcha. 56
Tabla 14. Lista de pines programados en arduino para los botones de operación. 61
Tabla 15. Formato trama de datos de módulo de almacenamiento 74
Tabla 16. Unidades de los datos obtenidos por el módulo de almacenamiento 79
Tabla 17. Comparación de valores del módulo de almacenamiento. 96
Tabla 18. Botones de operación 98
Tabla 19. Tabulación de datos almacenados y sus unidades 105
Tabla 20. Hábitos de conducción 106

ÍNDICE DE ABREVIATURAS

ATMEL	Compañía desarrolladora de microcontroladores.
CAN	Protocolo de comunicación desarrollado por Bosch.
CKP	Sensor de posición del cigüeñal.
DC	Corriente continua.
ECT	Sensor de temperatura del motor.
ECU	Unidad de control electrónico.
EEPROM	Memoria borrable y programable eléctricamente.
GND	Conexión a tierra (Ground).
GPS	Sistema de posicionamiento global.
IAT	Sensor de temperatura del aire de admisión.
IDE	Entorno de desarrollo integrado.
I2C	Bus de datos serial.
ISO	Siglas de la Organización Internacional de Estandarización.
JPG	Formato de imágenes.
KB	Kilobytes.
LCD	Pantalla de cristal líquido.
MAF	Sensor de flujo de aire.
MAP	Sensor de presión de aire de admisión.
MISO	Salida de datos esclavo y entrada al master.
MOSI	Salida de datos master y entrada al esclavo.
NTC	Coeficiente térmico negativo.

OBDII	Siglas del sistema de diagnóstico a bordo en vehículos segunda generación.
PID	Mecanismo de control codificado por retroalimentación.
PTC	Coeficiente térmico positivo.
RPM	Revoluciones por minuto.
RX	Línea de lectura y recepción de datos en serie.
SAE	Siglas de la Sociedad de Ingenieros Automotrices.
SCK	Envío de pulsos sincronizados.
SCL	Reloj del SPI.
SDA	Línea de datos serial.
SPI	Bus de interfaz periférico.
SRAM	Memoria estática de acceso aleatorio.
TFT	Transistor de películas finas.
TPS	Sensor de posición de la mariposa de aceleración.
TX	Línea de trasmisión y escritura de datos en serie.
TXT	Formato de archivos de texto.
USB	Bus de conexión universal en serie.
UTC	Tiempo universal coordinado.
V	Voltios.
VI	Instrumento Virtual en Labview.
5V	Voltaje de alimentación soportado.

CAPÍTULO I

1. Contextualización del problema.

1.1 Antecedentes.

La carrera de Ingeniería en Mantenimiento Automotriz de la Universidad Técnica Del Norte desde sus inicios ha formado estudiantes en todos los aspectos técnicos necesarios para su desarrollo profesional en el campo laboral, pero sin dejar de lado el aspecto social, creando conciencia acerca de los problemas que guardan relación directa con el campo automotriz como son: la contaminación y los accidentes de tránsito a los que se tienen que dar solución lo más pronto posible. Si bien la tecnología Automotriz ha ido avanzando en lo que tiene que ver con reducción de contaminación, confort y seguridad, no se da apertura total ya que existen conflictos de intereses que retrasan este proceso por ello no se lo difunde de manera correcta y su implementación va a pasos muy lentos.

Los aumentos considerables de accidentes vehiculares de unidades de taxis para el transporte urbano, ha llegado a ser un problema muy común entre la población que usa este sistema de transporte para movilizarse hacia los diferentes destinos dentro de la provincia, por esta razón es necesario realizar un monitoreo y seguimiento diario de todos los parámetros que puedan afectar a un buen funcionamiento del motor del taxi y así de esta manera disminuir el porcentaje de accidentes de tránsito por fallas mecánicas del vehículo o impericia del conductor.

Actualmente las cooperativas de taxis de Ibarra y del resto del país no cuentan con un dispositivo que almacene información de funcionamiento del vehículo. Esta información es de gran utilidad para determinar condiciones de manejo, causas accidentes de tránsito, etc.

1.2 Planteamiento del problema.

Al momento se cuenta con herramientas e instrumentos de medición de parámetros de funcionamiento de la parte electrónica de un vehículo, pero no se posee una herramienta que permita obtener y almacenar esta información. Al no contar con un dispositivo que almacene estos datos de funcionamiento del vehículo, no se puede dar seguimiento y control a las unidades de taxi.

Los datos obtenidos de este dispositivo pueden ser utilizados para determinar el desempeño del vehículo, condiciones de manejo e incluso puede servir al personal que realiza peritajes para determinar las causas por las que se produjo un accidente.

1.3 Formulación del problema.

¿Cómo implementar un módulo on-board que almacene datos de funcionamiento del vehículo, mediante Arduino-Labview para taxis?

1.4 Delimitación.

1.4.1 Temporal.

Este proyecto se llevará a cabo durante el periodo académico correspondiente al mes de junio del 2015 hasta el mes de febrero del 2017.

1.4.2 Espacial.

Este proyecto se llevará a cabo en la provincia de Imbabura, ciudad de Ibarra.

1.5 Objetivos.

1.5.1 Objetivo General.

- Implementar un módulo que almacene de manera sincronizada los datos de gestión electrónica del vehículo (ECU), GPS y registro fotográfico, para su aplicación en taxis, mediante las plataformas Arduino – Labview.

1.5.2 Objetivo específico.

- Diseñar un módulo de adquisición y almacenamiento de datos gestión electrónica, posición de marcha y estado de freno mediante la plataforma Arduino Mega.
- Integrar un sistema GPS para proporcionar datos de ubicación y ruta, además de una cámara que permita obtener un registro fotográfico de viaje.
- Desarrollar una interfaz mediante el Software Labview que permita al usuario visualizar y analizar los datos obtenidos de manera sincronizada y en tiempo real.

1.6 Justificación.

Este proyecto permite la implementación de un módulo de almacenamiento de datos on board para taxis, para la recolección y almacenamiento de datos de funcionamiento del motor, posición de marcha y estado de freno. Facilitando a los dueños de taxis realizar un seguimiento y control de sus unidades.

1.7 Tipo de Investigación.

Para el presente proyecto se hace uso tanto de la investigación tecnológica, así como de la investigación bibliográfica.

1.7.1 Investigación Tecnológica.

Este proyecto considera este tipo de investigación debido a que utiliza varios elementos tecnológicos para construir e implementar este módulo de almacenamiento de datos on-board para taxis. Se hace uso de equipos, herramientas y elementos tecnológicos existentes en el mercado para darles una nueva aplicación como lo es este módulo, así mismos programas como el que se requiere para programar la placa y el que servirá como interfaz para el usuario donde podrá visualizar toda la información recolectada por el módulo.

1.7.2 Investigación Bibliográfica.

Parte importante de este proyecto es este tipo de investigación bibliográfica, debido a que se recurre a libros y manuales para poder recolectar la información necesaria para realizar su fundamentación teórica y el uso de los programas de Arduino y Labview, de la misma forma cada equipo y herramienta poseen documentos que provee de información acerca de sus características e instrucciones de operación. Incluso herramientas como el internet permiten recolectar información como conceptos y definiciones empleados en el presente proyecto. Todas estas fuentes mencionadas permiten profundizar en los conocimientos de programación, utilización y sustentar su base teórica.

1.8 Métodos.

Así mismo este tipo de investigación cuenta con métodos empleados para su desarrollo como lo son: diseño, método analítico- sintético, programación, implementación, prueba-error.

a) **Diseño. -** Para realizar el Módulo de almacenamiento de datos on-board, se requiere tener un concepto e ideas claras de cómo será cuando haya culminado, así también que cosas necesita para hacerlo. Todo esto cabe dentro de la etapa de diseño.
b) **Método analítico-sintético. –** Este método se aplica ya que se requiere sustentar las bases teóricas y los conocimientos haciendo uso de fuentes de información como: libros, manuales, internet y todo aquello que proporcione información útil para este proyecto.
c) **Programación. -** Para que un módulo de estas características pueda recabar toda la información que se requiere, se debe declarar todas estas instrucciones de funcionamiento a través de un método de programación adecuado.
d) **Implementación.** - Si se toma en cuenta que este es un módulo creado para un tipo de vehículo en este caso taxis, se debe buscar la forma de cómo integrarlo a ese vehículo.
e) **Prueba-error. –** Una vez culminadas las anteriores etapas, se requiere poner a prueba el módulo, para conocer si el módulo es capaz de receptar, procesar y almacenar la información tal como se prevee que lo haga. En caso que ocurra algún fallo se debe corregir y continuar con las pruebas, hasta que se garantice su correcto funcionamiento.

1.9 Técnicas e instrumentos.

En el presente proyecto se emplearon técnicas como:

a) **Adaptación. –** Esta técnica se emplea puesto que todos los equipos usados requieren ser modificados para que puedan integrarse y trabajen conjuntamente, pasando a conformar lo que viene a ser el módulo. De la misma manera en el vehículo que se pretende implementarlo se requiere realizar unas pequeñas modificaciones.
b) **Análisis de datos. –** Todos los datos obtenidos a través de este deben ser analizados para comprobar si cumplen con el objetivo para el cual fue diseñado el módulo.
c) **Pruebas de funcionamiento. –** El módulo debe ser puesto a prueba en reiteradas ocasiones en busca de errores que puedan afectar su funcionamiento correcto. De encontrar fallos se debe tomar los correctivos respectivos.

CAPÍTULO II

2. Marco Teórico.

2.1 Transporte y movilidad.

La movilidad se entiende el deseo o necesidad de una persona de moverse de un lugar a otro. La cual debe ser satisfecha ya que se considera un derecho de cada ciudadano y esta no debe repercutir negativamente en la calidad de vida ni en el desarrollo económico, cultural y educativo. Por ello el transporte como tal es imprescindible dentro de una sociedad y el desarrollo de sus actividades económicas.

Conforme siga incrementándose la población seguirá tomando mayor importancia, por ello se hace necesario contar con sistemas de transporte complejos y adaptados a las necesidades para garantizar la movilidad de las personas y mercancías de una forma económica, eficientemente y segura.

El transporte no es necesario a nivel elemental de subsistencia. Si no que surge con el desarrollo económico y social de la comunidad y los países, por ello el volumen de transporte crece imparablemente. Pero esto no asegura su existencia, si no que deberá hacerlo con calidad y a costo competitivo. (Torres Mikel, 2013)

2.1.1 Servicio de taxi.

Denominado aquel vehículo de servicio de alquiler, utilizado para el transporte de un grupo máximo de 4 personas en diferentes rutas urbanas y rurales. Los sitios en el cual el vehículo recoge y finaliza su carrera se encuentran determinadas por el cliente, y de esta manera se determina el costo del viaje mayormente por sistema con taxímetro o por el contrario por criterio del conductor (Figura 1.).

Figura 1. Vehículo tipo taxi.

2.2 Análisis y determinación de causas de accidentes de tránsito.

El análisis para determinar las causas de un accidente de tránsito se entiende como la necesidad de identificar una serie de eventos que han estado presentes y conllevan a un incidente o accidente vehicular. De esta manera investigar y optar por seleccionar las medidas correctivas eficientes para evitar la manifestación y repetición de consecuencias más graves.

El análisis permite con una muy elevada fiabilidad conocer cómo aconteció el suceso y sus causas reales, el cual es el objetivo primordial del análisis ya que con el parte policial realizado del accidente no se alcanza la severidad del suceso. Proporciona de igual manera a las entidades encargadas de la movilidad fomentar su base para conocer la probabilidad de repetición y la gravedad potencial de los daños ocasionados tanto materiales como humanos y de esta manera considerar los factores tiempo y dinero que será invertidos para la implantación de medidas correctivas para disminuir los accidentes.

Figura 2. Vehículo accidentado.

Fuente:(Diario el Norte,2016)

2.2.1 Causas de accidentes de tránsito.

La entidad encargada de la movilidad determina una agrupación de causas entre las cuales cabe destacar 3 principales: imprudencia, irrespeto a las señales de tránsito y otras causas importantes. Como se puede observar en la tabla a continuación.

Tabla 1. Causas de accidentes de tránsito.

***POR IMPRUDENCIA**
• Del conductor
• De los pasajeros
• Del peatón
***IRRESPETO DE SEÑALES DE TRÁNSITO**
• Luz roja-amarilla (semáforo)
• Señal del vigilante
• Disco pare
• Prohibiciones de giros en U
• Prohibiciones de ingreso
***OTRAS CAUSAS IMPORTANTES**
• Exceso de velocidad
• Estado de embriaguez
• Impericia

• Órgano de seguridad en mal estado
• Casos fortuitos
• Distracción
• Causas desconocidas
• Otros

Fuente:(ANT,2016)

2.3 Módulos de almacenamiento.

Los módulos de almacenamiento de datos son dispositivos plug and play que guardan datos a través de la interacción entre un fenómeno físico o eléctrico y un software. Este facilita el acondicionamiento de las señales obtenidas y en algunos casos convierte la señal de analógica a digital para poder ser monitoreada más fácilmente por el software controlador que posee el computador, las recibe y las almacena en la memoria del módulo de almacenamiento de datos. Estos dispositivos pueden ser de 2 tipos:

- Sin comunicación
- Trasmisión de datos directa.

2.3.1 Dispositivos pasivos sin comunicación.

Poseen un funcionamiento similar al de una caja negra de un avión. Se instalan en el vehículo y almacenan cada cierto intervalo de tiempo programable. Recogen datos de ubicación, rumbo y velocidad del vehículo, indicando la fecha y hora. Estos datos son analizados posteriormente cuando el vehículo regresa y deben ser descargados directamente del dispositivo debido a que no tienen comunicación directa con el centro de control.

2.3.2 Módulos de almacenamiento con sistemas de control y monitoreo.

Estos módulos permiten la captura, registro y monitoreo de la cualquier variable del vehículo en tiempo real. Lo cuales permiten mejorar la eficacia y eficiencia en el transporte ofreciendo seguridad y confort a los usuarios. Para ello hacen uso de comunicaciones remotas para la trasmisión de datos tales como: GSM, GPRS, UMTS, HSDPA, WIFI.

2.3.3 Módulos de almacenamiento en el mercado.

a) **Drive Right 600e. -** Es una herramienta de gestión de flota para toda clase de vehículos. Trabaja con 2 dispositivos para obtener y almacenar la información, el drive Right y el carchip fleet. El carchip fleet va montado en el conector OBD II y puede acceder a todos los datos de gestión del motor, mientras que el drive Right se encargará de los registros de tiempo.

b) **TTX Data logger. -** Este módulo de almacenamiento es utilizado por Audi para la grabación de datos, así como el análisis de fallos durante las pruebas de funcionamiento. Este dispositivo cumple con los criterios para un registrador de datos de automoción. Tales como la facilidad de uso, así como la flexibilidad por medio de una amplia gama de interfaces y acompañado de un software completo adecuado para entornos de automoción.

c) Cosmos. - Este es un sistema de almacenamiento y monitoreo On board que trabaja conjuntamente con el sistema Atenea para trasmitir los datos en tiempo real. Destinado a cubrir una gran cantidad de necesidades, capaz de adaptarse a cada aplicación y proyecto en particular. Permite supervisar los parámetros del vehículo, registrar y almacenar esta información en bases de datos gestionadas. Este sistema se lo ha puesto en prueba en algunos países para monitorear como son las condiciones de manejo de cada conductor y en base a eso se les realiza el pago del salario.

2.4 Sistema OBDII

La sigla “OBD” se refiere al autodiagnóstico e informe de los diferentes parámetros de funcionamiento electrónico de un vehículo. Este sistema de diagnóstico a bordo brinda una ayuda fundamental tanto al propietario como a técnicos de taller de reparación de vehículos a conocer condiciones y estados de los sistemas del vehículo. Este sistema de diagnóstico a bordo ha estado sujeto a varios cambios para mejorar su estructura en cuanto a su funcionamiento.

Ahora implementa puertos de comunicación digital de 16 pines para la obtención de datos de funcionamiento del vehículo en tiempo real, integrando también la función de presentar una serie de códigos de fallos de diagnóstico estandarizados para todas los de vehículos, para permitir identificar el fallo presentado en el vehículo sin la necesidad de herramientas y softwares

específicos para cada marca automotriz. En el caso de presentar algún fallo informa al conductor a través de la luz de advertencia (check-engine) para que se tomen los correctivos necesarios.

2.4.1 Conector OBDII.

La ubicación de este conector en la mayoría de los vehículos se encuentra dentro del habitáculo, para facilitar la conexión de un implemento de diagnóstico automotriz y evitar inconvenientes con otros sistemas del motor al momento de realizar el análisis de fallos. Normalmente el conector OBDII suele encontrarse ubicado en la zona de los pies del conductor (Figura 3.), en la parte central del tablero, e incluso en la gaveta del tablero del vehículo.

Figura 3. Ubicación del conector OBDII de 16 pines.

2.4.2 Tipos de conectores y distribución de pines.

Para los conectores OBDII se ha determinado especificaciones dentro de la norma SAE J1962, la cual ofrece 2 interfaces estandarizadas distintas denominadas “Conector Tipo A” y “Conector Tipo B” ambos de tipo Hembra de 16 pines, ubicados en dos filas, cada una de 8 pines separadas por una ranura a excepción de la “Tipo B” la cual posee la ranura con un espacio en medio que la separa. Por esta diferencia un conector de “Tipo A” no se puede colocar en un conector de diagnóstico “Tipo B”, pero si se puede colocar un conector “Tipo B” en un “Tipo A”. El conector “Tipo A” es utilizado en aquellos vehículos que utilizan corriente de 12V mientras que el conector “Tipo B” es utilizado para vehículos con corriente de trabajo de 24V y posee una coloración azul (Figura 4.).

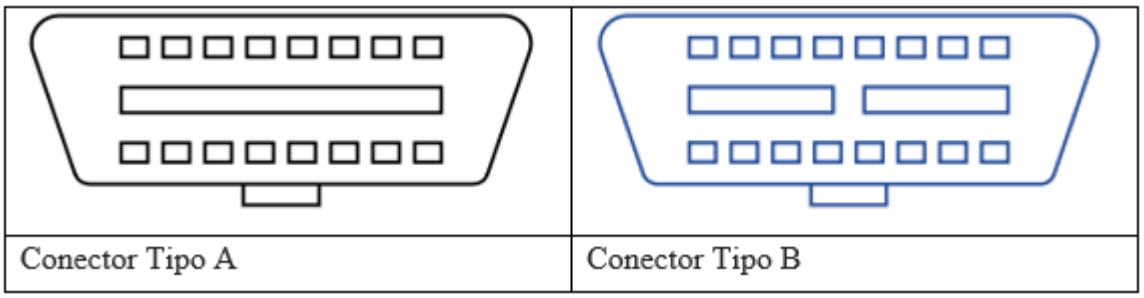


Figura 4. Tipos de conectores OBDII según SAE.

Cada marca hace uso de manera distinta de estos pines, aunque tampoco hace uso de todos ellos. Los únicos pines que son comunes entre los distintos protocolos son los que corresponde a tierra pines (4,5) y el de alimentación en el pin 16. La especificación SAE J1962 define los pines de salida de los conectores de la siguiente manera como se ve en la (Tabla 2)

Tabla 2. Distribución de pines conector OBDII estandarizado.

Pin 1	Reservado para el fabricante.
Pin 2	J1850 Bus+.
Pin 3	Reservado para el fabricante
Pin 4	Tierra chasis
Pin 5	Tierra señal.
Pin 6	Can/ High, J-2284.
Pin 7	Línea K, ISO 9141-2 y ISO/DIS 14230-4
Pin 8	Reservado para el fabricante.
Pin 9	Reservado para el fabricante
Pin 10	J1850 Bus-.
Pin 11	Reservado para el fabricante.
Pin 12	Reservado para el fabricante.
Pin 13	Reservado para el fabricante.
Pin 14	Can/Low, J2284-4.
Pin 15	Línea L, ISO 9141-2 y ISO/DIS 14230-4.
Pin 16	Alimentación batería.

Fuente: (SAE)

2.4.3 Protocolos de comunicación.

La forma en que cada marca hace uso de los pines del conector define varios protocolos con distinta forma de decodificar la información obtenida por el sistema. Existe variedad de protocolos que solo son variaciones, sin embargo, entre los más importantes tenemos los siguientes:

a) **ISO 9141-2. -** Lo usan la mayoría de vehículos asiáticos y europeos, e incluso Chrysler. El tipo de dato enviado en este protocolo es asíncrono de una sola dirección en el cual se envían datos sin previo aviso y el receptor debe estar preparado para aceptar los datos. En este protocolo se hace uso del pin 5 (tierra) ,16 (batería) y 7 (datos).
b) **ISO 14230. -** Este es el protocolo estándar de la norma europea EOBD, principalmente usado por Renault y descendientes de Opel comercializados por GM. Usa los pines 5 (tierra), 16 (batería), 7 (datos 1) y 15 (datos 2).
c) **SAE J1850 VPW. -** Usado por la marca automotriz GM en todos sus vehículos. Donde el ancho del pulso es siempre variable con un ralentí bajo y un voltaje máximo de 7v. Los pines usados en este protocolo son 5 (tierra) ,16 (batería) y 2 (datos).
d) **SAE J1850 PWM. -** Lo usa exclusivamente Ford en sus modelos OBDII, este protocolo encontramos una modulación de ancho de pulso que no se puede variar a nuestro gusto como el VPW. Soporta un voltaje de 5v al igual que el resto de protocolos OBDII. Los pines usados en este protocolo son 5 (tierra).16 (batería), 2 (datos1) y 10 (datos2).
e) **CAN (ISO 15765). -** Este es la variación del protocolo ISO con la adicción e intercomunicación del sistema CAN-bus aplicada en gran variedad de vehículos asiáticos y europeos. Usa los pines 5 (tierra), 16 (batería), 6 (datos high) y 14 (datos low).

2.4.4 Protocolos de vehículos tipo taxi de la ciudad de Ibarra.

En la ciudad de Ibarra existe una gran variedad de vehículos utilizados como taxis y cada una posee un protocolo de comunicación diferente. Las marcas de vehículos más comunes en la ciudad de Ibarra son: Kia, Hyundai, Chevrolet, Nissan, y en menor cantidad Renault, Mazda, Toyota, Greatwall. A continuación, se muestra una tabla de los vehículos utilizados como taxis y su protocolo de comunicación.

Tabla 3. Protocolos de vehículos utilizados como taxis en Ibarra.

Marca	Modelo	Protocolo de comunicación
Chevrolet	Aveo	ISO 14230
	Spark	ISO 14230
	Optra	ISO 14230
	Sail	ISO 14230
KIA	Xcite	ISO 9141_2
	Rio	ISO 9141_2
	Stylus	ISO 9141_2
	Cerato	ISO 9141_2
Nissan	Sentra	ISO 9141_2
	Almera	ISO 9141_2
Hyundai	Accent	ISO 9141_2
	Elantra	ISO 9141_2
	I10	ISO 9141_2
Mazda	Alegro	ISO 9141_2
	3	ISO 9141_2
Renault	Sandero	ISO 14230
Toyota	Yaris	ISO 9141_2
Greatwall	C30	ISO 9141_2

2.4.5 Datos de diagnóstico disponibles OBDII.

OBDII permite tener el acceso a todos los datos del módulo de control del motor (ECU) y así tener una fuente de información muy concreta para lo solución de problemas presentado en los vehículos. La norma SAE J1979 ha definido códigos para la obtención de datos de diagnóstico de la ECU y los parámetros estándar de funcionamiento. SAE J1979 pone a disposición una lista de parámetros determinados con números de identificación denominados PID.

La lista de PIDs que presenta la norma SAE J1979 es muy numerosa, pero dentro de ella se presenta una lista básica de PIDs para el funcionamiento de un motor de inyección electrónica convencional en la cual se determina su código, función y fórmula para convertirlo en una señal

a la salida del conector OBDII. Los fabricantes de automotores no tienen la obligación de poner a disposición todos los PIDs que determina la norma SAE J1979, ya que esto permite tener el acceso a datos del sistema en tiempo real, por tal motivo son ellos los que determinan que PIDs permitirán obtener la información mediante el conector OBDII.

2.4.6 PIDs del sistema OBDII.

Los PIDs se definen como códigos empleados para la obtención de datos del vehículo. Estos códigos son muy útiles ya que se utilizan como la herramienta de diagnóstico fundamental en el proceso de reconocimiento del fallo electrónico presentado. La norma SAE J1979 dispone de una lista grande de PIDs. Cada fabricante de automotores puede agregar muchos más PIDs ya que poseen códigos de sensores colocados en una marca especifica de vehículos.

Cualquier técnico automotriz puede hacer uso de un implemento de diagnóstico que tenga conexión OBDII para el vehículo, y gracias a esta herramienta acceder a los PIDs, en este caso el no podrá ver que código corresponde a cada PID, sino poder observar solo el nombres y valores de PIDs. La tabla presentada a continuación muestra todos los PIDs estándar que envían información a través del conector OBDII según la norma SAE J1979. Cada PID posee su codificación, conjuntamente con la información acerca de cómo se genera la conversión para la obtención de valor de señal de cada sensor (Tabla 4.).

Tabla 4. Lista de PIDs OBDII.

PID (hex)	**Data bytes**	**Descripción**	**Min valor**	**Max valor**	**Unidades**	**Fórmula**
04	1	Carga del motor	0	100	[%]	A*100/255
05	1	Temperatura del motor	-40	215	[°C]	A-40
0A	1	Presión de combustible	0	765	[kPa]	A*3
0B	1	MAP	0	255	[kPa]	A
0C	1	RPM	0	16383	[min^{-1}]	((A*256)+B)/4

Continúa

0D	1	Velocidad del vehículo	0	255	[km/h]	A
0E	1	Tiempo de avance	-64	63.5	[%]	(A-128)/2
0F	1	Temperatura del aire de admisión (IAT)	-40	215	[°C]	A-40
10	1	Flujo de aire de admisión (MAF)	0	655.35	[g/s]	((A*256)+B)/100
11	1	Posición de la mariposa de aceleración	0	100	[%]	A*100/255
2D	1	EGR error	-100	99.22	[%]	(A-128)100/128
2E	1	EVAP	0	100	[%]	A*100/255
2F	1	Nivel de combustible	0	100	[%]	A*100/255
42	1	Voltaje del módulo de control	0	65.53	[V]	((A*256)+B)/1000
43	1	Valor de carga absoluta	0	25.700	[%]	((A*256)*100)/255
46	1	Temperatura ambiente	40	215	[°C]	A-40
4C	1	IAC	0	100	[%]	A*100/255
52	1	% de etanol en el combustible	0	100	%	A*100/255
59	1	Presión del riel de combustible (absoluta)	0	655.350	[kPa]	((A*256)+B)*10
5B	1	Estado de la	0	100	[%]	A*100/255

		batería hibrida				
5C	1	Temperatura de aceite del motor	-40	210	[°C]	A-40
5D	1	Tiempo de inyección del combustible	-210	301.99	[°]	(((A*256)+B)-26680/128
5E	1	Consumo de combustible	0	3212.75	[l/h]	((A*256)+B)*0.55
61	1	% Torque del motor	-125	125	[%]	A-125
62	1	% Torque del motor actual	-125	125	[%]	A-125

Fuente: (Norma SAE J979)

2.5 Sistema GPS.

El GPS es un sistema de satélites que permite determinar la posición de cualquier objeto las 24 horas del día, en cualquier lugar del mundo en cualquier condición climática. Consiste en un conjunto de 24 satélites que rodean la tierra y envían señales de radio a la superficie y que cualquier receptor GPS puede captar estas señales para calcular su posición con un error máximo de 15 metros producidos por ruidos, desajustes, dilución geométrica, disponibilidad selectiva y retrasos atmosféricos.

2.5.1 Modo de operación de un sistema GPS.

El modo de operación de este sistema se basa en la medición de la distancia entre el receptor y el conjunto de satélites. Cada satélite envía permanentemente su posición exacta con respecto a la tierra, de este modo el receptor puede usar la posición del satélite para determinar la posición del mismo. De esta forma si se conoce la distancia y la posición del satélite, se puede trazar un circulo imaginario en la superficie de la tierra dentro del cual se encuentra el receptor, si tomamos tres círculos de posición de otros tres satélites distintos, estos círculos se interceptan formando una zona en cuyo centro se encuentra el receptor. (Arnalich Santiago; Urruela Julio, 2012)

Esta situación puede expresar de 2 formas:

a) **Situación en dos dimensiones (2D). -** Esta situación solo permite conocer la longitud y latitud del receptor. Este modo de determinar la situación con los círculos de posición de 3 satélites solo es posible cuando se conoce la altura, lo cual solo se produce a nivel del mar. Sin embargo, en la mayoría de situaciones no se conoce la altura para poder determinar la posición. En este caso se requiere establecer un posicionamiento en tres dimensiones para determinar la posición.
b) **Situación en tres dimensiones (3D). -** Los 3 satélites que bastaban para determinar una situación de dos dimensiones, no son suficientes para indicar la posición cuya altura no se conoce. Entonces se requiere un cuarto satélite que situé al receptor en tres dimensiones: latitud, longitud y altitud. El tiempo empleado para que la señal llegue al receptor define una esfera alrededor del satélite, en alguna parte de dicha esfera se encuentre el receptor, el segundo satélite define otra esfera y la intercepción de estas dos esferas define un circulo en el cual se halla el receptor, este círculo se intercepta con la esfera de un tercer satélite determinando 2 puntos de referencia. Dada esta situación el receptor requiere la intervención de un cuarto satélite que le permita eliminar uno de los puntos y ajustar el reloj, permitiendo que la intersección de las 4 esferas defina un volumen lo más reducido posible.

2.5.2 Receptores GPS.

Dentro del mercado existen gran variedad de receptores GPS. Los avances realizados en circuitos integrados y métodos de producción, han hecho que pasemos de grandes unidades a receptores cada vez más pequeños, más potentes y con amplias características con lo que podremos elegir el que mejor se adapte a nuestras necesidades. Estos receptores son capaces de detectar, receptar, decodificar y procesar las señales que reciben de los satélites permitiéndole determinar la posición de cualquier objeto en cualquier parte del mundo (Figura 5). Existen dos tipos de receptores:

a) Fijos. – Son aquellos que se instala en cualquier clase de vehículos.
b) **Portátiles. –** los portátiles pueden ser tan pequeños como un celular y se lo puede llevar a cualquier lugar.

Figura 5. Receptor GPS.
Fuente:(Arnalich Santiago; Urruela Julio, 2012)

2.5.3 Tipos de datos que puede proporcionar un sistema GPS.

a) **Posición. -** Determinar la posición es la función principal de un sistema GPS, el cálculo del mismo se lo realiza mediante triangulación, para ello se toma el valor de 3 puntos fijados como referencia. Los 3 valores corresponden a la distancia del receptor respecto del satélite 1, la distancia del satélite 1 respecto al satélite 2 y la distancia del satélite 2 respecto al satélite 3. Cada uno forma una circunferencia de rango y cuya intersección determina la posición del receptor. La precisión de la posición varía según el tipo de filtrado y los algoritmos de posicionamiento utilizados por el receptor. Algunos receptores pueden usar la información de más satélites con el fin de precisar sus cálculos y eliminar aquellos datos poco precisos.
b) **Velocidad. -** La velocidad de desplazamiento en un sistema GPS se puede hacer de varias maneras. El método más común y sencillo consiste en calcular la velocidad con los valores de dos mediciones sucesivas de la posición. Este método posee un error medio de 3.5 km/h que afecta a todos los receptores, introducido por la restricción de la disponibilidad selectiva. El segundo método hace uso del efecto Doppler que consiste en el desfase de la frecuencia de las señales recibidas en función de la velocidad relativa del receptor respecto del satélite. Este método es más preciso, sin embargo, mucho dependerá de la calidad del receptor. La mayor parte de receptores utilizan una combinación de ambos métodos para obtener un dato de mayor precisión.
c) **Hora y fecha. -** El receptor GPS puede determinar la hora y fecha si se conoce la posición geográfica exacta del receptor, tomando la señal de los satélites los cuales son

monitoreados y ajustados desde las estaciones de control terrestres. Al conjunto de relojes de los satélites y estaciones se los denomina reloj compuesto (C.C), el error de precisión de este conjunto es inferior a un microsegundo.

a) **Latitud, longitud y altitud. –** Cada satélite informa permanentemente su posición respecto de la tierra. De este modo al ubicar al receptor en la zona de intersección de las esferas de posición de los satélites, podrá calcular cuáles son sus coordenadas respecto a la superficie de la tierra. La altitud se determina con el tiempo que tarda en llegar la señal desde el satélite al receptor.

2.6 Arduino.

Es una plataforma electrónica que posee hardware y software de código abierto para múltiples aplicaciones y usos. Esta placa es capaz receptar señales de entrada desde una variedad de sensores y puede ser utilizado para controlar una serie de dispositivos. Consta básicamente de un circuito impreso que contiene un microcontrolador de la marca ATMEL, el microcontrolador de la placa se programa usando el software provisto por los mismos desarrolladores de la esta placa, el cual usa un lenguaje C++. Por medio de este dispositivo se pueden crear un sinnúmero de aplicaciones interactivas, obtener datos y enviar comandos.

Arduino es una plataforma electrónica y de programación en arquitectura abierta con amplia gama de aplicaciones en ciencias exactas e ingeniería. Hoy en día Arduino se considera una poderosa herramienta tecnológica en la industria, universidades y centros de investigación. (Cortez, Fernando; Monjaraz Jaime, 2015).

Figura 6. Placa Arduino Mega.

Fuente:(Arduino store, 2015)

2.6.1 Hardware.

Esta placa es una sencilla, así como versátil plataforma con entradas y salidas tanto analógicas como digitales, cuyo elemento principal es el microcontrolador Atmega 328 integrado, un chip sencillo y barato, que hace que esta placa sea muy utilizada para llevar a cabo una gran cantidad de proyectos y aplicaciones en diferentes campos. Para usar este tipo de placas se tiene que seleccionar de acuerdo a los requerimientos de la aplicación que se le quiere dar, puesto que existen gran variedad de placas que se diferencian por la cantidad de pines y la velocidad de procesamiento.

2.6.1.1 Entradas y salidas.

Según la aplicación que se le quiera dar, hay disponibles placas con mayor número de pines programables y prestaciones, en este caso se toma como referencia la placa Arduino Mega. Esta placa posee los siguientes pines programables:

a) **Pines digitales. -** Esta placa posee 14 pines digitales los cuales son alimentados con 5v. Cada pin puede proporcionar o recibir un máximo de 40 mA y poseen una resistencia pull up (desconectada por defecto) de 20-50 kΩ.
b) **Pines analógicos. -** También poseen 6 pines PWM (modulación por ancho de pulsos) de lectura y escritura analógicas. Mediante estas entradas se puede obtener señales de sensores en forma de variaciones continuas de voltaje. Estas señales son trasladadas a un conversor analógico/digital de 10 bits.
c) **RX y TX. -** Estos pines tienen unas funciones especiales que son las de trasmitir datos TTL en serie, estos pines están conectados a los pines correspondientes del chip FTDI USB-a-TTL serie.

2.6.1.2 Alimentación.

Esta placa puede ser alimentado a través del puerto USB o puede provenir de una alimentación externa desde un adaptador de AC y DC e incluso desde una batería. Pero el suministro a la placa no debe exceder los 12 v. Si se usa más de 12 V, el regulador de tensión puede sobrecalentarse y dañar la placa. El rango recomendado es de 7 a 12 voltios. Los pines de alimentación son los siguientes:

a) **VIN. -** Este pin se puede usar si la tensión de entrada a la placa Arduino va a ser suministrada desde una fuente de alimentación externa.
b) **5V. -** Este pin se lo utiliza para el suministro regulado de energía usado para alimentar al microcontrolador y otros componentes de la placa. Este puede venir o desde VIN a través de un regulador en la placa, o ser suministrado por USB u otro suministro regulado de 5 V.
c) **3V3. -** Este pin proporciona un voltaje de 3.3 V generado por el chip FTDI de la placa. La corriente máxima es de 50 mA.
d) **GND. -** Este pin corresponde a la masa o tierra requerida para cerrar el circuito.
e) **Entrada de fuente de alimentación. -** Permite conectar un adaptador a la placa el cual puede recibir un voltaje de 7 a 12v DC.

2.6.1.3 Memoria.

La placa Arduino Mega posee un microcontrolador ATMEGA 328 en un circuito integrado de alto rendimiento con 268 KB de memoria con la capacidad de leer mientras escribe. La memoria dentro de la placa se distribuye en 3 tipos de memoria que cumple varias funciones, además se puede adicionar memoria extra a través de módulos o pics SD. Los tipos de memoria son los siguientes:

a) **Memoria Flash. -** Esta memoria está destinada para almacenar el código y la programación, para ello se destina 256 KB de la memoria total de la placa, de los cuales 8 KB se usan para el arranque del dispositivo.
b) **Memoria SRAM (Static Random Access Memory). -** Está viene a ser una memoria de almacenaje temporal ya que solo almacenara datos mientras se mantenga alimentada sin necesidad de refrescamiento, para esta memoria se dispone de 8 KB de la memoria total.
c) **EEPROM (Electrically Erasable Programmable Read-only Memory). -** Es una memoria que solo puede ser borrada y programada a través de pulsos eléctricos también, se la conoce como memoria no volátil y es que pese a ser desconectada la energía que alimenta el dispositivo los datos almacenados en esta memoria no serán borrados y permanecerán intactos. Esta memoria tiene destinado para su empleo 4 kb de la memoria total. El resto de la memoria de la placa se lo emplea para el procesamiento de datos y señales.

d) **SD Shield. -** Cuando se requiere almacenar una gran cantidad de datos, y la memoria nativa de la placa Arduino no es suficiente. Se debe optar por una SD Shield que es un módulo TF compatible con memoria SD y de esta manera ampliar la capacidad de memoria.

2.6.1.4 Puerto USB y protección de sobrecarga.

Esta placa posee un módulo adaptador USB-serie que permite programar el microcontrolador desde cualquier PC o laptop de manera más cómoda, también permite hacer pruebas de comunicación con el propio chip. Este puerto USB además de permitir la comunicación también sirve de fuente de suministro de energía por tanto puede sufrir sobrecargas, por esta razón dispone de un fusible reseteable que protege tus puertos USB del ordenador de cortes y sobrecargas. Aunque la mayoría de los ordenadores proporcionan su propia protección interna, el fusible proporciona una capa de protección extra. Si más de 500 mA se aplican al puerto USB, el fusible automáticamente romperá la conexión hasta que el corte o la sobrecarga sean eliminados.

2.6.2 Software.

La placa Arduino Mega y todas sus versiones vienen con un software propio proporcionado por la empresa. Este software es de código abierto esto facilita que cualquier persona interesada pueda acceder a él. Este software fue creado bajo la plataforma Java y basada en el procesamiento avr-gcc. Para escribir y desarrollar un código de programación en este software se utiliza el lenguaje C++ que es el lenguaje que utiliza este software. Para poder establecer la comunicación entre la placa y la PC a través del puerto USB se requiere instalar los drivers para el chip FTDI de la placa los cuales también son proporcionados por la empresa que fabricó la placa para diferentes sistemas operativos y pueden ser descargados gratuitamente desde la web de Arduino (Figura 7).

Figura 7. Software IDE Arduino.

Fuente:(Arduino,2015)

2.6.3 Lenguaje de programación Arduino.

Un lenguaje de programación se entiende como un sistema de comunicación humano-maquina a través del cual podemos indicarle a la máquina que acción tienen que lleva

r a cabo y cuál es el modo para concretarla. Entonces en el software Arduino este lenguaje estructurado con cierta base sintáctica y semántica indicara todas las acciones que debe realizar la placa. Para posteriormente ser grabadas en la placa.

El lenguaje utilizado para programar el microcontrolador de la placa puede ser creado mediante librerías C/C++ o AVR C en el cual está basado. Este lenguaje permite comunicar y programar al microcontrolador para que este realice una tarea específica. Además, el software posee una herramienta que permite verificar el código en busca de errores antes de grabarlo en la placa. Para ello se hace uso de varios elementos en la programación.

a) **Estructura.** - Básicamente esta estructura de programación de Arduino es simple, contienen declaraciones, estamentos o instrucciones necesarias para que el programa funcione. Para ello se hace uso de funciones, las funciones se escriben para ejecutar tareas repetitivas y reducir el desorden y tamaño de un programa.
b) **Funciones. -** Una función es un bloque de código que tiene un nombre y un conjunto de estamentos que son ejecutados cuando se llama a la función, estos se repiten o no según el tipo de función. La declaración de una función incluye el tipo de datos que devuelve la

función. Las funciones principales dentro de esta programación se muestran a continuación (Tabla 5):

Tabla 5. Funciones programa Arduino.

Void Set up ()	Donde set up se encarga de recoger la configuración, es usada para asignar el pin mode o iniciar la comunicación en serie aquí se declara las variables que solo se ejecutara una sola vez al arrancar el programa.
Void loop ()	Es un bucle que contiene el programa en si el cual se ejecutará continuamente leyendo entradas, activando salidas, etc. Esta función es el núcleo de todos los programas Arduino y hace la mayor parte del trabajo.
Void Interruption ()	Este es un bucle especial que sirve para declarar los pines de interrupción, estos pines permiten receptar señales continuas y simultaneas. Esto es muy útil debido a que programas similares a este captan señales, pero las procesan en el orden que van llegando, esto causa que se pierda una gran cantidad de datos en ese lapso de tiempo que le toma leer la otra señal.
Sentencias condicionales	El lenguaje de Arduino permite usar este tipo de funciones para establecer condiciones de funcionamiento para una determinada situación tales como if, else, while, for.

Fuente: (Arduino,2015)

c) **Variables. -** Una variable es una manera de nombrar y almacenar un valor numérico para su uso posterior por el programa. Las variables son números que se pueden variar continuamente en contra de lo que ocurre con las constantes cuyo valor nunca cambia. Una variable debe ser declarada indicando el tipo de datos que desea almacenar (int, float, log, min y máx.).

Una variable no solo representa un identificador o un nombre, también significa una forma de almacenar en memoria un valor numérico para usarse después en el programa (Cortez, Fernando;

Monjaraz Jaime, 2015). Arduino permite manejar los siguientes tipos de datos mostrados a continuación (Tabla 6).

Tabla 6. Tipos de datos en Arduino.

int	Que devolverá a un valor entero de 16 bits.
byte	Almacena un valor numérico de 8 bits
long	Almacena un valor numérico de 32 bits
void	Es una función vacía que no devuelve ningún valor.
array	Cadena de datos numéricos.

Fuente: (Reyes Cortez, Cid Monjaraz, 2015)

d) **Comandos. -** Además de las funciones presentadas anteriormente existen estas funciones especiales las cuales permiten definir los pines de la placa que se usaran y cómo será su interacción dentro de la programación. Ya sean como pines digitales o analógicos. Las funciones usadas para definir los pines se detallan en la (Tabla 7):

Tabla 7. Comandos programables en Arduino.

pinmode ()	Para configurar un pin como entrada o salida, aunque no es necesario declarar una entrada porque esta viene dada por defecto.
digitalwrite ()	Introduce un nivel alto (high) o bajo (low) en el pin especificado ya sea como una variable o una constante.
digitalread ()	Lee el valor de un pin digital especifico devuelve un valor High o Low.
analogread ()	Lee el valor desde el pin analógico especificado con una resolución de 10 bits. A diferencia de los digitales no necesita ser declarado previamente como entrada o salida
analogwrite ()	Escribe un valor analógico usando modulación por ancho d pulso en un pin marcado como PWM. Esta función corresponde a los pines 3, 5, 6, 9, 10, 11.

Fuente: (Aranda, 2014)

e) **Librerías. -** En Arduino también se puede hacer uso de librerías que son ficheros de código que realizan una tarea concreta dentro del código principal, evitando alargar más el código principal con tareas secundarias. Las librerías son una colección de funciones que contienen el código proporcionado por el compilador por razones de eficiencia (Cortez, Fernando; Monjaraz Jaime, 2015).

Tabla 8. Ejemplos de librerías en Arduino

Librería	**Función**
Tiny GPS++.h	Configuraciones del receptor GPS.
ODB.h	Configuraciones OBD.
SPI.h	Permite la conexión con la SD.
SD.h	Gestiona el almacenamiento SD.
MultiLCD.h	Programa varias pantallas en la LCD.
Utouch.h	Configura la pantalla táctil
Timerone.h	Coloca un timer dentro de la programación.

2.7 Cámara.

La cámara OV2640 posee un pequeño chip, el cual es un sensor de imagen "CMOS de bajo voltaje", el cual favorece todas las funciones completas tanto de captura y procesamiento de imágenes en un chip de la cámara "UXGA". Ov2640 pone a disposición un fotograma muy completo, ya que se puede obtener sub muestras, imágenes de 8-10 bits con escala o con una vasta gama de formatos, que es controlada a través de su interfaz de bus que controla la cámara de serie.

OV2640 permite operar a una velocidad de captura de hasta 15 cuadros por segundo en la resolución UXGA, controlar el formato de calidad de la imagen y la transferencia de salida de datos. Pone a disposición la programación de diferentes funciones de procesamiento de imágenes mediante la interfaz SCCE, tales como control de exposición, gamma, balance de blancos, saturación de color entre otras. Esta cámara Omni-Visión posee tecnología de sensores que mejoran la calidad de las imágenes mediante un proceso de reducción y eliminación de iluminación o fuentes de ondas eléctricas que reduzca la calidad de imagen o genere manchas en

los pixeles de la imagen capturada. Para de esta manera obtener una imagen de excelente calidad, limpia y de un color estable.

2.8 Labview

Labview es un sistema de desarrollo basado en el leguaje tipo G o programación gráfica orientado a desarrollar aplicaciones para instrumentación que integra una serie de librerías para comunicación con instrumentos electrónicos, tarjetas de adquisición de datos, sistemas de adquisición, accionamiento y comunicación en redes TCP/IP. Labview puede ser utilizado por gente con poca experiencia en programación pues usa metodologías familiares a técnicos, ingenieros y a toda la comunidad científica

Los programas realizados en Labview se denominan VIs o instrumentos virtuales por sus siglas ya que tienen la apariencia de los instrumentos reales con funciones provenientes de leguajes de programación convencionales. Cada VI creado en Labview consta de dos partes o interfaces: el panel frontal y el diagrama de bloques cada una contiene paletas con instrumentos propios para construir y diseñar tareas en el entorno del programa (Figura 8).

Figura 8. Programa Labview 2015.

2.8.1 Panel de frontal.

Es la interface gráfica que se muestra al usuario, la cual simula al instrumento real, permite la entrada y salida interactiva de datos, puede contener una gran variedad de elementos entre controladores e indicadores. Normalmente se empieza por diseñar este panel frontal y

posteriormente se diseña el diagrama de bloques donde se asignarán las funciones de todos los controladores e indicadores puestos en el panel frontal (Figura 9).

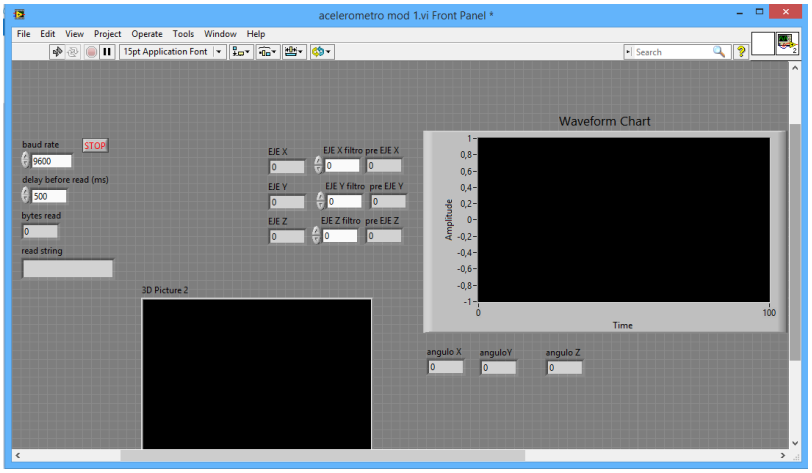


Figura 9. Vista del panel frontal de Labview.

Los controladores en el panel de control permiten ingresar datos o variables de entrada al programa y pueden ser manipulados por el usuario. Mientras los indicadores en un panel de control sirven para mostrar o representar los resultados o variables de salida que el diagrama de bloques genera o adquiere y no pueden ser manipulados por el usuario a diferencia de los controladores. Cada control e indicador tiene un tipo de dato asociado, los tipos de datos más usados son los numéricos, booleanos y de cadena de caracteres.

a) **Control e indicador numérico. -** El tipo de dato ingresado o mostrado puede representar números de varios tipos como entero o real.
b) **Control e indicador booleano. -** El dato manejado por estos solo tiene 2 estados posibles, en el caso del indicador (true, false) y el controlador (on, off).
c) **Control e indicador de cadena de caracteres. -** Cuando el tipo de datos es una secuencia ASCII, se requiere usar un controlador de caracteres para poder que el usuario pueda ingresar texto, mientras que se usa un indicador de caracteres para mostrar un texto prefijado al usuario.

2.8.2 Diagrama de bloques.

Contiene todos los códigos de la programación del VI. Los controladores e indicadores colocados en el panel frontal aparecen como terminales en el diagrama de bloques las cuales mediante cables y estructuras se las pueden relacionar creando así el flujo de datos requeridos.

En esta parte podemos ver la utilización del lenguaje tipo G, cada elemento colocado aquí tiene su propia programación y en esta pantalla solo se les da una aplicación a dichos elementos. Supone una solución grafica a un determinado problema de programación (Figura 10).

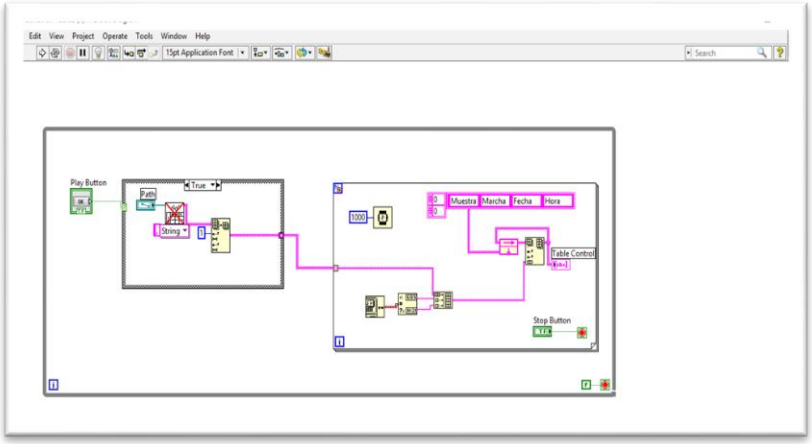


Figura 10. Diagrama de bloques de Labview.

2.8.3 Herramientas de Labview.

Una herramienta es un modo de operación especial que seleccionamos con el cursor del mouse, que es la única manera de interactuar con el entorno de programación de este programa. Labview ofrece una gran variedad de herramientas para crear, modificar y depurar Vis que hacen más fácil la utilización de este programa, incluso si no se tiene muchos conocimientos acerca del mismo. (National Instruments,2016). Las herramientas más comunes en Labview son:

a) **Herramientas de operación. –** Esta herramienta es usada tanto en el panel frontal como en el diagrama de bloques, permite cambiar los valores de un control, así como de constantes booleanas.
b) **Herramientas de posicionamiento.** – Esta herramienta se usa para seleccionar o cambiar el tamaño de los objetos colocados en el panel frontal y en el diagrama de bloques sirve para arrastrar los objetos y modificar el tamaño de las estructuras.
c) **Herramientas de cableado. –** Esta herramienta permite cablear y unir los diferentes elementos colocados en el diagrama de bloques.
d) **Herramientas del panel frontal y diagrama de bloques. –** Cada ventana tiene una barra de herramientas asociada que sirven para ejecutar o editar el VI.

e) **Herramientas de depuración.** – Labview cuenta con potentes herramientas de depuración para ayudar al programador a detectar e identificar problemas en su código y poder realizar los cambios adecuados.

2.8.4 Estructuras en Labview.

Las estructuras dentro de la programación de Labview tienen un papel muy importante por ello a pesar que fue mencionada anteriormente como parte del diagrama de bloques, se requiere analizarlas nuevamente para su mejor comprensión. Estas son utilizadas para definir secuencias, decisiones y ciclos que el programa debe seguir dentro de su funcionamiento. En general es un nodo que controla el flujo de datos del programa. En Labview podemos distinguir 5 tipos de estructuras que son:

a) **For loop. -** Esta estructura es un ciclo que repite el subdiagrama que contiene un número definido de veces. El terminal de interacción indica el número de veces que se ha ejecutado el ciclo. El control de interacción nos permite establecer el número de veces que se quiere que el programa contenido en la estructura se repita, aunque si no se define se estaría creando un ciclo auto indexado por defecto, esto quiere decir que se repetirá hasta que se completen todas las acciones o se terminen los datos (Figura 11).

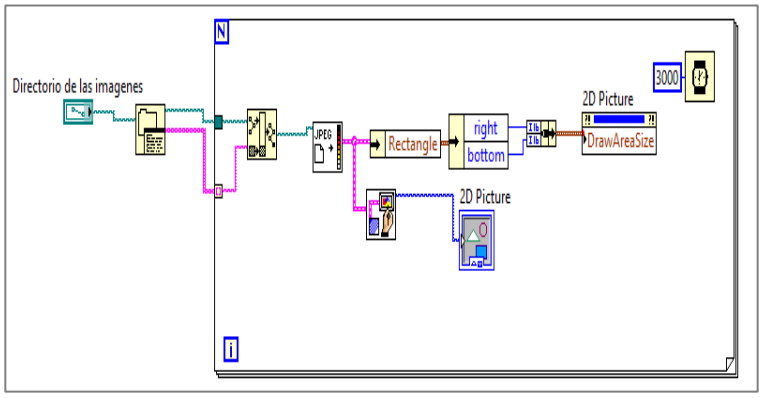


Figura 11. Estructura For loop en Labview.

b) **While loop. -** Esta estructura también es un ciclo que repite un subdiagrama que contiene, pero a diferencia del for loop solo lo repetirá hasta que se cumpla una condición determinada. Por defecto las instrucciones contenidas en el ciclo se repiten mientras que al terminal de condición llegue un valor verdadero. Si se desea cambiar la lógica del

terminal de condición para que el ciclo termine cuando a este le llegue un valor falso basta con cambiar la condición de la estructura de true a false en el terminal (Figura 12).

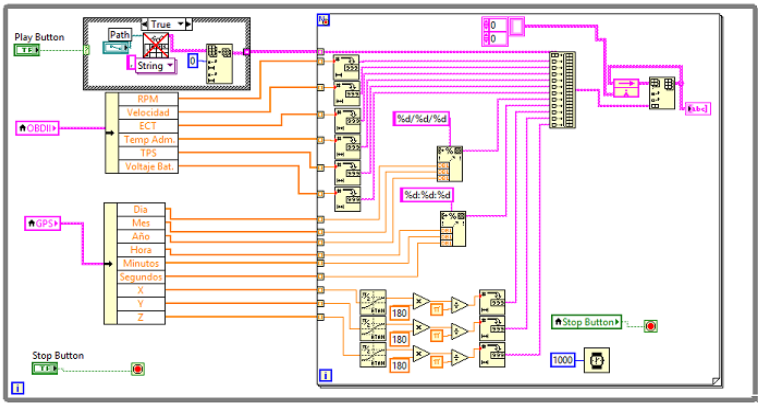


Figura 12. Estructura while loop de Labview.

c) **Estructura Case. -** La estructura case posee varios subdiagramas denominados casos de los cuales solo se ejecuta uno. Al igual que la estructura sequence solo es visible un subdiagrama a la vez. Dependiendo del tipo de variable asociada al terminal de selección la estructura se comporta de distinta forma. Si el valor cableado proveniente del terminal es un booleano las estructuras tiene 2 casos false y true, pero si es numérico o cadena la estructura tiene desde 2 hasta 2^{16} casos posibles. Esta estructura también posee un menú de opciones para añadir, borrar, duplicar y mover como lo tiene la estructura sequence (Figura 13).

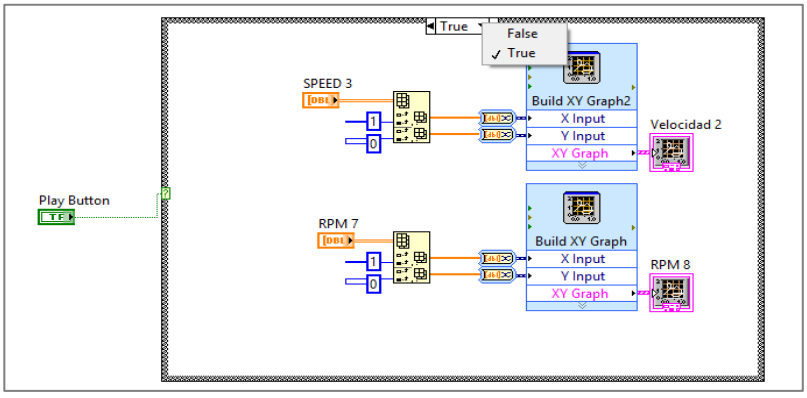


Figura 13. Estructura Case en Labview.

d) **Flat Sequence. -** Esta estructura permite ejecutar varios subdiagramas de manera ordenada y controlada por el ordenador. En lenguajes de programación convencionales

basados en códigos de líneas no se requiere y por lo tanto no existe una estructura análoga. Esta estructura posee varios subdiagramas denominados frames que se ejecutan en estricto orden y solo es visible uno a la vez (Figura 14).

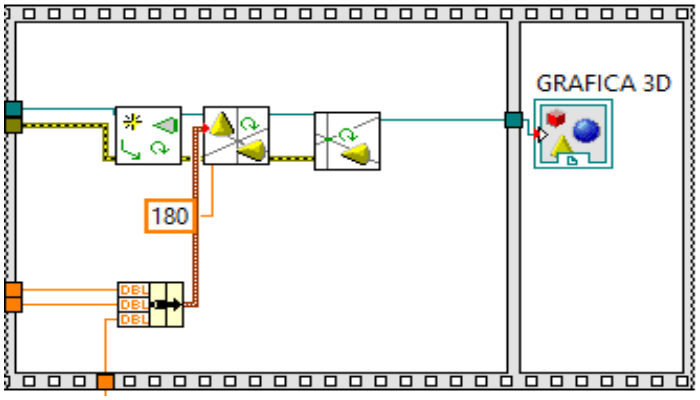


Figura 14. Estructura Flat sequence de Labview.

e) **Nodo de fórmula. -** Esta estructura es una caja redimensionada donde se alojan fórmulas matemáticas y lógicas para su evaluación. Los terminales de entrada y salida son variables numéricas, escalares y reales que se generan adicionándolas desde el menú de la estructura. El lenguaje usado es similar a la de C (Figura 15).

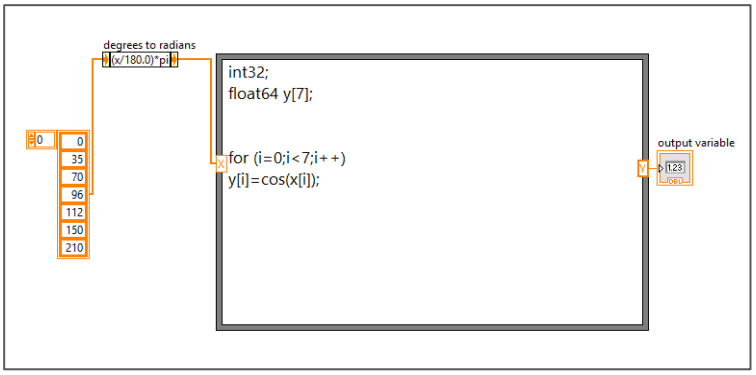


Figura 15. Nodo de fórmula en Labview.

2.8.5 Arreglos.

Un arreglo es una colección ordenada de variables del mismo tipo puede tener 1 o varias dimensiones y múltiples elementos por dimensión. Los elementos de los arreglos están ordenados desde la posición 0 hasta la posición n-1 donde n es el tamaño del arreglo. En Labview existen varios tipos de arreglos los más comunes son:

a) **Array size. -** Retorna el tamaño de n de arreglo de entrada. Si este es de n dimensiones, la salida será un valor de n elementos donde cada uno muestra el tamaño de cada dimensión.
b) **Index array. -** Retorna el elemento indicado de un arreglo. Se debe adicionar tantos elementos de índice como dimensiones tenga el arreglo. La salida de esta función también puede ser un arreglo cuando se cablean solo algunos de los índices
c) **Replace array subset. -** Reemplaza el elemento indicado en los terminales de índice por el elemento que este cableado al terminal. Se debe agregar tantos terminales de índice como dimensiones tenga el arreglo. El nuevo elemento debe ser del mismo tipo y dimensión del arreglo inicial.
d) **Insert into array. -** Inserta un arreglo o un elemento en la posición especificada por el terminal de índice. Cuando no se cablea este terminal, el arreglo o elemento se inserta al final del arreglo de entrada.
e) **Delet from array. -** Elimina un arreglo o elemento del arreglo de entrada. Además de volver el arreglo editado, también devuelve la porción de arreglo eliminada.
f) **Initialize array. -** Retorna el arreglo de n dimensiones, donde todos los elementos serán inicializados con el valor y tipo de cada cableado en elementos.
g) **Build array. -** Construye un arreglo de n dimensiones con los elementos de entrada que pueden ser de n o de n-1 dimensiones. Si se desea que la salida sea un arreglo de dimensión n conformado por la concatenación de todas las entradas se lo debe seleccionar en el menú de la función.
h) **Array subset. -** Retorna una porción del arreglo de entrada (sub-arreglo). Labview adiciona tantos de índice como dimensiones tenga el arreglo de entrada.
i) **Array max & min. -** Retorna los valores máximos y mínimos de un arreglo numérico con sus respectivas posiciones.

2.8.6 Tipos de datos en Labview.

Labview al igual que muchos otros lenguajes de programación maneja diferentes tipos de datos, estos pueden ser identificados por su color, cuando se cablea estos se podrá notar que la coloración es distinta según el dato manejado y no solo eso también se podrá ver que el cable tiene diferente grosor dependiendo de las dimensiones del dato (Figura 12). Esto es muy

importante dentro de esta programación, donde el problema más común es no poder conectar elementos entre sí, si el tipo de dato no corresponde o tiene más dimensiones que las permitidas por dicho elemento. Los tres tipos de datos que maneja Labview son numéricos, booleanos y string (Figura 16).

Figura 16. Tipos de datos usados en Labview.

Fuente: (National Instruments, 2016)

2.9 Señales analógicas y digitales.

Una señal se define como una variación utilizada para transmitir información y su valor se representa con un valor de magnitud. Existen señales que no transmiten información, las cuales son denominadas ruido eléctrico, siempre que provengan del canal en el cual se transmite información, y se denomina interferencias a las señales que no provienen de este canal. Estas dos señales dificultan la obtención y procesamiento de la información que se transmite por señales. (Areny, 2006).

2.9.1 Señal analógica.

En este tipo de señal los valores de voltaje y corriente sufren una variación constante, aumentando su valor de manera positiva en la mitad de un ciclo, y disminuyendo su valor de manera negativa en la mitad del ciclo siguiente, completando un ciclo completo. Esta variación de polaridad genera una forma de onda sinusoidal, las cuales son representadas en unidades de tiempo para su ciclo (Figura 17).

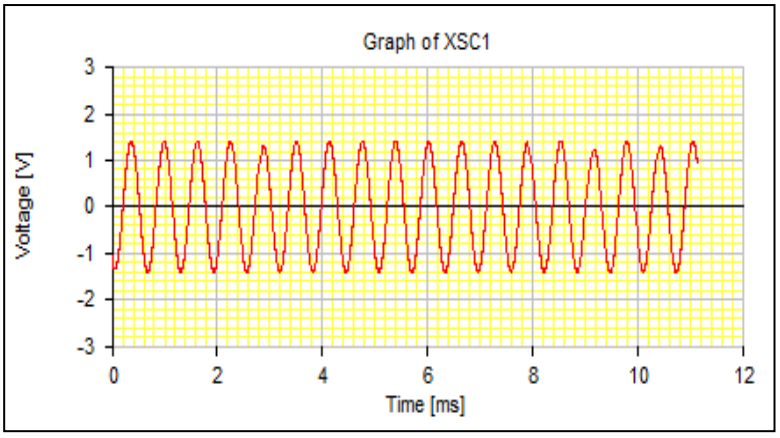


Figura 17. Forma de onda de una señal análoga simulada

2.9.2 Digitalización de la señal analógica.

La señal analógica varía entre valores máximos positivos y negativos fijados, en cambio en una señal digital solo existe un cambio de condición: hay paso de corriente o no hay paso de corriente y genera una forma de onda cuadrada en intervalos de tiempo. Al transformar una señal analógica en una señal digital, sus valores se convierten en números binarios, determinados por "0" en el caso que no existe pulsos eléctricos y "1" cuando exista pulso eléctrico.

a) **Conversión analógica. –** digital. Para convertir una señal analógica en digital, el primer paso consiste en realizar un muestreo (sampling) de ésta, o lo que es igual, tomar diferentes muestras de tensiones o voltajes en diferentes puntos de la onda senoidal. La frecuencia a la que se realiza el muestreo se denomina razón, tasa o también frecuencia de muestreo y se mide en hertz (Hz). (J. Espi Lopez, 2006)**.**

b) **Cuantificación de la señal analógica.** - Por tanto, la cuantificación representa el componente de muestreo de las variaciones de valores de tensiones o voltajes tomados en diferentes puntos de la onda sinusoidal, que permite medirlos y asignarles sus correspondientes valores en el sistema numérico decimal, antes de convertir esos valores en sistema numérico binario (Figura 18).

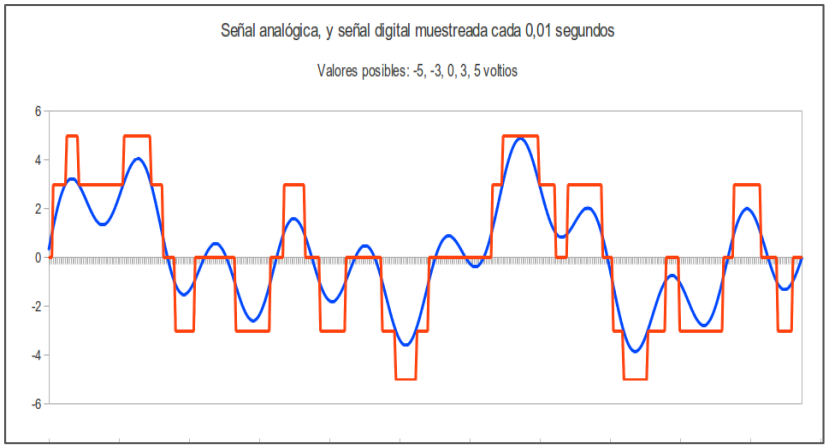


Figura 18. Conversión analógico-digital.

Fuente:(Monroy, 2015)

2.10 Sensores de gestión del motor.

Los sensores que intervienen en la gestión del motor juegan un papel muy importante a la hora de mejorar la eficiencia y prestaciones del motor, reducir el consumo de combustible y por ende las emisiones contaminantes. Estos sensores pueden enviar tanto señales analógicas como digitales a la ECM para que esta las procese y envíe una respuesta para controlar varias funciones en el motor. Estos sensores deben ser resistentes, precisos, sensibles y fiables para su aplicación en automoción.

2.10.1 Sensores de posición/velocidad.

Estos tipos de sensores miden el número de giros que se producen en una rueda dentada, al pasar a través de un imán que posee el sensor. Esta información recogida por el sensor es enviada a la ECU, la cual determina la velocidad del vehículo y la desaceleración que se produce. La información de velocidad del vehículo es aprovechada por la ECU para controlar diversos parámetros de gestión del motor del vehículo. Estos sensores pueden ser de 2 tipos inductivo y de efecto hall.

2.10.1.1 Sensor inductivo.

Según (Robert Bosch GmbH, 2002) “Los sensores inductivos aprovechan la inducción para medir la velocidad de rotación, generando en su salida bipolar una tensión, que es proporcional a la variación en el tiempo de un flujo magnético”. Los sensores inductivos están compuestos por

una bobina con imán permanente, que trabajan conjuntamente a una rueda dentada, y se encuentra colocado en el volante de inercia del motor.

Debido a que poseen una bobina, estos sensores generan un campo magnético, el cual es alterado cuando un diente de la rueda pasa por el imán del sensor. Este flujo magnético genera una onda entre los terminales positivo y negativo del bobinado del sensor, lo que la convierte en una señal analógica. Esta señal de onda se eleva hasta un punto máximo de rango positivo y luego desciende hasta hacerse negativa y asciende hasta llegar a un valor nulo, este es su ciclo de funcionamiento y se lo interpreta como frecuencia de onda generada por el sensor como se ven en la (Figura 19).

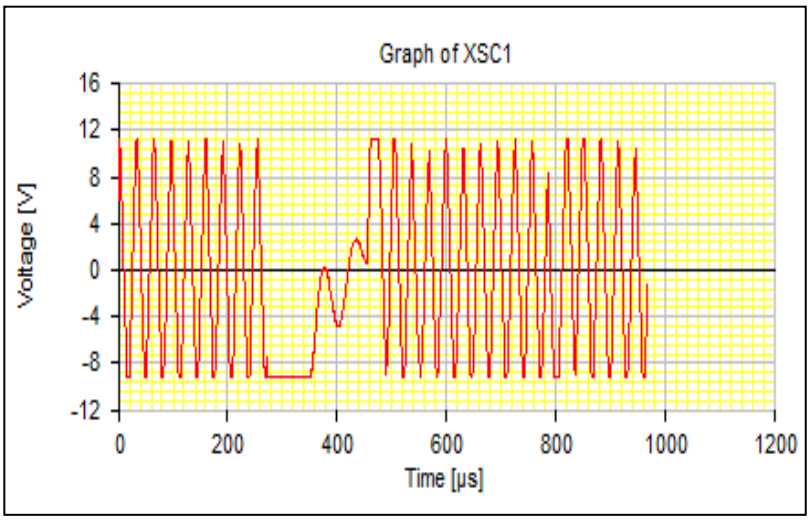


Figura 19. Señal de un sensor inductivo simulada en Livewire.

Un ejemplo de sensor inductivo es el CKP donde se presenta un flujo magnético alto cuando un diente de la rueda se coloca frente al sensor y un flujo magnético mínimo cuando pasa sobre el sensor un espacio vacío. La ECU usa esta señal para determinar la velocidad del motor, y otros parámetros entre los cuales tenemos tiempo de inyección, punto de encendido y posición y grados de giro del cigüeñal (Figura 20).

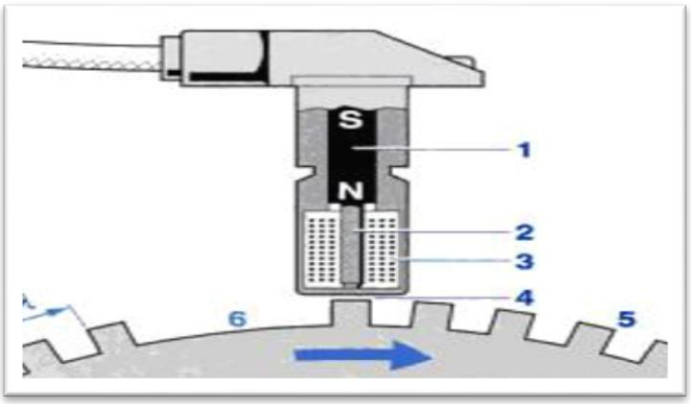


Figura 20. Sensor inductivo de velocidad.

Fuente:(Robert Bosch GmbH, 2002).

Mientras aumenta la velocidad de la rueda dentada, el número de ciclos aumenta, por ende, se genera una mayor frecuencia. El sensor tipo inductivo posee 3 conexiones entre las cuales tenemos: señal de alimentación de 5V al sensor, conexión a masa del sensor y señal de funcionamiento del sensor que es enviado a la ECU. Esta señal permite a la ECU controlar y gestionar varios parámetros de funcionamiento de los sistemas del motor.

2.10.1.2 Sensores de efecto hall.

Basan su funcionamiento en el efecto hall, que consiste en la aparición de un campo eléctrico en un conductor cuando es atravesado por una corriente estando dentro de un campo magnético. En el motor se lo puede encontrar en el árbol de levas a este sensor se lo conoce como CKP, donde un disco perforado gira solidario al árbol permite hacer llegar al sensor el campo magnético de un imán colocado al otro lado del disco cuando coincide un agujero con la trayectoria del flujo magnético y el sensor. Esta señal es interpretada por la ECU para determinar la posición de los pistones y la secuencia exacta de inyección de combustible en los cilindros. Y a través de esto generar el adelanto o retraso de inyección en el sistema de encendido.

Las características del efecto hall lo convierten en el método para detectar la posición y velocidad de rotación. Tiene una ventaja sobre la bobina captadora o inductiva, puesto que puede medir velocidades mínimas de rotación (Figura 21). (Rueda, Jesús, 2013).

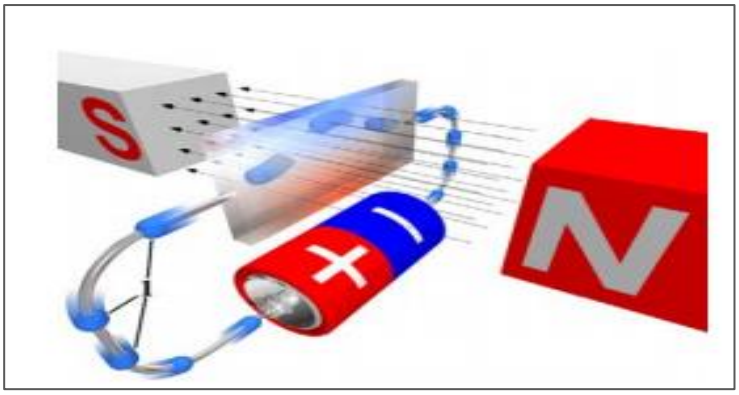


Figura 21. Esquema efecto hall.

Fuente:(Upcommons, 2012)

2.10.2 Sensor de temperatura.

Los sensores de temperatura se utilizan para medir la temperatura del refrigerante, aire de admisión, aceite, combustible y gases de escape. Esta información de estas temperaturas permite a la Ecu determinar diferentes acciones en función de estas temperaturas como por ejemplo encender o apagar el ventilador, la densidad del aire que ingresa entre otras. Existen 2 tipos de sensores de temperatura los termopares y termistores.

Los termopares son sondas que se utilizan para medir la temperatura de los gases de escape, pueden ser del tipo K, N, J, R, S, W, aunque las más utilizada es la sonda K. Las sondas son sensores activos que funcionan bajo el efecto Peltier-Seebeck, que consiste en que 2 metales distintos unidos se calientan, por uno de los extremos circula una corriente, esta corriente del orden de los (mv) debe ser inducida termalmente al extremo más frio produciéndose una diferencia de temperaturas. La sonda K usada en automoción está compuesta por cromo y alumel (AL+NI).

Los termistores en cambio están compuestos de una resistencia termo sensible de medición, de material semiconductor, esta resistencia forma parte de un circuito divisor de tensión alimentado con 5V. Normalmente tiene un coeficiente de temperatura negativo NTC en el cual el valor de su resistencia disminuye conforme aumenta la temperatura, raramente un coeficiente de temperatura positivo PTC en el cual su valor de resistencia aumenta conforme aumenta la temperatura (Robert Bosch GmbH, 2002).

2.10.2.1 Sensor ECT (Sensor de temperatura del refrigerante del motor).

Está ubicado en el sistema de enfriamiento, conjuntamente acoplado en el bock del motor, el cual mide la temperatura de refrigerante del motor mediante el aumento o disminución de una resistencia del sensor, gracias a esto disminuye el voltaje que recibe la computadora para ajustar los parámetros de funcionamiento tanto pulso de inyección y regulación de la mezcla A/C. El rango de operación del sensor ECT va desde los -40°C a los 130°C.

Figura 22. Sensor ECT.

2.10.2.2 Sensor IAT (Sensor de temperatura del aire de admisión)

Estes sensor se ubica montado en el colector de admision de aire, y es el que registra la temperatura del aire que ingresa al motor. Mayormente viene montado conjuntamente con un sensor de flujo de masa de aire para controlar la mezcla que ingresara a los cilindros del motor. El rango de funcionamiento comprende los valores de -40°C hasta los 120°C.

2.10.2.3 Sensor EOT (Sensor de temperatura del aceite del motor).

Este sensor tiene como función medir la temperatura del aceite, la Ecu utiliza esta temperatura para modificar el encendido según las necesidades. Por ejemplo, si el vehículo va arrancar en frio se requiere que el aceite tenga una temperatura adecuada para que este pueda fluir por todo el motor y así pueda realizarse la lubricación, por ello la Ecu mantendrá el vehículo en un estado de ralentí hasta que el aceite adquiera la temperatura optima y viceversa. El margen de temperaturas que viene programado este sensor es de -40°C hasta 170°C.

2.10.3 Sensor de flujo de aire (MAF).

Estos sensores juegan un papel importante a la hora de asegurar una óptima proporción de mezcla aire combustible y por ende reducir las emisiones de CO_2. Se sitúan directamente después del filtro de aire en el colector de admisión, en este sensor también pueden venir integrado el sensor IAT, de ser así este sensor no solo proporciona el dato del volumen de aire sino también la temperatura y humedad del aire.

El sensor MAF más común es el de hilo caliente, el cual trabaja bajo el principio de temperatura constante. El hilo caliente forma parte de un circuito que lo mantiene siempre a una temperatura constante, si aumenta el caudal de aire dentro del colector comenzará a enfriar el hilo y disminuirá su resistencia. Cuando esto sucede el circuito de regulación intenta corregir este desequilibrio aumentando la corriente de calentamiento hasta que el hilo recobre su temperatura nominal. La Ecu relaciona directamente este incremento de tensión para calcular el flujo de aire entrante.

2.10.4 Sensor de presión de aire (MAP).

Este sensor es situado en el colector de admisión, es el encargado de medir la presión del aire de admisión comparando la presión dentro del colector con la presión atmosférica. Esta información es muy importante para calcular la cantidad de combustible que se debe inyectar para lograr una mezcla óptima. Está compuesto de un chip de silicio con 2 partes, un transductor de membrana y la electrónica de acondicionamiento. La membrana de este sensor posee 4 piezo-resistores que varían cuando se le somete a un esfuerzo, la señal de salida de estos es del orden de los 100 mv, los cuales se hacen pasar por un amplificador de ganancia elevada que la convierte en una señal analógica con un rango de 0.5 a 4.5 v.

2.11 Sistema de Freno.

El sistema de frenos debe cumplir la función de detener el vehículo, evitar el exceso de velocidad cuando el vehículo desciende y mantenerlo cuando se detiene sobre declives. Se diseña de modo que el esfuerzo de frenado pueda ser cambiado por el conductor y así mantener el vehículo bajo control. (Rueda, Jesus,2010)

Todo vehículo está dotado de un sistema de frenado que permita detener o disminuir progresivamente la velocidad del vehículo según la voluntad del conductor. El frenado se produce cuando el conductor pisa el pedal de freno, esta fuerza aplicada sobre el freno se reparte por igual a las 4 ruedas actuando sobre el giro de las mismas. Sin embargo, esta fuerza solamente ejerce resistencia sobre el giro de la rueda mas no es la que desacelera el vehículo, el frenado real se produce cuando la rueda entra en contacto y fricción con el suelo.

2.11.1 Tipos de frenos.

Los frenos en el campo automotriz se pueden dividir en 2 grupos que son: el primer grupo toma en cuenta el tipo de accionamiento empleado, tenemos los frenos neumáticos utilizados en vehículos pesados, frenos mecánicos mejor conocidos como freno de mano o de servicio, freno hidráulico utilizado en vehículos livianos y la variación de este último denominado electrohidráulico mejor conocido como ABS. El segundo grupo toma en cuenta el elemento de fricción utilizado en este grupo se encuentra los frenos de disco y de tambor.

a) **Frenos neumáticos. -** El accionamiento de este freno se lo realiza mediante aire comprimido proporcionado por un compresor instalado en el propio vehículo. El aire comprimido acciona unos pistones dispuestos en cada rueda actuando como prensas neumáticas sobre los discos de frenos o tambores, el control y dosificado de aire suministrado se lo realiza mediante válvulas.
b) **Frenos mecánicos. -** Este tipo de frenos es accionado por la fuerza ejercida sobre el pedal de freno y trasmitido a las ruedas a través de palancas o cables debido a su poca fiabilidad y poca potencia de frenado son usados únicamente como frenos de estacionamiento para inmovilizar el vehículo. Requiere de frecuentes ajustes para equiparar su acción sobre cada rueda.
c) **Frenos hidráulicos. -** Este sistema se basa en el principio de Pascal, aprovecha que la fuerza de frenado aplicada sobre un líquido hidráulico es igual en cualquier dirección por tanto esta fuerza multiplicada se distribuirá a cada una de las ruedas para accionar los elementos de fricción. Sin embargo, cada rueda no está sometida a la misma fricción con el suelo por lo que la fuerza para frenar una rueda quizás no sea la óptima para detener la otra, produciéndose un frenado defectuoso.

d) **Frenos ABS. -** Este sistema es sin duda el mejor dispositivo desarrollado en pro de la seguridad activa, puesto que da solución a los problemas del sistema de frenado hidráulico convencional. Evita que las ruedas se bloqueen al momento de frenado permitiendo al conductor tener un mejor control sobre el vehículo. Este sistema no es más que una microcomputadora integrada en el vehículo la cual recibe las señales de los sensores de velocidad dispuestos en cada rueda, esta información permite conocer la velocidad y el estado de desaceleración de cada rueda, lo cual permite comparar esta información de cada rueda para detectar problemas en el frenado, de esta manera la computadora acciona válvulas para tomar las respectivas correcciones.

2.12 Caja de cambios.

La caja de cambios es un elemento intercalado entre el motor y las ruedas el cual convierte la velocidad mecánica de giro producida por el motor y la adapta a la velocidad requerida en las ruedas o invertir el giro de las mismas cuando así se lo requiera, dicho de otra manera, es quien administra la potencia del motor a diferentes regímenes de desplazamiento del vehículo. De esta manera se puede conseguir desplazar el vehículo a diferentes velocidades mientras que el motor seguirá funcionando bajo el mismo régimen. En lo que respecta a las cajas de cambios podemos distinguir 2 tipos de cajas de cambios manual y automáticas.

a) **Cajas de cambio manuales. -** En este tipo de cajas de cambios la selección de las marchas se las realiza mediante el mando mecánico, previo a insertar una marcha se debe pisar el embrague para desconectar la caja de cambios del motor. El acoplamiento interno se lo realiza mediante ejes, engranes, cojinetes y sincronizadores los cuales esta bañados en aceite debido al desgaste que puede sufrir por el continuo rozamiento.
b) **Cajas de cambios Automáticas. -** Este tipo de cajas usan engranajes epicicloidales y como elemento de conexión y desconexión entre el motor y la caja un convertidor de par en lugar de un embrague como la caja manual. Estas cajas de cambio seleccionan la marcha adecuada en función de la velocidad del vehículo y la posición del pedal del acelerador a este se lo denomina lumbral de paso de velocidad. Para evitar el fenómeno de bombeo donde la trasmisión no sabe qué velocidad elegir el umbral de paso de velocidad realiza un autoajuste de tal forma que durante el aumento de velocidad se retrasa y para el descenso de velocidad se adelanta.

CAPÍTULO III

3. Diseño y construcción del módulo de almacenamiento de datos on-board.

3.1 Diseño del Hardware para el módulo de almacenamiento de datos.

Para el diseño del hardware para el módulo de almacenamiento de datos on-board, se requiere seleccionar los elementos que mejor se ajusten a las necesidades del proyecto, estos elementos deben ser fiables y compatibles entre sí, de esta manera se asegura su buen funcionamiento y evita complicaciones que se puedan presentar.

En este caso el módulo cuenta con un cable OBDII para obtener los datos de sensores del motor directamente de este conector equipado en todos los vehículos modernos, de esta manera se trata de que el módulo sea lo menos invasivo posible. También se emplea un receptor GPS que proporcione al módulo datos como hora, fecha, latitud, longitud, una cámara para capturar fotos durante todo el trayecto a manera de un registro fotográfico y un sistema para adquirir la posición y el estado de marcha (Figura 23).

Figura 23. Diagrama de bloques (Módulo de almacenamiento de datos).

Para la construcción del de este módulo se requieren los siguientes elementos y dispositivos :

a) 1 placa Arduino Mega 2560, como placa principal encargado de recolectar, procesar y almacenar los datos provenientes del OBDII, GPS, marchas, freno, botones.
b) 1 placa Arduino Nano la cual actúa como esclavo del arduino principal y está destinado a procesar solo los datos provenientes de la cámara.
c) Un receptor GPS que proporcione al módulo datos como los de hora, fecha, latitud, longitud.
d) Un cable OBDII para hacer la lectura directa de datos de los sensores directamente desde el conector OBDII, de esta manera se trata de que el módulo sea lo menos invasivo posible.
e) Un sensor MPU6050 que proporciona información de la aceleración generada en los ejes X, Y, Z.
f) Un sistema de marcha/freno que permite brindar información de la posición en la cual se encuentre la palanca de cambios y la activación del pedal de freno del vehículo.
g) Una cámara de video/fotos adaptada para trabajar con las placas arduino, con esta cámara se puede tener un registro fotográfico de todo el trayecto realizado con el vehículo.
h) Pulsadores y fines de carrera, estos se usan como botones de acción y para detectar la posición de la marcha.
i) Una pantalla TFT 3.2” para indicar algunos datos útiles para el usuario de este módulo, además que posee entradas I2C y 2 Serial más una ranura para micro SD.

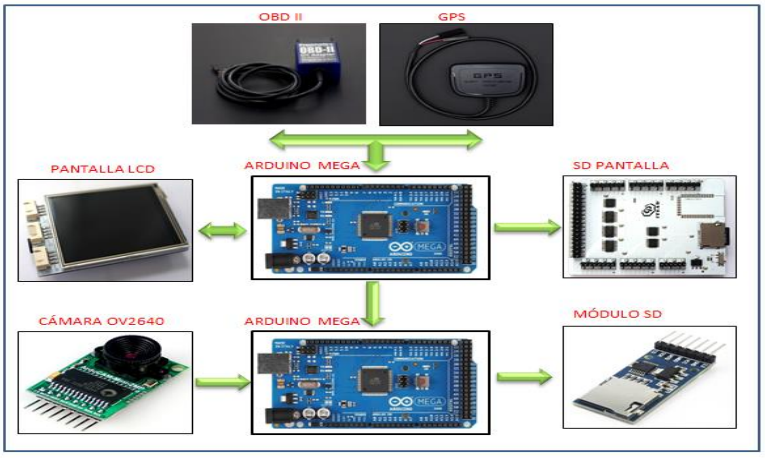


Figura 24. Elementos que componen el módulo de almacenamiento.

3.1.1 Selección de la placa arduino.

Existen varios tipos de placas arduino en el mercado, pero por las características de este proyecto donde se requiere conectar varios dispositivos a una misma placa arduino, entonces se hace necesario contar con una placa que posea muchos pines. El Arduino mega es una placa que cumple con estas necesidades cuenta con 54 pines digitales de los cuales 6 corresponden a pines seriales 3tx y 3rx útiles cuando se quiere conectar dispositivos en serie, 16 entradas análogos. Además, cuenta con 5 pines de interrupción. En la (figura 25) se puede ver el esquema de distribución de pines en la placa arduino mega.

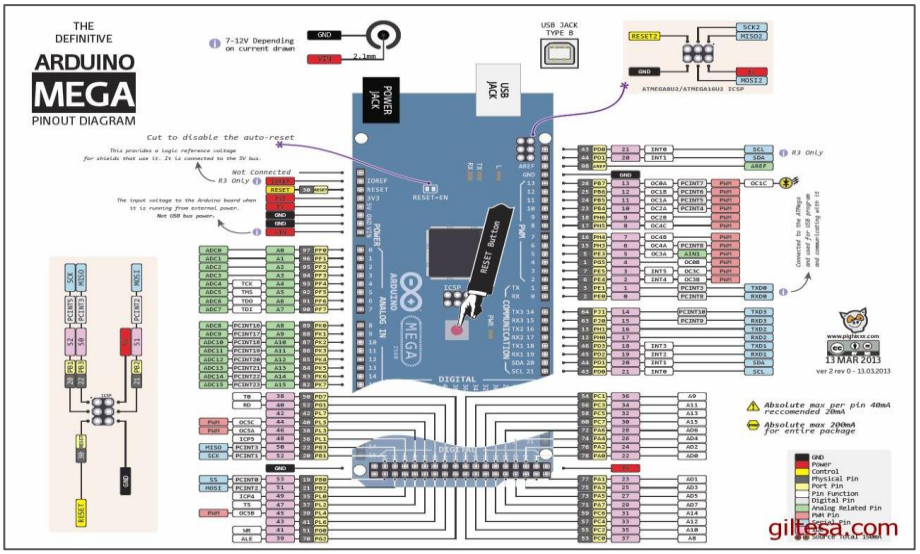


Figura 25. Esquema de pines Arduino Mega 2560.

Fuente: (Giltesa, 2013)

En cuanto a la programación esta placa al igual que todas las versiones de placas arduino cuenta con un software proporcionado por la misma empresa basado en lenguaje C++ a través de un computador ya que cuenta con un puerto usb para transferir datos. Se alimentan básicamente con voltajes en un rango de 7 a 12v inferiores a estos hacen que la placa no suministre el suficiente voltaje a los pines y voltajes superiores provocan sobrecargas y pueden dañar la placa. Posee 256 kb de memoria para lo que tiene que ver con programación y 3 tipos de memorias, la memoria flash que sirve para arrancar el dispositivo, la SRAM que sirve de memoria temporal mientras esté conectado el dispositivo y la EEPROM que es una memoria programable con pulsos eléctricos útil para guardar datos como registro para la siguiente ves que se inicie. En la (Tabla 9) se indica las características principales de las placas Arduino Mega 2560.

Tabla 9. Datos técnicos de la placa Arduino Mega 2560.

Características	Descripción
Microcontrolador	Atmega 2560
Voltaje de operación	5v
Voltaje de entrada recomendado	7 a 12v
Pines digitales	54
Pines análogos	16
Tensión máxima por pin I/O	40mA
Memoria total	256 kb
SRAM	8kb
EEPROM	4kb
Velocidad de reloj	16 mhz
Pines disponibles para interrupción	5 pines
Peso	37g
Dimensiones	101.52mm x 53,3mm
Otros	Posibilidad de configurar pines como pull_up

Fuente: (Trastejant, 2013).

3.1.2 Selección de Pantalla LCD.

La pantalla LCD seleccionada para el módulo es una TFT de 3.2” pulgadas multifuncional diseñada para el uso en la placa Arduino MEGA y Arduino DUE, se puede alimentar esta pantalla con voltajes de 3.3V a 5V. Posee una ranura micro SD en la parte posterior de la pantalla para el almacenamiento de datos, cuenta con dos extensiones UART de comunicación

serie y una extensión para comunicación I2C. Puede ser insertada directamente sobre la placa Arduino Mega.

Figura 26. Pantalla TFT 3.2”

Fuente:(Freematics,2015)

Características de la pantalla.

- Compatible con Arduino Mega 1280/ 2560/ ADK y Arduino DUE
- Voltaje de funcionamiento: 5V / 3,3V
- Resolución LCD 320 * 240 píxeles
- Colores LCD 262K
- Tamaño del LCD 3,2” pulgadas
- Retroiluminación del LCD: LED
- Controlador del LCD IC: SSD1289 (interfaz de datos de 16-bit)
- Almacenamiento micro SD de hasta 32GB
- Puertos de entrada: Serial1 (Rx1/ Tx1/ VCC/ GND), Serial2 e I2C.

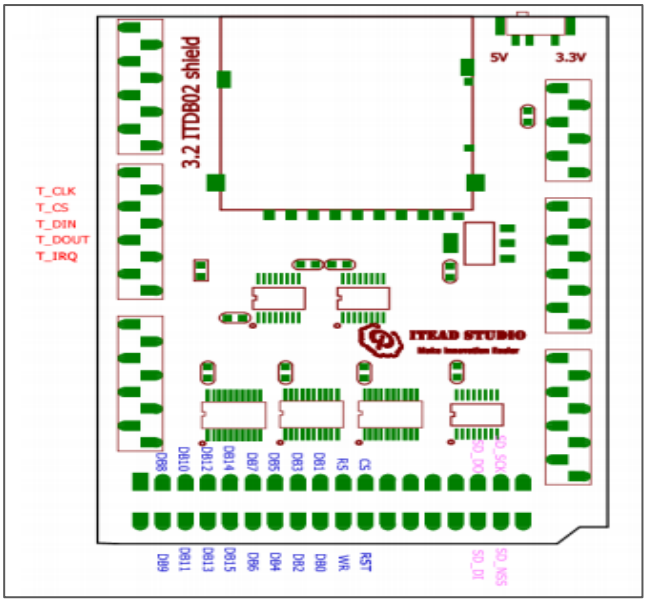


Figura 27. Diagrama de la pantalla TFT 3.2”

Fuente: (Freematics, 2015).

Esta pantalla tiene casi las mismas dimensiones que la placa arduino Mega y encajan perfectamente sobre ella. Pero no hace uso de todos los pines, los únicos pines que usa son los que sirve para el funcionamiento de la pantalla, los 2 puertos serie, el I2C, alimentación, GND y la ranura para la micro SD, los demás pines quedan libres para usarse para cualquier otra aplicación. En la siguiente (Tabla 10) se muestra la distribución de pines utilizados por la pantalla TFT 3.2” en la placa Arduino.

Tabla 10. Lista de pines usados por la pantalla TFT 3.2”

Arduino MEGA PIN	DESCRIPCIÓN
D2	T_IRQ
D3	T_DOUT
D4	T_DIN
D5	T_CS
D6	T_CLK
D22	DB8
D23	DB9
D24	DB10

D25	DB11
D26	DB12
D27	DB13
D28	DB14
D29	DB15
D30	DB7
D31	DB6
D32	DB5
D33	DB4
D34	DB3
D35	DB2
D36	DB1
D37	DB0
D38	RS
D39	WR
D40	CS
D42	RST
D50	SD_MISO
D51	SD_MOSI
D52	SD_SCK
D53	SD_NSS

Fuente: (Freematics, 2015)

3.1.3 Selección modulo GPS.

El receptor GPS empleado en este módulo es un UBX-M8030KA (Figura 27) puesto que es desarrollado para aplicaciones automotrices de gran precisión, debido a que posee una recepción simultánea de hasta 3 GNSS (GPS, Galileo, GLONASS). Este módulo funciona con un voltaje de 5V que recibe de la placa Arduino, además de poseer los pines de comunicación RX, TX que pueden ser conectados al Serial2 de la shield de la pantalla TFT 3.2" y que corresponden los pines 17(RX) y pin 16(TX).

Figura 28. Receptor GPS UBX-M8030KA.

Fuente: (Freematics, 2015).

Este módulo GPS UBX-M8030 posee las siguientes características:

a) Recepción simultánea hasta 3 GNSS.
b) Sensibilidad de navegación de -167dBm.
c) Bajo consumo de corriente.
d) Soporte a todos los sistemas de satélite.
e) Temperatura de funcionamiento de -40° a +105°C.
f) Comunicación UART.

La transmisión de datos de este receptor se da por comunicación serial directamente a la shield de la pantalla TFT 3.2" montada sobre la placa Arduino. Los pines que tiene este receptor corresponden a 5v de donde obtiene su alimentación por parte de la placa arduino, GND que corresponde a la conexión tierra necesaria para cerrar el circuito, D16 es la salida de datos hacia la placa denominada TX y D17 es la entrada de datos desde la placa hacia el receptor sirve de comunicación para indicarle al receptor cuando se inicie y que tarea realice.

Tabla 11. Conexiones del Receptor GPS UBX-M8030KA.

PIN ARDUINO	DEFINICIÓN
5V	Alimentación
GND	Tierra
D16	TX Transmisión
D17	RX Recepción

Fuente: (Freematics, 2015).

3.1.4 Selección de la cámara.

La cámara seleccionada para integrarla en este módulo es la cámara OV2640. Esta es una cámara SPI de alta definición diseñada para capturar tanto fotos como videos puede ser adaptada para trabajar con cualquier placa Arduino siempre y cuando tengan SPI y entradas I2C. Tiene una resolución de 2 y 5 megapíxeles. Posee un reloj sincronizado interno que permite programar el tiempo de disparo entre captura y captura, además se puede ajustar la calidad de la imagen en el lente de la misma.

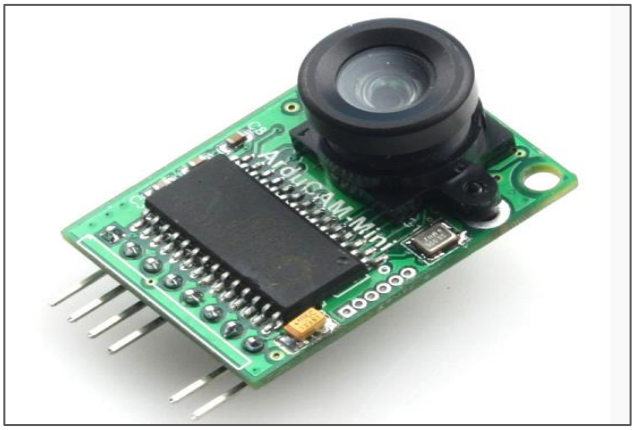


Figura 29. Cámara.

Fuente:(Arducam,2015)

Esta cámara posee las siguientes características:

- 2MP o 5MP sensor de imagen OV2640.
- M12 montaje o desmontaje CS porta lentes con opciones de lentes intercambiables.
- IR sensible con la combinación adecuada de la lente.
- Interfaz I2C para la configuración de sensor.
- Interfaz SPI para los comandos de cámara y flujo de datos.
- Todos los puertos del IO son 5V / 3.3V tolerantes.
- Compatibilidad con el modo de compresión JPEG, modo de disparo simple y múltiple, una operación múltiple captura en tiempo leer, explosión de la operación, el modo de baja potencia y leer, etc.
- 3 ~ 10 fps de salida de vídeo a baja resolución.
- Bien acoplado con tablas estándar de Arduino.
- Biblioteca de código fuente abierto para Arduino.

Tabla 12. Descripción de pines de la cámara OV2640.

Nº de pin	**Nombre del Pin**	**Tipo**	**Descripción**
1	CS	Entrada	SPI entrada de selección esclavo.
2	MOSI	Entrada	SPI salida master, entrada de esclavos.
3	MISO	Salida	SPI maestro entrada principal, salida del esclavo.
4	SCLK	Entrada	reloj en serie SPI
5	GND	Suelo	tierra de la fuente
6	+ 5V	PODER	alimentación de 5V de alimentación
7	SDA	Bidireccional	De dos hilos de serie de interfaz de datos de E / S
8	SCL	Entrada	Interfaz en serie de dos hilos Reloj

Fuente: (Arducam,2015)

Además, posee una serie de funciones adicionales para un amplio rango de aplicaciones.

- **Modo de captura individual. –** Este es el modo de captura por defecto de la cámara. Después de emitir una orden de captura a través del puerto SPI, la cámara espera un nuevo cuadro.
- **Modo de captura múltiple.** - Es el modo de captura avanzada. Al establecer el número de tramas en el registro de captura, la cámara captura consiguientes cuadros después de emitir el comando de captura. Hay que tomar en cuenta que el número de fotogramas debe ajustarse correctamente y asegúrese de que no superen el máximo espacio de memoria.
- **La compresión JPEG.** - La función de compresión JPEG se implementa en el sensor de imagen. Con valores de los registros adecuados al sensor, el usuario puede obtener una resolución diferente con la salida de imagen JPEG corriente. Se recomienda utilizar la salida JPEG para obtener una resolución más alta que en el modo RGB, debido a la limitación de la memoria intermedia de trama.
- **Normal de lectura y operación de lectura de ráfaga.** - El funcionamiento normal de lectura lee los datos de cada imagen mediante el envío de un comando de lectura en un SPI leer ciclo de operación. Mientras se completa la operación de lectura sólo es necesario enviar un comando de lectura a continuación, leer datos de imágenes múltiples

en un solo SPI leídos ciclo de operación. Se recomienda utilizar operación de lectura de ráfaga para obtener un mejor rendimiento de procesamiento.

- **Rebobinar operación de lectura.** - A veces usuario quiere leer la misma trama de datos de imagen varias veces para el procesamiento, el rebobinado operación de lectura está diseñado para este propósito. Al restaurar el puntero de lectura al principio de los datos de imagen, el usuario puede leer los mismos datos de la imagen desde el punto de inicio de nuevo.
- **Modo de bajo consumo.** – A veces es necesario ahorrar energía cuando en el estado de reposo, el cámara ofrece el modo de bajo consumo para reducir el consumo de energía, por el cierre de los circuitos de sensor y de la memoria.
- **Control de Sensor de Imagen.** - Función de control del sensor de imagen se implementa en el sensor de imagen. Al establecer conjunto adecuado de la configuración de registro, el usuario puede controlar la exposición, el balance, brillo, contraste, saturación de color blanco y etc.

El circuito de la cámara compone de los siguientes implementos: Arduino nano, placa constituida por relés, módulo y controlador de imágenes. Su funcionamiento se constituye en base a pulsos de contacto enviados por el Arduino Nano tanto para el encendido, cambio de estado de video a fotografía, captura de fotografía con intervalos de tiempo para el correcto procesamiento de la imagen y apagado del sistema.

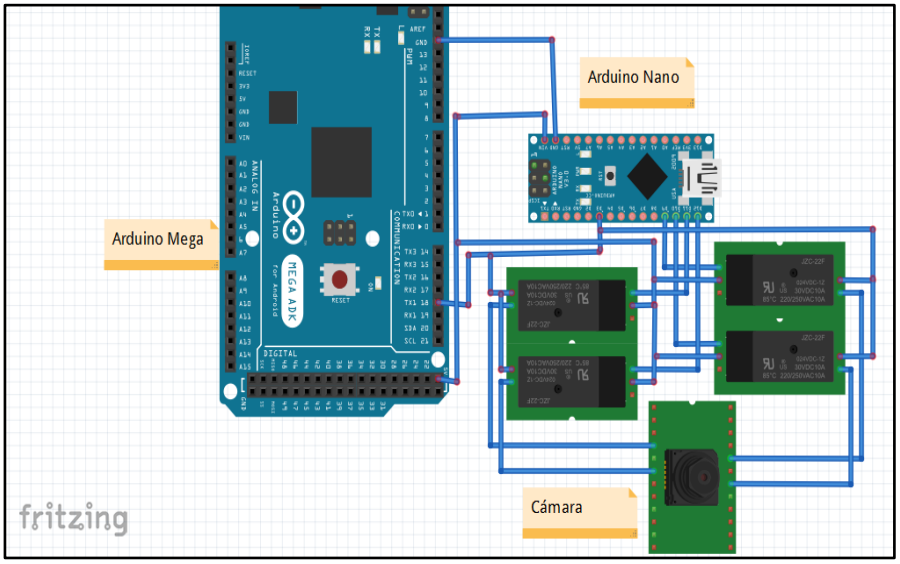


Figura 30. Diagrama de conexión.

3.1.5 Sistema de adquisición de estado de marcha.

Para poder conocer la posición de la marcha insertada se opta por utilizar sensores fines de carrera, los cuales puede trabajar como contactos normalmente abiertos o como contactos normalmente cerrados. Para esta aplicación en específico se requiere usarlos como contactos normalmente abiertos.

Características contacto NA.

Todas las entradas se consultan periódicamente para apreciar su estado de señal. Si se conecta a una entrada un contacto normalmente abierto, dicha entrada puede tener 2 valores lógicos 1 y 0. El estado de señal 1 solo se produce cuando se acciona el contacto y 0 cuando no se lo accione

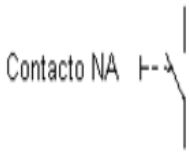
El emisor de señal es un	El emisor está	Tensión de entrada	Estado de la señal en la entrada
Contacto NA 	accionado 	existente	1
	no accionado 	no existente	0

Figura 31. Descripción de un contacto normalmente abierto.
Fuente: (Uniovi, 2014)

Entonces para conocer en que marcha se encuentra se requiere de 5 sensores fines de carrera conectados en circuito normalmente abierto, los cuales representan las 5 marchas del vehículo. Montados en un molde de la forma del ranurado de la palanca de cambios, cuando se inserte la marcha se accionará el sensor de carrera correspondiente a la marcha insertada cambiando su valor a 1 que es lo que leerá la placa arduino y conforme a la programación se le asigna un número y guarda el valor de la marcha. Mientras todos los sensores fin de carrera estén en estado 0 el programa cargado en la placa arduino entiende esto como un estado neutro.

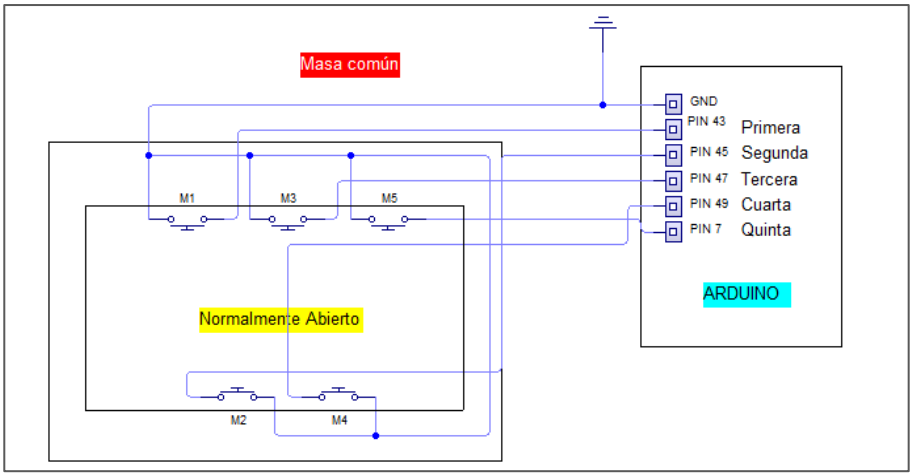


Figura 32. Diagrama de conexión del sistema del estado de marcha.

En la (Figura 32) se puede observar el esquema de conexión de los sensores fin de carrera, los cuales tiene 2 cables de conexión 1 es el positivo y el otro negativo. Todos los fines de carrera comparten masa la cual por seguridad hay que usar la de la placa Arduino y el chasis. Mientras que los cables positivos de cada uno se conectan directamente a sus pines respectivos en la placa arduino. En la tabla 13 se puede observar los pines asignados a cada marcha.

Tabla 13. Pines en Arduino usados para detectar cada marcha.

PINES DE ENTRADA	**MARCHA/FRENO**
D7	QUINTA
D43	PRIMERA
D45	SEGUNDA
D47	TERCERA
D49	CUARTA
D9	PEDAL FRENO

Estos sensores fin de carrera para esta aplicación deben trabajar en 2 estados low y high, se necesita estos 2 valores para determinar la marcha insertada. Es posible que, debido a diferentes factores como el ruido eléctrico o variaciones de voltaje, el valor caiga en un rango indefinido y

no sea posible determinar el estado, para evitar este inconveniente se puede utilizar resistencias Pull up y Pull down.

Resistencia Pull down.

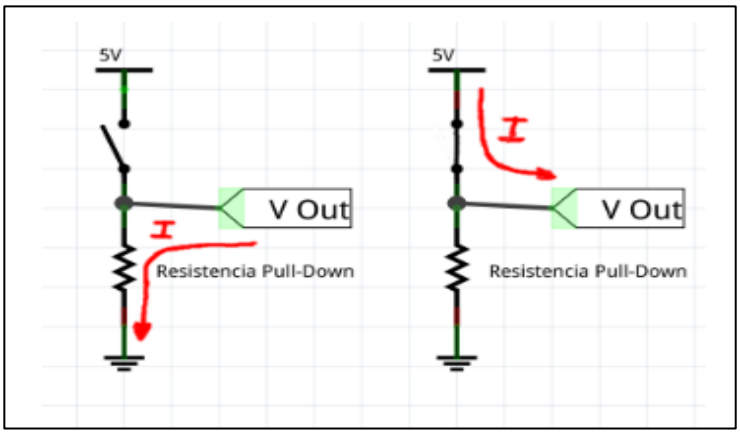


Figura 33. Resistencia Pull down.
Fuente: (OVTOASTER, 2015)

Como se observa en la (Figura 33) la resistencia se conecta a GND, de esta manera cuando el contacto este abierto la corriente es dirigida hacia la resistencia dejando un valor de salida 0 o low y cuando el contacto este cerrado la corriente es dirigida a la salida dejando un valor lógico 1 o high.

Resistencia Pull up.

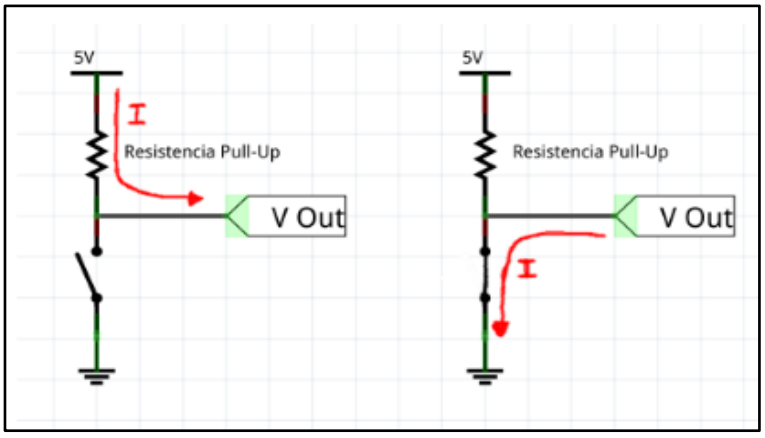


Figura 34. Resistencia Pull up.
Fuente: (OVTOASTER, 2015)

En la (Figura 34) pasa todo lo contrario, cuando el contacto está abierto la corriente va desde la fuente de alimentación hacia la salida dando un valor lógico high y cuando el contacto está cerrado toda la corriente fluye a la resistencia entregando un valor de salida 0 o low.

Para este caso se opta por usar las resistencias pull up que las placas arduino poseen por defecto y acondicionando la señal obtenida a la requerida dentro de la programación del arduino. Esta resistencia pull up también actúa como un circuito de protección en caso de producirse un pico de voltaje cortara el circuito evitando un daño en el pin por sobrecarga.

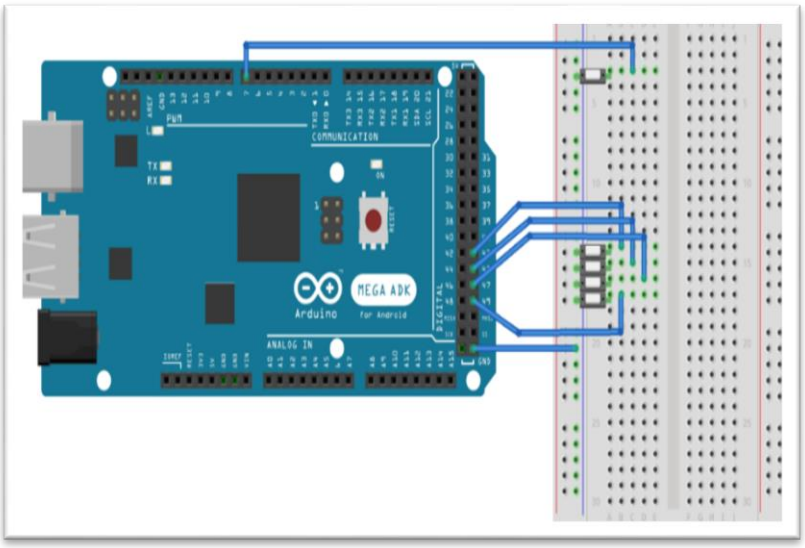


Figura 35. Esquema de conexión del sistema de estado de marcha.

3.1.6 Sistema de adquisición de estado del freno.

En el caso del freno se determina 2 estados posibles on y off. On cuando se accione el pedal de freno y off cuando no se actúa sobre el pedal. Su desarrollo es similar al utilizado en el sistema de estado de marcha. Pero a diferencia del anterior sistema aquí se opta por conectar un pulsador en condición normalmente cerrado. Esto es porque el pedal mantiene el pulsador presionado, mientras que cuando se presione el pedal del freno este deja libre el pulsador.

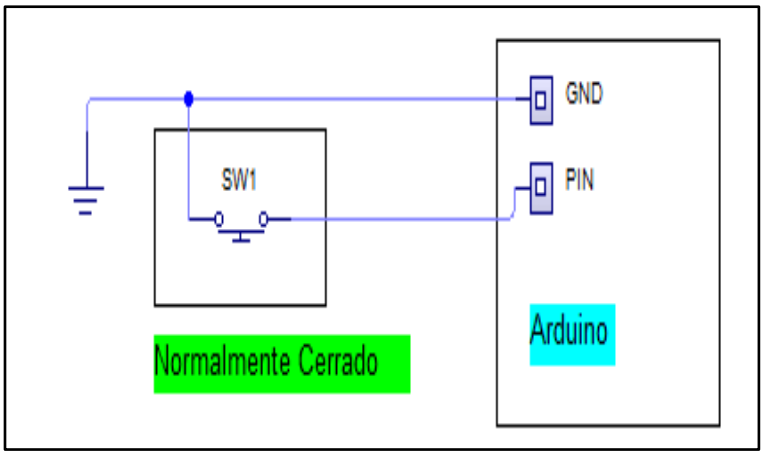


Figura 36. Diagrama de conexión del sistema de estado del

Características contacto NC.

Toda entrada conectada en circuito normalmente cerrado, su señal es 0 porque siempre esta accionada y cuando se deje de actuar sobre ella cambia a 1. Para distinguir entre NA y NC el programa debe incluir instrucciones de consulta del estado de señal en la entrada.

Contacto NC	accionado	no existente	0
	no accionado	existente	1

Figura 37. Cuadro de descripción de un contacto normalmente cerrado.

Fuente: (Uniovi, 2014)

Al igual que el sistema de estado de marcha, este sistema también debe ser conectado a una resistencia pull up para evitar los problemas anteriormente expuestos. De esta manera cuando el pulsador este presionado el programa interpreta como estado off y cuando se accione el pedal y este deje libre el pulsador lo interpretara como estado on (Figura 38).

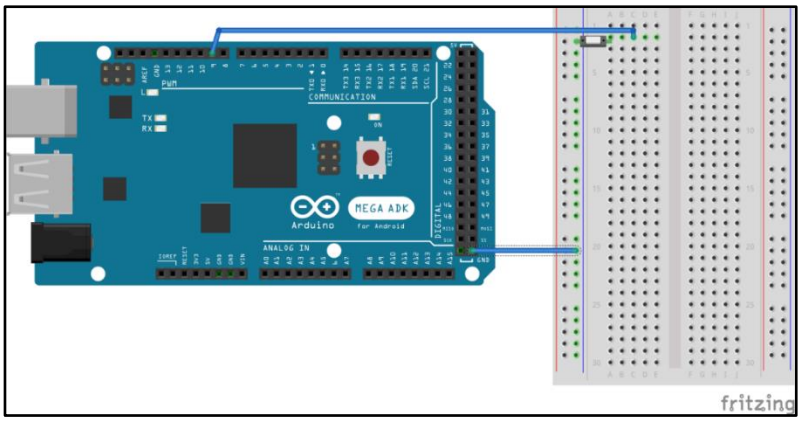


Figura 38. Esquema de conexión del sistema de estado del freno.

3.1.7 Botones de operación.

El módulo de almacenamiento de datos on-board está dotado de 3 botones de operación. Cada botón tiene una función en específico. El primer botón es el de inicio el cual al ser presionado da inicio a la grabación de los datos en su respectiva memoria SD. El segundo botón es el de detención el cual detiene la grabación de datos y el tercero un botón de reset cuya función es resetear el contador de datos para cuando se requiera volver a grabar nuevos datos.

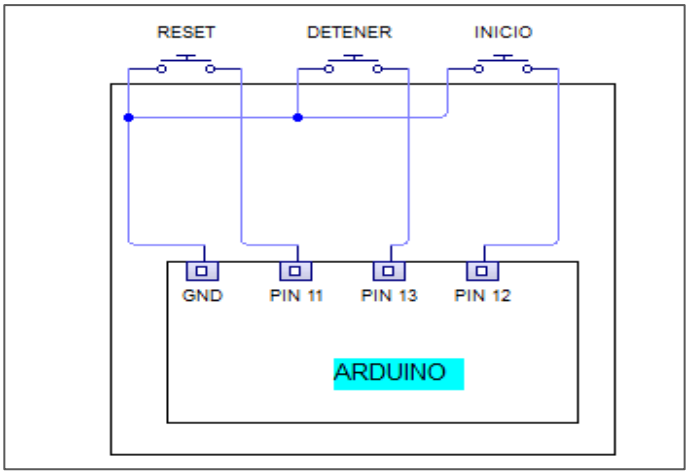


Figura 39. Esquema de conexión de los botones de operación.

Estos botones físicos esta colocados a un costado del módulo de almacenamiento de datos on-board, sus conexiones son sencillas poseen 2 salidas negativo y positivo. El negativo se conecta al GND de la placa arduino y es compartida por todos los botones y el positivo de cada botón se conecta a su respectivo pin del Arduino.

En la (Tabla 14) se muestra los pines asignados dentro de la programación y en la placa arduino y la función que realiza cada botón, además consta de una masa común compartida por todos los botones.

Tabla 14. Lista de pines programados en Arduino para los botones de operación.

Pines	Función
GND	Conexión a tierra compartida por todos los botones.
D12	INICIO
D11	DETENCIÓN
D13	RESETEO

3.2 Diseño de software para el módulo de almacenamiento de datos.

3.2.1 Fases de diseño

La construcción del módulo de almacenamiento de datos se realiza en dos fases, en la primera se integran los distintos elementos para la construcción del módulo en la plataforma Arduino MEGA el cual está conformado por sub funciones de trabajo como son el inicio-fin de almacenamiento de datos y reseteo de los valores para una nueva grabación, sincronizado y orden de inicio de captura de la cámara más los datos provenientes del GPS y acelerómetro.

La segunda fase es el desarrollo de software mediante el programa Labview para un análisis de todos los datos almacenados, que consta de 5 aplicaciones agrupados en un interfaz más la portada y el apartado de configuración e instrucciones, además de una herramienta de filtrado de datos por hora. Las 5 aplicaciones son las siguientes: una gráfica mixta que permite la comparación de curvas de funcionamiento de los datos de gestión electrónica del motor y otra con graficas individuales, una función donde se pueda simular todos los datos como si fueran en tiempo real más una tabla de valores máximos y mínimos, un visualizador de imágenes y una tabla donde se pueda mostrar los datos ordenados y un generador de reportes en Excel para que el usuario pueda imprimirlo.

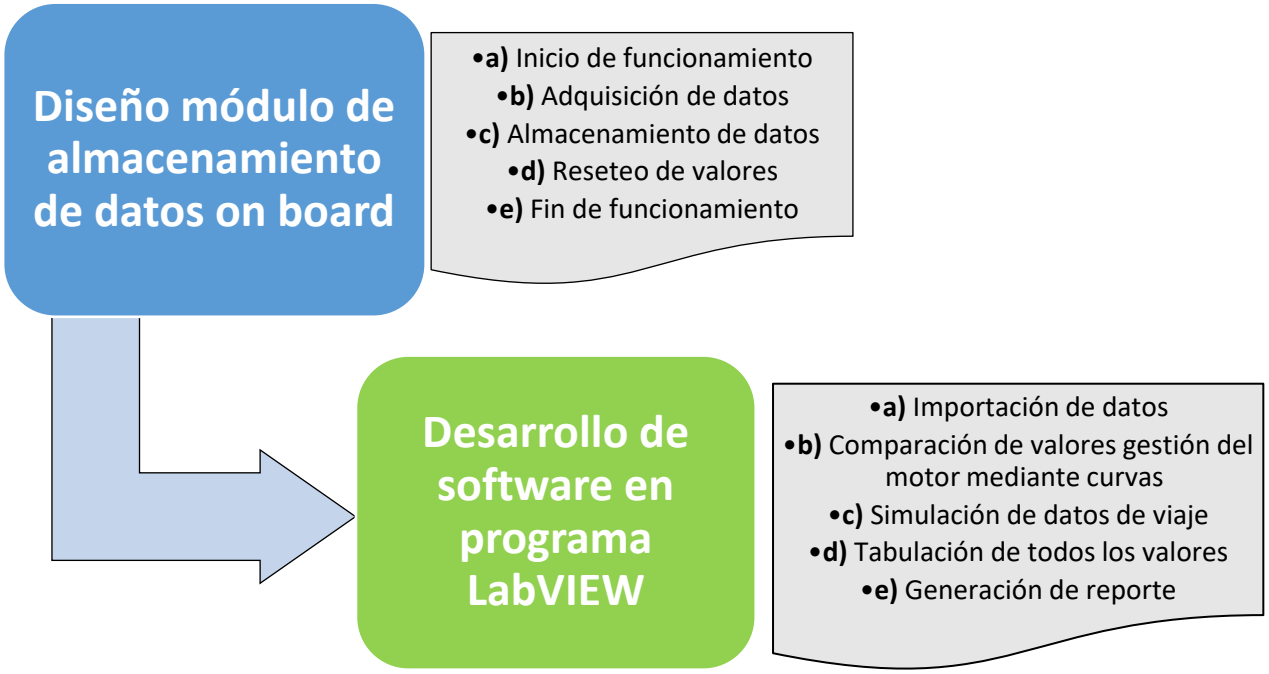


Figura 40. Diagrama de proceso Módulo/Interfaz Labview.

El módulo debe iniciar reconociendo el protocolo del vehículo para poder tener acceso a los datos, una vez establecida la comunicación debe indicar en la pantalla el mensaje OBD listo, en ese momento el módulo ya comienza a adquirir datos sin embargo no los almacena hasta que no reciba la orden por medio del botón de inicio de grabación.

Una vez accionado dicho botón el módulo comienza un proceso de sincronizado, donde espera el tiempo que le toma a la cámara encenderse y cambiar a su modo captura para iniciar la grabación al mismo tiempo y en ese momento comenzar guardar las cadenas de datos a 1 segundo en la tarjeta sd de la placa principal con el nombre de PIDS.txt, mientras que la cámara toma y guarda fotos a 1.5 segundos en un sd secundaria. Cuando se accione el botón de fin de grabación el módulo debe dejar de grabar datos y cerrar el archivo tanto de los datos como los de la cámara, de no usar el botón de reinicio al comenzar la siguiente grabación debe continuar con el último valor del registro, caso contrario debe regresar a 0 su contador.

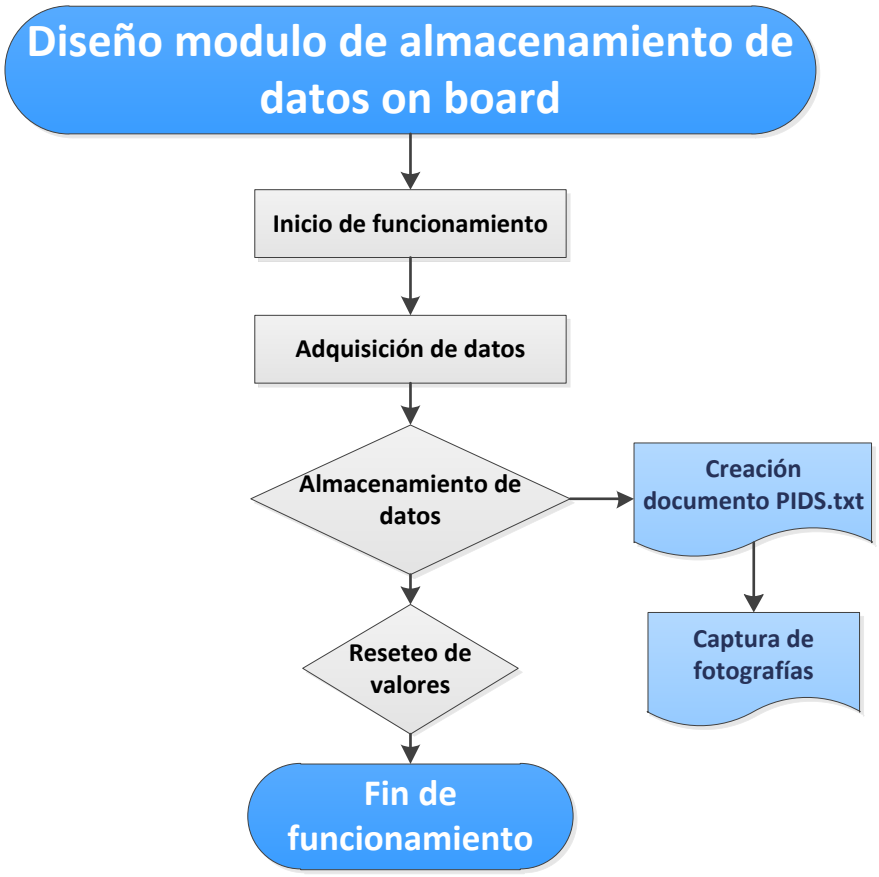


Figura 41. Flujograma de funcionamiento para módulo de almacenamiento.

Luego de extraer las 2 micro sd con las cuales cuenta el módulo, se importa esos datos a la computadora y por ende a la interfaz. Donde cada una recibe un tratamiento diferente dentro de la interfaz mientras que para la lectura de las fotos se debe definir el formato de la imagen que para este caso son jpg Labview no tiene problemas para leer este tipo de formatos de imagen, mientras que para los datos si se debe seguir 1 proceso, puesto que el archivo generado esta creado en forma de cadena de texto separado por comas cada dato y en un orden que se define en la programación.

Entonces una vez ingresado en el path para que Labview lea el archivo txt pasa por un proceso de clasificación de datos, para ello se deben crear arrays individuales para cada dato, en este caso son 21 datos. Cada array tiene el nombre del dato que le corresponde, una vez se indica a Labview que reconozca el carácter de la coma (,) y cada vez que lo encuentre tome los datos anteriores a este y los guarde en uno de los array y asi con cada dato hasta completar los 21

arrays. Una vez hecho esto se agrupa estos arrays en un clúster para poder llevarlos a cada una de las funciones como lo son las gráficas, el simulador, la tabla de datos mediante variables locales.

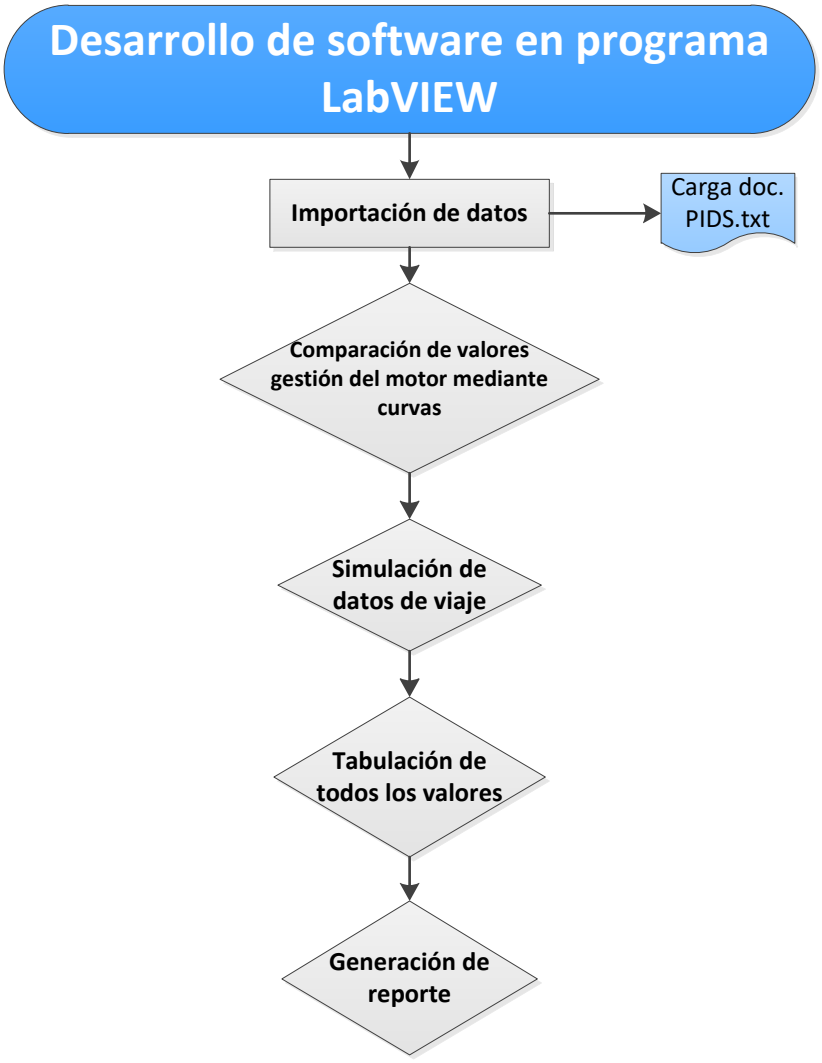


Figura 42. Flujograma de funcionamiento para la interfaz en Labview.

3.2.2 Programación para el módulo de almacenamiento de datos.

El código de programación mediante Arduino, permite la toma de datos a bordo del vehículo, mediante comandos, PIDs, librerías e instrucciones. De esta manera se puede interpretar la información proveniente de la ECU del vehículo y de los demás dispositivos conectados con lo son el GPS, acelerómetro, sistema de marcha y freno para posteriormente almacenar esta información en un documento ".txt", en el cual se guarda datos como: fecha, hora, ubicación,

aceleraciones de ejes x,y,z, velocidad, rpm, temperatura del motor, MAP, voltaje de batería, marcha y freno. Conjuntamente al módulo se programa una cámara en otra placa la cual almacena fotografías del trayecto en formato ".jpg".

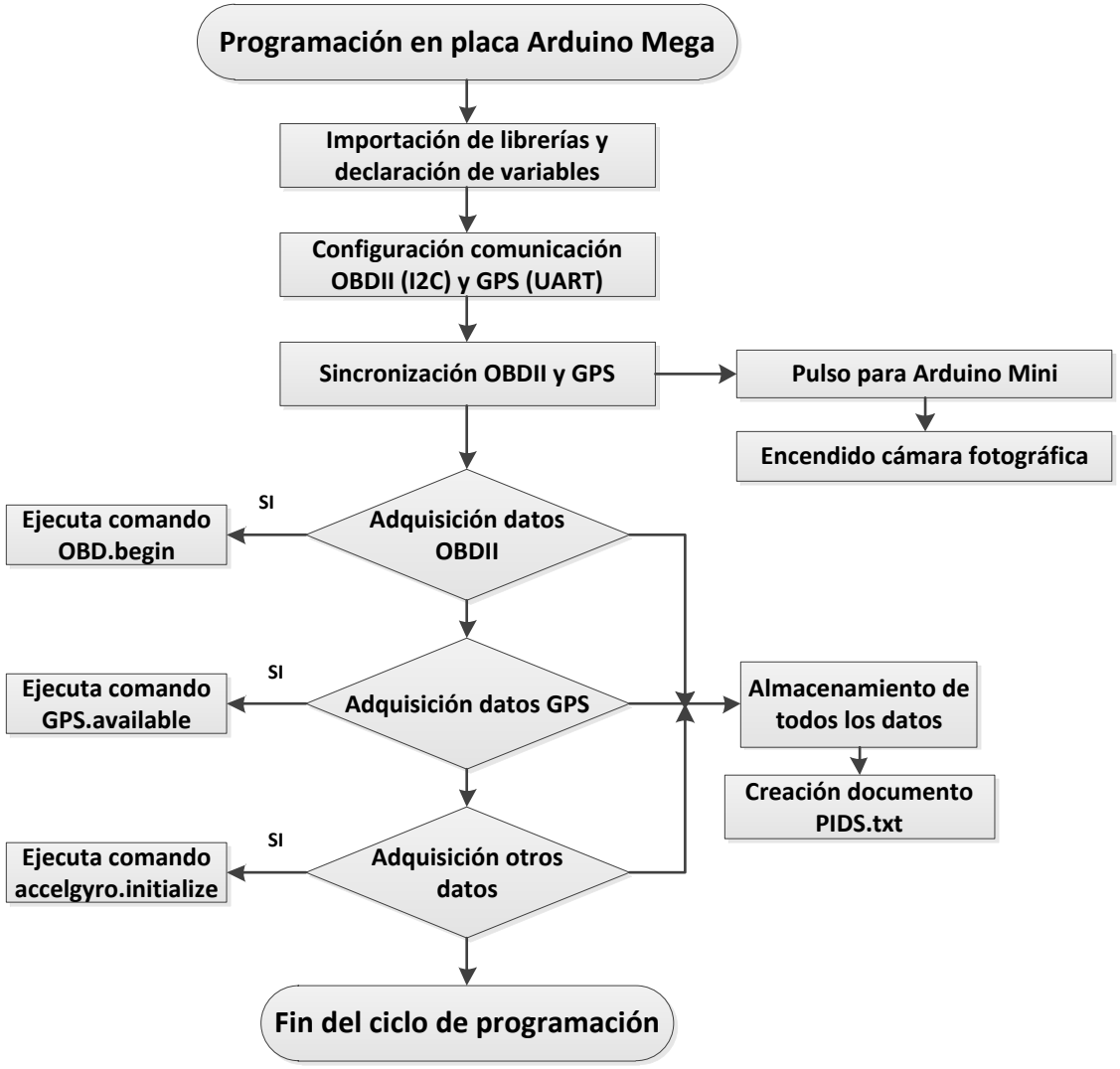


Figura 43. Flujograma de proceso para el módulo de almacenamiento de datos.

El funcionamiento del módulo OBDII conjuntamente con la placa Arduino Mega para la obtención de datos de gestión electrónica del motor del vehículo, se la realiza mediante un código de programación en la cual es necesaria la importación de ciertas librerías, que permiten realizar instrucciones de funcionamiento. También es necesario realizar la declaración de las variables que serán utilizadas durante todo el código para la obtención de valores (Figura 44).

rpm_led_uart | config.h | images.h

```
#include <EEPROM.h>
#include <TinyGPS++.h>
#include <Arduino.h>
#include <SPI.h>
#include <SD.h>
#include <MultiLCD.h>
#include <I2Cdev.h>
#include <MPU9150.h>
#include <UTouch.h>
#include <TimerOne.h>

#include <Wire.h>
#include "images.h"
#include "config.h"
#include <OBD2UART.h>

#define STATE_OBD_READY 0x2
#define STATE_MEMS_READY 0x10
#define STATE_GUI_ON 0x20
```

Figura 44. Librerías y variables incluidas en el código Arduino.

Los valores de gestión electrónica del motor se los obtiene mediante PIDS que dependiendo del vehículo que se emplee se pueden añadir más, estos datos se adquieren mediante los comandos de la librería <OBD2UART.h>. Para obtener estos valores es necesario conocer a detalle la lista de PIDS que conforman el sistema de diagnóstico a bordo del vehículo. Bajo la instrucción readPID () se indica a la placa que es lo que queremos leer, además se debe agregar dentro de los corchetes el PID correspondiente del cual deseemos obtener el valor, por ejemplo, readPID (PID_SPEED) como se muestra en la (Figura.45).

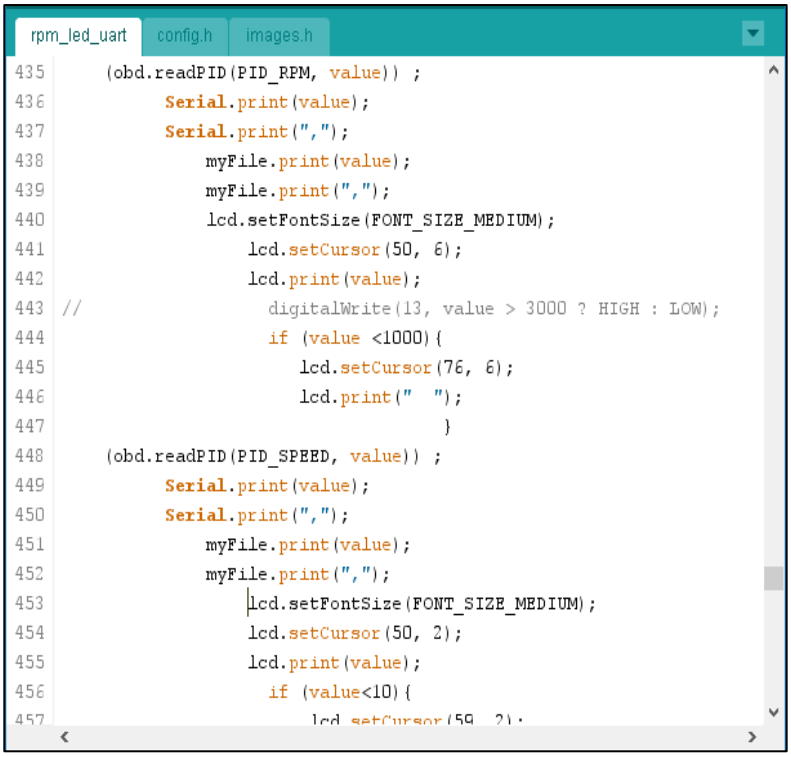


Figura 45. Llamado de funciones mediante PIDs.

Luego de enlazarse a la Ecu del vehículo, el módulo comienza a leer la lista de PIDs preestablecida en la programación y comparar con las señales del OBD una vez comparada, si la señal del OBD corresponde a alguno de la lista, empieza a tomar su valor correspondiente, este proceso lo realiza para cada uno de los PID de la lista. Todo este proceso se debe realizar en un lapso de 1 segundo que es el tiempo de captura y el tiempo en el que debe completar la cadena de datos con los valores de los demás dispositivos conectados.

Los valores son impresos en el archivo txt uno seguido del otro, solamente separados por una coma y guardados en la micro sd principal. La lista de PIDs puede ser modificada para incluir más o menos PIDs sin embargo mucho depende si la computadora del vehículo le permite acceder o no, dicho esta para este caso se opta por seleccionar e incluir aquellos PID más comunes y de los que se puede acceder en el vehículo Chevrolet Aveo como lo son: rpm, velocidad, ect, temperatura de admisión, tps, map.

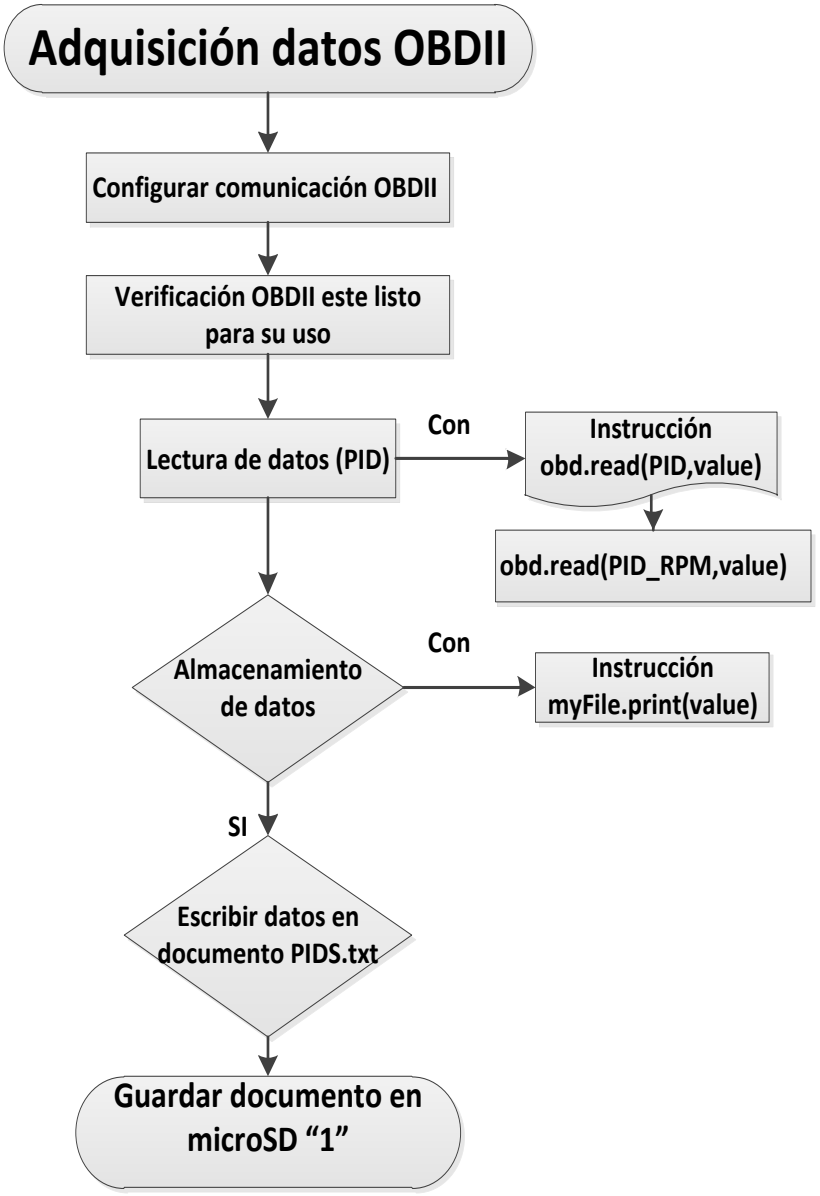


Figura 46. Flujograma de lectura de PIDs y adquisición de datos.

Los datos como fecha, hora, latitud, longitud y altitud son proporcionados por el receptor GPS externo pero que trabaja conjuntamente con módulo a través de instrucciones de la librería <TinyGPS++.h>. Para determinar la hora en nuestro país Ecuador se debe restar 5 horas en la programación ya que el dispositivo GPS nos brinda la hora del meridiano de Greenwich o GMT mientras que para Ecuador la hora corresponde al GMT-5. Los demás datos los obtenemos bajo la instrucción gps.date(), gps.time(), gps.location(), tal como se muestra en la (Figura 47).

```
sketch_dec03a §
1  static void smartDelay(unsigned long ms)
2  {  unsigned long start = millis();
3       do
4       {  while (Serial2.available())
5            gps.encode(Serial2.read());
6       } while (millis() - start < ms);
7  }           Serial.print(gps.date.day());
8               Serial.print(",");
9                    myFile.print(gps.date.day());
10                   myFile.print(",");
11              Serial.print(gps.date.month());
12              Serial.print(",");
13                   myFile.print(gps.date.month());
14                   myFile.print(",");
15              Serial.print(gps.date.year());
16              Serial.print(",");
17                   myFile.print(gps.date.year());
18                   myFile.print(",");
19                int h = (gps.time.hour());
20                    h = h - 5;
21                if (h < 0)
22                    h = h + 24;
```

Figura 47. Código de llamado para la función del receptor GPS.

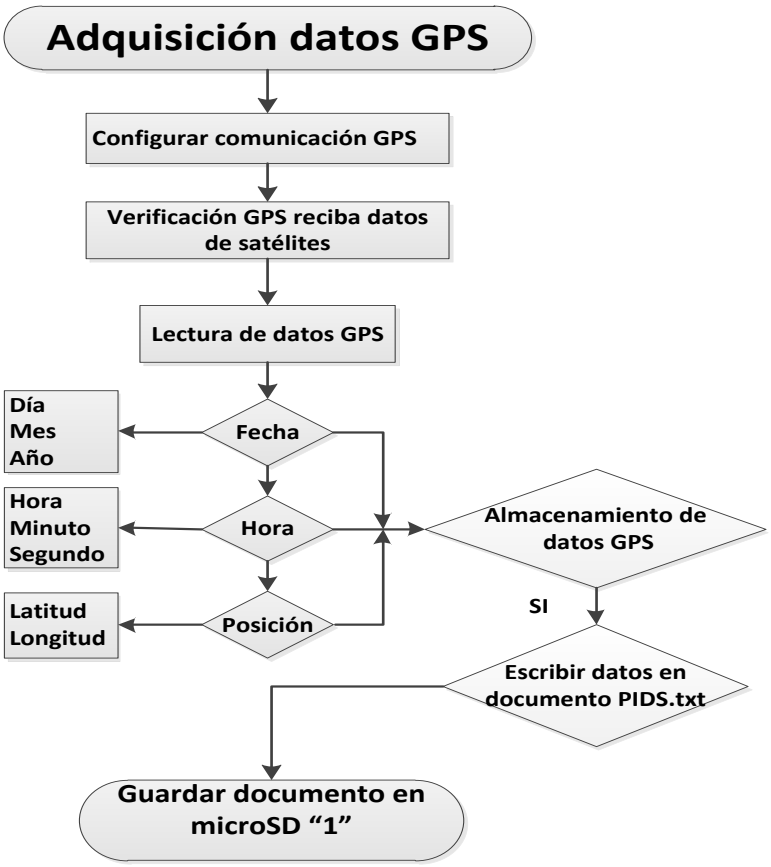


Figura 48. Flujograma de adquisición del GPS.

El receptor GPS al igual que los datos de los PID se programa para que capture datos cada 1 segundo y los tenga listos para enviar para completar la cadena de datos. La sincronización de este receptor es automática cuando se lo alimenta con los 5v necesarios para su funcionamiento. Los datos de hora y fecha que proporciona este receptor son separados en datos individuales de esta forma (día, mes, año, hora, minuto, segundos) más los datos de latitud y longitud, esto para facilitar su extracción y clasificación cuando sea importado a la interfaz de Labview.

Los valores de aceleraciones en los 3 ejes son proporcionados por un micro controlador integrado en el conector OBDII, mediante los comandos e instrucciones de las librerías <MPU9150.h> e <I2Cdev.h>. Para la lectura de los valores de aceleraciones en los 3 ejes es necesario utilizar la instrucción readAccel (ax, ay, az) tal como se muestra en la (Figura.49).

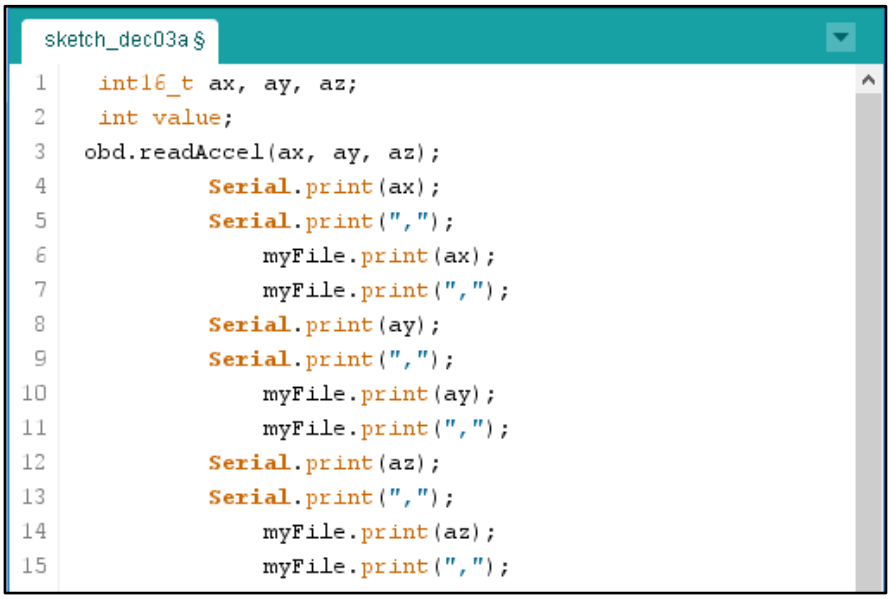


Figura 49. Código de llamado para la función del acelerómetro.

Los datos de aceleración se pueden obtener de 2 maneras, la primera es hacer una lectura directa y conjunta con la librería del OBDII mediante un comando incluido en esta, sin embargo, si la lista de PIDs incluidos en la programación es larga puede causar que la lectura del acelerómetro no se complete por saturación. Por la razón expuesta se opta usar la segunda opción de lectura que es hacerlo independiente del OBD mediante su propia librería y anexándola a la cadena del resto de datos.

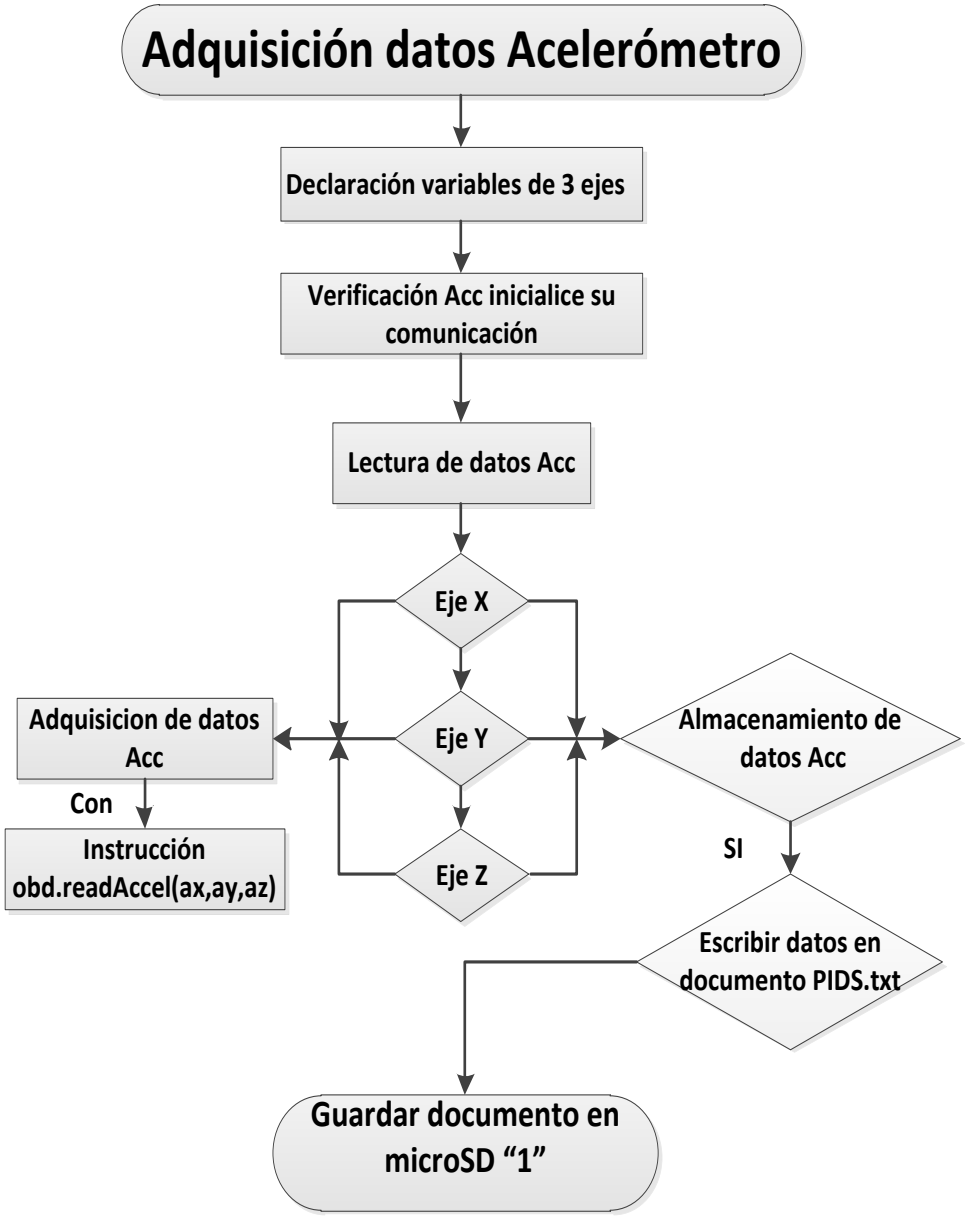


Figura 50. Flujograma de adquisición de datos del acelerómetro.

Para conocer la posición de cada marcha y la activación del pedal de freno se hace uso de sensores de fin de carrera y pulsadores con configuración pull up. La programación del código nos brinda la información de activación mediante la configuración de pines digitales en la placa Arduino Mega. Determinando el cambio de estado del sensor y pulsador de LOW a HIGH conocemos la marcha o pulsación de freno. El cambio de estado lo podemos observar en la siguiente imagen.

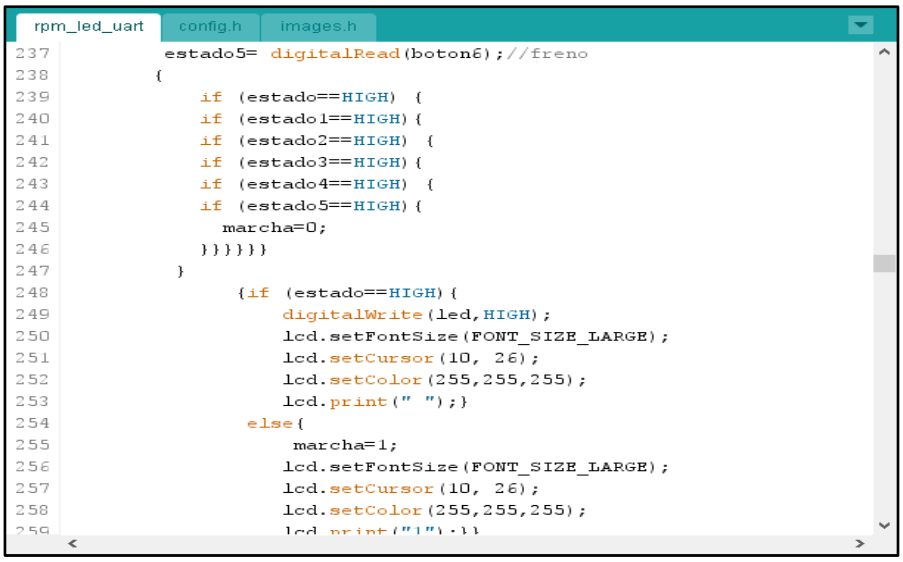


Figura 51. Código para la función de marchas y freno.

A estos códigos se les agrega comandos que gobiernen su ejecución tanto para el OBD, GPS, freno/marcha y acelerómetro, para esto se deben crear botones de acción y mensajes de comprobación. El primer mensaje de comprobación incluido en la programación tiene por objetivo indicar al usuario el estado de la conexión entre la Ecu del vehículo y módulo, para ello se verifica si los pines utilizados tanto para entrada y salida y sus valores iniciales antes de cambiar de estado responden a la comunicación serial para de esta manera realizar el envío de datos de ser así debe mostrar el siguiente mensaje “OBD listo” esto se consigue mediante la instrucción if.

Para pasar al estado grabación se requiere programar a un botón, esto se consigue metiendo todos los códigos indicados anteriormente dentro de un ciclo infinito que se repita únicamente cuando este botón cambie de estado y para comprobar que su ejecute muestre un mensaje de “GRABANDO” en la pantalla. Para los demás botones de acción como lo son el de detención y el de reinicio se realiza un proceso similar en cual se debe ingresar el código dentro de un bucle donde detecte el cambio de estado del botón y que muestre su respectivo mensaje para comprobar su ejecución “FIN DE GRABACION”, “REINICIAR”.

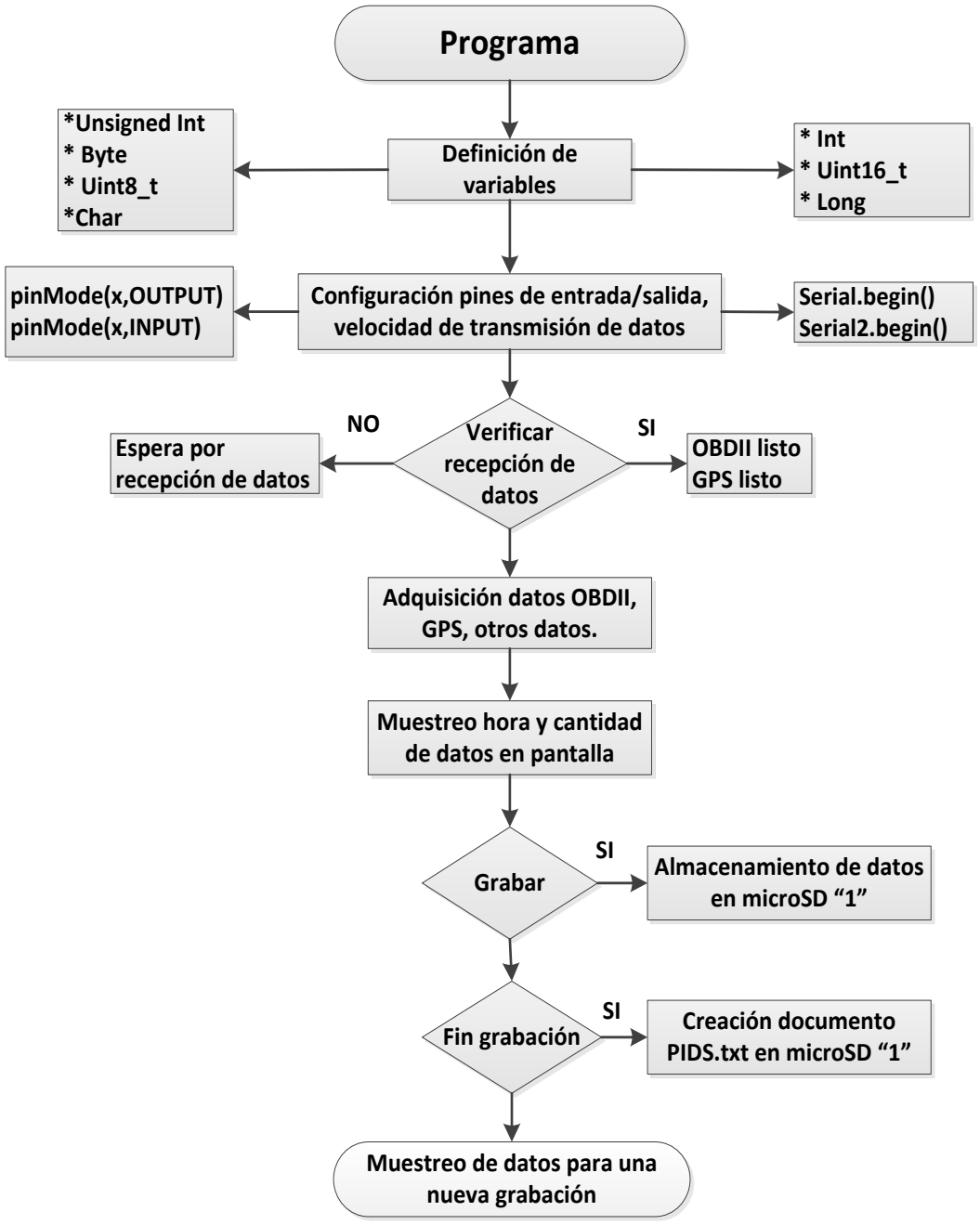


Figura 52. Flujograma de ejecución para el módulo de almacenamiento de datos

Los datos se guardan en el archivo de texto según el siguiente orden separado únicamente por comas (,): N° datos, día, mes, año, hora, minutos, segundos, aceleración en x [m/s^2], aceleración en y [m/s^2], aceleración en z [m/s^2], latitud, longitud, revoluciones del motor [min^{-1}], velocidad [km/h], ECT [°C], temperatura de admisión [°C], TPS [%], voltaje de batería [Voltios], MAP [kg/cm^2], posición de la marcha y estado de freno [on/off].

Tabla 15. Formato trama de datos de módulo de almacenamiento

FORMATO TRAMA DE DATOS DEL MÓDULO DE ALMACENAMIENTO
1,23,1,2017,10,36,36,13088,13619,847,0.0000,0.0000,825,0,76,35,0,13.50,34,1,1,
2,23,1,2017,10,36,37,13088,13619,847,0.3344,-78.1202,853,0,76,35,0,13.50,33,1,1,
3,23,1,2017,10,36,38,13088,13619,847,0.3344,-78.1202,838,0,76,35,0,13.50,33,1,1,
4,23,1,2017,10,36,39,13088,13619,847,0.3344,-78.1202,854,0,76,35,0,13.50,33,1,1,
5,23,1,2017,10,36,40,13088,13619,847,0.3344,-78.1202,826,0,76,35,0,13.50,34,1,1,
6,23,1,2017,10,36,41,13088,13619,847,0.3344,-78.1202,848,0,76,35,0,13.50,33,1,1,
7,23,1,2017,10,36,42,13088,13619,847,0.3344,-78.1202,836,0,76,35,0,13.50,33,1,1,
8,23,1,2017,10,36,43,13088,13619,847,0.3344,-78.1202,831,0,76,35,0,13.50,33,1,1,
9,23,1,2017,10,36,44,13088,13619,847,0.3344,-78.1202,838,0,76,35,3,13.50,37,1,0,
10,23,1,2017,10,36,45,13088,13619,847,0.3344,-78.1202,1197,0,77,35,0,13.50,38,1,0,
11,23,1,2017,10,36,46,13088,13619,847,0.3344,-78.1202,722,0,77,35,2,13.50,47,1,0,
12,23,1,2017,10,36,47,13088,13619,847,0.3344,-78.1202,1036,3,77,35,5,13.50,39,1,0,
13,23,1,2017,10,36,48,13088,13619,847,0.3344,-78.1202,1382,6,77,35,6,13.50,36,1,0,
14,23,1,2017,10,36,49,13088,13619,847,0.3344,-78.1202,1613,9,77,35,0,13.50,24,1,0,

3.3 Diseño de la interfaz en Labview.

Esta interfaz se diseña para que el usuario pueda visualizar y analizar los datos obtenidos por el módulo de almacenamiento de datos on-board para taxis. Cuenta con funciones como filtrado de datos por hora, gráficas, simulador, visualizador de fotos, tabla de datos e incluso puede generar un reporte en Excel con todos los datos recolectados. Cada función dentro de esta interfaz posee botones de acción (play, stop, reset), para que al iniciar el programa todas estas funciones solo estén activas cuando el usuario lo requiera. Además, evita problemas de ralentización o congelamiento del programa, esto es debido a que Labview es un programa que hace uso de muchos recursos del pc para su funcionamiento.

Figura 53. Flujograma de funciones para la interfaz de Labview.

3.3.1 Panel frontal de la interfaz en Labview.

Esta es la parte donde el usuario interactúa con el programa y los datos recolectados. El panel frontal de la interfaz esta echa en un tab control de Labview, cuenta con 7 páginas o ventanas cada una con diferentes funciones. Las 7 páginas que componen este programa son:

a) **Portada. –** Es la página principal del programa, contiene algunos datos informativos acerca del proyecto.

b) **Guía/Configuración/Filtrado. –** Aquí se presenta una breve guía de uso para el uso del programa, además en el apartado de configuración permite seleccionar los archivos de datos obtenidos por el módulo para cargarlos en el programa y la herramienta de filtrado por hora.

c) **Gestión electrónica del motor. –** Con los datos precargados, esta función permite visualizar en una gráfica las curvas de comportamiento de 2 sensores que el usuario seleccione.

d) **Graficas Comparativas. –** Al igual que en la página anterior aquí también se puede visualizar las curvas de comportamiento de los sensores por separado.

e) **Simulador. –** Esta función permite realizar una simulación de los datos obtenidos, tal como si fueran en tiempo real.

f) **Visualizador de fotos. –** Esta función permite ver todas las fotos captadas con la cámara de tráfico incorporada al módulo de almacenamiento de datos, si se desea hacer un análisis más detallado de ellas.

g) **Tabla de datos. –** Esta función permite organizar y clasificar todos los datos recolectados en una tabla, que posteriormente puede ser trasformados en un reporte de Excel.

Figura 54. Panel Frontal de la interfaz en Labview.

3.3.2 Configuración y filtrado de datos por hora.

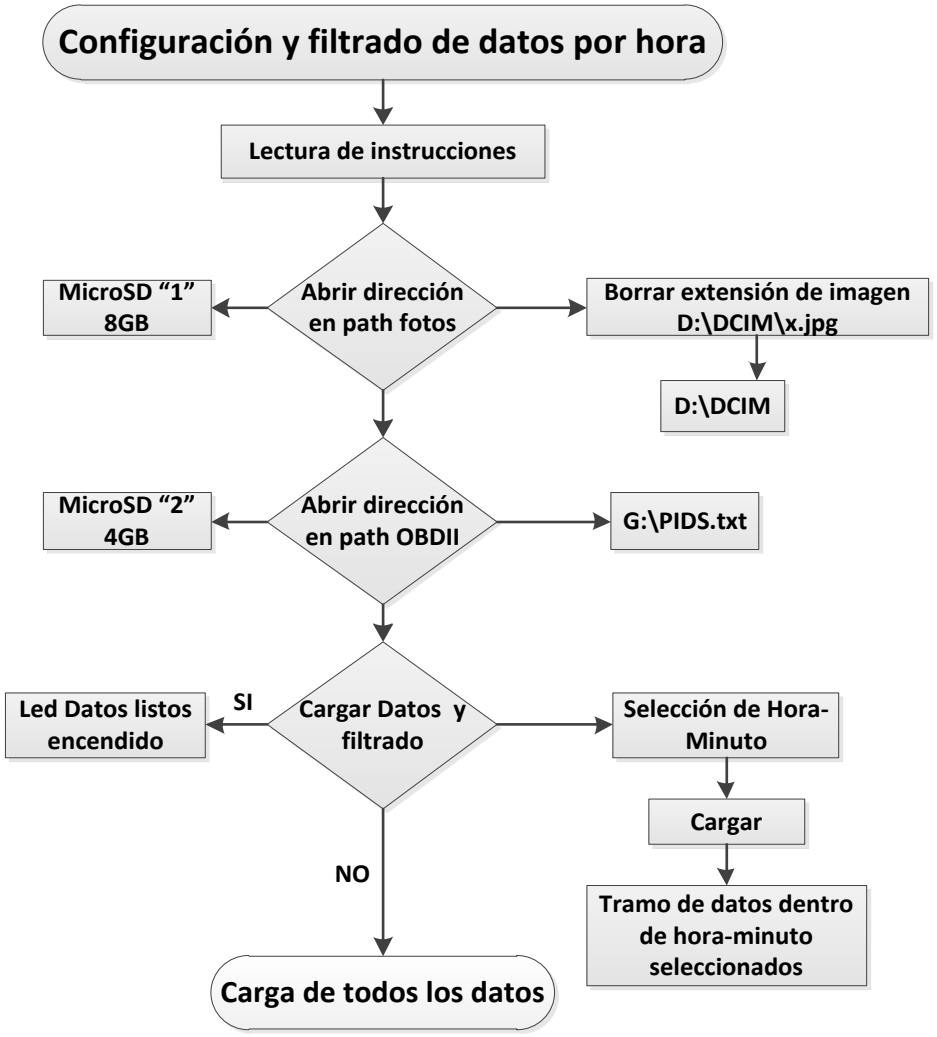


Figura 55. Flujograma de configuración y filtrado de datos.

Esta parte es la base del funcionamiento de todo el interfaz, puesto que aquí se selecciona las rutas de los archivos necesarios para la ejecución de todas las funciones de la interfaz. En la parte izquierda se encuentra una mini guía y consejos para el correcto uso de la interfaz, en la parte derecha se encuentre el apartado de configuración el cual consta de 2 path para seleccionar la ruta del archivo txt y la ruta de la carpeta que alojan todas las fotografías captadas por la cámara. Bajo esta se encuentra la herramienta de filtrado de datos por hora, que permite al usuario seleccionar el tramo de tiempo del que desea ver los datos, la hora inicial y final deben ser colocados por el usuario manualmente y para que este sepa el tiempo en el que puede realizar este filtrado en la parte superior se indica la hora de inicio y de fin de la prueba (Figura 56).

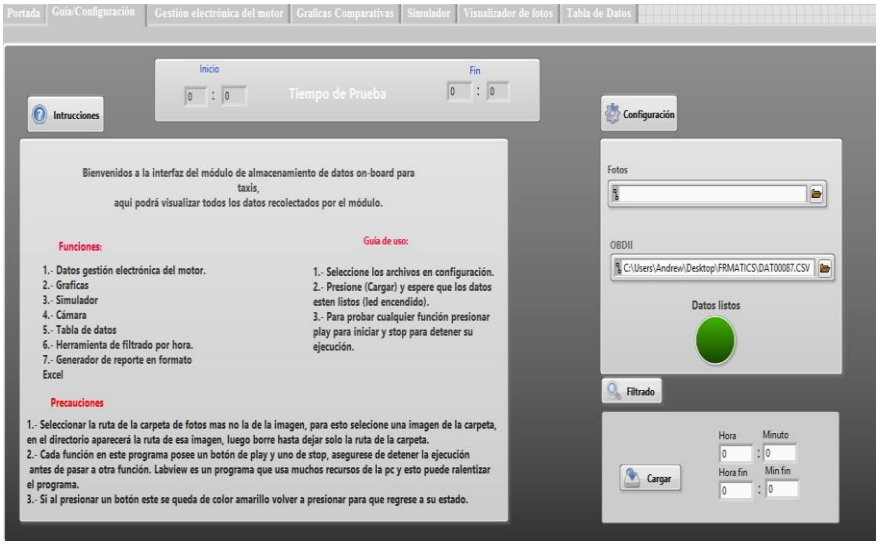


Figura 56. Configuración y filtrado por hora (vista panel frontal).

El código para la herramienta de filtrado por hora se realiza mediante una serie de comparaciones para definir el rango de datos a continuación se detalla las comparaciones realizadas:

Si (Hora inicial = valor ingresado y Minuto inicial ≥ valor ingresado) la sentencia 1 es verdadera.

Si (Hora final = valor ingresado y Minuto final ≤ valor ingresado) la sentencia 2 es verdadera.

Si (sentencia 1 y sentencia 2 son verdaderas) continúe, caso contrario repetir el proceso.

De esta manera se identifica los datos que componen ese tramo de datos, para posteriormente escribirlos en un nuevo clúster de datos que se refrescara cada vez que se ingresen nuevos valores en la herramienta de filtrado. Cada dato ingresa en un nuevo array y se lo conecta a un shift register que incrementa la posición para que el siguiente dato se escriba debajo y no se sobre escriba, este proceso hay que hacerlo con cada dato y asignarles 1 shift register a cada uno (Figura 57).

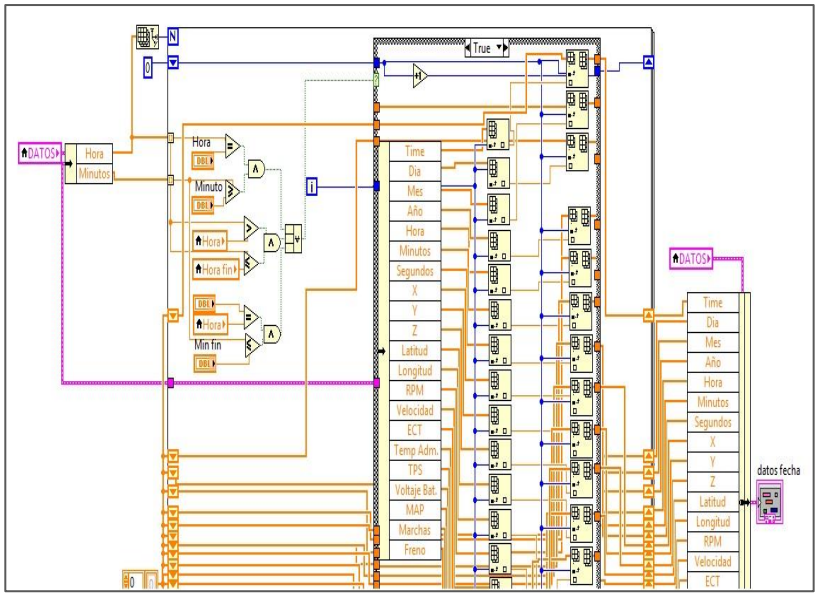


Figura 57. Código de filtrado de datos por hora.

3.3.3 Importado y almacenamiento de datos en la interfaz de Labview.

Los datos obtenidos por el módulo se almacenan en la SD en un archivo de formato texto denominado “PIDS.txt”, todos los datos están ordenados en cadena y separados con comas. El orden de los datos esta dado de la siguiente manera: contador, día, mes, año, hora, minuto, segundos, X, Y, Z, latitud, longitud, rpm, velocidad, ECT, Temperatura de admisión, TPS, voltaje de batería, carga del motor, MAP, marcha, freno (on/off).

Tabla 16. Unidades de los datos obtenidos por el módulo de almacenamiento

Valor	**Unidad**
Dato	[°N]
Fecha	[D/M/A]
Hora	[H: M: S]
Latitud/longitud	[Coordenadas]
Aceleración en los ejes X, Y, Z	[m/s^2]
Revoluciones del motor	[min^{-1}]
Velocidad	[km/h]
ECT	[°C]
Temperatura de admisión	[°C]

TPS	[%]
Batería	[V]
MAP	[kg/cm2]
Marchas	Posición
Freno	On/off

Estos datos son ingresados al programa por medio de file paths, los cuales permiten seleccionar la ruta del archivo. Una vez seleccionados los archivos se deben clasificar y almacenar en clusters. Una vez ingresado el archivo tiene que pasar por un programa que actúa como decodificador reconociendo y eliminando las comas dejando los datos sueltos para ingresarlos en su array correspondiente. Estos arrays están agrupados en clusters, para ello se crearon 3 clusters denominados: “DATOS”, “Estados” donde se manejan datos numéricos luego de realizar la clasificación y extracción de comas más un clúster “datos fecha” que contienen los datos filtrados por hora, estos son los que se usan en todas las funciones del programa.

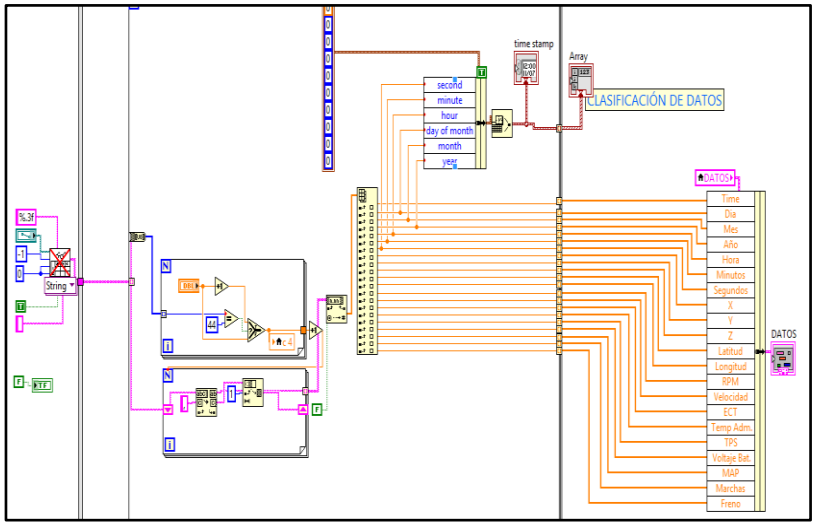


Figura 58. Código para importar y almacenar los datos en Labview.

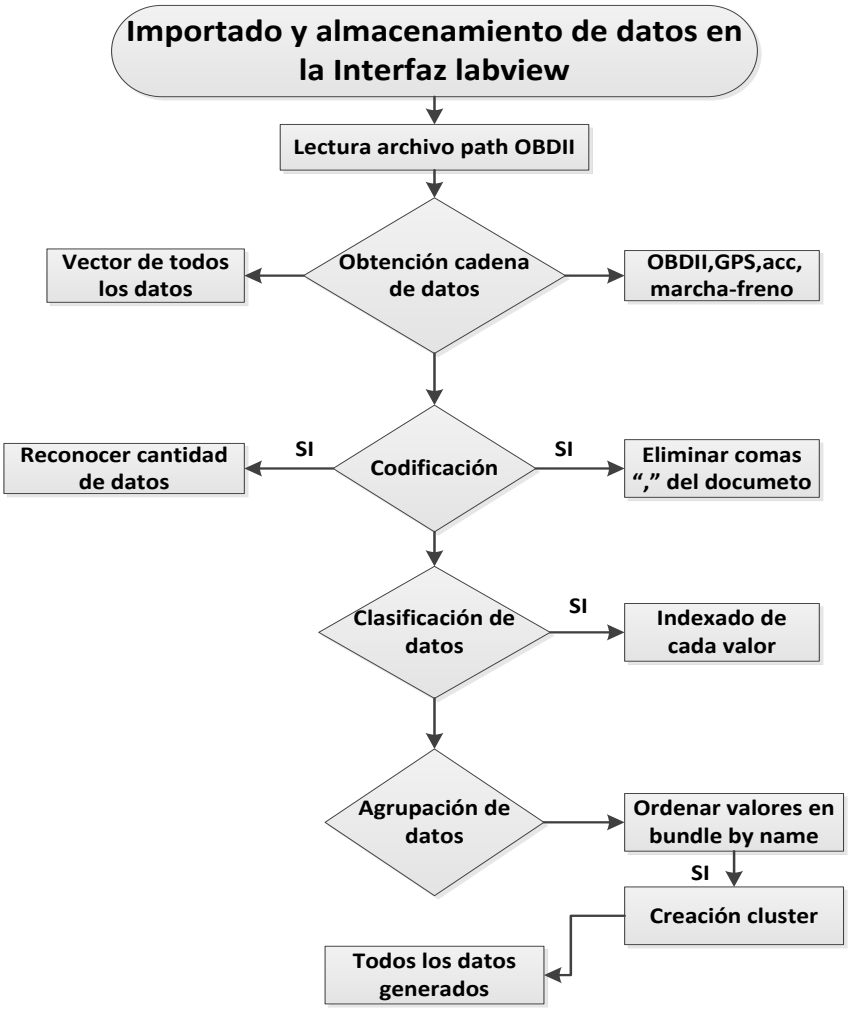


Figura 59. Flujograma de importado y almacenamiento de datos.

3.3.4 Graficas.

Estas graficas permiten ver el comportamiento de la curva de datos de los sensores monitoreados por el módulo. Para realizar estas gráficas se debe crear 2 sub-menús que contenga el nombre de todos de todos los sensores y asignarles un caso con un "case structure" en total se crean 8 casos. Una vez hecho esto se crea una variable local del cluster OBDII que contiene todos los datos de estos sensores, luego se les asigna los valores de los sensores correspondientes a cada uno de los casos creados y finalmente se conecta un grafica "x y" a cada salida de los casos. Debido a que esta grafica requiere la entrada de 2 datos para graficar se crea un caso especial que contiene los datos del contador, quedando en el eje y los valores de los sensores y en el eje x el contador como escala de tiempo.

De esta manera al seleccionar algún sensor en los sub-menús este llama inmediatamente al caso que contenga los datos de ese sensor y las gráficas. En las propiedades de cada gráfica se debe definir el nombre del sensor, así como las unidades en las que están sus valores.

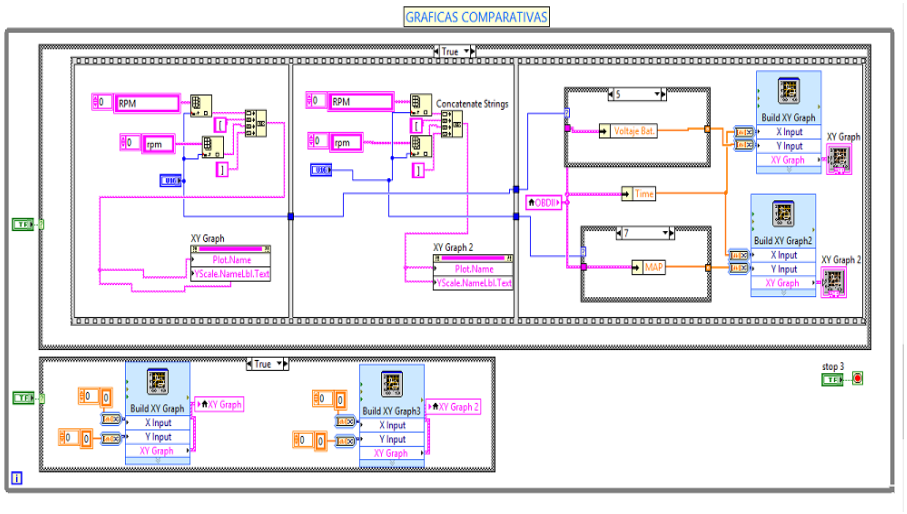


Figura 60. Código en Labview para graficar los datos.

En la siguiente figura se puede observar 2 tipos de botones, el botón play que sirve para cargar los datos en la gráfica, mientras que el botón refresh permite borrar los datos de la gráfica. El código de funcionamiento del botón refresh consiste en ingresar un arreglo de ceros a la misma gráfica, como si estos fueran nuevos valores de este modo se consigue borrar los datos anteriores.

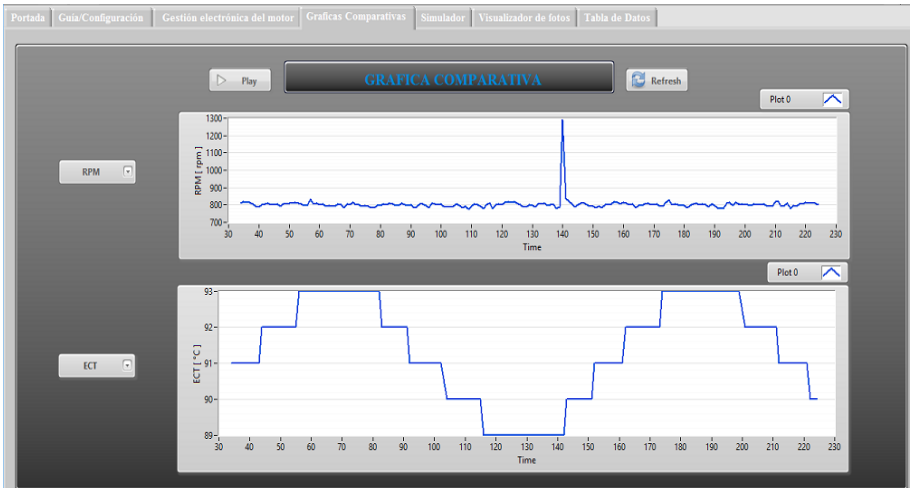


Figura 61. Grafica de datos en el panel frontal.

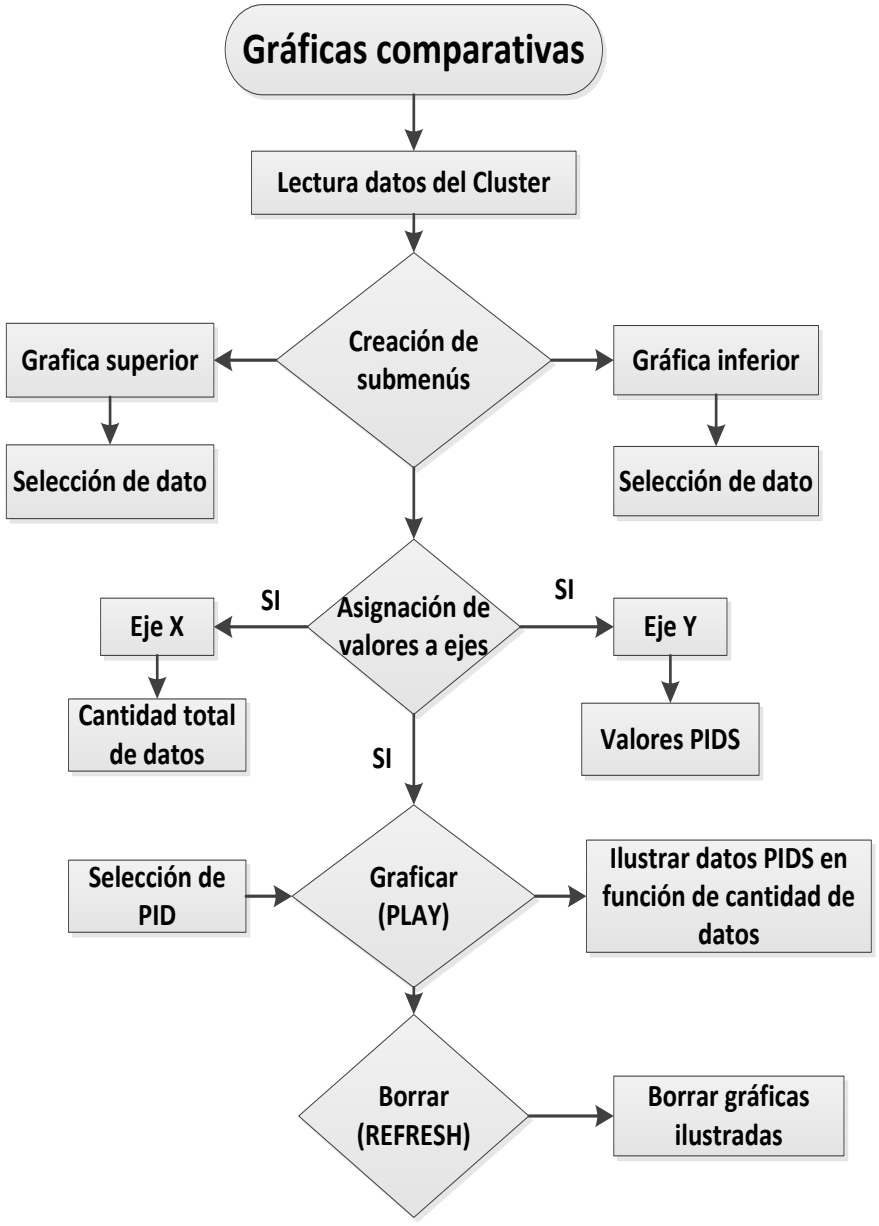


Figura 62. Flujograma para graficar los datos en Labview.

3.3.5 Simulador.

Esta función se lo crea para poder simular todos los datos recolectados por el módulo. El simulador visto en el panel frontal tiene la apariencia de un tablero de instrumentos de un vehículo. Consta de 2 indicadores gauge como tacómetros de velocidad y rpm, 1 indicador de temperatura, 8 indicadores numéricos donde se muestran los valores de (rpm, velocidad, temperatura del motor, latitud, longitud, ángulos x y z) y 8 indicadores led modificados para simular las diferentes marchas y el estado del freno.

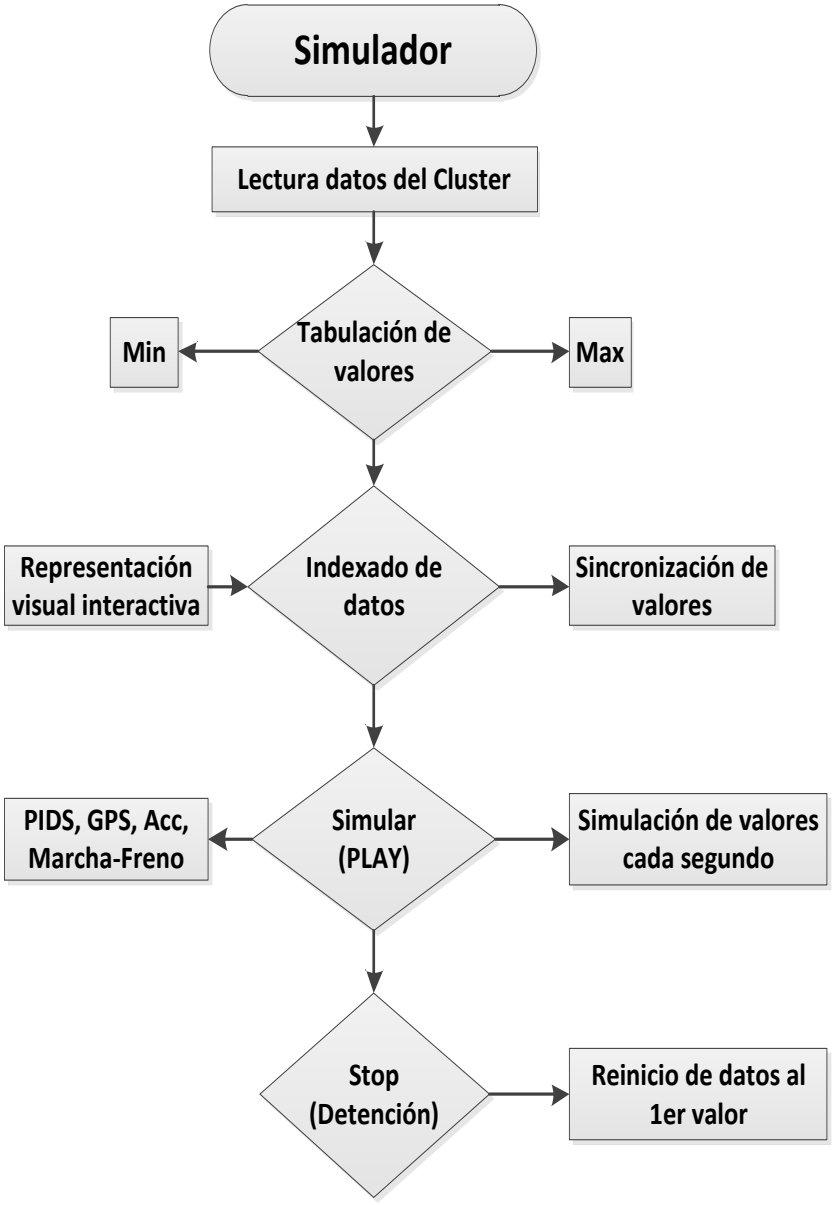


Figura 63. Flujograma de ejecución para el simulador.

También posee 1 indicador 2D para mostrar las fotos captadas por la cámara en modo secuencia y un indicador 3D donde se visualiza en un cubo como varían los ángulos proporcionados por el acelerómetro incorporado en el módulo. Para poder iniciar y detener la ejecución del simulador se debe presionar los botones de play y stop dispuestos en la parte superior del simulador, mientras que en la parte derecha se encuentra una tabla donde se muestran los valores mínimos y máximos como se ve en la (Figura 64).

Figura 64. Apariencia del simulador en el panel

Todos los datos usados en el simulador provienen de los clusters OBII y GPS, para hacer llegar estos datos se crean variables locales que contienen estos datos y así poder conectarlos a su respectivo indicador. Mediante estructuras “for loop” auto-indexadas se asegura que se vayan mostrando dato por dato hasta que se terminen los mismos, esta es una parte fundamental para la simulación.

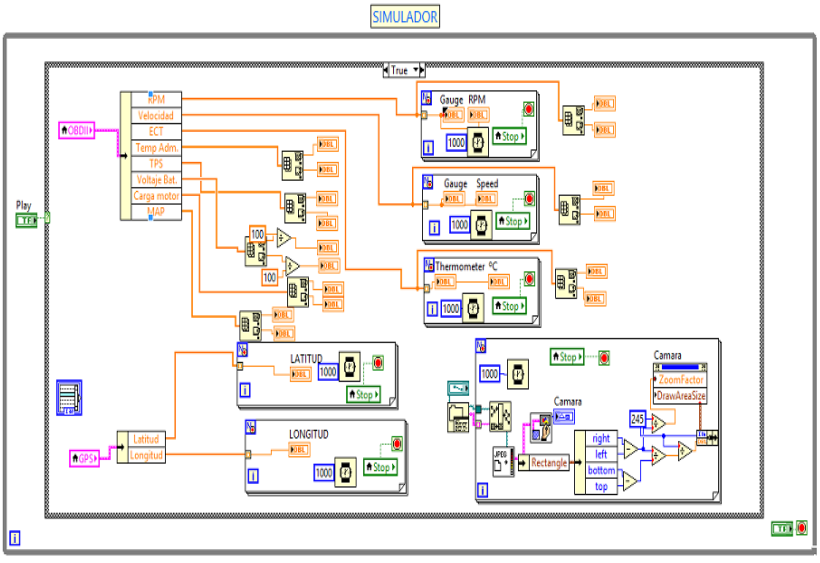


Figura 65. Código principal del simulador.

En las Figuras 66 y 67 se indica los códigos utilizados tanto para el acelerómetro como para el indicador de estado de marchas y freno, los cuales funciona independientes del código principal del simulador sin embargo al presionar el botón de play del simulador los 3 se iniciarán al mismo

tiempo, esto es debido a que están gobernados por un botón de inicio y de stop común para los 3 códigos.

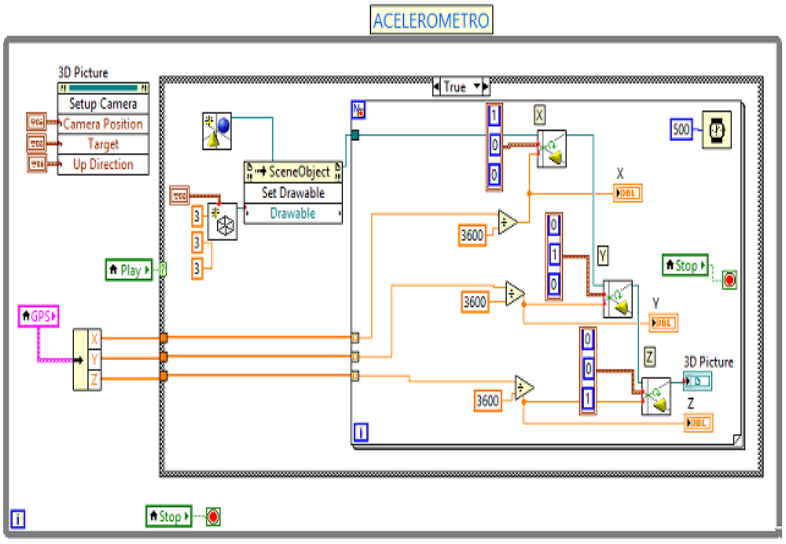


Figura 66. Código del acelerómetro.

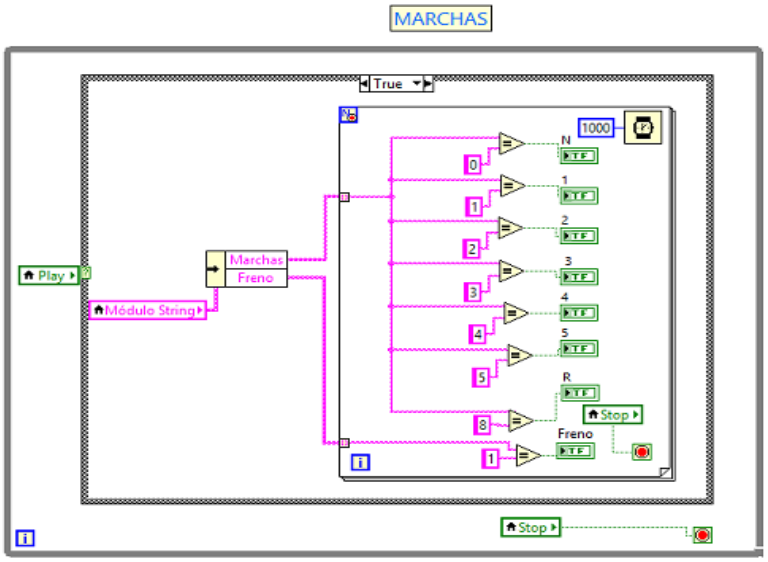


Figura 67. Código del indicador de marchas y freno para el simulador.

3.3.6 Visualizador de fotos.

Esta función dentro del programa permite visualizar las fotos captadas por la cámara OV240 instalada en el módulo. En la pantalla frontal se puede observar una pequeña pantalla donde se visualizan las fotos y 3 botones para interactuar con la interfaz. El botón play carga las fotos

desde la ruta de la carpeta especificada, mientras que los botones anterior y siguiente permite ir cambiando de foto en foto para un análisis detallado de cada uno de ellas.

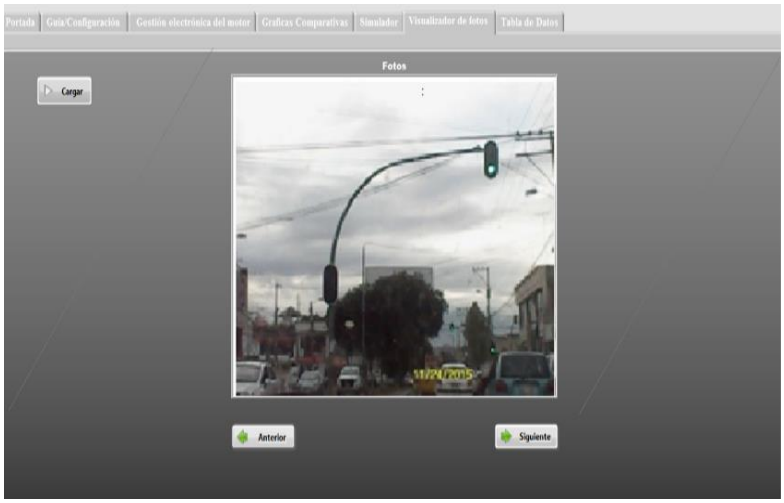


Figura 68. Visualizador de fotos (vista del panel frontal).

En la siguiente figura se puede ver el código hecho en Labview para recrear esta función en el programa. Para lo cual se usa una estructura case y se crea 5 casos posibles. El primer caso es el encargado de leer la ruta de las fotos, el segundo caso determina la función de cada botón, el tercer caso es el encargado de leer y redimensionar las fotos para mostrarlas en la pantalla pequeña del panel frontal, el cuarto caso es un caso vacío que indica que el programa no debe hacer nada si no se cumple alguna condición del segundo caso y el quinto caso solo actúa cuando se produce algún error.

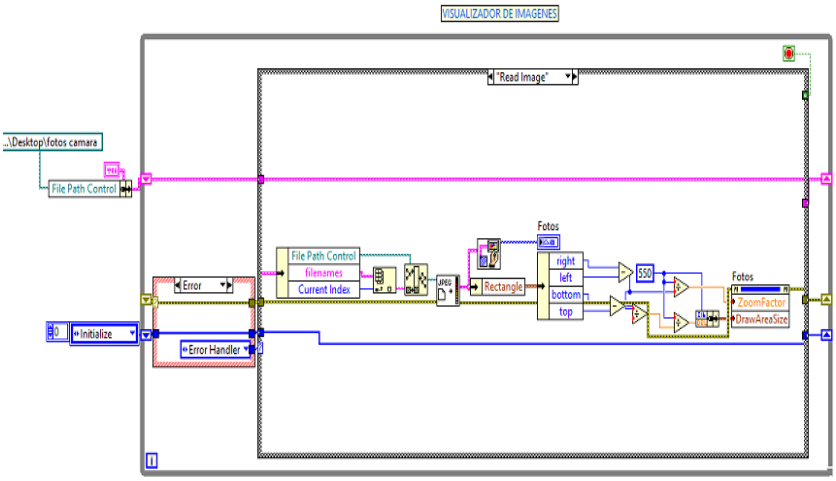


Figura 69. Código en Labview para el visualizador de fotos.

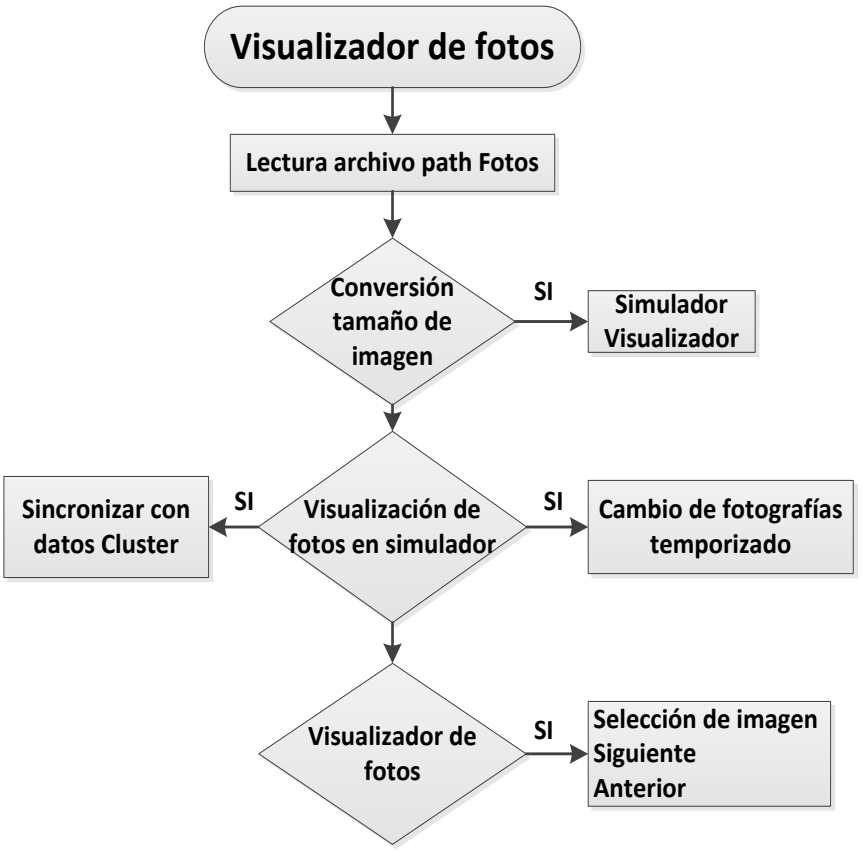


Figura 70. Flujograma para el visualizador de fotos.

3.3.7 Tabla de datos.

Como se menciona en la parte del almacenamiento de datos el archivo de datos utilizado en este programa viene ordenado y separado por comas a manera de codificación. La función de esta tabla dentro de este programa es decodificar estos datos y clasificarlos en cada una de las columnas asignadas con cada uno de los nombres de cada sensor y dato que el módulo puede obtener y almacenarlo.

La siguiente figura muestra la tabla de datos creada en el panel frontal del programa, a cada una de las columnas se les asigno el nombre del dato correspondiente. Posee 4 botones de interacción, el botón Datos permita cargar todos los datos y visualizarlos en la tabla, el botón limpiar vacía la tabla de datos, el botón de reporte permite importar todos los datos de la tabla en Labview a una tabla de Excel para generar un reporte con todos los datos y el botón de borrar elimina el archivo de la tarjeta sd.

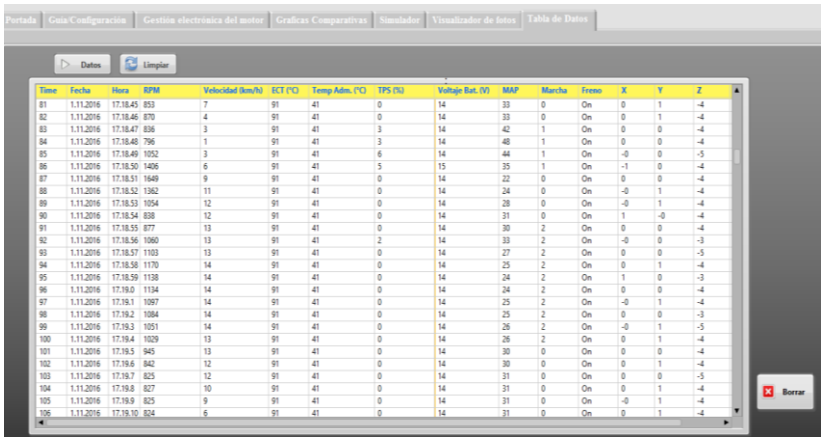


Figura 71. Tabla de datos en Labview (Panel frontal).

El dato "time" de la primera columna corresponde a los datos del contador, el cual sirve como referencia del tiempo de muestra, puesto que muestra que el intervalo de toma de datos se realiza cada 1 segundo. En la figura 72 se puede observar cómo llegan los datos a la tabla, a través de variables locales de los clusters creados en Labview que contienen todos los datos cargados al programa. Los datos llegan a su respectiva columna mediante un "index array", todos llegan de manera directa a la tabla a excepción de los datos de fecha/hora que se les da un formato antes de ingresar a la tabla y los datos que correspondientes a los ejes x, y, z los cuales se deben dividir para 3600, esto debido a que el dato llega en grados/segundo.

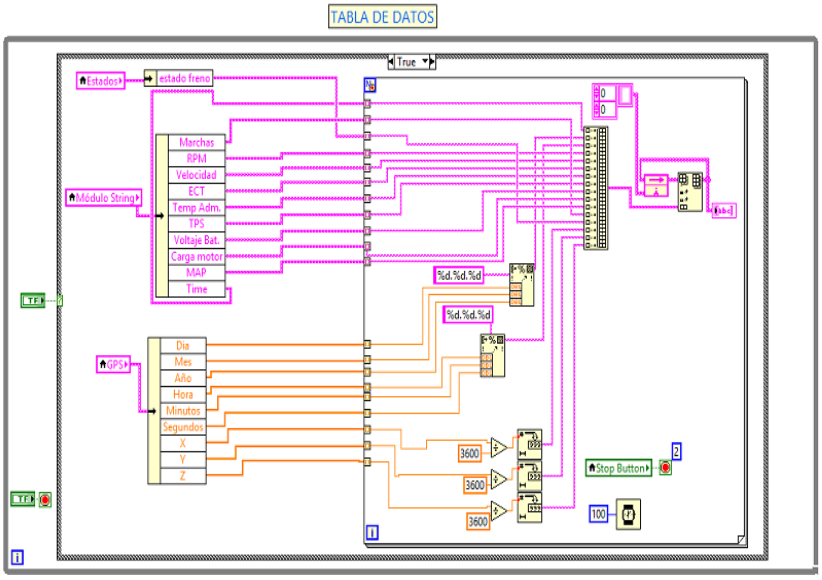


Figura 72. Código de la tabla de datos en Labview.

La (Figura 73) muestra el código empleado para generar el reporte en Excel, esto se puede lograr mediante el uso de las librerías del “Report Generation Tool” de Labview. Este reporte solo se genera al presionar el botón reporte de la pantalla frontal.

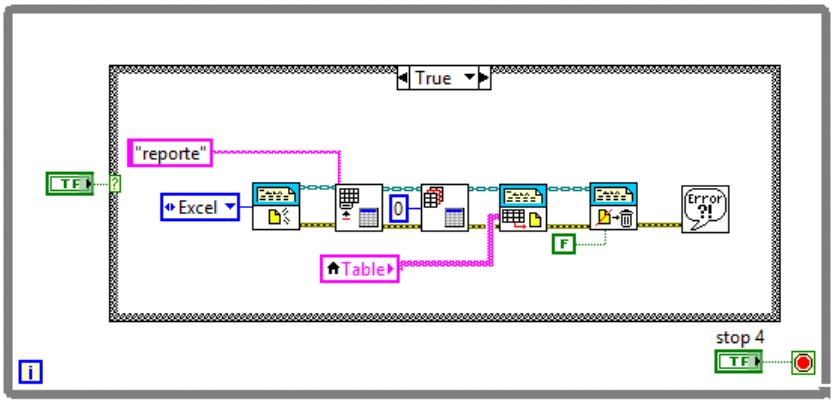


Figura 73. Código en Labview para generar el reporte en Excel.

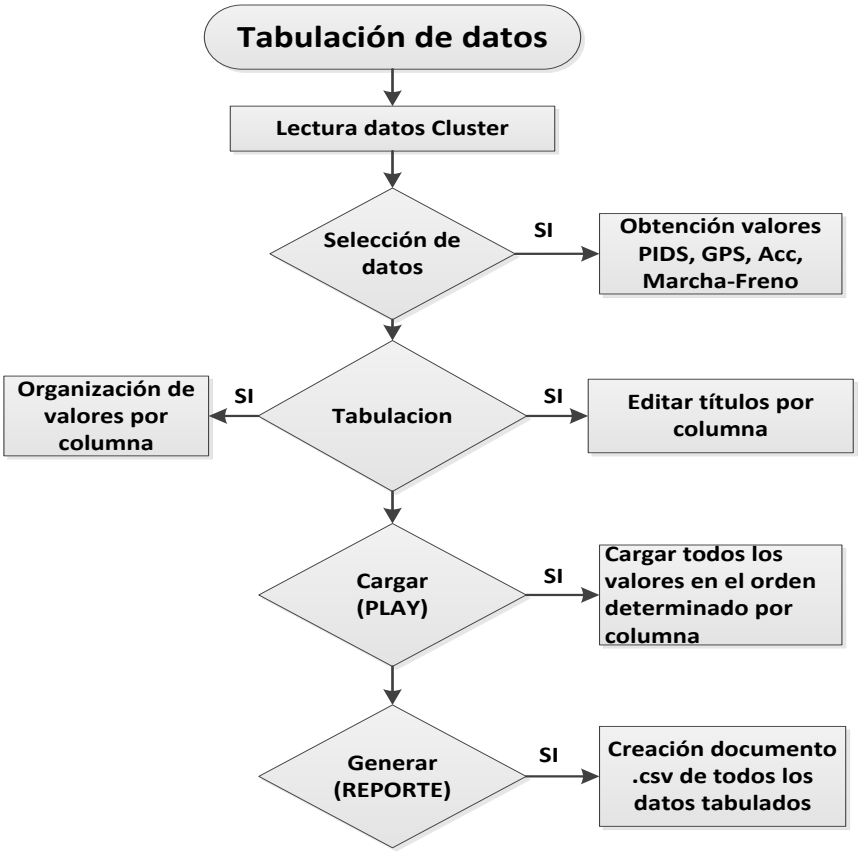


Figura 74. Flujograma para generar la tabla de datos.

3.4 Adaptabilidad a sistemas instalados en taxis.

Si bien este dispositivo está diseñado para recolectar datos del vehículo, cuenta con varios sistemas acoplados a él, como GPS y la cámara a bordo. Partiendo de este punto de vista se puede adaptar este módulo a los sistemas que los taxis ya poseen instalados para obtener un registro de datos más amplio. Los sistemas disponibles en los taxis y que puede integrarse al módulo de almacenamiento son:

- Taxímetro
- Radio de comunicación.

Un taxímetro basa su funcionamiento en el cobro por distancia recorrida, para conocer la distancia requiere de los datos de velocidad y tiempo para calcular dicha distancia. Si se integra este módulo al sistema de taxímetro no se requiere realizar estos cálculos porque este módulo permite obtener el dato de distancia directamente del conector OBDII del vehículo mediante la instrucción (PID_DISTANCE). Únicamente se requiere enviar este dato al taxímetro para que este calcule el valor de cobro.

Para poder integrar este módulo de almacenamiento a la radio de comunicación, se requiere implementar un módulo de grabación de audio (HiLetgo ICSH041A 4 Channel) al pulsador del radio de comunicación para comenzar la grabación de la conversación entre el conductor y la central de servicio y almacenar el audio en una memoria. Para usar estos datos conjuntamente con los datos del módulo, evitando así la invasión de conexiones en el sistema de radio comunicación existente en taxis, similar a las grabaciones de audio de una caja negra.

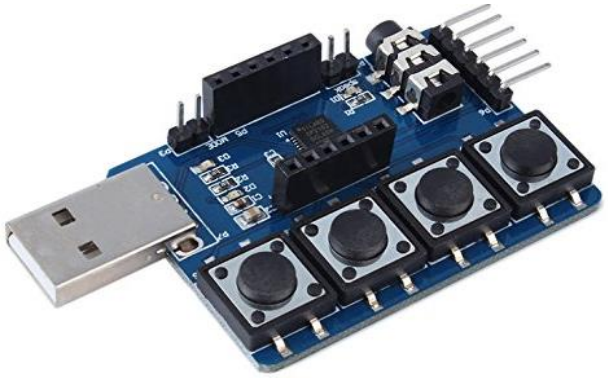


Figura 75. Grabador de audio HiLetgo ICSH041A 4 Channel.

Fuente: (Ebay, 2016)

La adaptabilidad del sistema de cámaras monitoreado por el ECU 911 al módulo es posible, pero requiero del desarrollo de un software y programación de un bucle independiente solo para el procesamiento de video y esto genera un elevado costo de desarrollo y un lapso de tiempo para ejecutar un poco extenso. Por esta razón se desarrolla un sistema independiente de captura de imágenes del entorno frontal del vehículo con un muy bajo costo y una calidad de captura considerablemente buena.

3.5 Mejoras para el módulo de almacenamiento de datos.

Si bien este módulo está diseñado para adquirir y almacenar una trama compuesta de 21 datos y cumple con este objetivo, la posibilidad de ampliar este registro de datos y hacer de este dispositivo un sistema más completo es una buena opción. Entre los sistemas que pueden mejorar la funcionalidad de este módulo tenemos:

- Sistema de cámaras para el interior del vehículo.
- Grabadora de audio.
- Taxímetro.
- Sensores de proximidad (para aparcamiento).
- Indicadores led para advertencia (sobrepasar límites de temperatura, rpm, velocidad).
- Sensor de contacto (para controlar el uso cinturón de seguridad).
- Sistema de bloqueo del vehículo.

CAPITULO IV

4. Pruebas de Funcionamiento y análisis de datos obtenidos.

4.1 Pruebas de funcionamiento de la pantalla.

El módulo de almacenamiento de datos cuenta con una pantalla LCD de 3,2” pulgadas. Esta pantalla se alimenta del voltaje que obtiene del conector OBDII, cuando la pantalla se ha encendido aparece un menú con datos informativos como: Universidad Técnica del Norte, Ingeniería Automotriz, FICA, Módulo de Almacenamiento de datos On-board más un aviso que indica que se está estableciendo la conexión al OBDII del vehículo e indicando cuando el dispositivo está conectado y listo para su uso (Figura 76). Este menú solo será visible cuando se conecte el módulo al vehículo luego de presionar el botón inicio pasa automáticamente a la siguiente página.

Figura 76. Portada del Módulo de almacenamiento de datos.

La segunda pantalla solo es visible cuando se presione el botón de inicio su funcionamiento, esta permanece activa mientras el módulo este grabando, aquí se puede observar algunos datos como hora y la cantidad de datos que ha guardado con una imagen de fondo de un taxi (Figura 77).

Figura 77. Aspecto de la segunda pantalla.

4.2 Prueba de comunicación del OBDII.

Efectuada la conexión del conector OBDII a la ranura I2C de la pantalla LCD, se comprueba el envío y recepción de datos requeridos para el almacenamiento de los diferentes parámetros en el módulo, ya que mediante este conector OBDII se adquiere datos de velocidad, revoluciones, temperatura del motor, apertura mariposa aceleración, voltaje de batería que son los PIDs genéricos habilitados. Además, los datos de aceleraciones en los ejes x, y, z provenientes de acelerómetro integrado en el conector. Estos valores que son muy importantes para analizar el estado del vehículo luego de un viaje. Los valores se visualizan mediante comunicación serial entre el módulo OBDII y la placa Arduino Mega 2560. En la (Figura 78) se puede observar los datos recibidos del conector OBDII.

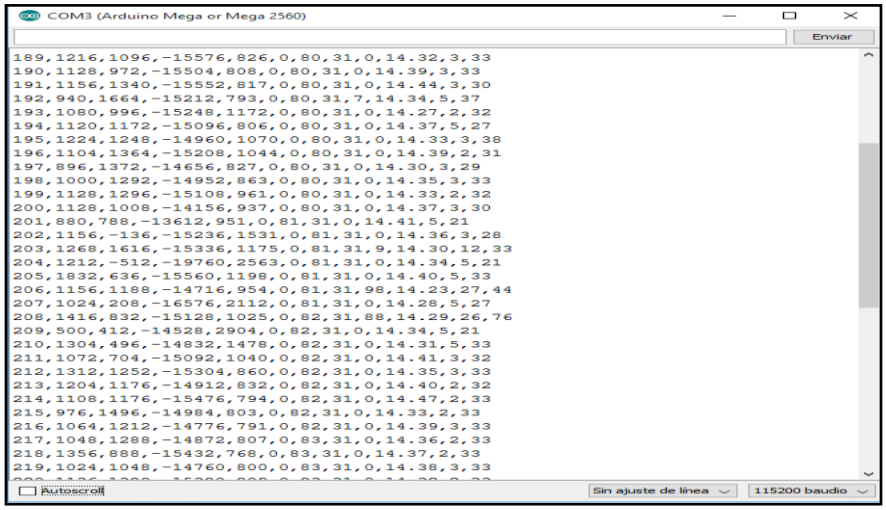


Figura 78. Datos OBDII a través de monitor serial

4.3 Prueba de comunicación del sistema de estado de marcha y freno.

Efectuadas las conexiones de cada sensor fin de carrera usando pines digitales en la placa Arduino Mega, se comprueba la recepción de datos a través de lectura digital de cada componente. Con este sistema se reconoce la posición de la palanca de cambios y pulsación del pedal de freno. Los valores son visualizados mediante comunicación serial entre la placa Arduino Mega 2560 y el sistema marcha-freno. Para cada marcha se le asigna el valor correspondiente a la marcha, mientras que para el estado del freno solo hay 2 valores posibles 1 cuando esta accionado y 0 cuando no lo está.

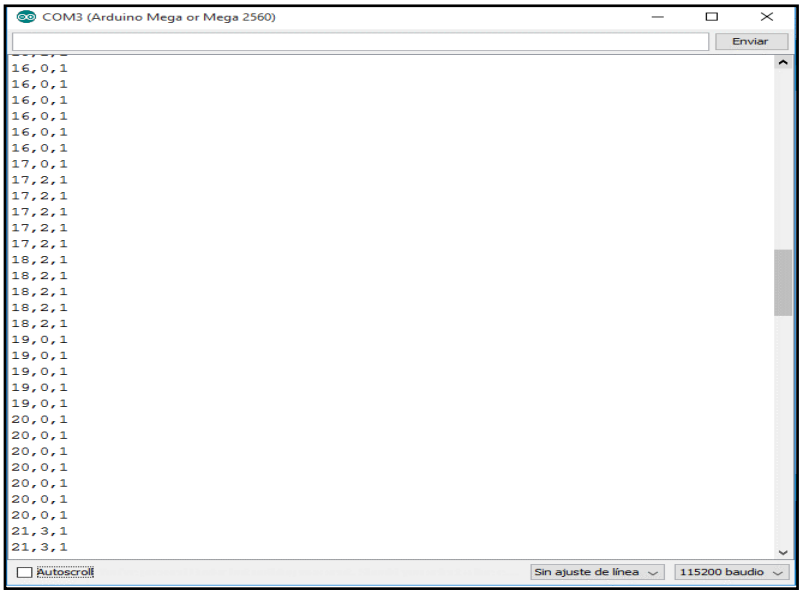


Figura 79. Datos sistema marcha-freno a través de monitor serial.

4.4. Prueba de comunicación del módulo GPS.

Efectuada la conexión del módulo GPS a la ranura Serial2 de la pantalla LCD, se comprueba el envío y recepción de datos requeridos puesto que, mediante GPS se adquiere valores de fecha, hora y ubicación, valores importantes para organizar todos los datos obtenidos por el OBDII. Los valores se visualizan mediante comunicación serial entre el módulo GPS y la placa Arduino Mega 2560.

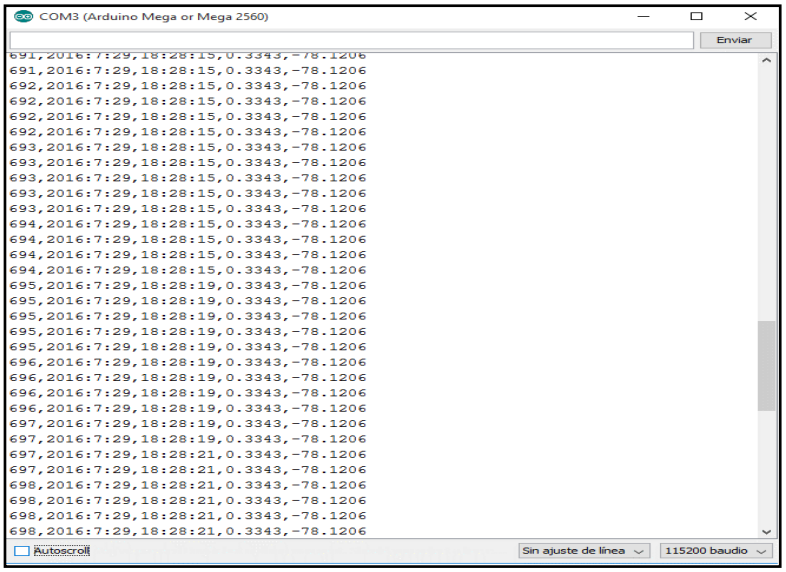
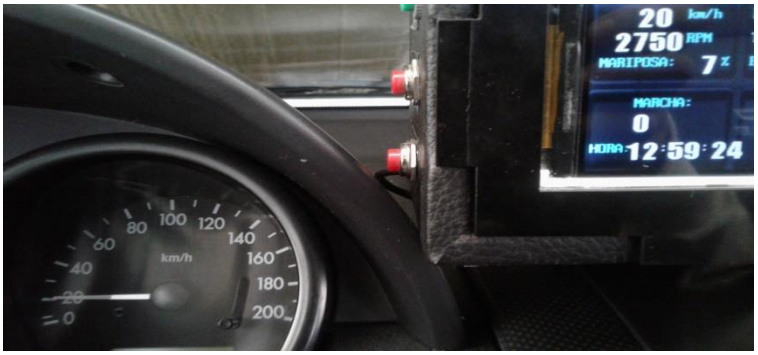


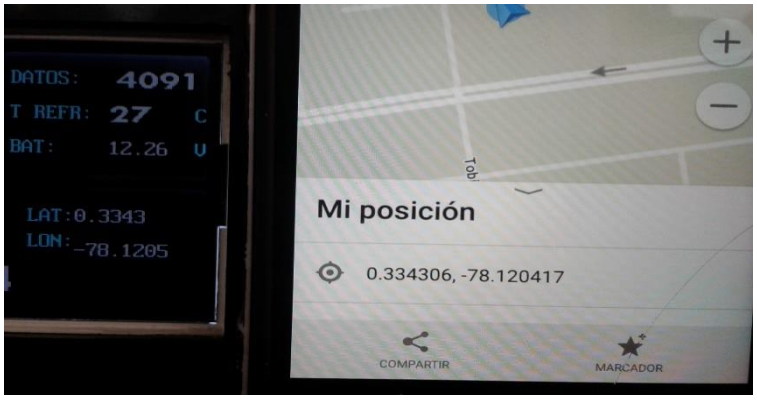
Figura 80. Datos GPS a través de monitor serial

4.5 Prueba de comparación de valores.

Efectuadas las conexiones de todos los implementos que conforman el módulo de almacenamiento de datos, se comprueba la recepción de datos mediante la visualización de valores adquiridos de cada implemento en la pantalla LCD, y se verifica en el tablero de instrumentos del vehículo si los datos ilustrados en la pantalla poseen variación con los valores que brinda el vehículo al conductor. Como se puede observar en la siguiente tabla comparativa.

Tabla 17. Comparación de valores del módulo de almacenamiento.

COMPARACIÓN DE VALORES MÓDULO DE ALMACENAMIENTO	
VELOCIDAD	**DIFERENCIA [km/h]**
	Velocidad de 21 [km/h] en el vehículo. Velocidad de 20 [km/h] en el módulo. Diferencia de 1 [km/h].

REVOLUCIONES	DIFERENCIA [min^{-1}]
	Vehículo 3400 [min^{-1}]. Módulo 3459 [min^{-1}]. No existe diferencia significativa entre los 2 valores.
TEMPERATURA REFRIGERANTE	**DIFERENCIA [º C]**
	Temperatura 90 [ºC] aproximadamente en el vehículo ya que no cuenta con valor digital. Temperatura 93 [ºC] en el módulo. No existe diferencia significativa entre los 2 valores.
POSICIÓN LAT - LONG	**DIFERENCIA [º]**
	Posición del dispositivo GPS Lat.: 0.3343, Lon: -78.1205. Posición Google Earth Lat: 0.334306, Lon: -78.120417. No existe diferencia entre los 2 valores.

4.6. Prueba de funcionamiento de los botones de operación.

El módulo de almacenamiento de datos cuenta con 3 botones fiscos de operación ubicados en lado izquierdo del módulo, 1 de color verde y 2 de color rojo. Estos botones permiten seleccionar los 3 tipos de estado en los que opera el módulo de almacenamiento de datos. Los tres estados de operación se pueden observar en la (Tabla 18).

Tabla 18. Botones de operación

BOTÓN	FUNCIÓN
1er Botón (VERDE)	Muestra en la pantalla el texto "GRABACIÓN ON", abre el documento "txt" y da comienzo a la grabación de datos.
2do Botón (ROJO)	Muestra en la pantalla el texto "GRABACIÓN OFF", cierra el documento con todos los valores obtenidos.
3er Botón (ROJO)	Muestra en la pantalla el texto "REINICIA" y devuelve todos los valores del contador a su estado inicial 0 para una nueva grabación.

El primer botón verde es el encargado de iniciar la grabación de datos, cuando este botón es accionado pasa primero a una pantalla con el mensaje "SINCRONIZANDO" esta pantalla tiene como objetivo retrasar la captura de datos, mientras la cámara se enciende y cambia de modo video a modo fotografía, esta pantalla esta activa durante 9 segundos que es lo que le toma a la cámara estar lista para la grabación. De esta manera se asegura que la tanto la cámara como el módulo empiecen a grabar al mismo.

Figura 81. Mensaje de sincronizando en la pantalla.

Después de esos 9 segundos pasa automáticamente a la pantalla de inicio de grabación donde se muestra el mensaje “GRABANDO” en la segunda pantalla. Al accionar este botón la pantalla también se activa e inicia la captura de fotos mediante la cámara integrada al módulo.

Figura 82. Botón de inicio y test de mensaje en la pantalla.

Figura 83. Flujograma de ejecución del botón de inicio.

El segundo botón rojo es el encargado de detener la grabación de datos. Este botón detiene el contador de los datos hasta ese instante para posteriormente volver a contar desde ese valor cuando se inicie de nuevo la grabación. Cuando es accionado muestra un mensaje el siguiente mensaje en la pantalla “FIN DE GRABACION”.

Figura 84. Botón de detención y test de mensaje en la pantalla.

Figura 85. Flujograma de ejecución del botón Fin de grabación.

El tercer botón de color rojo tiene como función resetear el valor del contador, usado como número de registro. Cuando se inicia una grabación nueva el contador empieza en valor 0 y se irá incrementando segundo a segundo que es el tiempo de muestreo entre toma y toma de datos. Cuando se detenga la grabación el contador se detiene, pero no se borra el valor del contador, si no que se almacena el ultimo valor, para cuando se vuelva a iniciar la grabación continúe con el valor del último registro, solo cuando se accione este botón se regresa a 0 este valor del contador. Este botón muestra al ser accionado muestra el siguiente mensaje “REINICIANDO”.

Figura 86. Botón de reinicio y test de mensaje en la pantalla.

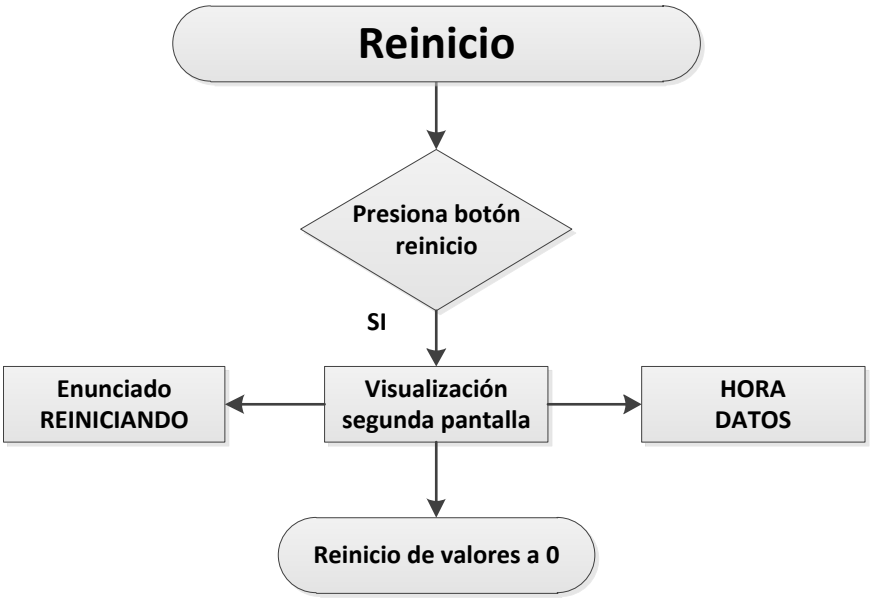


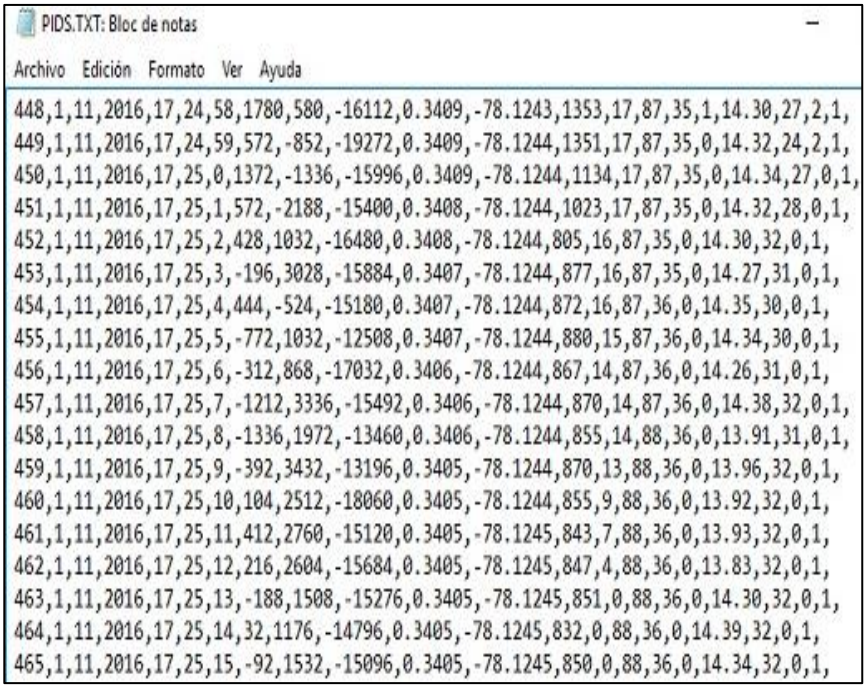
Figura 87. Flujograma de ejecución del botón de reinicio.

4.7 Prueba de funcionamiento módulo de almacenamiento de datos

El módulo de almacenamiento de datos basa su funcionamiento en la adquisición de datos en tiempo real y almacenamiento de los mismos más los elementos acoplados a éste. En la pantalla se muestran un fondo con una imagen de un taxi, y en la parte inferior la hora actual que entrega el receptor GPS y la cantidad de datos que va obteniendo esto es útil saber si se quiere reiniciar el historial de datos a 0 para iniciar una nueva prueba

Figura 88. Funcionamiento módulo de almacenamiento

Este módulo tiene la capacidad de almacenar hasta 2dias de grabación continua, esto es posible al usar la capacidad de la memoria EEprom de la placa arduino mega. Esta memoria permite mantener almacenado un valor de registro para continuar desde ese punto si el usuario detuvo la grabación y vuelve a activar la misma.


PIDS.TXT: Bloc de notas

Archivo Edición Formato Ver Ayuda

```
448,1,11,2016,17,24,58,1780,580,-16112,0.3409,-78.1243,1353,17,87,35,1,14.30,27,2,1,
449,1,11,2016,17,24,59,572,-852,-19272,0.3409,-78.1244,1351,17,87,35,0,14.32,24,2,1,
450,1,11,2016,17,25,0,1372,-1336,-15996,0.3409,-78.1244,1134,17,87,35,0,14.34,27,0,1,
451,1,11,2016,17,25,1,572,-2188,-15400,0.3408,-78.1244,1023,17,87,35,0,14.32,28,0,1,
452,1,11,2016,17,25,2,428,1032,-16480,0.3408,-78.1244,805,16,87,35,0,14.30,32,0,1,
453,1,11,2016,17,25,3,-196,3028,-15884,0.3407,-78.1244,877,16,87,35,0,14.27,31,0,1,
454,1,11,2016,17,25,4,444,-524,-15180,0.3407,-78.1244,872,16,87,36,0,14.35,30,0,1,
455,1,11,2016,17,25,5,-772,1032,-12508,0.3407,-78.1244,880,15,87,36,0,14.34,30,0,1,
456,1,11,2016,17,25,6,-312,868,-17032,0.3406,-78.1244,867,14,87,36,0,14.26,31,0,1,
457,1,11,2016,17,25,7,-1212,3336,-15492,0.3406,-78.1244,870,14,87,36,0,14.38,32,0,1,
458,1,11,2016,17,25,8,-1336,1972,-13460,0.3406,-78.1244,855,14,88,36,0,13.91,31,0,1,
459,1,11,2016,17,25,9,-392,3432,-13196,0.3405,-78.1244,870,13,88,36,0,13.96,32,0,1,
460,1,11,2016,17,25,10,104,2512,-18060,0.3405,-78.1244,855,9,88,36,0,13.92,32,0,1,
461,1,11,2016,17,25,11,412,2760,-15120,0.3405,-78.1245,843,7,88,36,0,13.93,32,0,1,
462,1,11,2016,17,25,12,216,2604,-15684,0.3405,-78.1245,847,4,88,36,0,13.83,32,0,1,
463,1,11,2016,17,25,13,-188,1508,-15276,0.3405,-78.1245,851,0,88,36,0,14.30,32,0,1,
464,1,11,2016,17,25,14,32,1176,-14796,0.3405,-78.1245,832,0,88,36,0,14.39,32,0,1,
465,1,11,2016,17,25,15,-92,1532,-15096,0.3405,-78.1245,850,0,88,36,0,14.34,32,0,1,
```

Figura 89. Archivo de texto generado por el módulo de

4.8 Prueba de funcionamiento de la cámara.

La cámara que trabaja conjuntamente con el módulo de almacenamiento de datos, inicia la captura de imágenes cuando recibe la orden de la placa principal al iniciar la grabación, un disparo de un relé cambia de modo video que es su modo por defecto al de fotos. Las fotos que captura las guarda en formato .jpg con calidad de 2megapixeles y las almacena en una sd independiente de la de los datos, con un lapso de tiempo entre captura y captura de 1.5 segundos, para facilitar el procesamiento de imagen del controlador que posee el sistema y brindar una imagen clara del panorama que se encuentre frente al vehículo, la cual sirve de evidencia para verificación de rutas, acontecimientos que se puedan suscitar en un viaje.

Figura 90. Calidad de imagen (2mp).

El módulo de almacenamiento de datos posee el pin digital 18 del arduino Mega como salida, para enviar el pulso de encendido cuando se presiona el botón verde de grabación de datos para una correcta sincronización de encendido para el almacenamiento de todos los datos, además el sistema fotográfico se encuentra configurado con el pin digital 3 del arduino Nano como entrada para leer el pulso y comenzar con el proceso de funcionamiento indicado a continuación.

Figura 91. Etapas de funcionamiento de la cámara.

4.9 Uso de datos y dinámica del equipo para futuras aplicaciones.

El módulo de almacenamiento de datos provee información que permite monitorear cuál fue el comportamiento de funcionamiento del motor, marcha, estado de freno, ubicación del vehículo, aceleraciones en ejes X, Y, Z. Adquirida toda la información se almacena en un documento .txt, se separa los valores y organiza mediante columnas para una mejor interpretación valor-unidad como se puede observar en la tabla. Estos valores se pueden utilizar para evaluar ciertas condiciones como: rendimiento del vehículo, condiciones de manejo.

Tabla 19. Tabulación de datos almacenados y sus unidades

Datos	Día	Mes	Año	Hora	Minuto	Segundo	Acel X [m/s^2]	Acel Y [m/s^2]	Acel Z [m/s^2]	Latitud	Longitud
1	23	1	2017	10	36	36	880	1588	-14780	0.3344	-781.202
2	23	1	2017	10	36	37	820	1364	-14960	0.3344	-781.202
3	23	1	2017	10	36	38	804	1304	-14936	0.3344	-781.202
4	23	1	2017	10	36	39	1008	1588	-15048	0.3344	-781.202
5	23	1	2017	10	36	40	812	1268	-15064	0.3344	-781.202
6	23	1	2017	10	36	41	884	1208	-15072	0.3344	-781.202

Revoluciones RPM	Velocidad km/h	Temperatura Refrigerante °C	Temperatura Admisión °C	Apertura mariposa %	Voltaje Batería V	Preson Admisión kg/cm^2	Marcha	Freno
825	0	76	35	0	13.50	34	1	1
853	0	76	35	0	13.50	33	1	1
838	0	76	35	0	13.50	33	1	1
854	0	76	35	0	13.50	33	1	1
826	0	76	35	0	13.50	34	1	1

4.9.1 Desempeño del vehículo

Los datos como velocidad, revoluciones, posición de marcha, activación freno, temperatura del refrigerante del motor y tiempo se puede utilizar para generar informes estadísticos del comportamiento del vehículo, así como el desempeño realizando pruebas como:

- Tiempo de aceleración entre rangos 0-100 [km/h] 0-160 [km/h]
- Tiempo de respuesta frenado del vehículo en rangos 100-0 [km/h] 160-0 [km/h]
- Comportamiento dinámico del vehículo
- Registro de temperatura a altas revoluciones
- Respuesta de aceleración [apertura mariposa-aumento de revoluciones]
- Rangos de velocidad-rpm por cada marcha [km/h-rpm / marcha]

Se realiza estos tipos de pruebas para conocer de manera fiable el comportamiento del vehículo a la altura en la cual se desempeña, y así se compara con la información del vehículo proporcionada por la casa comercial ya que esta información se encuentra determinada con el vehículo desempeñándose a nivel del mar.

4.9.2 Hábitos de conducción

Para realizar una guía la cual contenga puntos a considerar para llevar un buen hábito de manejo, se toma los datos del módulo de almacenamiento como velocidad, revoluciones, posición de marcha, estado de freno, hora y fecha. Con estos datos se realiza una tabla de comparación en la cual se puede observar fecha-hora y los datos obtenidos durante ese transcurso de tiempo de la prueba.

En la tabla generada se determina los rangos de velocidad que alcanzo el conductor, los rangos de revoluciones a las cuales sometió al vehículo, la marcha posicionada y la cantidad de veces que fue pulsado el freno. Con la tabulación de estos valores determinamos excesos de velocidad, aceleraciones bruscas, sobreesfuerzos de revoluciones al motor, posicionamiento de marcha fuera de revoluciones optimas de desempeño y abuso de pulsación de freno ya que todos estos parámetros generan malos hábitos de manejo y un deterioro temprano de los sistemas del vehículo.

Tabla 20. Hábitos de conducción

Malos hábitos de conducción	
Parámetro	**Hábito**
Velocidad	Aceleraciones bruscas
Revoluciones	Alcanzar alto régimen de revoluciones
Marcha	Mantener una misma marcha en altas revoluciones
Freno	Abuso de pulsación

4.9.3 Ciclos de conducción.

Otra aplicación que se le puede dar al módulo de almacenamiento es usar los datos obtenidos para definir ciclos de conducción. Un ciclo de conducción permite conocer el comportamiento de un vehículo en una región determinada. Basada en datos estadísticos de la velocidad en función del tiempo.

Tramo Urbano

En un sector urbano la velocidad máxima que puede alcanzar es 40km/h. Durante la prueba la velocidad máxima alcanzada según los datos es de 36km/h y una velocidad media de 18km/h.

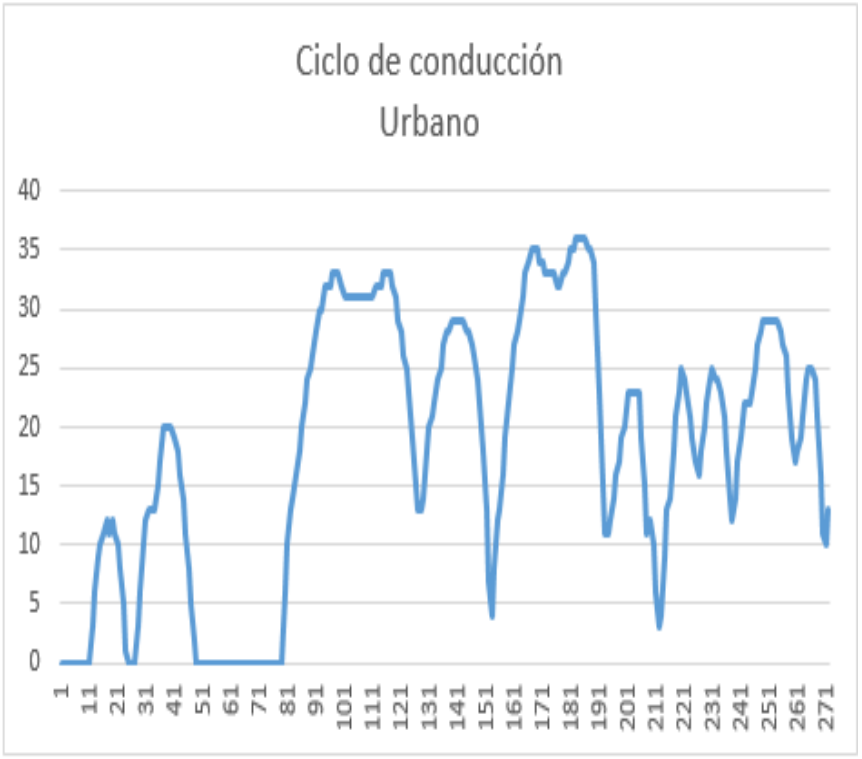


Figura 92. Modelo de ciclo de conducción urbano.

Tramo Interurbano

En un tramo interurbano la velocidad 120 km/h. Durante la prueba la velocidad máxima según los datos es de 85km/h y una velocidad promedio de 42.3 km/h.

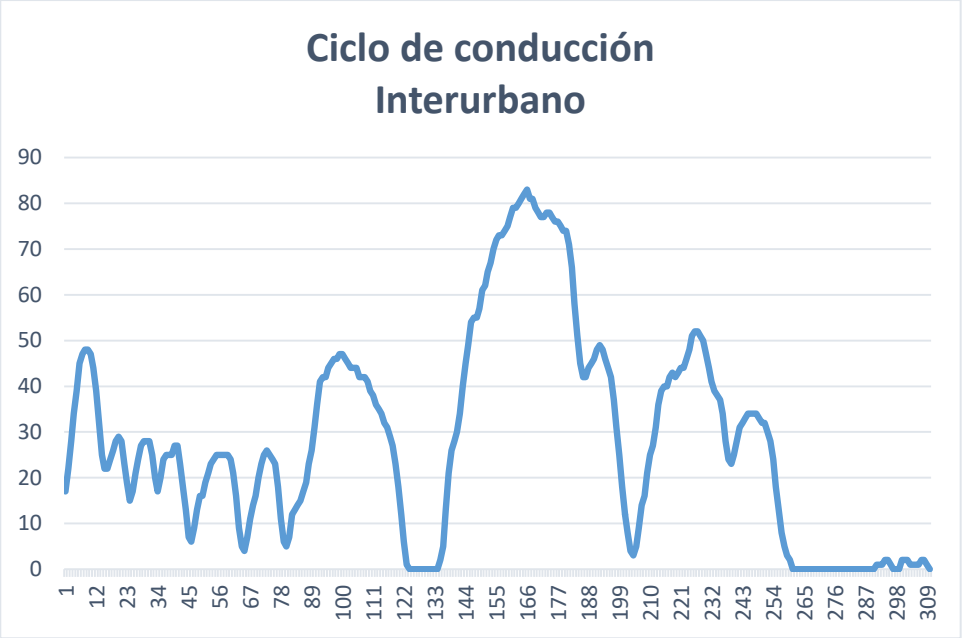


Figura 93. Modelo de ciclo de conducción interurbano.

De esta manera se pueden construir varios modelos de conducción con diferentes vehículos y compararlos entre sí para obtener un perfil de conducción único para vehículos que se desplazan en la misma zona, estos perfiles son únicamente aplicables a dicha zona porque varían según la tipografía. Estos ciclos de conducción pueden servir para calcular emisiones generadas, consumo de combustible, etc.

4.10 Guía de Uso Módulo de Almacenamiento de datos On-board para taxis.

Figura 94. Módulo de almacenamiento de datos on-board.

4.10.1 Introducción.

El módulo de almacenamiento de datos on-board para taxis se crea para poder recolectar información y almacenar datos de viaje de cualquier vehículo que disponga de un conector OBDII y que el protocolo de comunicación sea compatible con el conector. Con el objetivo de entregar al usuario una herramienta que permita realizar un análisis detallado de las condiciones de funcionamiento a las que se expone el vehículo durante un viaje. Puede servir de base para determinar las causas del deterioro o daños en el motor por condiciones de manejo, como también condiciones que puedan influir en un accidente de tránsito por un inadecuado manejo del conductor durante el viaje.

4.10.2 Objetivo del manual.

El presente manual de uso tiene como objetivo indicar al usuario las instrucciones de manejo del módulo de almacenamiento de datos on-board, proporcionándole la información necesaria que pueda despejar todas las dudas que puedan existir. Comprende:

- Modo de instalación y conexión al vehículo.
- Modo de operación del módulo.
- Almacenamiento de datos del módulo en una tarjeta micro SD
- Importado de datos a la interfaz de Labview
- Generación de reporte en Excel.

4.10.3 Especificaciones técnicas.

Para un correcto funcionamiento del módulo de almacenamiento de datos on-board para taxis es necesario cumplir con las siguientes condiciones:

- El módulo está diseñado para vehículos livianos que dispongan de un conector OBDII con protocolos de comunicación compatible con el conector.
- El voltaje de funcionamiento del módulo de almacenamiento es de 5V, en ningún caso debe superarse este valor o proporcionarle un voltaje inferior porque podría causar un funcionamiento deficiente y generar daños en los componentes del módulo.
- Verificar mediante la visualización en la pantalla del módulo de almacenamiento la cantidad de datos recopilados en cada prueba, así el conductor puede estar seguro de los datos adquiridos y tomar la decisión de encerar estos valores para un nuevo viaje.

- Retirar las memorias tanto del módulo de almacenamiento de datos como del sistema fotográfico.
- Tener instalado los programas necesarios para el análisis y compilación de códigos (Arduino y Labview).

Nota: Este módulo puede ser adaptado a cualquier vehículo con conector OBDII con protocolo de comunicación compatible, realizando algunas modificaciones en el sistema de adquisición de posición de marcha y activación de pedal de freno.

4.10.4 Modo de instalación y conexión en el vehículo.

A continuación, se detalla el proceso de instalación del módulo de almacenamiento de datos y sistema de marcha - freno:

- Verifique la ubicación del conector de diagnóstico del vehículo OBD, "la posición de este conector varía de vehículo en vehículo".

  Nota: Si no encuentra el conector diríjase al manual de propietario para encontrar la información necesaria de la ubicación del conector en el vehículo.
- Conecte el cable OBDII en el socket de diagnóstico del vehículo y verifique si el led incorporado en el cable parpadea para conocer que se encuentra bien conectado. Seguido conecte el cable USB al módulo de almacenamiento en la entrada hembra ubicada en la parte inferior derecha.
- Instale el sistema de posición de marcha y activación de pedal de freno en la palanca de cambios del vehículo. En este sistema contamos con 5 sensores de fin de carrera, los cuales nos proporciona 6 cables, uno común- masa de todos los sensores y los 5 restantes nos indica la activación de cada marcha.
- Conecte los 6 cables que provienen del sistema de marcha – freno en la parte inferior izquierda del módulo, allí se encuentra un socket que se conecta directamente con pines digitales de la placa Arduino Mega.

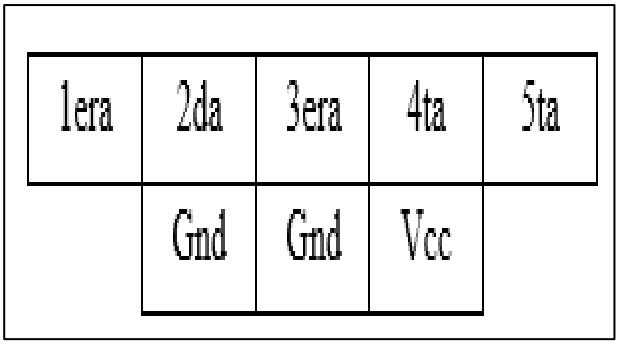


Figura 95. Socket parte inferior izquierda

- Ubique la antena del módulo GPS en el lugar más estratégico del vehículo que usted considere. Tomar en cuenta que debe ser un lugar donde permita a la antena obtener una buena recepción de comunicación con los satélites.
- Conecte los 2 cables de alimentación de corriente del sistema de captura de fotografías en la parte inferior izquierda del módulo, allí se encuentra un socket que se conecta directamente con pines del Arduino Nano.

4.10.5 Modo de operación del módulo.

A continuación, se detalla los pasos para poner en funcionamiento el módulo de almacenamiento de datos:

- Inserte las memorias micro SD en las ranuras correspondientes de la cámara y la pantalla, la primera memoria micro SD almacenara todos los datos de funcionamiento, mientras la segunda SD está destinada únicamente para guardar las fotografías.
- Realice la conexión del cable OBDII al socket de diagnóstico del vehículo y el cable a la ranura USB del módulo, enciéndalo mediante la pulsación de un conmutador ubicado en la parte izquierda del tablero de instrumentos y mostrará un mensaje en la pantalla, mientras el sistema reconoce el protocolo y verifica la comunicación con el OBD.

Figura 96. Pantalla de inicio.

- Una vez que la pantalla nos muestra el enunciado OBD LISTO, se procede a pulsar el botón de inicio de grabación, este genera un pulso de encendido al sistema de captura fotográfico además nos mostrara una nueva pantalla en la cual nos muestra un enunciado que dice SINCRONIZANDO esta pantalla permanece activa durante 9 segundos mientras la cámara se enciende y lista para iniciar las capturas.

Figura 97. Mensaje de sincronizando en la pantalla.

- Luego de esos 9 segundos cambiara el mensaje por el de GRABANDO en ese instante tanto la cámara como el módulo comienzan a guardar datos en sus respectivas micro sd.

- Para detener la grabación basta con presionar el segundo botón, lo cual detendrá tanto la captura de datos, crea el documento PIDS.txt y se genera un pulso para apagar la cámara fotográfica, así mismo se le indicará un mensaje en la pantalla “Fin de grabación” confirmando que se detuvo toda grabación. En este modo el valor último del contador se almacena hasta dar inicio a una nueva grabación desde este mismo valor manteniendo la secuencia de registro de datos.
- Caso contrario si se desea realizar una nueva sesión de grabación de datos, se debe presionar el tercer botón contando desde la parte superior. Este botón reiniciara los datos a 0 y mostrara en la pantalla un enunciado REINICIANDO.

Figura 98. Mensajes en la pantalla al presionar los botones de acción.

- Retirar las tarjetas micro SD tanto del módulo de almacenamiento de datos como de la cámara fotográfica.

4.10.6 Importación de datos a la interfaz de Labview.

A continuación, se detallan los pasos para ingresar los datos recolectados a la interfaz de Labview:

- Retire las 2 tarjetas micro SD tanto del módulo de almacenamiento de datos como de la cámara.
- Con la ayuda de un adaptador micro SD/USB pase los archivos de datos a su PC o laptop la cual debe tener instalado los programas de Arduino/Labview.
- Ejecute el programa “Interfaz 8.7” y espere a que cargue el programa y sus complementos.

- Una vez dentro del programa se puede ver un menú con 7 ventas, cuya primera ventana corresponde a la portada de la interfaz, la cual contiene los datos informativos acerca del módulo.
- En la segunda ventana, se encuentra en la parte izquierda una mini guía de uso de la interfaz.

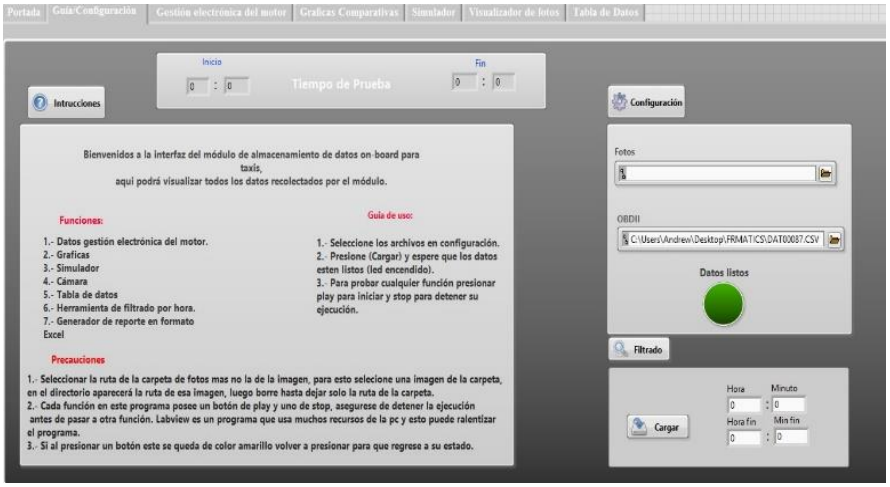


Figura 99. Pestaña de guía y configuración de la interfaz.

- En la parte derecha de la misma ventana se encuentra el apartado de configuración. En la sección de “fotos” seleccione la ruta de la carpeta donde se encuentran las fotos que desea visualizar en la interfaz. Para seleccionar simplemente dar click en una de las fotos, de esta manera se creará automáticamente la ruta.

Figura 100. Path de selección de la carpeta de fotos.

- En la sección OBDII seleccione la ruta del archivo de datos que desea ingresar en la interfaz. Cuando las rutas estén seleccionadas el indicador led “Datos listos” se enciende.

Figura 101. Path de selección del archivo txt.

- Bajo este apartado de configuración se encuentra la herramienta de filtrado por hora que permite al usuario indicar tanto la hora de inicio y la hora fin en la que desea analizar los datos.

Figura 102. Herramienta de filtrado por hora.

- En la tercera ventana denominada “Gestión del motor” se puede visualizar y comparar 2 valores en una misma gráfica, los valores disponibles son: Velocidad, RPM, TPS, MAP, ECT, Temp. Admisión, Voltaje de batería. Estos se pueden seleccionar en un submenú dispuesto en la parte izquierda.

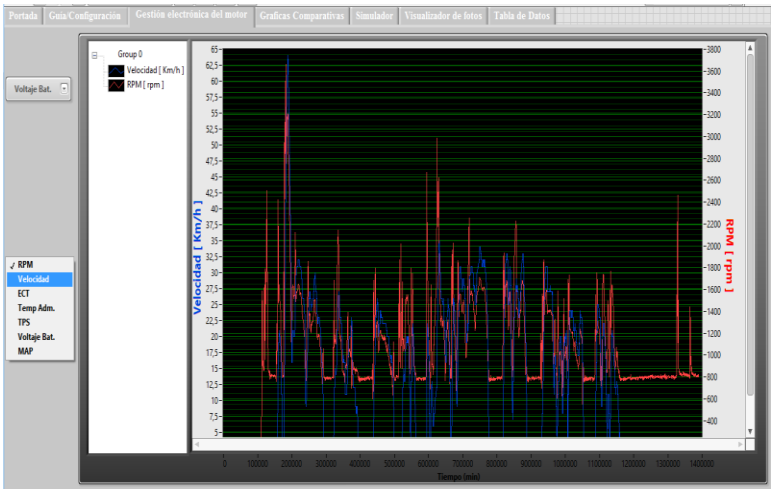


Figura 103. Gráfica mixta (ventana Gestión del motor).

- La cuarta ventana denominada “Graficas comparativas” permite realizar una gráfica individual de cada valor que se seleccione del submenú de la parte izquierda. Una vez seleccionada presionar el botón Play y se genera las curvas de los valores del PID indicado en función de la cantidad de datos adquiridos. Para generar una nueva comparación de curvas con parámetros distintos presione en el botón Refresh y las gráficas se limpian para una nueva comparación.

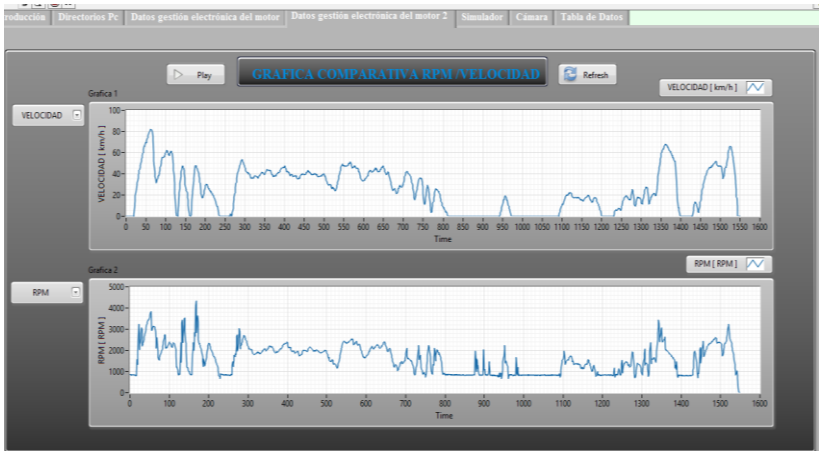


Figura 104. Ventana Gráficas comparativas.

- En la quinta pestaña encontramos una aplicación didáctica e interactiva que permite simular todos los datos recolectados en una interfaz, consta de un tacómetro digital y analógico, medidor de velocidad tanto digital como analógico, así mismo un medidor de

temperatura de refrigerante del motor analógico – digital, activación de pedal de freno, marcha colocada y una tabla de valores máximos y mínimos adquiridos durante la prueba.

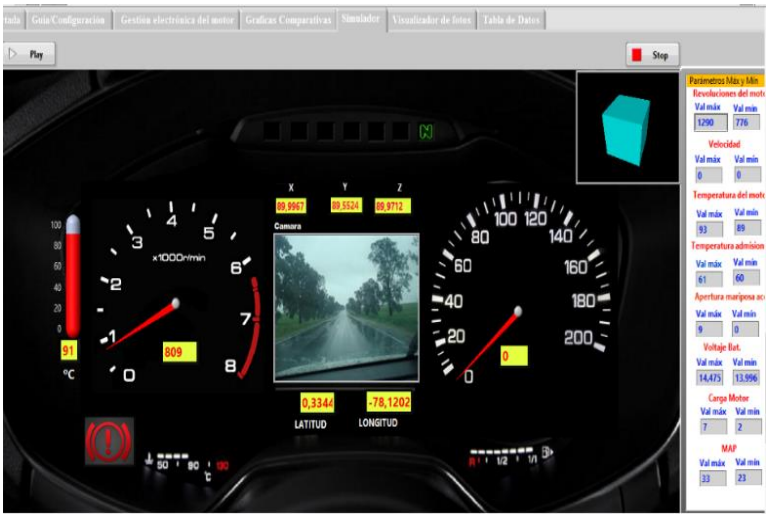


Figura 105. Ventana del simulador.

- Si se desea visualizar las fotografías una a una para detectar alguna anomalía en ellas, se puede hacer uso de la herramienta de la 6 pestaña de la interfaz que no es más que un visualizador de imágenes similar al de windows de las computadoras. Para su uso contamos con botones para el cambio de imágenes, tanto para cambiar y regresar a la imagen anterior.

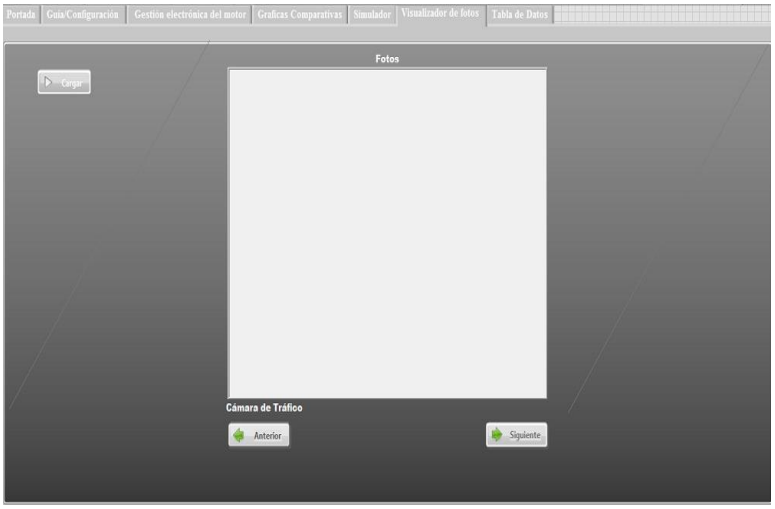
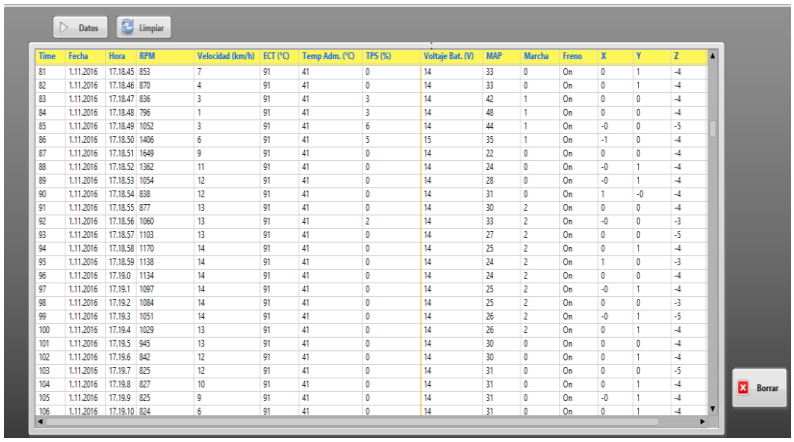


Figura 106. Ventana del visualizador de imágenes.

- El archivo de texto creado por el módulo almacena los datos y en la ventana 7 se encuentra la función para generar una tabla de los datos en un orden asignado de la siguiente manera: cantidad de datos, fecha, hora, Rpm, velocidad, Ect, Iat, Tps, voltaje de batería, Map, marcha, freno, ángulos x,y,z. Tenemos un botón play para generar los datos en la tabla una vez que lo pulsemos, además de un botón Reporte, el cual su función es la de crear un documento Excel el cual el usuario puede hacer uso o si desea imprimir el documento para analizar más detenidamente.



Time	Fecha	Hora	RPM	Velocidad (km/h)	ECT (°C)	Temp Adm. (°C)	TPS (%)	Voltaje Bat. (V)	MAP	Marcha	Freno	X	Y	Z
81	1.11.2016	17.18.45	853	7	91	41	0	14	33	0	On	0	1	-4
82	1.11.2016	17.18.46	870	4	91	41	0	14	33	0	On	0	1	-4
83	1.11.2016	17.18.47	836	3	91	41	3	14	42	1	On	0	0	-4
84	1.11.2016	17.18.48	796	1	91	41	3	14	48	1	On	0	0	-4
85	1.11.2016	17.18.49	1052	3	91	41	6	14	44	1	On	-0	0	-5
86	1.11.2016	17.18.50	1406	6	91	41	5	15	35	1	On	-1	0	-4
87	1.11.2016	17.18.51	1649	9	91	41	0	14	22	0	On	0	0	-4
88	1.11.2016	17.18.52	1362	11	91	41	0	14	24	0	On	-0	1	-4
89	1.11.2016	17.18.53	1054	12	91	41	0	14	28	0	On	-0	1	-4
90	1.11.2016	17.18.54	838	12	91	41	0	14	31	0	On	1	-0	-4
91	1.11.2016	17.18.55	877	13	91	41	0	14	30	2	On	0	0	-4
92	1.11.2016	17.18.56	1060	13	91	41	2	14	33	2	On	-0	0	-3
93	1.11.2016	17.18.57	1103	13	91	41	0	14	27	2	On	0	0	-5
94	1.11.2016	17.18.58	1170	14	91	41	0	14	25	2	On	0	1	-4
95	1.11.2016	17.18.59	1138	14	91	41	0	14	24	2	On	1	0	-3
96	1.11.2016	17.19.0	1134	14	91	41	0	14	24	2	On	0	0	-4
97	1.11.2016	17.19.1	1097	14	91	41	0	14	25	2	On	-0	1	-4
98	1.11.2016	17.19.2	1084	14	91	41	0	14	25	2	On	0	0	-3
99	1.11.2016	17.19.3	1051	14	91	41	0	14	26	2	On	-0	1	-5
100	1.11.2016	17.19.4	1029	13	91	41	0	14	26	2	On	0	1	-4
101	1.11.2016	17.19.5	945	13	91	41	0	14	30	0	On	0	0	-4
102	1.11.2016	17.19.6	842	12	91	41	0	14	30	0	On	0	1	-4
103	1.11.2016	17.19.7	825	12	91	41	0	14	31	0	On	0	0	-5
104	1.11.2016	17.19.8	827	10	91	41	0	14	31	0	On	0	1	-4
105	1.11.2016	17.19.9	825	9	91	41	0	14	31	0	On	-0	1	-4
106	1.11.2016	17.19.10	824	6	91	41	0	14	31	0	On	0	1	-4

Figura 107. Ventana de la tabla de datos en Labview.

CAPITULO VI

5. Conclusiones y Recomendaciones.

5.1 Conclusiones.

- El módulo de almacenamiento de datos on board para taxis, permite almacenar datos de gestión electrónica del vehículo, posición de marcha y estado de freno, aceleración en 3 ejes y GPS, organizados en una cadena de texto compuesta por 21 datos adquiridas cada 1 segundo y almacenados en un archivo txt.
- El modulo es capaz de almacenar hasta 1044480 datos consecutivos, es decir 12 días 8 horas, a través del valor datos que se encuentra ilustrado en la pantalla, el cual incrementa cada segundo que se adquiere toda la trama de datos.
- El receptor GPS que se encuentra integrado al módulo proporciona datos de tiempo como hora, fecha y ubicación latitud, longitud con una precisión de (+ - 3m).
- La cámara realiza capturas del entorno frontal del vehículo en intervalos de 1,5 segundos con una calidad de imagen de 2 megapíxeles en formato .jpg y un peso promedio de 100kb.
- La interfaz diseñada en LabView, permite al usuario importar los datos almacenados por el módulo y la cámara de manera sincronizada, para su visualización y análisis mediante submenús de selección de parámetros en graficas comparativas, simulación de viaje interactivo, además de brindar información estadística de los datos y valores adquiridos.
- La particularidad adicional de la interfaz desarrollada es el filtrado de todos los datos sincronizados por hora, así de esta manera es posible la visualización de datos dentro de los rangos de hora iniciales y finales que establezca el usuario.

5.2 Recomendaciones.

- Se recomienda el uso de este módulo de almacenamiento de datos para obtener información que permita al usuario que realiza el control, conocer los valores de funcionamiento de los distintos parámetros de gestión electrónica del vehículo y así se pueda establecer las causas probables presentes en un accidente.
- Es recomendable realizar pruebas de compatibilidad y envió de datos entre el vehículo y el módulo de almacenamiento antes de su instalación, debido a que los vehículos poseen distintos protocolos de comunicación y algunos no son reconocidos por el conector OBDII.
- Se recomienda descargar los datos almacenados en las tarjetas microSD diariamente para que el usuario realice un óptimo control cuando el vehículo haya culminado su movilización diaria por las diferentes rutas.
- Es recomendable colocar el receptor GPS en un lugar del vehículo donde aseguremos una buena recepción de datos de los satélites, así de esta manera cuando iniciemos la grabación se sincronicen ambos dispositivos.
- Es recomendable utilizar la plataforma de hardware libre Arduino ya que nos suministra librerías y herramientas compatibles para usarlas con distintos elementos electrónicos como la cámara fotográfica OV2640, facilitando su programación.
- Se recomienda el uso del software LabView para la importación de datos desde las tarjetas de almacenamiento, ya que nos permite la manipulación, transformación, decodificación y simulación de valores de cada parámetro adquirido.

BIBLIOGRAFÍA

- Arducam. (2015). ArduCAM-M-2M Camera shield. Obtenido de http://www.arducam.com/downloads/shields/ArduCAM_Mini_2MP_Camera_Shield_Hardware_Application_Note.pdf.
- Rueda, J. (2011). Técnico en mecánica & electrónica automotriz. Tomo 3. Colombia: Editorial Diseli. Código Biblioteca UTN: 629.287/. R84/Tec.
- Rueda, J. (2011). Manual técnico de fuel injection. Tomo 1. Colombia: Editorial Diseli. Código Biblioteca UTN: 629.53/. R84/Man.
- Reyes, F., Cid, J. (2015). Arduino: Aplicaciones en robótica, mecatrónica e ingenierías. México: Editorial Alfaomega. Código Biblioteca UTN: 629.892/. R49/Ard.
- Aranda, D. (2014). Electrónica: plataformas Arduino y Raps Berry PI. Argentina: Editorial Foxandina. Código Biblioteca UTN: 621.381/. A73/Ele.
- Del Rio, J., Sharial, S., Sarría, D. (2013). Labview: Programación para sistemas de instrumentación. México: Editorial Alfaomega. Código Biblioteca UTN: 005.13/. M65/Lab.
- Molina, J., Jiménez, M. (2012). Programación gráfica para Ingenieros. España: Editorial Alfaomega. Código Biblioteca UTN: 620.004/. M65/Pro.
- Bolton, W. (2013). Mecatrónica: Sistemas de control eléctrico en la ingeniería mecánica eléctrica. España: Editorial Alfaomega. Código Biblioteca UTN: 621.381/. B65/Mec.
- Villaseñor, J., Hernández, F. (2013). Circuitos eléctricos y aplicaciones digitales. Editorial Pearson. Código Biblioteca UTN: 621.319/. V55/Cir.
- Figliola, R., Beasley, Donald. (2009). Mediciones mecánicas: teoría y diseño. España: Editorial Alfaomega. Código Biblioteca UTN: 681.2/. F54/Med.
- Proakis, J., Manolakis, D. (2007). Tratamiento digital de señales. Editorial: Pearson.
- Muñiz, J. (2014). Android: Manual práctico para todos los niveles. Colombia: Ediciones de la U. Código Biblioteca UTN: 005.4/.M86/Adr.
- Foro de Labview. (2015). Obtenido de http://forums.ni.com/t5/La-Comunidad-en-Espa%C3%B1ol/ct-p/ESAForum.
- Tutorial Arduino. (2015). Obtenido de https://www.arduino.cc/en/Tutorial/BuiltInExamples.

- Correia, P. (2002). Guía práctica del GPS. España: Editorial Marcombo.
- Lawrence, L. (2001). GPS fácil: Uso del sistema de posicionamiento global. España: Editorial Paidotribo.
- Lajara, J., Pelegrí, J. (2011). Labview: entorno gráfico de programación. España: Editorial Marcombo.
- Arnalich, S., Urruela, J. (2012). GPS, Google Earth y Cooperación. Primera Edición. Arnalich, Water and hábitat.
- Quintero, C., Oñate, J., Arias, H. (2011). Instrumentación Electrónica Aplicada: prácticas de laboratorio. Colombia: Editorial Universidad del Norte.
- Adafruit. (2016). Obtenido de https://cdn-learn.adafruit.com/downloads/pdf/adafruit-micro-sd-breakout-board-card-tutorial.pdf.
- Areny, R. (2006). Instrumentos electrónicos básicos. Editorial: Marcombo SA.
- Camps, G., Espí, J., Muñoz, J. (2006). Fundamentos de la Electrónica Analógica. Universidad de Valencia.
- Zabler, E. (2002). Los sensores en el vehículo. Alemania: Editorial Roberth Bosch GmBH serie amarilla.

ANEXOS.

ANEXO 1. TABLA DE COSTOS

Item	Unidad	Costo unidad	Costo
Arduino Mega 2560	1	$ 60	$ 60
Arduino nano	1	$ 15	$ 15
Pantalla TFT 3.2”	1	$ 70	$ 70
Receptor GPS UBX 8030	1	$ 65	$ 65
Cámara	1	$ 80	$ 80
Cable OBDII UART V2	1	$ 80	$ 80
Sensores fin de carrera	6	$ 1	$ 6
Botones (Pulsadores)	4	$ 1.25	$ 5
Adaptador para encendedor de auto con regulador de salida de voltaje	1	$ 28	$ 28
Cable USB macho/hembra	1	$ 4	$ 4
Cables	1	$ 6	$ 6
Case Módulo/Cámara	1	$ 30	$ 30
Otros		$ 52	$ 52
Total			$ 501

ANEXO 2. DATASHEET UBX-M8030

UBX-M8030

Standard | Professional | Automotive

u-blox M8 concurrent GNSS chips

Highlights

- Concurrent reception of up to 3 GNSS (GPS, Galileo, GLONASS, BeiDou)
- Industry leading –167 dBm navigation sensitivity
- Industry lowest current consumption
- Superior position accuracy in urban canyons
- Security and integrity protection
- Support for all satellite augmentation systems
- Operating temperature range of -40° to +105°C for automotive grade chip

UBX-M8030-CT
2.99 x 3.21 x 0.36 mm

UBX-M8030-KT
5.00 x 5.00 x 0.59 mm

UBX-M8030-KA
5.00 x 5.00 x 0.59 mm

Product description

The UBX-M8030 high performance standard precision GNSS chips from u-blox, provide exceptional sensitivity and acquisition times for all GNSS systems. The chips utilize concurrent reception of up to three GNSS systems (GPS/Galileo together with Beidou or GLONASS). Reception from more than one constellation simultaneously allows extraordinary positioning accuracy in urban canyons, even with weak signals and high dynamics.

The UBX-M8030 chips feature low power consumption in concurrent reception mode and support advanced Power Save Modes for all GNSS, the power consumption remains low even for weak signals. The UBX-M8030 chips also support message integrity protection, geofencing and spoofing detection with configurable interface settings to easy fit to customer applications. The firmware supports QZSS, GAGAN and IMES together with WAAS, EGNOS, MSAS.

UBX-M8030 chips are available in miniature WL-CSP and QFN packages. Featuring built-in LNA, LDOs and DC/DC converter, and a small external BOM, the UBX-M8030 enables ultra-small solutions with a footprint of only 30 mm². Supporting TCXOs or lower price oscillators further ensures a minimal Total-Cost-of-Ownership.

The ultra small UBX-M8030-CT is a perfect choice for portable consumer applications with demanding size and cost constraints. Including rigorous automotive quality and manufacturing standards, extended testing and low failure rate make the UBX-M8030-KA chip ideal for automotive applications. With UBX-M8030-KA's operational tempature from -40° to +105°C, a new industry standard is set.

Migration from existing FW2 based u-blox M8030 chip designs are simple, since the upgraded UBX-M8030 offers backward compatibility.

Product selector

Model	Package	Category				GNSS					Supply	Interfaces				Features						Grade		
	Package	Standard Precision GNSS	High Precision GNSS	Dead Reckoning	Timing	GPS / QZSS	GLONASS	Galileo	BeiDou	Number of Concurrent GNSS	1.4 V – 3.6 V	UART	USB	SPI	DDC (I²C compliant)	Programmable (Flash)	Data logging	RTC crystal	Oscillator	Antenna supply and supervisor	Timepulse	Standard	Professional	Automotive
UBX-M8030-CT	WL-CSP47	•				•	•	•	•	3	•	•	•	•	•	S	S	S	C/T	S	2			
UBX-M8030-KT	QFN40	•				•	•	•	•	3	•	•	•	•	•	S	S	S	C/T	S	2			
UBX-M8030-KA*	QFN40	•				•	•	•	•	3	•	•	•	•	•	S	S	S	C/T	S	2			

C/T = Crystal and TCXO supported

S = supported, may require external components

* = Operating temperature -40° to +105°C

Standard
Professional
Automotive

Features

Receiver type	72-channel u-blox M8 engine GPS/QZSS L1 C/A, GLONASS L10F BeiDou B1, Galileo E1B/C SBAS L1 C/A: WAAS, EGNOS, MSAS, GAGAN
Time to first fix[1]	
Cold start:	26 s
Aided start:	2 s
Hot start:	1 s
Sensitivity[1]	
Tracking & Nav.	–167 dBm
Reacquisition	–160 dBm
Cold start	–148 dBm
Hot start	–157 dBm
Max nav. update rate[2]	
Single GNSS	up to 18 Hz
2 Concurrent GNSS	up to 10 Hz
Horizontal Pos. Accuracy[1]	2.0 m CEP
Multi-GNSS Assistance	AssistNow Online AssistNow Offline (up to 35 days) AssistNow Autonomous (up to 6 days)
Oscillator	Supports crystal or TCXO
LNA	Built-in
RTC input	32.768 kHz (optional), RTC can be derived from GNSS Crystal or TCXO
Antenna supervision	Short and open circuit detection supported with external circuit
DC/DC converter	Built-in, external component required
Anti Jamming	Active CW detection and removal
SQI Flash (optional) for	FW update AssistNow Offline AssistNow Autonomous
Raw Data	Code phase output
Odometer	Integrated in navigation filter
Geo-fencing	Up to 4 circular areas GPIO for waking up external CPU
Spoofing detection	Built-in
Signal integrity	Signature feature with SHA 256
Data-logger[3]	For position, velocity, time, and odometer data

1 For default mode: GPS/SBAS/QZSS+GLONASS with TCXO
2 ROM
3 External Flash required

Electrical data

Supply voltage	1.4 V to 3.6 V
Digital I/O voltage level	1.65 V to 3.6 V
Power consumption (2 concurrent GNSS)	21 mA @ 3.0 V (Continuous) 5.3 mA @ 3.0 V (PSM, 1 Hz)
Backup Supply	1.4 V to 3.6 V

Legal Notice

u-blox reserves all rights to this document and the information contained herein. Products, names, logos and designs described herein may in whole or in part be subject to intellectual property rights. Reproduction, use, modification or disclosure to third parties of this document or any part thereof without the express permission of u-blox is strictly prohibited.

The information contained herein is provided "as is". No warranty of any kind, either express or implied, is made in relation to the accuracy, reliability, fitness for a particular purpose or content of this document. This document may be revised by u-blox at any time. For most recent documents, please visit www.u-blox.com.

Copyright © 2016, u-blox AG

Packages

UBX-M8030-CT	47 Pin WL-CSP, 2.99 x 3.21 x 0.36 mm
UBX-M8030-KT/KA	40 Pin QFN, 5.00 x 5.00 x 0.59 mm

Environmental data, quality & reliability

Operating temp.	–20°C to +70°C (UBX-M8030-CT) –40°C to +85°C (UBX-M8030-KT) –40°C to +105°C (UBX-M8030-KA)
Storage temp.	–40°C to +125°C
Humidity	JEDEC MSL 1

RoHS compliant (lead-free) and green (no halogens)

Qualification according to AEC-Q100

Manufactured in ISO/TS 16949 certified production sites

Interfaces

Serial interfaces	1 UART 1 USB V2.0 compatible 1 DDC (I²C compliant) 1 SPI
Digital I/O	2 configurable time pulses 2 EXTINT interrupt inputs 2 PIO for antenna supervision
Memory	SQI interface for optional Flash

Support products

u-blox M8 Evaluation Kits:

Easy-to-use kits to get familiar with u-blox M8 positioning technology, evaluate functionality, and visualize GNSS performance.

EVK-M8N	u-blox M8 GNSS Evaluation Kit, which supports TCXO-based u-blox M8 designs
EVK-M8C	u-blox M8 GNSS Evaluation Kit, which supports crystal-based u-blox M8 designs

Product variants

UBX-M8030-CT	u-blox M8 GNSS chip, 47 Pin WL-CSP
UBX-M8030-KT	u-blox M8 GNSS chip, 40 Pin QFN
UBX-M8030-KA	u-blox M8 GNSS chip, 40 Pin QFN

Further information

For contact information, see www.u-blox.com/contact-us.

For more product details and ordering information, see the product data sheet.

ANEXO 3. CÓDIGO DE PROGRAMACIÓN ARDUINO

```
//*************//*****************//

//*** Ingenierìa Automotriz***//

//***Francisco Ormaza - Jefferson Enrìquez**//

//***Mòdulo de almacenamiento de datos ***//

#include <EEPROM.h>

#include <TinyGPS++.h>

#include <Arduino.h>

#include <Wire.h>

#include <OBD.h>

#include <SPI.h>

#include <SD.h>

#include <MultiLCD.h>

#include <I2Cdev.h>

#include <MPU9150.h>

#include <TimerOne.h>

#include "config.h"


LCD_SSD1289 lcd;


#define STATE_OBD_READY 0x2

#define STATE_MEMS_READY 0x10

#define STATE_GUI_ON 0x20

#define OBD_ADAPTER_I2C 0

#define OBD_ADAPTER_UART 1
```

```
#if USE_MPU6050 || USE_MPU9150

MPU6050 accelgyro;

#endif

extern uint8_t SmallFont[];

extern uint8_t BigFont[];

extern uint8_t SevenSegNumFont[];

extern unsigned int taxi[0x3B60];

extern unsigned int logo[0x3B60];


static const PROGMEM uint8_t tick[16 * 16 / 8] =

{0x00,0x80,0xC0,0xE0,0xC0,0x80,0x00,0x80,0xC0,0xE0,0xF0,0xF8,0xFC,0x78,0x30,0x00,0x00,0x01,0x03,
0x07,0x0F,0x1F,0x1F,0x1F,0x0F,0x07,0x03,0x01,0x00,0x00,0x00,0x00};


File myFile;// doc

int cont=0; // variable para conteo 1s

int comienzo=12;/////

int fin =13;//////

int barrido=11;//////Reinicio

//////18 y 19 Serial 1 para el obd

int on=44;///// encendido camara


int grabar=0;


byte contx1=0;///constantes eeprom

byte contx2=0;
```

```
////////////////////////////////caja//////////////
const int boton = 43; //primera
const int boton2= 45; //segunda
const int boton3= 47; //tercera
const int boton4= 49; //cuarta
const int boton5= 7;  //quinta
const int boton6= 9;  //freno
int m;
int freno;
int f;
int marcha=0;
int value;
int estado=0;  //1era
int estado1=0; //2da
int estado2=0; //3era
int estado3=0; //4ta
int estado4=0; //5ta
int estado5=0; //freno
byte state = 0;
//variables x, y posiciones pantalla
int x, y;
int x_1, y_1;
int led =8 ;//pin  de salida para encendido arduino 2
```

```
void processAccelerometer();

TinyGPSPlus gps;

COBDI2C obd;


String stringDato;


int dia = 0;

int mes = 0;

int anio = 0;

int hora = 0;

int minu = 0;

int segu = 0;


long anterior;

long intervalo;


void processAccelerometer()

{

#if USE_MPU6050 || USE_MPU9150

  int16_t ax, ay, az;

  int16_t gx, gy, gz;

#if USE_MPU9150

  int16_t mx, my, mz;

#endif

  int temp;
```

```
#if USE_MPU9150
  accelgyro.getMotion9(&ax, &ay, &az, &gx, &gy, &gz, &mx, &my, &mz);
#else
  accelgyro.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);
#endif

  temp = accelgyro.getTemperature();

  ax /= ACC_DATA_RATIO;
  ay /= ACC_DATA_RATIO;
  az /= ACC_DATA_RATIO;
  gx /= GYRO_DATA_RATIO;
  gy /= GYRO_DATA_RATIO;
  gz /= GYRO_DATA_RATIO;
#if USE_MPU9150
  mx /= COMPASS_DATA_RATIO;
  my /= COMPASS_DATA_RATIO;
  mz /= COMPASS_DATA_RATIO;
#endif
#if USE_MPU9150
#endif
#endif
}
```

```
void setup()
{
  stringDato = String();
  Serial.begin(115200);
  Serial2.begin(115200);
  pinMode(on, OUTPUT);// encendido/apagado càmara
  pinMode(led, OUTPUT);//salida para encendido
  pinMode(comienzo, INPUT_PULLUP);//////////13
  pinMode(fin, INPUT_PULLUP);///////////12
  pinMode(barrido,INPUT_PULLUP);///////////11
  pinMode(boton, INPUT_PULLUP);// interrupcion pin 2 "primera"
  pinMode(boton2, INPUT_PULLUP);// interrupcion pin 3 "segunda"
  pinMode(boton3, INPUT_PULLUP);// interrupcion pin 18 "tercera"
  pinMode(boton4, INPUT_PULLUP);//interrupcion pin 19 "freno"
  pinMode(boton5, INPUT_PULLUP);// pulsador cuarta
  pinMode(boton6, INPUT_PULLUP);//pulsador quinta
  pinMode(SS,OUTPUT);
  if (!SD.begin(SD_CS_PIN))
  {
    Serial.println("FALLO");
  }
  lcd.begin();
  lcd.clear();
  lcd.fillScr(255,255,255);//fondo de pantalla
  lcd.drawBitmap (220,145,80,80,logo,1);
```

```
lcd.setBackLight(255);
lcd.setBackColor(255, 255, 255); //fondo de letras
lcd.setColor(255,0,0);
lcd.setFontSize(FONT_SIZE_MEDIUM);
lcd.println();
lcd.println("   UNIVERSIDAD TECNICA DEL NORTE");
lcd.println();
lcd.println("     INGENIERIA AUTOMOTRIZ");
lcd.println();
lcd.println("            FICA");
lcd.println();
lcd.setColor(0,0,255);
lcd.println(" MODULO DE ALMACENAMIENTO DE DATOS");
lcd.println("       ON BOARD PARA TAXIS ");
lcd.println();
lcd.println();
lcd.println();
lcd.setColor(0,0,0);
lcd.println("CONECTANDO OBD-II.....");

#if USE_MPU6050 || USE_MPU9150
  Wire.begin();
  accelgyro.initialize();
  if (accelgyro.testConnection()) state |= STATE_MEMS_READY;
#endif
```

```
  obd.begin();
  //  while (!obd.init());
  while (!obd.init(OBD_PROTOCOL));    //omentario/////////////////
  state |= STATE_OBD_READY;

  lcd.setColor(0,128,0);
  lcd.setFontSize(FONT_SIZE_MEDIUM);
  lcd.println("OBD LISTO!");

char buf[OBD_RECV_BUF_SIZE];
if (obd.getVIN(buf))
  {
    lcd.setColor(RGB16_WHITE);
    lcd.print("VIN:");
    lcd.setColor(RGB16_YELLOW);
    lcd.print(buf);
  }

  while(grabar==0)
  {
    pin_inicio();
  }
  //delay(1000);
  lcd.clear();
```

```
  lcd.fillScr(255,255,255);//Pantalla en blanco
  lcd.drawBitmap (0,0,160, 95, taxi,2);

}
void grabando()
{
  lcd.setColor(255, 0, 0);              //rojo
  lcd.setBackColor(255, 255, 255);      //Fondo de las letras en blanco
  lcd.setFont(BigFont);                 //Cambio de Estilo de letra
  lcd.print2("GRABANDO...", CENTER, 120); //Mensaje en la posicion 120
}

void fin_grabacion()
{
  lcd.setColor(255, 0, 0);
  lcd.setBackColor(255, 255, 255);
  lcd.setFont(BigFont);
  lcd.print2("FIN DE GRABACION", CENTER, 120);
}

void reiniciando()
{
  lcd.setColor(255, 0, 0);
  lcd.setBackColor(255, 255, 255);
  lcd.setFont(BigFont);
```

```
  lcd.print2("REINICIANDO", CENTER, 120);
}

void lee_hora_fecha()
{
  dia = gps.date.day();
  mes = gps.date.month();
  anio = gps.date.year();
  hora = gps.time.hour();
  minu = gps.time.minute();
  segu = gps.time.second();
}

void pin_inicio()
{
  if(digitalRead(comienzo)==LOW)
  {
    delay(10);
    while(digitalRead(comienzo)==LOW);
    grabar=1;
    digitalWrite(on, HIGH);///
    delay(100);
    digitalWrite(on, LOW);///
    delay(80);
  }
```

```
}
void loop()
{
 int16_t ax, ay, az;
 int16_t gx, gy, gz;
 int16_t mx, my, mz;
 byte pids; //variable para llamado pids
 contx1=(byte)EEPROM.read(0);
 contx2=(byte)EEPROM.read(1);
 cont=(contx1<<8)+contx2;

 estado = digitalRead(boton); //1era
 estado1= digitalRead(boton2);//2da
 estado2= digitalRead(boton3);//3era
 estado3= digitalRead(boton4);//4ta
 estado4= digitalRead(boton5);//5ta
 estado5= digitalRead(boton6);//freno

 if (estado==HIGH) {
 if (estado1==HIGH){
 if (estado2==HIGH){
 if (estado3==HIGH){
 if (estado4==HIGH){
 if (estado5==HIGH){
 marcha=0;
```

```
}}}}}}

////////////////////////////////////////////////
////////////////////////////////////////////////


 if (estado==HIGH){}
 else{
   marcha=1;}

 if (estado1==HIGH){}
 else{
   marcha=2;}

 if (estado2==HIGH){}
 else{
   marcha=3;}

 if (estado3==HIGH){}
 else{
   marcha=4;}

 if (estado4==HIGH){}
 else{
   marcha=5;}
```

```
 if (estado5==LOW){
  freno=0;
  digitalWrite(led,HIGH);}
 else{
  freno=1;}
 m=marcha;
 f=freno;
////////////////////////////////////////////////
////////////////////////////////////////////////
 pin_inicio();
 lee_hora_fecha();              //lee hora y fecha

////////////////////////////////////////////////
////////////////////////////////////////////////
 if(digitalRead(fin)==LOW)
 {
  delay(10);
  while(digitalRead(fin)==LOW);
  fin_grabacion();
  grabar=0;
  digitalWrite(on, HIGH);///
  delay(100);
  digitalWrite(on, LOW);///
  delay(1000);
```

```
   lcd.drawBitmap (0,0,160, 95, taxi,2);
  }
////////////////////////////////////////////////
////////////////////////////////////////////////
  if(digitalRead(barrido)==LOW)
  {
   delay(10);
   while(digitalRead(barrido)==LOW);
   reiniciando();
   grabar=0;
   EEPROM.write(0,0);
   EEPROM.write(1,0);
   cont=0;
   contx1=0;
   contx2=0;
   delay(1000);
   lcd.drawBitmap (0,0,160, 95, taxi,2);
  }
////////////////////////////////////////////////
////////////////////////////////////////////////
////////////////////////////////////////////////

  if(grabar==1) ////////////modificacion txt/////////
  {
   long actual = millis();
```

```
if (x<9) {
x++;
  lcd.setColor(255, 0, 0);
  lcd.setBackColor(255, 255, 255);
  lcd.setFont(BigFont);
  lcd.print2("SINCRONIZANDO", CENTER, 120);
delay (9000);
 lcd.drawBitmap (0,0,160, 95, taxi,2);
}
if (x=9){
grabando();}

    myFile=SD.open("pids.txt",FILE_WRITE);
    cont++;
    contx1=cont>>8;
    contx2=cont&255;
    EEPROM.write(0,contx1);
    EEPROM.write(1,contx2);

    float v = obd.getVoltage();
    lee_hora_fecha();

    myFile.print(cont);          myFile.print(",");
    myFile.print(gps.date.day());  myFile.print(",");
    myFile.print(gps.date.month());myFile.print(",");
```

```
    myFile.print(gps.date.year()); myFile.print(",");


    Serial.print(cont);           Serial.print(",");
    Serial.print(gps.date.day());  Serial.print(",");
    Serial.print(gps.date.month());Serial.print(",");
    Serial.print(gps.date.year()); Serial.print(",");


    hora = hora - 5;
    if (hora < 0)
      hora = hora + 24;


    myFile.print(hora);             myFile.print(",");
    myFile.print(gps.time.minute()); myFile.print(",");
    myFile.print(gps.time.second()); myFile.print(",");


    Serial.print(hora);           Serial.print(",");
    Serial.print(gps.time.minute());Serial.print(",");
    Serial.print(gps.time.second());Serial.print(",");


  accelgyro.getMotion9(&ax, &ay, &az, &gx, &gy, &gz, &mx, &my, &mz);
    myFile.print(ax);    myFile.print(",");
    myFile.print(ay);    myFile.print(",");
    myFile.print(az);    myFile.print(",");
    myFile.print(gps.location.lat(), 4);    myFile.print(",");
    myFile.print(gps.location.lng(), 4);    myFile.print(",");
```

```
Serial.print(ax);    Serial.print(",");
Serial.print(ay);    Serial.print(",");
Serial.print(az);    Serial.print(",");
Serial.print(gps.location.lat(), 4);    Serial.print(",");
Serial.print(gps.location.lng(), 4);    Serial.print(",");

(obd.read(PID_RPM,value ));
Serial.print(value);    Serial.print(",");
myFile.print(value);    myFile.print(",");

(obd.read(PID_SPEED, value)) ;
Serial.print(value);    Serial.print(",");
myFile.print(value);    myFile.print(",");

(obd.read(PID_COOLANT_TEMP, value)) ;
Serial.print(value);    Serial.print(",");
myFile.print(value);    myFile.print(",");

(obd.read(PID_INTAKE_TEMP, value));
Serial.print(value);    Serial.print(",");
myFile.print(value);    myFile.print(",");

(obd.read(PID_THROTTLE, value)) ;
Serial.print(value);    Serial.print(",");
```

```
 Serial.print(v);       Serial.print(",");
 myFile.print(value);    myFile.print(",");
 myFile.print(v);       myFile.print(",");

 (obd.read(PID_INTAKE_MAP,value));
 Serial.print(value);    Serial.print(",");
 myFile.print(value);    myFile.print(",");
 myFile.print(m);       Serial.print(m);
 myFile.print(",");     Serial.print(",");
 myFile.print(f);       myFile.println(",");
 Serial.print(f);       Serial.println(",");

 lcd.setCursor(50,25);           //Posicion
 lcd.setColor(0,0,0);            //Color Negro
 lcd.setFontSize(FONT_SIZE_XLARGE);//Tipo de letra
 myFile.close();

 intervalo=actual-anterior;
 anterior=actual;
 //delay(0);
}


lee_hora_fecha();               //lee hora y fecha
lcd.setBackLight(255);          // va desde 0 a 255
```

```
lcd.setBackColor(255, 255, 255);  //Fondo de las letras en blanco


lcd.setCursor(2,25);            //Posicion
lcd.setColor(RGB16_RED);        //COlor
lcd.setFontSize(FONT_SIZE_MEDIUM);//Tipo de letra
lcd.print("HORA: ");            //Mensaje


lcd.setCursor(50,25);           //Posicion
lcd.setColor(0,0,0);            //Color Negro
lcd.setFontSize(FONT_SIZE_XLARGE);//Tipo de letra


int hora = (gps.time.hour());
hora = hora - 5;
if (hora < 0)
  hora = hora + 24;


lcd.printInt(hora);
lcd.print(":");
lcd.setCursor(95,25);           //Posicion
lcd.printInt(gps.time.minute(),2);
lcd.print(":");
lcd.setCursor(138,25);          //Posicion
lcd.printInt(gps.time.second(),2);


lcd.setColor(RGB16_RED);
```

```
  lcd.setFontSize(FONT_SIZE_MEDIUM);
  lcd.setCursor(180,25);
  lcd.print("DATOS: ");
  lcd.setColor(0,0,0);
  lcd.setFontSize(FONT_SIZE_XLARGE);
  lcd.printInt(cont,4);

  smartDelay(99);
}
static void smartDelay(unsigned long ms)
{
  unsigned long start = millis();
  do
  {
    while (Serial2.available())
    gps.encode(Serial2.read());
  }
  while (millis() - start < ms);
}
```