UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

ADQUISICIÓN Y REGISTRO DE PARÁMETROS VEHÍCULARES PARA EL DESARROLLO DE CICLOS DE MANEJO EN EL VALLE DE MÉXICO

T E S I S

QUE PARA OBTENER EL TÍTULO DE:

INGENIERO EN COMPUTACIÓN

P R E S E N T A :

ROBERTO GIOVANNI RAMÍREZ CHAVARRÍA

DIRECTOR DE TESIS:
M.I. LAURO SANTIAGO CRUZ

Ciudad Universitaria 2013

D E D I C A T O R I A

A mi abuela Isabel González, *IN MEMORIAM*. Porque desde donde estés, nunca me has dejado solo.

A mis papás, María Eugenia Chavarría y Roberto Ramírez, por ser el pilar de mi vida, por su apoyo incondicional y su eterno amor. Por su paciencia de aguantar mis locuras. Sin ustedes nada de esto hubiera sido posible. Pero aquí está el fruto de todos los esfuerzos. **¡Para ustedes! Los amo.**

A mi hermano Hugo, quien siempre me ha apoyado en las buenas y en las malas, eres mi cómplice de toda la vida, mi mejor amigo. **Gracias hermano, te quiero mucho.**

A mi Naty, por todo tu amor, por ser mi eterna compañera, por estar conmigo en las buenas y en las malas, por brindarme los mejores días, por aguantarme y siempre darme las palabras necesarias para seguir adelante con mis sueños. ¡**Llegaste en el mejor momento!**

A G R A D E C I M I E N T O S

A Dios, por todas sus bendiciones y permitirme llegar hasta este punto en mi vida.

A mi ALMA MATER, la Universidad Nacional Autónoma de México, por permitirme realizar mis estudios en su Facultad de Ingeniería.

Al Laboratorio de Control de Emisiones, encabezado por el Dr. Rogelio González Oropeza, a quien agradezco por todo el apoyo brindado durante la realización de esta tesis, su excelente calidad humana y su amistad. También agradezco a todo el grupo de trabajo del LCE, especialmente al M.I. Pedro I. Rincón Gómez, por compartir su espacio de trabajo conmigo.

A mi director de tesis, M.I. Lauro Santiago Cruz, por su paciencia, apoyo incondicional, sus conocimientos transmitidos, su valiosa amistad y por su orientación durante la elaboración de este trabajo.

Agradezco al Departamento de Ingeniería de Control y Robótica de la Facultad de Ingeniería, principalmente al M.I. Ricardo Garibay Jiménez, por permitirme colaborar con su grupo de trabajo durante la recta final de mi carrera, lo cual contribuye enormemente a mi formación profesional.

A los sinodales, por su tiempo y dedicación en la revisión de este trabajo.

A mi Tía, Rosario Chavarría y mi abuela Bernarda Zapata, por su hospitalidad, sus atenciones, su cariño.

A la Dirección General de Asuntos del Personal de Académico (DGAPA), de la UNAM, por el apoyo brindado a través del Programa de Apoyo a Proyectos de Investigación e Innovación Tecnológica (PAPIIT), en el proyecto IT102812-2 denominado: ***"Actualización y desarrollo de ciclos de manejo en el Valle de México".***

A todos aquellas personas quienes directa o indirectamente tuvieron alguna contribución para que esto sea posible.

ÍNDICE GENERAL

Capítulo 3. DESARROLLO DEL SISTEMA

ÍNDICE DE FIGURAS

Capítulo 1. INTRODUCCIÓN

Capítulo 2. GENERALIDADES

Capítulo 3. DESARROLLO DEL SISTEMA

Capítulo 4. INTEGRACIÓN, EVALUACIÓN Y PUESTA A PUNTO

ÍNDICE DE TABLAS

Capítulo 2. GENERALIDADES

Capítulo 3. DESARROLLO DEL SISTEMA

Capítulo 4. INTEGRACIÓN, EVALUACIÓN Y PUESTA A PUNTO

P R E F A C I O

En el Laboratorio de Control de Emisiones (LCE) de la Facultad de Ingeniería (FI) de la Universidad Nacional Autónoma de México (UNAM), se encuentra un grupo multidisciplinario de trabajo enfocado a las áreas de la ingeniería mecánica y eléctrica-electrónica y cuyos objetivos principales son:

- Desarrollo e investigación en temas relacionados con motores de combustión interna.
- Homologación y certificación vehicular a prototipos y/o modelos de vehículos que se pretenden comercializar en el país.
- Evaluación de las emisiones contaminantes provenientes de motores de combustión interna.
- Docencia y experimentación en los campos temáticos relacionados con: termodinámica, termofluidos y motores de combustión interna.
- Desarrollo de proyectos institucionales, con los sectores público y privado.

Como resultado de lo anteriormente mencionado, durante el año 2012, el personal del LCE participó en el proyecto institucional Programa de Apoyo a Proyectos de Investigación e Innovación Tecnológica (PAPIIT) IT102812-2 denominado: ***"Actualización y desarrollo de ciclos de manejo en el Valle de México"***, de donde este trabajo se desprendió.

En el presente trabajo se describe el desarrollo y la implementación de un sistema que permitirá resolver la propuesta de actualizar y desarrollar los ciclos de manejo en el Valle de México. Dicho sistema concuerda con las técnicas con las que actualmente se cuenta en el ámbito de la instrumentación electrónica y que generalmente permite al operador del instrumento poder visualizar y registrar los datos obtenidos como resultado del experimento de manera sencilla. Bajo este contexto, el LCE en conjunto con la Coordinación de Instrumentación (CI) del Instituto de Ingeniería (II), ambos de la UNAM, se dieron a la tarea de desarrollar un sistema que permitirá obtener los parámetros vehiculares más importantes (velocidad, distancia recorrida, inclinación del vehículo, altitud y emisiones contaminantes), los cuales permitirán solucionar el problema propuesto.

El Sistema de Adquisición y Registro de Datos (SARD), comprende desde la lectura de parámetros mediante distintos sensores, su registro en una unidad de memoria externa y el procesamiento de los datos obtenidos como resultado del proceso de adquisición.

En cuanto a la motivación del proyecto, desarrollo de ciclos de manejo, cabe comentar, de manera breve, que se pueden definir como un perfil de velocidades trazado en una gráfica velocidad-tiempo, que representa una forma típica de conducir por los distintos tipos de calles y autopistas, tomando en cuenta factores como la tecnología del vehículo, las características del tráfico, condiciones

climáticas y geográficas, así como las características de conducción de los vehículos por parte de los usuarios.

El uso de vehículos instrumentados mediante el SARD, permite un registro de los datos adquiridos en los recorridos realizados por el vehículo, los cuales al ser analizados permitirán desarrollar íntegramente los ciclos de manejo deseados y de esta manera sean una herramienta para evaluar dispositivos en relación con las emisiones contaminantes y el consumo de combustible.

Este trabajo se divide en cinco capítulos, organizados de la siguiente manera:

Capítulo 1. En este capítulo se introduce al lector a los conceptos básicos del desarrollo de ciclos de manejos y se describe un panorama general de la solución al problema presentado.

Capítulo 2. Dentro de este capítulo se abordan los temas necesarios que permitirán al lector comprender el desarrollo del SARD, como son: instrumentación electrónica, microcontroladores, Sistemas de Posicionamiento Global así como las tarjetas de memoria y su sistema de archivos.

Capítulo 3. Aquí se detallan los componentes utilizados en el desarrollo del sistema propuesto, basados en las especificaciones y necesidades del proyecto en cuestión. De igual manera se describe la programación del microcontrolador utilizado. Finalmente se aborda detalle el diseño del *software* de procesamiento de los datos, el cual tendrá como objetivo tener un registro histórico de los muestreos realizados mediante el SARD.

Capítulo 4. Aquí se describen la integración del equipo, las pruebas realizadas a cada uno de los componentes que lo integran, así como al software de procesamiento con el objetivo de evaluar su correcto funcionamiento.

Capítulo 5. Finalmente, dentro de este capítulo se detallan los resultados y conclusiones obtenidos en el desarrollo del trabajo y la implementación del sistema propuesto.

C A P Í T U L O 1

INTRODUCCIÓN

En este capítulo se introducirá al lector al origen y antecedentes de los Ciclos de Manejo, el proceso para su desarrollo, así como su importancia en México. Posteriormente se describen los motivos que originaron el desarrollo del sistema y una breve descripción de éste.

1.1. Antecedentes

En 1898 entró a México el primer automóvil. El auto era francés, marca Delaunay Belleville, hecho a mano en las fábricas de Couvier. En 1930 habían 88 443 vehículos en la República Mexicana, que contaba entonces con 16, 588,522 habitantes, esto es, 187.6 habitantes por vehículo; en 1938 la relación era de 150 y en 1946 de 114. Posteriormente, entre las décadas de los 70's y 90's esta relación ha oscilado entre 16 y 12, lo que representa un aumento desmesurado del parque vehicular, en comparación con el poco tiempo de su aparición de manera comercial. Precisamente por estas circunstancias de la creciente demanda de vehículos, las legislaciones se obligan a ser más estrictas en términos de emisiones contaminantes, y parece que ahora también en el consumo de combustible. Esta situación, a su vez, obliga a mantener instrumentos vehiculares de evaluación dinámicos, es decir, formas de evaluación de emisiones contaminantes que contemplen los avances tecnológicos de los vehículos, así como los cambios que van sufriendo las condiciones de equilibrio ecológico de una determinada población, o el desarrollo de centros urbanos, de acuerdo a su propio crecimiento, y en forma paralela, a sus necesidades, dígase suministro de servicios y de transporte.[1]

En la ciudad de México se han realizado estudios por parte de la Secretaría de Transporte y Vialidad enfocados a monitorear los volúmenes de tránsito en las principales avenidas de esta gran ciudad, así como las emisiones contaminantes provenientes principalmente de autobuses de transporte urbano. Este enfoque del problema de tránsito y emisión de contaminantes, aunque importante, es un enfoque parcial, que debe y tiene que ser complementado y reforzado con estudios más completos, a fin de obtener una prueba representativa y confiable del problema de emisión de contaminantes-tránsito, de aquí surge la necesidad de desarrollar Ciclos de Manejo.

[1] **González Oropeza R., Galván Zacarías A.** (2003). *"Desarrollo de ciclos de manejo característicos de la ciudad de México",* IX Congreso SOMIM. pp. 535-544.

Tomando en cuenta el panorama descrito anteriormente, el personal del LCE desarrolló entre 1998 y el año 2000, ciclos de manejo que corresponden a las zonas Noroeste, Noreste, Centro, Suroeste y Suereste de la Zona Metropolitana del Valle de México. También cabe mencionar, que en el año 1998 el LCE propuso al Gobierno del Distrito Federal un proyecto para diseñar y construir un banco de motocicletas, así como el desarrollo de ciclos de manejo para dichas motocicletas y vehículos ligeros.[2] Dichos ciclos de manejo fueron desarrollados siguiendo una metodología propia.

1.2. Ciclos de Manejo

1.2.1. Definición

Un ciclo de manejo es un perfil de velocidades trazado en una gráfica velocidad-tiempo, que representa una forma típica de conducir por los distintos tipos de calles en ciudades y autopistas, tomando en cuenta factores como la tecnología del vehículo, las características del tráfico, condiciones climáticas (velocidad y dirección de viento, temperatura y presión atmosférica) y geográfica (latitud, longitud, altitud), velocidad, así como las características de conducción de los vehículos por parte de los usuarios.[2]

Los ciclos de manejo son de gran importancia a nivel mundial, entre otros fines, para planear adecuadamente el desarrollo de alguna ciudad, en el desarrollo de las nuevas tecnologías para automóviles, en la validación de modelos que predicen el comportamiento de los vehículos en la vía pública y las estadísticas de emisiones contaminantes en grandes ciudades.

Las condiciones de tráfico cambian permanentemente debido a varios factores, como son: el crecimiento de la población, la tecnología de los vehículos, los cambios en el transporte público, las modificaciones en la red de carreteras, las nuevas legislaciones para mantener el equilibrio ecológico, entre otros. Esto hace que las formas de conducir sean dinámicas, es decir, que vayan cambiando conforme a las demandas en cuanto a transporte se refiere. Debido a esta dinámica de cambio, los ciclos de manejo deben ser actualizados y de no ser así dejan de ser una herramienta confiable para estimar los inventarios de emisiones, medición del consumo de combustible, homologación de vehículos, etcétera.

Los ciclos de manejo pueden ser la ventana que nos permita conocer la realidad de la contaminación atmosférica causada por el parque vehicular, y por supuesto, permita tomar las medidas

[2] **González Oropeza R. (2005)** *"Los ciclos de manejo, una herramienta útil si es dinámica para evaluar el consumo de combustible y las emisiones contaminantes del auto transporte"*, INGENIERÍA Investigación y Tecnología VI.3. pp. 147-162.

necesarias para controlar las emisiones contaminantes de los vehículos automotores que son los responsables principales de esta situación.

1.2.1. Importancia de los ciclos de manejo en México

Para ayudar a resolver el problema de tráfico en las ciudades, se requiere saber el funcionamiento de los vehículos en condiciones de uso cotidianas, además, esta información puede utilizarse también para otros propósitos, como son los desarrollos tecnológicos, por ejemplo, para probar combustibles, diseñar partes automotrices, elaborar inventarios de emisiones, homologar vehículos, entre otros, estos son algunos de los objetivos del desarrollo de ciclos de manejo.

En relación con la homologación de vehículos, en México se utiliza el ciclo de procedimiento de prueba federal *(FTP: Federal Test Procedure)* para determinar las emisiones, en gramos por kilómetro recorrido, de los vehículos nuevos en planta.[3] Este procedimiento emplea un ciclo de manejo que se obtuvo en la ciudad de los Angeles California, en un recorrido típico de casa al trabajo por la mañana, y se desarrolló a mediados de los años 60's. El procedimiento ha sufrido diversas modificaciones con el paso del tiempo, hasta llegar a la versión actual llamada *Urban Dynamometer Driving Schedule (UDDS)* y que sirvió como referencia para certificar los valores de las emisiones de vehículos y camiones ligeros de modelo 1972 y posteriores. Con el paso del tiempo, este ciclo ha sufrido ligeras modificaciones, de tal forma que se repite una etapa inicial que dura 505 segundos, seguida de un período de 10 minutos de reposo del vehículo, la diferencia estriba en que la primera etapa se inicia con el motor totalmente frío.

Desde el año 1975 el *UDDS* ha sido utilizado como procedimiento de prueba y ha sido adoptado en México por la Norma Oficial Mexicana (NOM); sin embargo, se ha cuestionado por diversos sectores la representatividad de la forma de conducir en México, incluso en los Estados Unidos de América (EUA), la Agencia de Protección Ambiental de EUA *(EPA: Enviroment Protection Agency)* realizó un estudio exhaustivo en diversas ciudades, con más de 100 vehículos para validar el ciclo que tradicionalmente se ha utilizado y paralelamente se intentó determinar los factores que influyen en la emisión de contaminantes, como pueden ser el tipo de combustible, la temperatura de operación del vehículo, la altitud y la tecnología del propio vehículo. Además de estos factores se tiene el creciente problema de tránsito, del cual se pueden citar los siguientes fenómenos más comunes:

- La circulación de diferentes vehículos en una misma vialidad.
- Automóviles con diferentes dimensiones, velocidades y características de aceleración.
- Circulación de tránsito motorizado en vialidades inadecuadas.

[3] **INSTITUTO NACIONAL DE ECOLOGÍA (INE), "***Determinación de los factores de emisión para fuentes vehiculares circulando en la zona metropolitana del Valle de México en unidades de gramos por kilómetro"*, México, 2004.

- Pocos cambios en el trazo urbano, calles angostas, torcidas, desfiguradas y pronunciadas pendientes, carreteras que definitivamente no han evolucionado.
- Falta de planificación de tránsito; calles, carreteras y puentes se siguen construyendo con especificaciones inadecuadas, sin previsión de estacionamientos en centros comerciales.
- Ubicación inapropiada de zonas residenciales con relación a zonas industriales o comerciales.
- Falta de responsabilidad cívica que implica tener un automóvil, además del desconocimiento total de los reglamentos de tránsito y una completa falta de educación vial.

1.3. Metodología para el desarrollo de ciclos de manejo representativos

Un ciclo de manejo es representativo cuando todos los factores que intervienen en el proceso de desarrollo son lo más cercanos posible a las condiciones reales. Dichos factores son:

- Características del conductor
- Características del vehículo
- Características del tráfico:
    a) Ruta de manejo
    b) Medio ambiente
    c) Tiempo de manejo (hora y fecha)

Para encontrar el grado y la relación en que los factores anteriormente mencionados afectan a los patrones de conducción, es necesario definir métodos estadísticos en el uso del vehículo. Eventualmente, las estadísticas que no son conocidas, resultan de los procesos de medición de los parámetros vehiculares. Para lograr las estadísticas de los factores, el proceso se puede dividir en dos formas:

1) Medición de los parámetros representativos:

En este caso, las mediciones son organizadas de tal manera que los datos resultantes de éstas pueden ser considerados como representativos. Este resultado es obtenido durante el proceso de adquisición de los datos.

2) Creación de los parámetros representativos:

En este caso, los datos son procesados de tal manera que, el resultado obtenido es considerado como representativo. La única condición para este caso es que los datos debieron ser recolectados y almacenados previamente. Este resultado es obtenido durante el procesamiento de los datos referenciados a una prueba determinada. A continuación se describen los pasos para desarrollar un ciclo de manejo representativo.

1.3.1. Adquisición de los datos

El proceso para desarrollar ciclos de manejo representativos, comienza con la adquisición y registro de los datos provenientes de las condiciones reales en que el automóvil se encuentra. El uso de vehículos instrumentados para registrar los datos correspondientes a la velocidad y el tiempo en distintas condiciones de tráfico vehicular, permitirán el correcto desarrollo de ciclos de manejo. Cabe mencionar que los datos deberán ser almacenados y organizados correctamente para su posterior análisis.[4]

1.3.2. Procesamiento de los datos

Para crear un ciclo de manejo que sirva como referencia, los datos obtenidos durante el proceso de medición deben ser combinados de tal manera que, el conjunto de ellos pueda definir una buena representación del vehículo del cual los datos fueron adquiridos. En esta fase, todos los parámetros que fueron tomados en cuenta para la creación de un ciclo representativo deben ser considerados. Durante el procesamiento de datos, estos deben ser separados y analizados estadísticamente.

1.3.3. Creación del ciclo de manejo

Conociendo todas las características del ciclo de manejo tomado como representativo, obtenidas en los procesos anteriormente descritos, el siguiente paso es construir un ciclo de manejo que se asemeje lo más posible a las características del ciclo representativo. Para ello se presenta un perfil estadístico que consta de las siguientes cuatro etapas:

1) Análisis del ciclo de manejo representativo
2) Creación de muestras del ciclo de manejo
3) Selección y comparación estadística de los datos
4) Finalización del ciclo de manejo

En resumen, un requerimiento esencial para determinar, de manera experimental, el consumo de energía y las emisiones de los vehículos, radica en el uso de ciclos de manejo, los cuales representan las condiciones de manejo en la vida diaria de una ciudad. Estos ciclos también son relevantes en el uso de simulaciones por computadora, usadas en el desarrollo y evaluación de nuevas tecnologías automotrices.

---

[4] **Debbie A. Niemeier, Ph.D**. (1999). *"Data Collection for Driving Cycle Development: Evaluation of Data Collection Protocols"*, Department of Civil and Environmental Engineering, University of California, United States of America.

1.4. Sistemas de Adquisición de Datos

Actualmente podemos encontrar una gran variedad de aplicaciones en las que es necesario adquirir, registrar y procesar variables físicas o datos analógicos, para ello dichas variables deben ser convertidas a información digital y así poder realizar las acciones antes mencionadas. El proceso mediante el cual las señales analógicas son adquiridas y digitalizadas para su registro y/o procesamiento comúnmente se realiza mediante un Sistema de Adquisición de Datos (SAD).

Un SAD es un sistema eléctrico-electrónico, el cual básicamente está compuesto por sensores o transductores para adquirir el valor de la variable de interés en el proceso de medición, un módulo acondicionador de la señal proveniente del sensor, para que posteriormente pueda ser muestreada y convertida a su equivalente digital mediante el convertidor analógico-digital (CAD) y finalmente es necesario el uso de una unidad de registro para poder almacenar los datos que posteriormente podrán ser procesados y analizados. En la figura 1.1 se muestra un diagrama de bloques simple de un SAD.

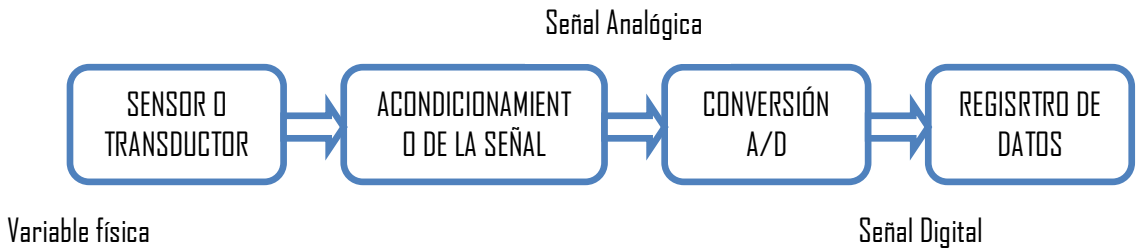


Figura1.1. Diagrama de bloques de un Sistema de Adquisición de Datos.

Cabe resaltar la diferencia entre un sistema de adquisición de datos puro y uno en donde el registro de datos se lleva a cabo, esto debido a que en diversas ocasiones sólo se realiza el proceso de adquisición de datos sin considerar registro alguno; por ejemplo, el monitoreo de la temperatura en cierto lugar, en donde sólo interesa saber si se ha alcanzado cierto valor de temperatura para accionar un dispositivo de enfriamiento. Por el contrario y siguiendo con el ejemplo de la temperatura, tal vez los valores de temperatura deberán ser almacenados con una tasa de registro de 1[s], en este caso el SAD también realiza el proceso de almacenamiento de información.

Por consiguiente, podemos hacer una clasificación en cuanto a procesos de adquisición de datos se refiere, lo cual nos permitirá diferenciarlos. Por una parte tenemos a los sistemas de adquisición en donde los datos no son registrados, a éstos les llamaremos típicamente Sistemas de Adquisición de Datos (SAD) y por otro lado a los sistemas de adquisición en donde el proceso de almacenamiento de los datos adquiridos si se lleva a cabo, los cuales comúnmente son llamados Sistemas de Adquisición y Registro de Datos (SARD).

1.5. Descripción general del SARD

El sistema desarrollado para obtener ciclos de manejo se denominará *CYCLE-DAQ*, por lo que, de éste momento en adelante, se asumirá que éste fungirá como un SARD. Dicho sistema estará basado en herramientas de instrumentación tradicional y virtual. El sistema dará información detallada de la posición, velocidad, inclinación y emisión de gases contaminantes del vehículo en el cuál éste se encuentre instalado.

1.5.1. Alcance

El sistema *CYCLE-DAQ* puede coadyuvar y participar en el desarrollo de ciclos de manejo para el Valle de México, estableciendo trayectorias de crucero reales y representando el flujo vehicular en zonas urbanas, semiurbanas y carretera foránea. Se pretende también que los ciclos de manejos obtenidos, mediante los datos adquiridos por el sistema, sean una base para la realización de pruebas dinamométricas, a fin de tener una visualización más exacta de los índices de emisión de los vehículos automotores, y asimismo, se puedan obtener datos para incidir en la Normativa Mexicana respectiva, para homologación de los mismos. Por otro lado, dichos ciclos de manejo pueden incidir en la planeación y reestructuración de los programas de verificación vehicular, en pruebas específicas de combustibles existentes y reformulados, y en el análisis de partes en el diseño de motores.

Existen hoy en día modelos que pueden reproducir casi a la perfección patrones de manejo, valores de las emisiones y el consumo de combustible; sin embargo, siempre existe la desventaja de que los ciclos generados por dichos modelos no incluyan algunos comportamientos reales de las variables, lo cual puede causar resultados y conclusiones mal fundamentadas. Los ciclos de manejo generados corren el riesgo de estar suavizando algunos aspectos en comparación con los datos reales del tráfico. En este sentido los ciclos que se establezcan servirán de antecedente para promover cambios sustanciales en las legislaciones respectivas, pero no sólo para los vehículos nuevos, sino también para aquellos que están en uso.

Entre los principales objetivos del *CYCLE-DAQ* destacan los siguientes:

- ✓ Evaluar los parámetros monitoreados para establecer relaciones promedio congruentes que permitan una visión clara de las trayectorias, su duración, las condiciones de la puesta en marcha del vehículo, la distancia recorrida, la frecuencia de uso, etc.
- ✓ Desarrollar ciclos de manejo representativos de las condiciones reales de conducción en vías principales, secundarias y autopista.
- ✓ Evaluar las emisiones contaminantes, producto de la prueba vehicular, para establecer la incidencia en la normativa mexicana respectiva para la homologación de vehículos ligeros.

- ✓ Hacer pruebas de condiciones en frío y en caliente, que muestren versatilidad, flexibilidad y objetividad, así como la funcionalidad que se requiere para diferentes propósitos, ya que los ciclos de manejo (o partes de ellos) pueden requerirse para evaluar vehículos en circulación, estimar desempeño de un combustible, aditivo, sistema, dispositivo, entre otras, para mejorar el desempeño de vehículos y motores.

1.5.2. *Hardware* del sistema

El *CYCLE-DAQ* se desarrollará alrededor de un microcontrolador de 8 bits. Los sensores y dispositivos a incorporar son: un módulo receptor GPS; un acelerómetro, utilizado como sensor de inclinación, y un módulo analizador de los 5 principales gases contaminantes (CO, CO_2, HC, NO_X y O_2) que el vehículo emite durante diversas condiciones. También habrá que considerar la adquisición de datos correspondientes a las características de arranque en frío, el frenado y la aceleración. Para operar el analizador de gases se requerirá del uso de un inversor de corriente directa a corriente alterna CD-CA, para que el analizador pueda ser energizado con la batería del vehículo.

Adicionalmente, los datos obtenidos como resultado de la adquisición, y que son de mayor relevancia para el usuario, son mostrados en un *display* de cristal líquido. Finalmente el equipo desarrollado se instalará en un gabinete portátil.

Con el objetivo de tener un registro de los datos adquiridos, éstos son almacenados de manera dinámica, en una memoria tipo micro SD CARD.

1.5.3. *Software* de procesamiento de datos

Una vez que los datos han sido adquiridos y registrados, la información es procesada por medio de un instrumento virtual (interfaz de usuario) en el entorno de desarrollo de LabVIEW. Dicho *software*, denominado *DRIVE-SOFT*, muestra, la fecha de adquisición de los datos, el tiempo del recorrido y la distancia recorrida por el automóvil, además, presenta las gráficas correspondientes:

- Velocidad – tiempo
- Recorrido del automóvil (posición)
- Concentraciones de gases contaminantes

Finalmente, se obtiene como resultado los datos ordenados en una hoja de cálculo, lo que facilitará el posterior análisis estadístico por parte de los ingenieros mecánicos del Laboratorio de Control de Emisiones.

En el siguiente capítulo se presenta el panorama general y actual, en el ámbito de la Instrumentación Electrónica, así como las descripciones, diferencias, ventajas y desventajas del uso de la instrumentación tradicional y de la instrumentación virtual, con el objetivo de comprender el desarrollo del equipo *CYCLE-DAQ*.

10 Introducción

C A P Í T U L O 2

GENERALIDADES

En este capítulo se abordan los temas básicos que permitirán al lector comprender el desarrollo del equipo CYCLE-DAQ. La instrumentación electrónica y los sistemas de adquisición de datos, los microcontroladores, Sistemas de Posicionamiento Global, las tarjetas de memoria y su sistema de archivos, así como los principios de las bancas analizadoras de gases contaminantes.

2.1. Aspectos generales sobre los sistemas de instrumentación

Los procesos de instrumentación y la medición automática, hoy en día son ampliamente usados tanto en laboratorios como en la producción, ya sea aplicado a la investigación o a la industria.

Uno de los principales objetivos de la instrumentación electrónica radica en tomar medidas de distintas variables físicas, mediante la detección y procesamiento de éstas para realizar el monitoreo y control de procesos, empleando dispositivos y tecnologías electrónicas. La instrumentación se enfoca en los sistemas en los que su uso está dirigido a la medición de magnitudes de fenómenos físicos del medio ambiente o los generados por el ser humano, y presenta la información deseada en un dispositivo de despliegue.

El concepto de medir lo podemos resumir como asignar un valor numérico a una magnitud concreta (voltaje, corriente, potencia, resistencia, etc.), de acuerdo con una regla predeterminada que esté basada en la experimentación.

Toda medida implica cuando menos tres funciones:

1) Detectar la magnitud de interés, empleando si hace falta un transductor, o un sensor que ofrezca una señal eléctrica útil a partir de la señal de entrada;
2) Procesar la señal obtenida por el detector para extraer la información deseada y ofrecerla al indicador en forma de una señal adecuada.
3) Presentar la lectura, almacenarla, o transmitirla, o varias acciones a la vez.

El conjunto de instrumentos que hacen posible la realización de la medida o medidas recibe el nombre de **sistema de instrumentación**, el cual consta de diversos instrumentos, un dispositivo de interconexión de instrumentos y un controlador inteligente que gestiona el funcionamiento de todo el sistema y da las órdenes para que una medida se realice correctamente.[9]

Las características por las que la tecnología electrónica es la más utilizada por los sistemas de instrumentación, son:

- Las señales eléctricas permiten manejar señales en un rango dinámico de tiempos muy amplio.
- Las señales eléctricas pueden ser transmitidas muy fácilmente a través de cables metálicos, sistemas radiados, o fibra óptica.
- Las señales eléctricas pueden ser amplificadas por circuitos electrónicos de forma muy eficientes, y pueden manejarse rangos de señal muy amplios.
- Los sistemas electrónicos permiten complejas transformaciones funcionales de las señales eléctricas.
- Las señales eléctricas son las más apropiada para ser introducidas en computadoras personales, los cuales representan el medio más potente de registro, transformación y presentación de la información.
- La tecnología electrónica actual es la que presenta mejor relación prestaciones /costo.

2.2. Componentes básicos de un sistema de instrumentación

El esquema básico de cualquier instrumento comprende tres unidades fundamentales: transductor, acondicionador y dispositivo de despliegue, aunque es posible que se utilicen diversos dispositivos e instrumentos, desde el propio detector de la variable, el dispositivo de presentación de datos, hasta los que proporcionan la potencia para el suministro eléctrico. De manera general, en la figura 2.1 se muestra un sistema de instrumentación.

[9] **Mata Hernández Gloria**, "TEMAS SELECTOS DE INSTRUMENTACIÓN VIRTUAL", Depto. De Ingeniería de Control y Robótica, FI-UNAM, 2012.

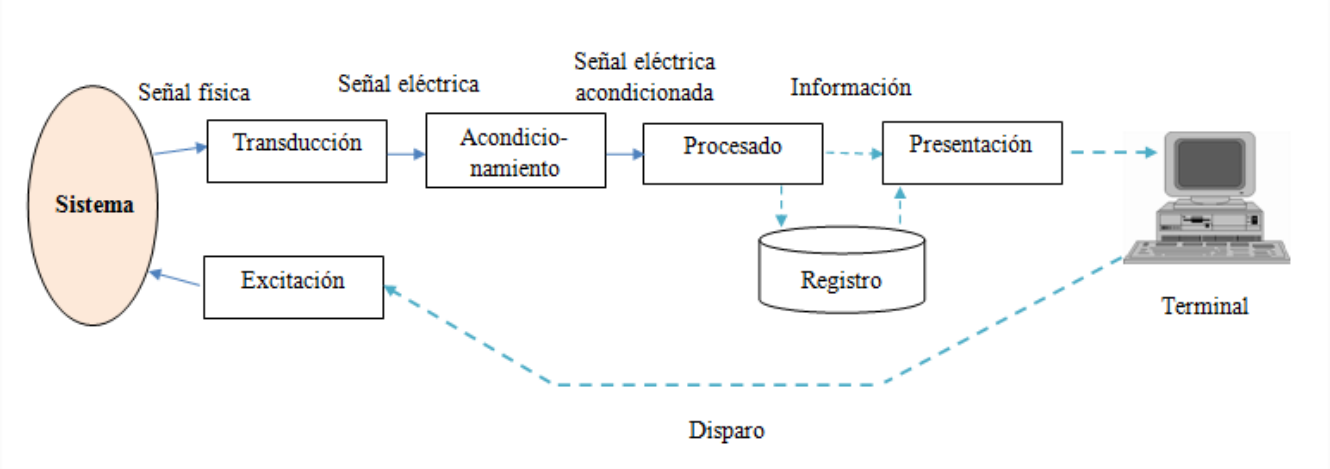


Figura 2.1. Diagrama general de un sistema de instrumentación.

Como se puede observar en la figura anterior, existen diversas unidades fundamentales en el esquema del sistema de instrumentación, las cuales se describen brevemente a continuación.

Transductor. Es el componente que convierte una magnitud física a medir en una señal eléctrica. En este componente se puede diferenciar del sensor, que es el elemento sensible primario que responde a las variaciones de la magnitud que se mide, y el transductor que es el que lleva acabo la conversión energética entre la magnitud de entrada y de salida.

Los transductores se suelen clasificar en dos grupos:

1) Analógicos: la salida es un valor de voltaje o corriente comprendida en un rango de valores, por ejemplo, medidas de temperatura, caudal y posición.
2) Digitales: La salida del sensor toma dos valores únicamente, estado lógico alto (5 [V]) y estado lógico bajo (0 [V]). Ejemplos de éstos son: pulsador, sensor de presencia, sonda lambda, sensor de fin de carrera.

Acondicionador. Esta unidad incluye todas aquellas transformaciones que deben realizarse sobre la señal eléctrica que resulta en la salida del transductor. Existen dos razones por las que las señales de salida del transductor deban ser acondicionadas:

1) Cuando el tipo de señal eléctrica que se proporciona el transductor no es un voltaje, se utiliza un convertidor desde el tipo de señal de que se trate, a voltaje. Así en transductores resistivos, es normal que se utilice un circuito puente para convertir el valor de resistencia a voltaje. Cuando el transductor es de tipo capacitivo o inductivo, se suele montar como parte de un oscilador, y la magnitud de salida es una frecuencia, y debe utilizar un convertidor frecuencia/voltaje.

2) La señal debe ser acondicionada para llevar la relación señal/ruido hasta niveles adecuados. Esto tipo de acondicionamiento implica:

- Amplificar: es el proceso por el cual las señales son llevadas hasta niveles que sean suficientemente superiores al nivel de ruido eléctrico aleatorio.
- Filtrar: se realiza con el objetivo de eliminar ruidos introducidos por interferencia eléctrica en la señal.
- Cuando el procesamiento de la señal es digital, el acondicionamiento corresponde a la conversión Analógica/Digital.

Procesamiento de la señal. Incluye el conjunto de transformaciones a que debe ser sometida la señal eléctrica, a fin de extraer de ella la información que se busca. El procesamiento de la señal suele contener muy diversas operaciones, ya sean lineales, no lineales, de composición de múltiples señales, o de procesado digital de las señales. Este último comúnmente se realiza a través de un microprocesador, microcontrolador, procesador digital de señales (DSP) o una computadora personal.

Dispositivo de despliegue. La información resultante del proceso de medida debe ser presentada de forma comprensible al usuario, o de forma integrada, para que pueda ser interpretada por un sistema supervisor automático. Los sistemas de presentación de información eléctrica analógica tradicionales, han sido: los indicadores de aguja, los registradores gráficos de papel y los tubos de rayos catódicos. Actualmente, los terminales alfanuméricos y gráficos basados en microcomputadoras o mediante software en computadoras personales, suelen ser el método más utilizado para presentar todo tipo de información.

Registro de la señal. Consiste en el almacenamiento permanente o temporal de las señales para su posterior análisis o supervisión. Esta operación es necesaria si el flujo de información que se adquieren supera la capacidad de procesamiento de que se dispone. El método tradicional de registro ha sido el basado en cinta magnética, ya sea a través de grabación analógica o utilizando codificación digital. Actualmente, los métodos de registro que se utilizan están basados en computadoras personales y medio extraíbles de almacenamiento masivo de información, como son: memorias flash, memorias USB, discos duros, CD-ROM, tarjetas SD/MMC/XD, entre otros.

2.3. Los Microcontroladores

Desde la invención del circuito integrado, el desarrollo constante de la electrónica digital ha dado lugar a dispositivos cada vez más complejos. Entre ellos los microprocesadores y los microcontroladores.

El microcontrolador es un circuito integrado que contiene todos los componentes de una computadora. Se emplea para controlar el funcionamiento de una tarea determinada y, debido a su reducido tamaño, suele ir incorporado en el propio dispositivo al que éste controla. Esta última característica es la que le confiere la denominación de "controlador empotrado o incrustado" (*embedded controller*). El microcontrolador es una computadora dedicada. En su memoria sólo reside un programa destinado a gobernar una aplicación determinada, sus líneas de entrada/salida soportan la conexión de los sensores y actuadores del dispositivo a controlar. Una vez programado y configurado el microcontrolador solamente sirve para gobernar la tarea asignada. Se dice que es la solución en un chip porque su reducido tamaño minimiza el número de componentes y el costo.

En la figura 2.2 se muestra la imagen de un microcontrolador, el cual, dispone normalmente de los siguientes componentes:

- Procesador o CPU (Unidad Central de Proceso).
- Memoria RAM para contener los datos.
- Memoria para el programa tipo ROM/EPROM/EEPROM/Flash.
- Líneas de E/S para comunicarse con el exterior.
- Diversos módulos para el control de periféricos (temporizadores, puertos serie y paralelo, convertidores Analógico/Digital, convertidores Digital/Analógico, etc.).
- Generador de impulsos de reloj que sincronizan el funcionamiento de todo el sistema.

Figura 2.2. Microcontrolador ATMega8 de la empresa ATMEL.

2.3.1. Clasificación

Podemos clasificar a los microcontroladores en dos grandes grupos, por su arquitectura interna y por su juego de instrucciones, a continuación se describen brevemente dichas clasificaciones.

1) Por su arquitectura interna: Debido a que los microcontroladores son un tipo especial de computadoras, se pueden clasificar de acuerdo a su arquitectura interna, lo cual implica la forma en que estos acceden a los recursos con los que dispone. La clasificación es la siguiente:

a) Arquitectura Von-Neumann:

- ✓ Tiene un único bus de datos para datos e instrucciones.
- ✓ El programa y los datos se almacenan en la misma memoria principal.
- ✓ Para ejecutar una instrucción, primero se busca la instrucción y a continuación el dato correspondiente (dos búsquedas consecutivas).
- ✓ El tamaño de la palabra de memoria está fijado por el *bus* que la comunica con la CPU.
- ✓ Su velocidad de ejecución es lenta.

En la figura 2.3, se muestra un diagrama de bloques de la estructura interna de un microcontrolador con arquitectura Von-Neumann.

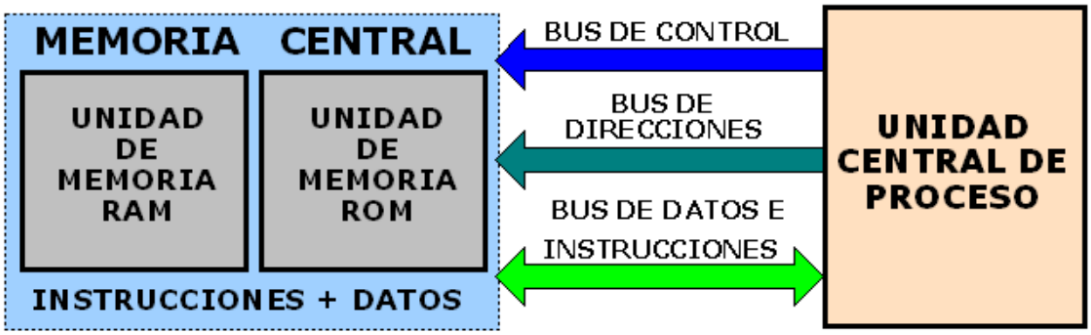


Figura 2.3. Arquitectura Von- Neumann.

b) Arquitectura Harvard:

- ✓ Tiene el bus de datos y bus de instrucciones separados, lo cual permite que existan dos memorias, una para datos y otra para programa.
- ✓ Permite búsquedas de instrucciones y datos simultáneas: se adelanta a realizar la búsqueda de la siguiente instrucción en paralelo con acceso a los datos de la instrucción en ejecución.
- ✓ El tamaño de las instrucciones no está relacionado con el de los datos, y por lo tanto puede ser optimizado para que cualquier instrucción ocupe una sola posición de memoria.
- ✓ Mayor velocidad de ejecución.

En la figura 2.4, se muestran los elementos que conforman a un procesador con arquitectura Harvard.

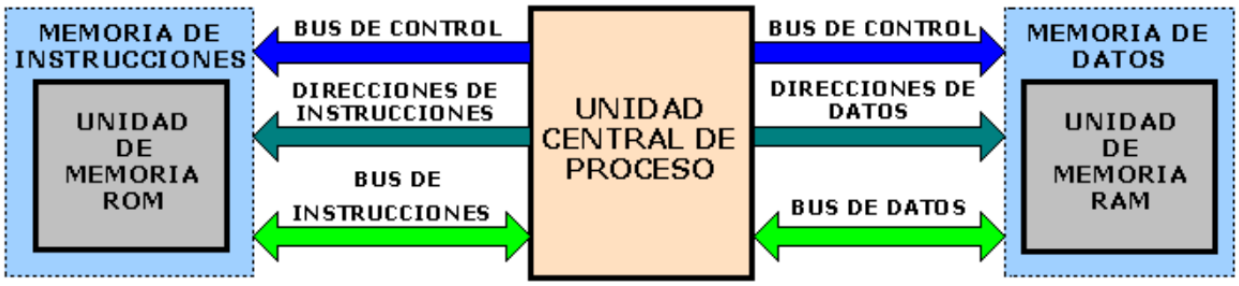


Figura 2.4. Arquitectura Harvard.

2) Por su juego de instrucciones: El juego de instrucciones está relacionado directamente con la unidad central de procesamiento o procesador, el cual, es el elemento más importante del microcontrolador y determina sus principales características, tanto a nivel hardware como software. Con base en ello, los microcontroladores, se pueden clasificar por el juego de instrucciones de su procesador en:

a) CISC: Un gran número de procesadores usados en los microcontroladores están basados en la filosofía CISC (*Complex Instruction Set Computer*). Disponen de más de 80 instrucciones máquina en su repertorio, algunas de las cuales son muy sofisticadas y potentes, requiriendo muchos ciclos para su ejecución. Una ventaja de los procesadores CISC es que ofrecen al programador instrucciones complejas que actúan como macros. La microprogramación es una característica importante y esencial de casi todas las arquitecturas CISC.[10]

b) RISC: Tanto la industria de los computadores comerciales como la de los microcontroladores están decantándose hacia la filosofía RISC (*Reduced Instruction Set Computer*). En estos procesadores el repertorio de instrucciones máquina es muy reducido y las instrucciones son simples y, generalmente, se ejecutan en un ciclo. La sencillez y rapidez de las instrucciones permiten optimizar el hardware y el software del procesador. Debido a que se tiene un conjunto de instrucciones simplificado, éstas se pueden implantar por hardware directamente en la CPU, lo cual elimina el microcódigo y la necesidad de decodificar instrucciones complejas.

2.3.2. Aplicación de los microcontroladores

En la actualidad, los microcontroladores son empleados en multitud de sistemas presentes en nuestra vida diaria, como son: juguetes, hornos de microondas, refrigeradores, televisores, computadoras personales, impresoras, módems, telefonía celular, dispositivos de audio portátiles,

[10] **Vega Luna, J. Ignacio et. al**, *"Arquitectura RISC vs CISC", UAM Unidad Azcapotzalco DCBI*, México.

etcétera; y otras aplicaciones en el ámbito de la ingeniería y las ciencias aplicadas, como la instrumentación electrónica, para el proceso de adquisición de datos.

En resumen, los microcontroladores se encuentran por todas partes:

- Sistemas de comunicación: en grandes automatismos como centrales y en teléfonos fijos, móviles, fax, etc.
- Electrodomésticos: lavadoras, hornos, refrigeradores, televisores, reproductores DVD, equipos de música, consolas de videojuegos, etc.
- Industria informática: Se encuentran en casi todos los periféricos; ratones, teclados, impresoras, escáner, etc.
- Automotriz: climatización, seguridad, frenos ABS, computadora del auto, etc.
- Industria: Automatización y control de procesos, sistemas SCADA.
- Sistemas de supervisión, vigilancia y alarma: ascensores, calefacción, aire acondicionado, alarmas de incendio, robo, etc.
- Otros: Instrumentación electrónica y científica, ingeniería biomédica, sistemas de autenticación de personalidad, sistemas de navegación, robótica, etc.

En la práctica, cada fabricante de microcontroladores oferta un elevado número de modelos diferentes, desde los más sencillos hasta los más poderosos. Es posible seleccionar la capacidad de las memorias, el número de líneas de E/S, la cantidad y potencia de los elementos auxiliares, la velocidad de funcionamiento, etc. Por todo ello, un aspecto muy destacado del diseño es la selección del microcontrolador a utilizar.

2.3.3. Tarjetas de desarrollo basadas en microcontroladores

Las tarjetas de desarrollo basadas en microcontroladores son una gran herramienta tanto en el aspecto didáctico, para la enseñanza de ingeniería electrónica, así como para el diseño de aplicaciones que utilizan a los microcontroladores como unidad central de procesamiento. Las tarjetas de desarrollo cumplen un importante rol dentro de la industria electrónica, permitiendo reducir los tiempos involucrados en el diseño de una solución, aumentando la confiabilidad y velocidad de fabricación de un prototipo y, en ocasiones, transformándose en la base del producto final mismo. Con estas tarjetas, el desarrollador puede concentrarse en afinar las prestaciones de su diseño, más que en implementar funcionalidades de bajo nivel, pues éstas ya han sido resueltas por los fabricantes de la plataforma. Una de las ventajas de utilizar tarjetas de este tipo, radica en que éstas cuentan con un buen número de módulos periféricos, ampliando el rango de aplicaciones de los microcontroladores y facilitando el proceso de probar los programas. Además de los módulos integrados en ellas, es posible utilizar sus terminales de E/S a conveniencia del usuario. Otro aspecto importante a considerar es que cuando se

cuenta con una tarjeta de desarrollo es posible eliminar el uso de un programador externo y que el entorno de desarrollo para la programación del microcontrolador asociado a ella, es más sencillo de utilizar.

Considerando la gran cantidad de microcontroladores disponibles en el mercado, existe una variedad aún mayor de modelos y versiones de placas de desarrollo, que se diferencian entre sí por las funcionalidades que incorporan y que compiten para ofrecer el mayor número de prestaciones, terminales de E/S, memoria, interfaces de comunicación, entre otras, y al menor costo posible, lo que las vuelve más atractivas para los usuarios y desarrolladores de aplicaciones electrónicas.

Los criterios para escoger una tarjeta de desarrollo están íntimamente ligados a las necesidades y perfil del usuario. Sin embargo, lo ideal es que la plataforma tenga los atributos necesarios para el proyecto en cuestión (velocidad del procesador, memoria disponible, consumo de energía, etc.) y que incorpore los circuitos y puertos para los dispositivos externos (sensores, módems GSM, Bluetooth, etc.) requeridos por el proyecto en cuestión, ya que esto permitirá un ahorro sustancial en el tiempo de desarrollo del producto definido por el usuario.

Entre las marcas de microcontroladores que más se incorporan en tarjetas de desarrollo fabricadas en el mundo se encuentran: *Microchip, Atmel, Freescale, Intel, Texas Instruments,* etcétera.

Las tarjetas de desarrollo también evolucionan a medida que avanza la tecnología que las sustenta. En este sentido, los cambios y tendencias que se manifiestan en el campo de la electrónica se reflejan en las prestaciones y capacidades de estas plataformas. En este sentido, las plataformas de desarrollo electrónico continúan abriendo el mundo de la electrónica a nuevos grupos de usuarios, permitiéndoles concentrarse en los aspectos del diseño con los que están más familiarizados y plasmar de manera rápida y sencilla sus ideas en un prototipo totalmente funcional, que posteriormente se transformará en un producto final.

2.4. Instrumentación virtual

El diseño de los instrumentos siempre se ha basado en el aprovechamiento de la tecnología del momento. En la antigüedad se usaron el capacitor, resistor e inductor variables, junto con el tubo de vacío, para construir los primeros instrumentos electrónicos. La tecnología del tubo de rayos catódicos permitió el diseño de los osciloscopios y los analizadores. Las computadoras, que son indispensables para el cálculo, procesamiento, captura y almacenamiento de datos y presentación de la información, han dado lugar, mediante la digitalización electrónica de las mediciones, a la integración de un nuevo tipo de instrumentos.

La introducción de las computadoras en el campo de la instrumentación inició como un simple acoplamiento con un instrumento individual, tal como un sensor, habilitando el despliegue de los datos medidos sobre un panel de instrumento virtual, mostrado mediante *software* sobre el monitor de la computadora, y conteniendo botones y otros medios para controlar la operación del sensor. Entonces, este *software* de instrumentación habilitó la creación de un instrumento físico simulado, teniendo capacidad de controlar los componentes del sensado físico de un sistema de control.

Las nuevas computadoras de propósito general son bastante rápidas para trabajar en tiempo real, y con la adición de *software* y *hardware* especializado cualquier computadora de laboratorio puede ser capaz de realizar lo que hasta hace pocos años solamente las computadoras especializadas podían lograr: el instrumento virtual.

Así, surge el instrumento virtual, un dispositivo que se compone de subunidades especializadas, de una computadora de propósito general y de *software*. Los instrumentos virtuales pueden ser simples o muy complejos. El usuario de un instrumento virtual puede cambiar su funcionamiento de acuerdo con sus necesidades, para poder abarcar un amplio rango de aplicaciones.

La principal ventaja de estos sistemas de instrumentación radica en que proporcionan un fácil desarrollo y reconfiguración para diferentes tareas, aunado a la estandarización de los componentes usados en su desarrollo.

Con los instrumentos virtuales, los ingenieros y científicos construyen sistemas de medición y automatización que se ajustan exactamente a sus necesidades (definidos por el usuario) en lugar de estar limitados por los instrumentos tradicionales de funciones fijas (definidos por el fabricante).

2.4.1. Instrumentación tradicional versus instrumentación virtual

Los instrumentos autónomos tradicionales, tales como osciloscopios y generadores de ondas, son muy caros y diseñados para llevar a cabo una o más tareas específicas definidos por el fabricante. Sin embargo, el usuario por lo general no puede extender o personalizar esas tareas. Debido a esta situación, los ingenieros y científicos cuyas necesidades, aplicaciones y requerimientos varían muy rápidamente, necesitan flexibilidad para crear sus propias soluciones. Con el uso de instrumentos virtuales. El usuario puede adaptarlo a sus necesidades particulares, sin necesidad de reemplazar todo el instrumento, dado que posee el *software* de aplicación instalado en la computadora y al amplio rango disponible de hardware para instalar en ella.

Instrumento tradicional. En un instrumento tradicional el bloque sensor contiene el transductor para detectar la variable de interés. El acondicionamiento se lleva a cabo mediante circuitos

electrónicos para adecuar la variable a un nivel que sea posible visualizar en un osciloscopio, pantalla, dial, registro en papel, etc., representando éstos el medio de despliegue de la señal.

Estos instrumentos tradicionales, por lo general, miden sólo algunas variables y no se puede modificar la arquitectura del dispositivo. Se pueden considerar como un instrumento 1 a 1.

La complejidad de la instrumentación tradicional, la convergencia tecnológica y los sistemas de mediciones y pruebas más flexibles, han conducido a los sistemas de instrumentación convencional o autónoma y a los sistemas de instrumentación inteligentes**,** también llamados virtuales, ambos basados en procesadores y en computadoras.

Los sistemas de instrumentación tradicionales se conciben como un conjunto de diversos instrumentos conectados por un *bus* común, comandado habitualmente por un sistema de control general. Comúnmente se utilizan para la realización de pruebas automatizadas de equipos especializados o de procesos.

Instrumento virtual. En este tipo de instrumentos, el sensor sigue siendo el medio de transducción para detectar el fenómeno físico, el acondicionamiento de señal la adecua a niveles apropiados para posteriormente realizar una conversión analógico-digital a través de tarjetas de adquisición de datos, equipos con interfaz GPIB o bien con USB y RS232 como medio de comunicaciones hacia la PC.

La ventaja de usar una PC en la instrumentación virtual radica en que se aprovechan sus capacidades de velocidad y procesamiento para manipular la señal (escalamiento, linealización, análisis, etc.) según se requiera y visualizarla en diversas representaciones como en el tiempo, en la frecuencia, en formato digital o continuo, entre otras, a través de interfaces gráficas y/o reportes impresos.

Así, los sistemas de instrumentación han evolucionado, convirtiéndose en sistemas de operación manual o automática, de medición y prueba, en el cual se tiene un control exacto de los parámetros involucrados.

Los instrumentos virtuales son dispositivos para la medición, generación y control de señales, presentando las salidas y respuestas en uno o en varios despliegues a través del monitor de la computadora, es decir, se considera un instrumento M a 1 (Muchas variables – un instrumento), en donde se puede utilizar solo la PC como instrumento virtual para visualización y despliegue de la información. La modularidad de estos sistemas les hace ser instrumentos muy flexibles.

En este sentido, los instrumentos virtuales forman sistemas de instrumentación en donde no sólo se puede medir las variables de interés, sino procesarlas, almacenarlas, presentar la información en diversos formatos para una mejor interpretación.

La gran disponibilidad y capacidades de los sistemas de cómputo han permitido cada vez más su incorporación en los procesos de adquisición de datos, de mediciones automatizadas y de simulación de procesos.

El esquema del sistema de instrumentación basado en PC, figura 2.6, haciendo uso de la instrumentación virtual, se compacta notablemente, ya que la PC integra los diferentes instrumentos de medición y generación de señales.

Figura 2.5. Sistema de instrumentación virtual.

Actualmente, muchos de los sistemas de instrumentación están basados en computadora, la cual resuelve todos los aspectos relativos al procesado de la señal, al registro, a la transferencia y a la presentación de la información. A estos equipos basados en computadora se les llama también *sistemas de instrumentación inteligente o* bien *sistemas de instrumentación virtual*.

En resumen, la principal diferencia entre los sistemas de instrumentación tradicionales y los virtuales, es que mientras que en los primeros los datos de medida son generados uno a uno y deben ser interpretados por el usuario, en los sistemas de instrumentación virtual se pueden registrar grandes cantidades de información de forma automática y presentarla de forma integrada y amigable al usuario.

Los sistemas de instrumentación virtual están basados en computadora y requieren el uso de transductores y circuitos que acondicionan la señal a los niveles adecuados, para ser codificada en las mejores condiciones para la conversión A/D; sin embargo, el procesado, elaboración y presentación del resultado de la medida, se realizan mediante *software*.

Una característica importante de resaltar es que, los sistemas de instrumentación virtual definidos por *software* permiten que los datos sin procesar desde el *hardware* estén disponibles para los usuarios, los cuales pueden definir sus propias mediciones e interfaz de usuario. Esta característica permite definir los sistemas de instrumentación por *software*, en donde los usuarios pueden realizar sus mediciones de forma personalizada, llevar a cabo esquemas de prueba específicos, mediciones para estándares nuevos o modificar el sistema si este requiere cambios (agregar instrumentos, canales o medidas). El *software* definido por el usuario puede ser utilizado en *hardware* autónomo de aplicaciones específicas, pero en realidad es ideal para emplearlo en *hardware* modular de uso general.

La combinación de *software* flexible, definido por el usuario, y los componentes de *hardware* escalables, es el núcleo de los sistemas de instrumentación virtual con base modular.

El *software* utilizado para la programación de los sistemas de instrumentación virtual es un lenguaje de programación gráfica, llamado **LabVIEW**, de la empresa norteamericana **National Instruments (NI).** En la figura 2.6 se muestra la interfaz gráfica de un instrumento virtual desarrollado bajo la plataforma LabVIEW.

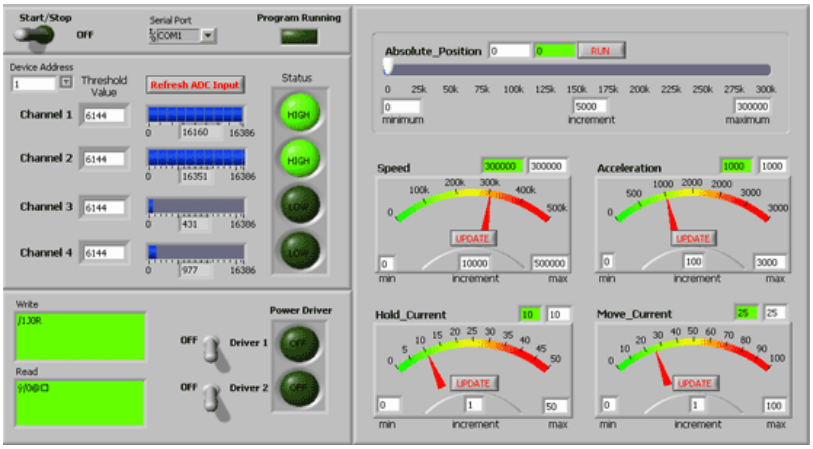


Figura 2.6. Instrumento virtual desarrollado bajo el entorno de LabVIEW.

2.5. El entorno de desarrollo LabVIEW

El software de NI, LabVIEW es un entorno de programación para el desarrollo de aplicaciones de propósito general, está basado en el paradigma de programación gráfica ó programación G, que utiliza íconos, terminales y líneas de unión en lugar de texto, lo cual ayuda a programar de manera muy intuitiva, y de esta manera facilita la creación de interfaces gráficas. Tal como aprender cualquier software de programación nuevo, aprender cómo programar en LabVIEW requiere saber cómo navegar

en el entorno. LabVIEW se encarga de gestionar los recursos de la computadora a través de un entorno sencillo, rápido y eficiente. De esta forma se reducen enormemente los tiempos de desarrollo al momento de realizar los programas. Entre sus objetivos están el reducir el tiempo de desarrollo de aplicaciones de todo tipo (no sólo en ámbitos de prueba, control y diseño) y el permitir la entrada al mundo de la informática a programadores no expertos.

Los programas creados en LabVIEW son llamados "instrumentos virtuales" (VI*s: Virtual Instruments*), debido a que su apariencia y operación es similar a la de instrumentos físicos reales, como osciloscopios y multímetros. De ahora en adelante, se referirá a los programas creados en LabVIEW como VIs.

Los ingenieros, científicos, estudiantes, principalmente lo utilizan para desarrollar aplicaciones como:

- Adquisición de datos
- Control de instrumentos
- Automatización industrial
- Sistemas de control supervisorio y adquisición de datos (SCADA)
- Diseño de controladores digitales
- Diseño con microcontroladores y FPGA
- Robótica, visión artificial y reconocimiento de patrones
- Procesamiento digital de señales
- Monitoreo de sistemas en tiempo real
- Procesamiento de información

2.5.1. Características principales

La principal característica de la plataforma LabVIEW es la facilidad de uso, válido para programadores profesionales como para personas con pocos conocimientos en programación, los cuales pueden hacer programas relativamente complejos, imposibles para ellos de hacer con lenguajes tradicionales. También es muy rápido hacer programas con LabVIEW y cualquier programador, por experimentado que sea, puede beneficiarse de él. Con LabVIEW pueden crearse programas de miles de VIs (equivalente a millones de páginas de código texto) para aplicaciones complejas, programas de automatizaciones de decenas de miles de puntos de entradas/salidas, etc. Incluso existen buenas prácticas de programación para optimizar el rendimiento y la calidad de la programación. Las principales características de LabVIEW son las siguientes:

- Entorno de desarrollo gráfico; desaparece el código en formato texto que estamos acostumbrados a utilizar. Con esto se consigue una forma de programación más intuitiva.

- Diseño de la interfaz gráfica del instrumento virtual, utilizando elementos como controles e indicadores numéricos, gráficas, etc.
- Herramientas convencionales para la depuración de los programas (VI's).
- Programación modular.
- Interfaces de comunicaciones.
- Capacidad de interactuar con otros lenguajes y aplicaciones. Visualización y manejo de gráficas con datos dinámicos.
- Tiempo Real estrictamente hablando.
- Sincronización entre dispositivos.

2.5.2. Programación en LabVIEW

Como se ha dicho es una herramienta de programación gráfica, esto quiere decir que los programas no se escriben, sino que se enlazan bloques (SubVI), facilitando su comprensión y una estructura ordenada de la aplicación. En general, un programa se divide en *Panel Frontal* y *Diagrama de bloques*.

El Panel Frontal, figura 2.7, es la interfaz con el usuario, en él se definen los *controles* e *indicadores* que se muestran en pantalla; esta es la forma final de la aplicación.

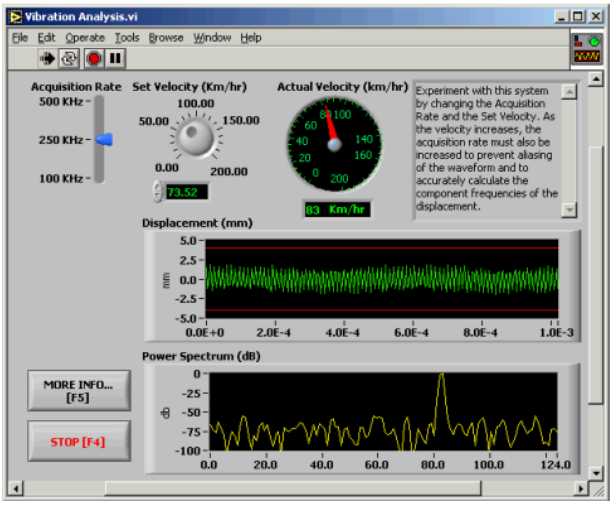


Figura 2.7. Panel frontal de un Instrumento virtual desarrollado con LabVIEW.

El Diagrama de Bloques, figura 2.8, es donde se realiza la programación (en forma gráfica), donde se define su funcionalidad, aquí se colocan iconos (librerías dinámicas) que realizan una determinada tarea y se interconectan para cumplir con los requerimientos de la aplicación.

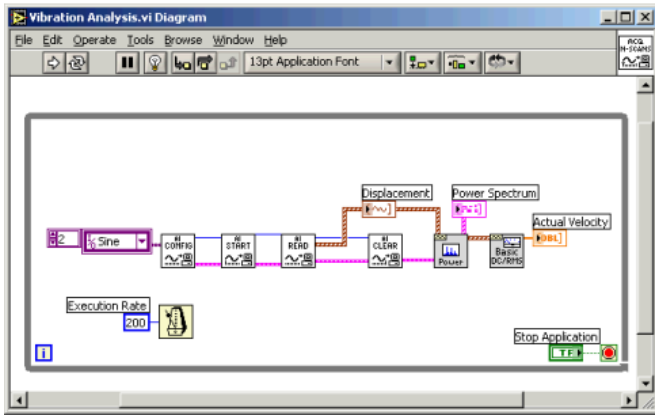


Figura 2.8. Diagrama de bloques de un Instrumento virtual desarrollado con LabVIEW.

En conclusión podemos decir que, la instrumentación virtual está *motorizada* por la siempre creciente tecnología computacional que le ofrece al usuario el poder de crear y definir su propio sistema basado en un marco de trabajo abierto. Este concepto no sólo le asegura que el trabajo será utilizable en el futuro sino que también la provee la flexibilidad de adaptarlo y extenderlo a medida que cambian las necesidades. LabVIEW fue diseñado teniendo en mente a los científicos e ingenieros, provee herramientas poderosas y un medio ambiente de desarrollo familiar creado específicamente para el diseño de instrumentos.

2.6. Sistema de Posicionamiento Global (GPS)

Sistema de Posicionamiento Global *(GPS: Global Position System)*, es un sistema que puede trabajar con medida directa, el cual tienen cobertura en cualquier parte del mundo, ya sea por el día o por la noche. Fue puesto en funcionamiento desde 1973 y se desarrolló a partir de los satélites de la constelación *NAVigation Satellite Timing And Ranging (NAVSTAR).* Fue desarrollado por el Departamento de Defensa de los Estados Unidos y se lanzó el primer satélite el 22 de Febrero de 1978.

El GPS es un sistema de posicionamiento por satélites, diseñado para apoyar los requerimientos de navegación y posicionamiento precisos, con fines militares. En la actualidad es una herramienta versátil para aplicaciones de navegación y determinación de posiciones en tierra, mar y aire.

La constelación GPS está formada por 24 unidades operacionales (sin incluir satélites de respaldo) con órbitas circulares, ubicados a una distancia de 26 560 [km] de la Tierra y que se desplazan a una velocidad aproximadamente de 4 [km/s]; sin embargo, su posición instantánea puede

estimarse con un error de unos cuantos metros. Los satélites están organizados en seis planos orbitales con cuatro satélites por órbita, como se aprecia en la figura 2.9.

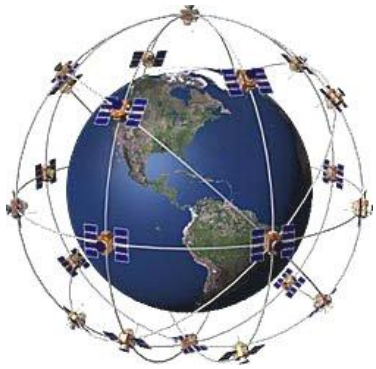


Figura 2.9. Constelación de satélites GPS.

En la práctica, actualmente los GPS constituyen un nuevo estándar internacional que permite determinar ubicaciones y distancias. Asociado a otras tecnologías permite también localizar objetos y personas.

El GPS está integrado por tres segmentos o componentes, que se describen a continuación:

1. **Segmento Espacial:** Los satélites del GPS transmiten dos señales de radio de baja potencia, llamadas "L1" y "L2". Cada señal GPS contiene tres componentes de información: un código pseudoaleatorio, los datos de efemérides de satélite y datos de almanaque. El código pseudoaleatorio identifica al satélite que transmite su señal. Los datos de efemérides de satélite proporcionan información sobre la ubicación del satélite en cualquier momento. El almanaque contiene información sobre el estado del satélite y la fecha y hora actuales. Para cada satélite, el tiempo es controlado por los relojes atómicos a bordo que son cruciales para conocer su posición exacta.

2. **Segmento de Control:** Es una serie de estaciones de rastreo, distribuidas en la superficie terrestre que continuamente monitorea a cada satélite, analizando las señales emitidas por estos y a su vez, actualiza los datos de los elementos y mensajes de navegación, así como las correlaciones de reloj de los satélites. Las estaciones se ubican estratégicamente cercanas al plano ecuatorial y en todas se cuenta con receptores con relojes de muy alta precisión.

3. Segmento de Usuario: Lo integran los receptores GPS que registran la señal emitida por los satélites para el cálculo de su posición, tomando como base la velocidad de la luz y el tiempo de viaje de la señal, así se obtienen las pseudodistancias entre cada satélite y el receptor en un

tiempo determinado, observando al menos cuatro satélites en tiempo común; el receptor calcula las coordenadas X, Y, Z y el tiempo.

2.6.1. Funcionamiento de los receptores GPS

El GPS es un sistema de radiolocalización que, además de la posición, también permite conocer la velocidad del movimiento, la orientación del desplazamiento y la traza del recorrido que se ha efectuado. El GPS proporciona un sistema de navegación por toda la superficie de la Tierra utilizando señales de radio.

Las señales de radio son emitidas de forma ininterrumpida por un conjunto de satélites. Estas señales contienen datos relativos a la posición del satélite en el espacio y a la hora en un formato internacional *(UTC: Universal Time Coordinated).*

Para el buen funcionamiento del sistema, cada satélite debe mantener una órbita extremadamente precisa y estable, es decir que el receptor y el satélite tengan una línea de vista correcta, con el objetivo de que la señales emitidas por el satélite puedan ser captadas por el receptor.

El receptor GPS es un dispositivo electrónico móvil, en su mayoría portátil, que recibe las señales emitidas por los satélites y que al procesar la información que contienen dichas señales, calcula la posición en la que el receptor se encuentra.

El principio funcional del GPS se basa en medir el tiempo empleado por la señal transmitida por los satélites en llegar al receptor, y este intervalo de tiempo es multiplicado por la velocidad de la luz para obtener la distancia satélite-receptor.

El receptor GPS calcula su posición realizando una triangulación; para poder realizar un cálculo preciso se debe contar como mínimo con señales provenientes de cuatro satélites distintos. Debido al número de satélites (24) y el tiempo que tardan en dar la vuelta a la Tierra (12 [h]), desde cualquier parte del planeta siempre se encontraran visibles seis satélites. Cabe mencionar que aunque con cuatro satélites es suficiente para poder realizar el cálculo, recibir la señal de seis satélites aporta mayor precisión.

El receptor GPS calcula su distancia con respecto a cada satélite a partir de la relación entre distancia, velocidad y tiempo. La distancia es igual a la velocidad multiplicada por el tiempo, la velocidad es conocida, pues la señales de radio viajan a la velocidad de la luz, 300 000 [km/s] y el tiempo se puede calcular, pues el receptor compara el dato sobre la hora en que el satélite emitió la señal con la hora actual, que obtiene de su propio reloj.

De lo anterior se deducen dos consideraciones importantes. La primera es que, es fundamental que los relojes de los satélites y el receptor estén sincronizados y sean precisos. La segunda es que, una señal que llegue rebotada, introducirá un error en el cálculo.

Fórmula de Haversine

Calcular la distancia entre dos puntos sobre un plano podría llegar a ser relativamente sencillo; sin embargo, cuando estos dos puntos los ubicamos sobre la esfera terrestre, es decir, lo que pretendemos es calcular la distancia lineal entre dos posiciones dadas (coordenadas de latitud y longitud, dados por un receptor GPS), la situación se torna complicada.[11]

Para el cálculo de la distancia entre ambas posiciones debemos contemplar la curvatura terrestre. Es aquí donde entra en escena la Fórmula de Haversine, ampliamente utilizada en receptores GPS comerciales, para determinar la distancia recorrida por el usuario, la cual en términos matemáticos, se expresa mediante la ecuación (2.1):

$$d = R * c \quad \text{..... (2.1)}$$

Donde:

d: distancia entre los dos puntos

R: radio de la Tierra

$c = 2 \cdot \text{atan } 2 \left[\sqrt{a}, \sqrt{1-a} \right]$

$a = sin^2\left(\frac{\Delta\, lat}{2}\right) + \cos(lat1) \cdot \cos(lat2) \cdot sin^2\left(\frac{\Delta\, long}{2}\right)$

$\Delta\, lat = lat2 - lat1$; diferencia de latitudes

$\Delta\, long = long2 - long1$; diferencia de longitudes

Al utilizar la fórmula de Haversine se debe tener en cuenta que:

- ✓ La fórmula asume que la Tierra es completamente redonda, con lo que se espera una ligera tasa de error (0.3 %) que se podría llegar a despreciar.

[11] **Kaula, William M**., "THEORY OF SATELLITE GEODESY", Blaisdell Publishing Co., 1996.

- ✓ Para el cálculo de la distancia es necesario considerar que el radio de la tierra tiene dos valores distintos debido a que no posee una forma totalmente esférica. Dichos valores son: 6378 [km] para el radio ecuatorial y 6357 [km] para el radio polar.

De acuerdo a la ubicación de México en el globo terráqueo, el radio ecuatorial será el valor empleado para el cálculo de la distancia recorrida por el vehículo mediante la fórmula de Haversine.

2.6.2. Confiabilidad de los datos

La mayoría de los receptores GPS, en la medición de la posición, tienen una precisión de entre 10 y 20 metros, dependiendo de de la ubicación de los satélites disponibles para la transmisión de señales hacia el receptor; sin embargo, algunos receptores utilizan un método llamado *Wide Area Argumentation System* (WAAS) que coordina la posición con un segundo receptor terrestre fijo. Este receptor proporciona una posición con una diferencia de un metro a la posición exacta y consiste en una red de torres receptoras que corrigen la señal de los satélites y la envían al receptor GPS.

La posición calculada por un receptor requiere instantáneamente la posición del satélite y un retraso medido de la señal recibida. Al introducir el retraso, el receptor compara una serie de bits provenientes del satélite, lo cual introduce un retraso de tiempo de aproximadamente 10 [ns].

Fuentes de Error

Debido a que las unidades receptoras GPS requieren el contacto con cuatro satélites al menos para calcular la posición, la precisión se ve afectada por la ubicación de éstos y por las condiciones atmosféricas a través de las cuáles debe desplazarse la señal. Además, los obstáculos naturales como las montañas y zonas forestales o alto edificios en las ciudades pueden interferir las señales, por esta razón, la precisión es mayor en lugares abiertos. En la tabla 2.1 se describen los errores más comunes en receptores GPS.

FUENTE DE ERROR	*DESCRIPCIÓN*
Error de reloj	Entre menos sincronizados estén los relojes de los satélites y el receptor, menos precisan serán las mediciones.
Retrasos por Ionósfera y Tropósfera	La densidad de la atmósfera causa retrasos que afectan la medición.
Señales Múltiples	Ocasionado por el rebote de la señal en edificios u objetos altos.
Errores Orbitales	Es la imprecisión del receptor para obtener la localización que el satélite transmite.
Número de satélites visibles	Cuantos menos satélites pueda ver el receptor GPS, más débil será la señal de sincronización.
Alineación Satelital	Si los satélites están mal alineados para realizar la triangulación, los datos pierden confiabilidad.

Tabla 2.1. Fuentes de error en receptores GPS.

2.6.3. Protocolo NMEA 0183

La *National Marine Electronics Association* (NMEA), es una asociación conformada por: fabricantes, distribuidores, comerciantes, instituciones educativas y demás interesados en sistemas electrónicos usados en el ámbito de la navegación marítima.

En la década de los 80 NMEA creó un protocolo único para el intercambio de datos digitales entre los sistemas de instrumentación marítimos, denominado NEMA 0183. Dicho estándar es muy aceptado por la mayor parte de los receptores GPS del mundo.[12]

Los dispositivos cuyas señales que generalmente utilizan el protocolo NMEA son:

- ✓ Receptores GPS
- ✓ Compás magnético
- ✓ Radares
- ✓ Ecosondas
- ✓ Sensores de velocidad
- ✓ Instrumentos meteorológicos

[12] www.nmea.org

- ✓ Sistemas de navegación integrados
- ✓ Comunicaciones satelitales

El protocolo NMEA 0183 también establece una interfaz eléctrica para su conexión con otros dispositivos como microcontroladores o computadoras personales. Dicha interfaz es de tipo serial, bajo el estándar RS-232. Las especificaciones de la comunicación se muestran en la tabla 3.2.

PARÁMETRO	*VALOR*
Baudaje	4800 bauds
Numero de bits de datos	8
Bits de parada	1
Paridad	Ninguna
Handshake	Ninguna

Tabla 2.2. Parámetros de comunicación del protocolo NMEA 0183.

Todos los datos son transmitidos en tramas de caracteres de tipo *American Standard Code for Information Interchange* (ASCII), cada sentencia comienza con el carácter "$" y termina con un fin de línea, <CR> <LF> (*CR: Carriage Return, LF: Line Feed*). Los primeros dos caracteres después del "$" identifican al equipo, en el caso de los receptores GPS, se encuentran los caracteres "GP" y lo siguientes tres caracteres definen el identificador del tipo de sentencia que se está enviando. Los tres tipos de sentencias NMEA que existen son, los de envío (*Talker Sentences*), los de origen del equipo (*Propietary Sentences*) y los de consulta (*Query Sentences*). Los datos están delimitados por una coma y al final de la trama se encuentra una suma de verificación (*Checksum*) para corroborar la transmisión.

A continuación se describen brevemente las sentencias de consulta más utilizadas en receptores GPS y que serán utilizadas para la sincronización y adquisición de datos por parte del *CYCLE-DAQ.*

GGA (Fix information) - Estatus del receptor. El mensaje GGA incluye el tiempo UTC, la posición y datos relacionados al estado del receptor.

GGL (Lat/Lon data) - Posición geográfica. Latitud/Longitud. El mensaje GLL contiene los datos de latitud, longitud, el tiempo UTC y el estatus del receptor.

GSV (Detailed Satellite data) - Satélites visibles. El mensaje GSV enumera los satélites visibles por el receptor.

RMC (recommended minimum data for gps) - Información mínima de navegación. Este mensaje contiene la hora, fecha, posición, dirección y velocidad del receptor GPS.

VTG (Vector track an Speed over the Ground) - Rastreo bien hecho y velocidad en tierra. Este mensaje transmite el rastreo actual y la velocidad del receptor GPS.

ZDA (Date and Time) - Hora y fecha. El mensaje ZDA contiene al tiempo UTC.

2.7. Memorias Flash *Secure Digital*

Las memorias de almacenamiento masivo se caracterizan por dispositivos de bajo costo y que trabajan a bajas velocidades, y las memorias semiconductoras o de estado sólido, más caras pero con la ventaja de ser más veloces. En las computadoras personales comúnmente se utilizan las memorias *Read Only Memory* (ROM), las cuáles permiten un almacenamiento no volátil y son usadas en el sistema de arranque de la computadora. Aunque este tipo de memoria se concibió para que su contenido no fuera modificado, actualmente existen diferentes tipos de memorias ROM en las cuales su contenido es modificable. Un ejemplo de estas memorias es la *Electric Erasable Programmable Read Only Memory (*EEPROM*)*. Las memorias EEPROM son borradas y programadas eléctricamente byte a byte a través de *software.*

2.7.1. Memorias Flash

Las memorias flash son un tipo específico de EEPROM, que son borradas y programadas por bloques de bytes. Debido a sus características, como el bajo costo y su gran velocidad de almacenamiento, se han convertido en la tecnología dominante en aplicaciones en donde se requiere gran capacidad de almacenamiento de información. Permiten que múltiples posiciones de memoria sean escritas o borradas en una misma operación de programación mediante impulsos eléctricos, caso contrario a lo que sucede con memorias anteriores que sólo permite escribir o borrar una única celda cada vez, como la memoria EPROM. Por ello, la memoria flash permite funcionar a velocidades muy superiores cuando los sistemas emplean lectura y escritura en diferentes puntos a la vez. Las aplicaciones más habituales son: USB, PC *Card*, tarjetas de memoria flash.

En cuanto a las características que ofrecen este tipo de memorias cabe destacar su gran resistencia a los golpes, son de bajo consumo y de reducido tamaño.

En lo referente a defectos hay que mencionar que sólo permite una cantidad finita de escrituras y borrados, generalmente entre 10,000 y un millón, dependiendo de la celda.

Actualmente, las memorias Flash que podemos encontrar, vienen en conjunto con microcontroladores embebidos, que se encargan de las tareas requeridas, ofreciendo la posibilidad de trabajar con sectores y de utilizar sólo una fuente de alimentación.

Por sus características las memorias flash ocupan gran parte del mercado de computadoras personales, teléfonos celulares, electrodomésticos, equipo médico, video juegos, cajas negras de aviones y en general de todo dispositivo electrónico que requiera almacenamiento de datos. En las aplicaciones multimedia las memorias flash se han convertido en la tecnología emergente frente a los métodos magnéticos y ópticos de grabación.

Una de las formas más comunes en la que podemos encontrar memorias flash es en las tarjetas de memoria. Existen muchos tipos de formatos de tarjetas de memoria que compiten y son incompatibles (casi una por fabricante). En la tabla 2.3 se muestran los formatos más comunes de tarjetas de memoria flash que existen.

Formato	Sigla	Capacidad máxima
PC Card	PCMCIA	8 GB
Compact Flash I	CF-I	12 GB
Start Media	SM/SMC	128 MB
Memory Stick	MS	128 MB
Memory Stick Duo	MSD	256 MB
Memory Stick PRO Duo	MSPD	16 GB
Memory Stick Micro M2	M2	32 GB
MultiMedia Card	MMC	4 GB
Reduce Size Multimedia Card	RE-MMC	2 GB
MMC micro Card	MMC micro	2 GB
Secure Digital Card	SD	2 GB
miniSD Card	miniSD	4 GB
microSD Card	microSD	16 GB
SD Card High Capacity	SDHC	32 GB
Xd-Picture Card	Xd	2 GB

Tabla 2.3. Formatos y capacidad de memorias flash.

2.7.2. Especificaciones de las memorias SD Card

Para conocer las características y especificaciones técnicas de las memorias *Secure Digital* (SD Card), se hizo uso de un compendio simplificado llamado *simplified Version of phisical layer specification,* el cual se puede obtener de la página de internet de la *SD Association.*

La tecnología y el estándar de las tarjetas de memoria *Secure Digital* han sido desarrolladas por el grupo SD, el cual tiene como participantes a las compañías Panasonic, SanDisk y Toshiba, quienes comparten derechos de autoría con la asociación SD Card.

Las dimensiones físicas, interfaz eléctrica y protocolo de comunicación, son parte de las especificaciones de la SD Card. Las principales diferencias entre las distintas tarjetas disponibles en la actualidad son: el consumo de energía, las dimensiones y su manipulación. La interfaz eléctrica de la SD Card es sencilla, su consumo de energía es de 100[mA] cuando se encuentra en actividad y su tamaño reducido es una cualidad que le permite tomar ventaja ante los competidores fabricantes de tarjetas de memoria.

Básicamente cuando se habla de memorias SD se pueden distinguir tres tipos de éstas (SD, miniSD y microSD), las cuales se diferencian por sus dimensiones principalmente, ya que su principio de funcionamiento es el mismo. En la figura 2.10 se muestran los tres tipos de SD CARD y sus dimensiones.

Figura 2.10. Tipos de SD CARD y sus dimensiones.

Para establecer la comunicación con memorias SD existen dos modos, estos son, el *SD Bus Mode* y el *SPI Bus Mode. E*l primero es el protocolo de comunicación propio de la SD CARD, utiliza nueve líneas para la comunicación: cuatro para transmisión de datos en forma paralela, una para la señal de reloj y tres más para la alimentación de la tarjeta. El modo serie o modo SPI, es un protocolo

de comunicación secundario, usa un extracto del protocolo oficial de la SD CARD y una parte del total de comandos. Esta restricción de comando no es de gran trascendencia, puesto que el modo SPI no los utiliza. Para el desarrollo del equipo *CYCLE-DAQ* se usa este último protocolo, el cual se describe detalladamente en el capítulo siguiente.

De acuerdo con las especificaciones utilizadas en el desarrollo de este proyecto, llamadas *Physical Layer Simplified Specification Version 2.00,* éstas dividen a las SD CARD de distintas formas, para el fin de nuestro desarrollo es de suma importancia la clasificación referida a la capacidad de almacenamiento. Bajo este contexto las memorias SD se dividen en: SD estándar (SD) y SD *High Capacity* (SDHC).

Para poder comprender la capacidad de almacenamiento de las memorias SD es necesario conocer las unidades que se utilizan para representar la cantidad de información almacenada. Las computadoras guardan la información digital en potencias de dos, ya que utilizan como base el sistema numérico binario, pero los usuarios de las computadoras comúnmente preferimos la numeración decimal para expresar cantidades en este caso de información.

En el caso de la información, la unidad de medida de la información es el bit, pero por cuestiones de utilidad se utiliza el "Byte", el cual representa a un grupo de 8 bits. Puede abreviarse como b ó B, pero aún no se ha estandarizado su forma de representarlo, por lo que en este trabajo se utiliza la B para referirnos al Byte.

Al igual que las demás unidades de medida, para el Byte se utilizan múltiplos decimales para determinar las cantidades. Sin embargo, un Kilobyte no es exactamente 1000 Bytes, sino 1024 Bytes. Hay un problema con respecto al uso de los prefijos como el Kilo y Mega, ya que estos no coinciden con el Sistema Internacional de Unidades de medida (SI). Para el sistema internacional de medida 1 Kilo = 1000, mientras que en informática 1 KiloByte = 1024 [B]. Por tal motivo se el *Institute of Electrical and Electronics Engineers* (IEEE) ha propuesto una nueva nomenclatura para evitar lo anterior, dejando a un lado que Mega signifique 1000 Kilos como siempre ha sido y creando un nuevo prefijo basado en las letras "bi" que indicaría "binary", ello para usarse solamente en algunos ámbitos de la informática. Por ejemplo el prefijo Mega se convierte en Mebi. Así un Megabyte será 10^6 bytes y un Mebibyte será 2^{20} bytes.

De acuerdo con las especificaciones, las memorias SD soportan capacidades hasta de 2[GB], las memorias SDHC soportan capacidades mayores a 2[GB] y hasta 32[GB]. En la primera versión simplificada de las especificaciones de las SD CARD, se establecía que dichas tarjetas de memoria soportaban capacidades menores a 2[GB]. Al aparecer la segunda versión simplificada en 2006, se establece que las memorias SD y SDHC pueden soportar capacidades de hasta 32[GB]. En 2009, las

más recientes especificaciones se pusieron alcance de los usuarios y comprenden tarjetas con capacidades mayores a 32[GB] y hasta 2[TB].

En la tabla 2.4, se muestra un resumen de los parámetros y capacidades en los distintos tipos de SD CARD.

Tipo de tarjeta	Año de aparición	Capacidad limite	Velocidad de escritura	Formato de archivo
SD	2000	4 [GB]	0.9-20 [MB/s]	FAT16
SDHC	2006	32 [GB]	2-40 [MB/s]	FAT32
SDXC	2009	2 [TB]	max 300 [MB/s]	exFAT

Tabla 2.4. Parámetros de SD CARD.

La mayoría de las tarjetas SD disponibles en el mercado son vendidas ya pre-formateadas, con un sistema de archivos basado en tablas de memoria. La popularidad de este sistema de archivos permite que la tarjeta de memoria pueda ser accesada en computadoras personales y dispositivos electrónicos mediante un lector de tarjetas SD. A continuación se explica detalladamente dicho sistema de archivos.

2.7.3. Sistema de Archivos FAT

Un sistema de archivos es la forma en que el Sistema Operativo (SO) organiza la información en unidades de almacenamiento masivo para su grabación y su posterior recuperación. La mayoría de los SO utiliza su propio sistema de archivos y permiten que éstos sean compatibles con otro. A nivel mundial el SO *Windows* de *Microsoft* es el más utilizado y utiliza el sistema de archivos *File Allocation Table* (FAT) para el manejo de información.

FAT es un sistema de archivos que se encuentra disponible en la mayoría de las computadoras personales y tarjetas de memoria, por su facilidad de uso. El sistema de archivos FAT es técnicamente sencillo y soportado por todos los sistemas operativos disponibles en computadoras personales, lo cual lo convierte en un formato útil para las tarjetas de memoria flash y como una conveniente manera para almacenar información.

En las memorias SD CARD, como en la mayoría de los medio de almacenamiento, los datos se almacenan en bytes, éstos se agrupan en sectores, generalmente de 512 bytes, los cuales, al agruparse entre 1 y 64 conforman las clústeres.

El sistema de archivos FAT se caracteriza por la tabla de asignación de archivos, la cual es una lista de valores digitales que describen la asignación de los clústeres cuando la unidad de almacenamiento es particionada. Para proteger el volumen, se conservan dos copias de la FAT por si una de ellas resulta dañada. Además, las tablas de asignación de archivos y el origen del cual se partirá para comenzar el almacenamiento de información, llamado directorio raíz, deben almacenarse en una ubicación fija para que se puedan encontrar correctamente los archivos de inicio del sistema.

En un disco formateado con FAT se asignan clústeres, los cuales agrupan una cierta cantidad de sectores y cuyo tamaño está determinado por el tamaño del dispositivo de almacenamiento. Para el sistema FAT un clúster es la unidad mínima de almacenamiento que se puede asignar a un archivo. Cuando se crea un archivo, se crea una entrada en el directorio y se establece el primer número de clúster que contiene datos. Esta entrada de la tabla FAT indica que éste es el último clúster del archivo o señala al clúster siguiente.

Actualmente se utilizan dos tipos de archivos FAT: FAT16 y FAT32. La diferencia entre éstos radica en el tamaño en bits de las entradas en la región FAT en la unidad de almacenamiento; esto es, para FAT16 hay 16 bits en una entrada FAT, mientras que para FAT32 hay 32 bits.

FAT 16

El sistema de archivos FAT16 es un sistema de 16 bits. Esto implica que las direcciones de clúster no pueden ser mayores a 16 bits. El número máximo de clústeres al que se puede hacer referencia con el sistema FAT es, por consiguiente, 2^{16} (65536) clústeres. Ahora bien, ya que un clúster se compone de un número fijo (4, 8, 16, 32,...) de sectores de 512 bytes contiguos, el tamaño máximo de la partición FAT se puede determinar multiplicando el número de clústeres por el tamaño de un clúster. Con clústeres de 32Kb, el tamaño máximo de un archivo en este formato es por lo tanto de 2 [GB].

FAT32

Mantiene la misma estructura de sectores y tablas, pero disminuye el tamaño de los sectores. Los valores de los clústeres son representados por números de 32 bits, de los cuales 28 son usados para el número de clúster, por lo cual se tienen aproximadamente 268 millones de clústeres. En general admite unidades de almacenamiento de hasta 2 [TB] de tamaño, como es el caso de discos duros. Utiliza clústeres menores (de 4Kb a 8Kb), lo que significa entre un 10 y un 15 % de mejora en el uso del espacio con respecto a unidades grandes con sistemas de archivos FAT o FAT16. El tamaño de máximo para una archivo en un sistema FAT32 es de 4 [GB].

En la tabla 2.5 se muestra una comparativa entre los formatos de archivos según el tamaño del clúster y el tamaño de la partición correspondiente a cada sistema.

Tamaño del clúster	FAT 16	FAT 32
512 [B]	32 [MB]	64 [MB]
1 [KB]	64 [MB]	128 [MB]
2 [KB]	128 [MB]	256 [MB]
4 [KB]	256 [MB]	8 [GB]
8 [KB]	512 [MB]	16 [GB]
16 [KB]	1 [GB]	32 [GB]
32 [KB]	2 [GB]	2 [TB]

Tabla 2.5. Sistema de Archivos FAT.

En el sistema de archivos FAT, el primer sector físico es el *Master Boot Record* (MBR) o registro de arranque maestro. Este registro contiene la información sobre las divisiones lógicas de la tarjeta de memoria, denominadas particiones. Para el caso particular de las memorias SD CARD, éstas tienen una partición activa, es decir, todo el espacio de almacenamiento se realiza dentro de una sola partición, y no en varias como puede ocurrir en los discos duros de las computadoras personales.

El sistema de archivos FAT está compuesto por cuatro regiones:

Región 0. Región reservada
Región 1. Región FAT
Región 2. Región del directorio raíz (no existe para FAT32)
Región 3. Región de datos, directorios y archivos.

En la Región 0, el primer sector de la partición es el *boot sector* o sector de arranque. Contiene la información básica sobre el sistema de archivos, la cual es proporcionada por el SO cuando la tarjeta es formateada y sirve para tener acceso al medio de almacenamientos, es decir, leer o escribir en él.

La Región 1, o también llamada Región FAT, indica cómo están distribuidos los clústeres en la región de los datos. Cada clúster es identificado por un número con el que se crea una entrada en la región FAT. Por lo general hay dos copias en esta región, una de las cuales sirve como respaldo en caso que los datos puedan estar dañados.

La Región 2, o directorio raíz, sólo se utiliza en FAT16. No es utilizado en FAT32 debido a que en éste sistema el directorio no tiene una zona especial, si no que es una cadena de clústeres que actúa como cualquier otro archivo y puede ser colocado en cualquier lugar de la región de datos. El directorio guarda información de los datos que se almacenan en la tarjeta de memoria, esto es, archivos y subdirectorios creados por el usuario para almacenar la información. El directorio nos dice en donde comienza un archivo, de acuerdo con el primer número del clúster del archivo.

Las tres regiones descritas anteriormente forman el área del sistema, lo cual implica que la capacidad de almacenamiento del dispositivo, y que sirve para guardar los datos, no sea el total de la capacidad del de la memoria.

Finalmente, la Región 3, o de datos, es el espacio de memoria que queda disponible para el usuario, en ella se almacenan los archivos y para el caso de FAT32 también se almacena el directorio.

Para comprender mejor el funcionamiento de los sistemas de archivos anteriormente mencionados, en la figura 2.11, se muestran los mapas de memoria correspondientes a cada uno de ello y de esta manera las diferencias entre ambos.

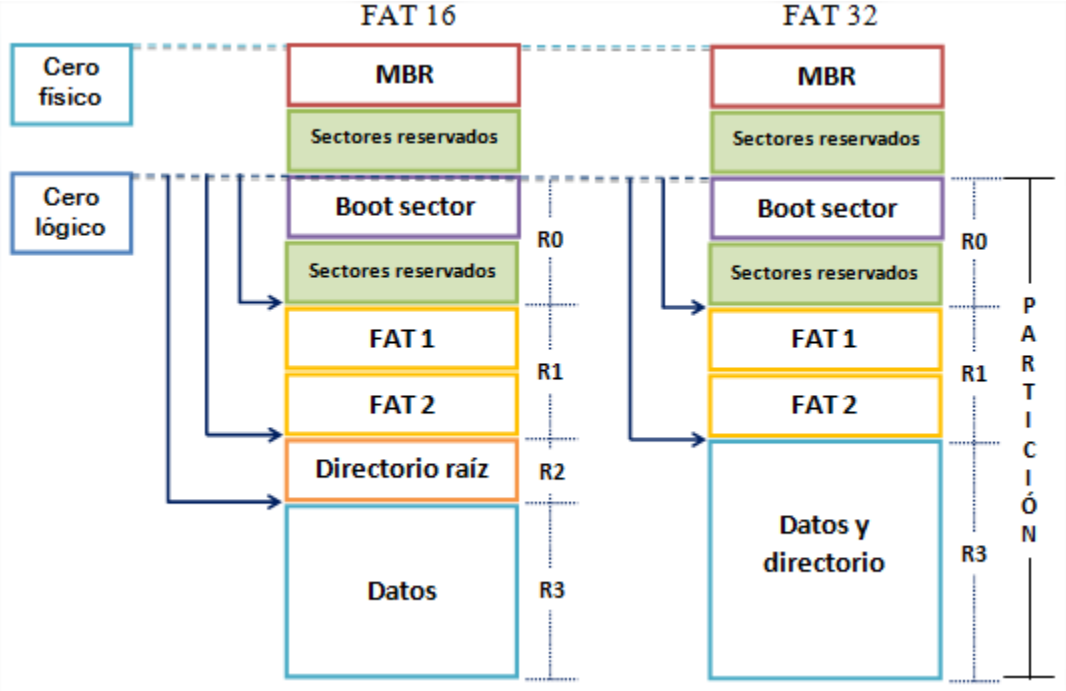


Figura 2.11. Mapas de memoria FAT16 y FAT32.

2.8. Bancas Analizadoras de Gases

Las bancas analizadoras de gases son instrumentos que se utilizan para medir la concentración de los gases de escape producto de la combustión que se realiza en el motor de un automóvil. El análisis de la muestra de gas indicará si ésta contiene contaminantes en exceso o si el motor se encuentra operando mecánicamente bien. Estas mediciones en el tubo de escape pueden ayudar a determinar el comportamiento del motor, el rendimiento, el encendido, el sistema de combustible y el control de emisiones.[13]

Una función principal de los analizadores de gases es corroborar que la combustión en el motor haya sido adecuada, pues cuando la mezcla entre combustible y aire no se efectúa en las proporciones correctas, existen perjuicios en contra del rendimiento del motor del automóvil. Los analizadores miden las concentraciones de cinco principales gases considerados contaminantes para el medio ambiente, éstos son: hidrocarburos (HC), monóxido de carbono (CO), oxígeno (O_2), dióxido de carbono (CO_2) y óxidos de nitrógeno (NOx).

Aunque existen diferentes tipos de analizadores, generalmente éstos se componen de cuatro partes básicas: elementos de filtrado, sistema neumático de transporte, un conjunto de sensores y los dispositivos electrónicos de control[14].

Las características principales de los elementos mencionados son:

a) **Elementos de filtrado**. Se encargan de eliminar los elementos sólidos y moléculas de agua que se encuentran suspendidos en el gas. Esto permite mantener limpios los conductos del sistema y el correcto funcionamiento de los sensores a los que generalmente el agua y las partículas sólidas causan estragos.

b) **Sistema neumático de transporte**. Son los dispositivos responsables del transporte de la muestra de gas por los conductos del sistema. Generalmente está compuesto por bombas, válvulas, mangueras, etc.

c) **Conjunto de sensores**. Existen diferentes técnicas para la detección y análisis de gases dependiendo de su naturaleza. Ciertamente la utilización de fuentes luminosas (generalmente de

[13] **R.G. Ramírez Chavarría, L. Santiago Cruz, R. González Oropeza, P.I. Rincón Gómez**, "*DESARROLLO DE UN SISTEMA DE ADQUISICIÓN Y REGISTRO DE PARÁMETROS VEHÍCULARES PARA EL DESARROLLO DE CICLOS DE MANEJO EN EL VALLE DE MÉXICO*", Memorias del Congreso, SOMI XXVII, México, 2012.

[14] **Manual Técnico de Verificación automotriz.** Jiménez Editores e Impresores S.A., 2002.

rayos infrarrojos) es altamente utilizada en este campo. La óptica es pues, una de las principales materias que auxilian a la detección y análisis de ciertos gases como el bióxido de carbono, monóxido de carbono e hidrocarburos. Esta técnica ha evolucionado notoriamente y es la que se utiliza para poder evaluar las emisiones contaminantes de los vehículos que poseen motores de combustión interna. La técnica se conoce como método "NDIR" (Non Dispersive Infrared) y el principio de operación es el siguiente: se evalúa la cantidad de luz que absorbe un gas al ser expuesto a una fuente luminosa infrarroja. Como la concentración de un gas se encuentra en función a la cantidad de moléculas que tiene por unidad de volumen, la absorción de luz infrarroja aumenta considerablemente cuando la concentración de algún gas aumenta y viceversa.

Algunos sistemas poseen dos cámaras de análisis independientes, una en donde se encuentra un gas de referencia y otra por donde el gas por medir se hace circular. Esto permite hacer una comparación de información utilizando la misma fuente luminosa. Para ello el sistema hace una serie de muestreos que permitirán obtener valores diferenciales en las mediciones.

Los resultados del análisis de la información adquirida arrojan datos sobre la concentración de diferentes gases, como se muestra en la figura 2.12. En ella se aprecia cómo a diferentes frecuencias de la fuente luminosa es posible obtener una respuesta para cada tipo de gas, que permite no sólo identificarlo si no además poder conocer la concentración del mismo.

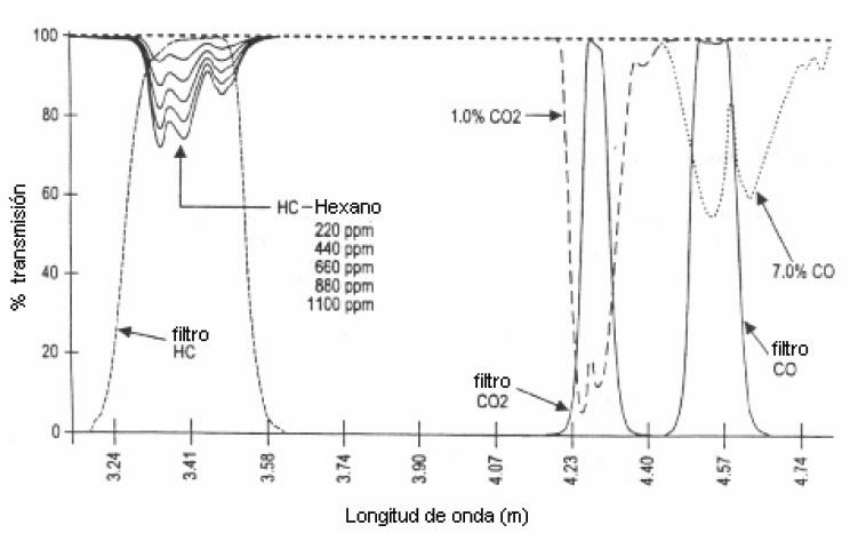


Figura 2.12. Resultados del análisis usando la técnica NDIR.

Otros gases como el oxígeno y los gases nitrosos pueden ser detectados mediante el uso de biosensores, explotando la propiedad de oxido-reducción de los materiales.

d) **Dispositivos electrónicos de control y despliegue.** Se refiere al sistema que controla la instrumentación del sistema. También es responsable del despliegue e interpretación de datos. Los analizadores de gases pueden emplearse para verificar si la combustión fue completa. La mezcla entre combustible y aire pueden cambiar esta situación modificando la eficiencia de un motor de combustión. El residuo de gases producto de la combustión refleja si un dispositivo trabaja en el punto óptimo, conocido como punto estequiométrico.

Existe una variedad de analizadores de gases en función del principio que se utilice para realizar las mediciones de los gases contaminantes. Estas técnicas incluyen el análisis por NDIR (*Non Dispersive Infrared*), FTIR (*Fourier Transform Infrared*) y espectrometría de masas, entre otras. El proceso de análisis por NDIR, que se ha sido explicado anteriormente, es la técnica que utiliza la banca analizadora de gases contaminantes ANDROS 6600, la cual se explica a detalle en el capítulo siguiente.

Una vez comprendidos los conceptos generales en los cuales este trabajo tiene su base, en el siguiente capítulo se describe detalladamente el diseño y desarrollo del equipo *CYCLE-DAQ*.

C A P Í T U L O 3

DESARROLLO DEL SISTEMA

En este capítulo se describen detalladamente los elementos que integran al sistema de adquisición y registro de datos CYCLE-DAQ, esto es, sensores, unidad de procesamiento, almacenamiento de los datos y el dispositivo de despliegue de información para el usuario, así como la programación del microcontrolador y el desarrollo del software de procesamiento de datos, DRIVE-SOFT .

3.1. Descripción del Hardware

El diseño del sistema de adquisición y registro de datos, denominado CYCLE-DAQ, comienza con la definición de las especificaciones dadas por el grupo de trabajo del LCE, de la UNAM; las cuales se dan en función de las necesidades planteadas al inicio del proyecto, con el objetivo de poder desarrollar íntegramente ciclos de manejo en el Valle de México. En la tabla 3.1 se muestran las características generales del equipo CYCLE-DAQ.

CARACTERÍSTICAS DEL EQUIPO CYCLE-DAQ
Independiente del funcionamiento mecánico del automóvil
Modular y de fácil uso
Registro de datos en memorias SD Card, compatible con formatos FAT16 Y FAT32
Fuente de alimentación independiente
Despliegue de información para el usuario en un *display* LCD
Periodo de muestreo igual a 1 segundo
Variables a medir: velocidad, distancia recorrida, latitud, longitud, altitud, pendiente, voltaje en la batería del automóvil y concentración de emisiones de gases contaminantes (CO_2, CO, O_2, HC y NOx)

Tabla 3.1. Características generales del equipo CYCLE-DAQ.

En la etapa de definición de necesidades, se proporcionó una tabla con los requerimientos de los sensores a utilizar en el desarrollo del equipo para la adquisición de las variables de interés, así como la precisión de los mismos. En la tabla 3.2 se muestran dichos requerimientos.

VARIABLE A MEDIR	*SENSOR*	*PRECISIÓN*
Velocidad	Receptor GPS	± 5 [km/h]
Distancia recorrida	Receptor GPS	±100 [m]
Latitud, longitud, altitud	Receptor GPS	No aplica
Pendiente (inclinación del automóvil)	Acelerómetro	± 2 [°]
Voltaje en la batería del automóvil	Sensor de voltaje	± 0.5 [V]
Emisiones de gases contaminantes	Analizador de gases	No aplica

Tabla 3.2. Requerimientos de los sensores del CYCLE-DAQ.

3.1.1. La tarjeta de desarrollo

De acuerdo con las necesidades planteadas anteriormente, el diseño del equipo CYCLE-.DAQ deberá estar basado en un microcontrolador, cuya ventaja principal es proporcionar autonomía al sistema. Es por ello que al inicio del proyecto se contaba en el laboratorio con una tarjeta de desarrollo llamada Arduino y que actualmente es muy común para el desarrollo de sistemas basados en microcontroladores. Además de la ventaja de que ya se contaba con dicha tarjeta, ésta posee beneficios a comparación de otras tarjetas de desarrollo que existen en el mercado. De esta manera, la tarjeta Arduino fue la seleccionada para ser el cerebro del equipo CYCLE-DAQ. Las ventajas y las características más importantes de Arduino, que fundamentan la selección de ésta para el desarrollo del CYCLE-DAQ, se describen a continuación.

Arduino es una tarjeta electrónica de plataforma abierta, de fácil uso, que consta básicamente de una placa de circuito impreso que contiene un microcontrolador de la familia ATmega de la marca Atmel, fabricante a nivel mundial de semiconductores. Arduino posee una gran cantidad de ventajas a comparación de otras tarjetas de desarrollo, características que lo han convertido en una de las herramientas preferidas de los desarrolladores de proyectos relacionados con electrónica, telecomunicaciones, control, automatización y computación. Entre las características y ventajas más importantes de la tarjeta Arduino se encuentran las siguientes:

- ✓ El sistema se integra en una placa donde se encuentra todo listo para comenzar a ser utilizado.

- ✓ No requiere de un programador externo, en vez de ello, cuenta con la posibilidad de ser "autoprogramado" mediante el puerto USB de la computadora.
- ✓ Es multiplataforma, puede ser utilizado en ambientes Windows, Mac y Linux.
- ✓ Su lenguaje de programación es intuitivo y flexible.
- ✓ El costo no es tan elevado, a comparación de otras tarjetas de desarrollo basadas en microcontroladores.
- ✓ Es un sistema de *hardware* y *software* libre, es decir, puede utilizarse libremente para desarrollar cualquier tipo de proyecto sin tener que adquirir ningún tipo de licencia.

La tarjeta Arduino se encuentra disponible en el mercado en diferentes versiones, la diferencia entre dichas versiones radica principalmente en las características técnicas del microcontrolador ATmega en el cual están basadas, lo que a su vez implica que el número de entradas y salidas, la capacidad de memoria y el tamaño de la tarjeta sean diferentes para cada versión.

La elección de la tarjeta Arduino como unidad de procesamiento del equipo CYCLE-DAQ se fundamenta en los siguientes puntos:

- ✓ Ya se contaba con una tarjeta en el LCE, lo que permitió que el desarrollo del equipo comenzara inmediatamente.
- ✓ En el LCE no se cuenta con un programador de microcontroladores.
- ✓ La programación se realiza mediante el puerto Universal Serial Bus (USB) de la computadora y a su vez funciona como fuente de alimentación para la tarjeta.
- ✓ Uso de lenguajes de alto nivel para la implementación del programa en el microcontrolador.

Siguiendo los puntos anteriormente mencionados, el equipo CYCLE-DAQ en su primera versión, prototipo CYCLE-DAQ V1.0, se desarrolló bajo la plataforma Arduino UNO, debido a que dicha tarjeta era la que se encontraba en el LCE. Sin embargo, debido a las especificaciones y necesidades establecidas para el desarrollo del equipo, el número de entradas y salidas para controlar los dispositivos periféricos que se fueron integrando paulatinamente al sistema, no fue suficiente, y por tal motivo fue necesario actualizar a la versión CYCLE-DAQ V2.0, basada en la tarjeta Arduino MEGA. Ésta tarjeta cuenta con un mayor número de líneas de entrada y salida, puertos de comunicación y capacidad de la memoria para implementar el programa. Por tal motivo, en el presente trabajo se enfatiza en el estudio de la tarjeta Arduino MEGA y su microcontrolador asociado.

Arduino MEGA

La tarjeta de desarrollo Arduino MEGA, figura 3.1, posee:

- 54 terminales de entrada/salida digitales, de los cuales 14 pueden ser usados como salidas en modulación de ancho de pulso (*PWM: Pulse Width Modulation*).
- 16 terminales para entradas analógicas.
- 4 puertos seriales por *hardware* (*UART: Universal Asynchronus Receiver-Transmitter*).
- Un oscilador de cristal de 16 [MHz].
- Un conector USB tipo B.
- Un conector para alimentación externa.
- Un conector para programadores externos *(ISP: In System Programming).*
- Un botón de *Reset.*

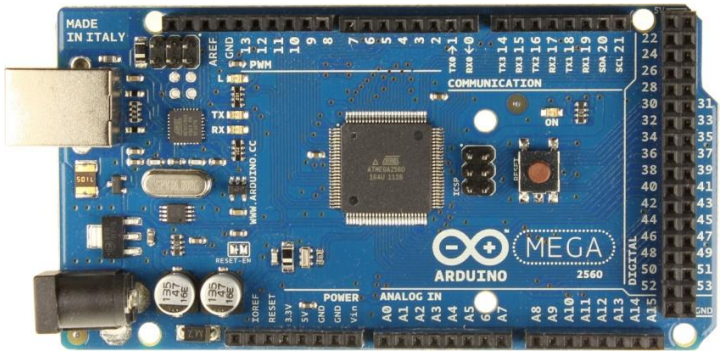


Figura 3.1. Tarjeta de desarrollo Arduino MEGA.

En la tabla 3.3 se describen las características generales de la misma.

Microcontrolador	*ATmega2560*
Voltaje de operación	5 [V]
Voltaje de alimentación recomendado	7-12 [V]
Terminales E/S digitales	54 de los cuales 15 pueden ser PWM
Terminales Analógicos	16
Corriente DC en terminales de E/S	40 [mA]
Memoria Flash	256 [KB]
Memoria EEPROM	4 [KB]
Memoria RAM	8 [KB]
Oscilador	Cristal de cuarzo a 16 [MHz]

Tabla 3.3. Características de la tarjeta Arduino MEGA.

Microcontrolador ATmega2560

El microcontrolador ATmega2560 es la unidad de procesamiento principal de la tarjeta Arduino MEGA, el cual sigue la arquitectura AVR de ATMEL, los cuales son una familia de microcontroladores tipo RISC. El ATmega2560 es una unidad de procesamiento central (CPU) de arquitectura Harvard. Tiene 32 registros de 8 bits. Algunas instrucciones sólo operan en un subconjunto de estos registros. Está basado en la tecnología *Complementary metal-oxide-semiconductor* (CMOS*). P*ara su programación este microcontrolador cuenta con 135 instrucciones, las cuales son ejecutadas en un solo ciclo de reloj, además cuenta con de 256 [KB] de memoria *Flash*, de los cuales 8 [KB] son utilizados por el *"bootloader"* de Arduino. La memoria EEPROM del dispositivo es de 4 [KB], sobre la cual se permiten hasta 100,000 ciclos de lectura/escritura y cuenta con una memoria RAM estática *(SRAM)* de 8 [KB].

Las características de sus periféricos son las siguientes:

- 100 terminales (encapsulado de montaje superficial).
- Dos contadores/temporizadores de 8 bits con pre-escalador separado y modo de comparación.
- Cuatro contadores/temporizadores de 16 bits con pre-escalador separado, modo de comparación y modo de captura.
- Cuatro canales de 8 bits con salida de modulación en ancho de pulso PWM.
- Doce canales PWM programables para resoluciones de 2 a 16 bits.
- Dieciséis canales para conversión analógica-digital (CAD) con una resolución de 10 bits.
- Cuatro puertos seriales USART
- Interfaz *SPI: Serial Periphal Interface* en configuración Maestro/Esclavo.
- Interfaz I^2C: *Inter-Integrated Circuit.*
- Interfaz *JTAG: Joint Test Action Group.*
- Comparador analógico integrado en el mismo circuito integrado.
- *Watchdog Timer* progamable.
- Fuentes de interrupción internas y externas.

Las características eléctricas del microcontrolador son:

- Voltaje de Operación: 1.8 [V] a 5.5 [V].
- Rango de temperatura: -40 [°C] a 85 [°C].
- Velocidad de reloj: 0-16 [MHz] @ 4.5 - 5.5 [V].
- Consumo de corriente en operación: 500 [µA].

El diagrama de bloques del microcontrolador ATmega2560 se muestra en la figura 3.2.

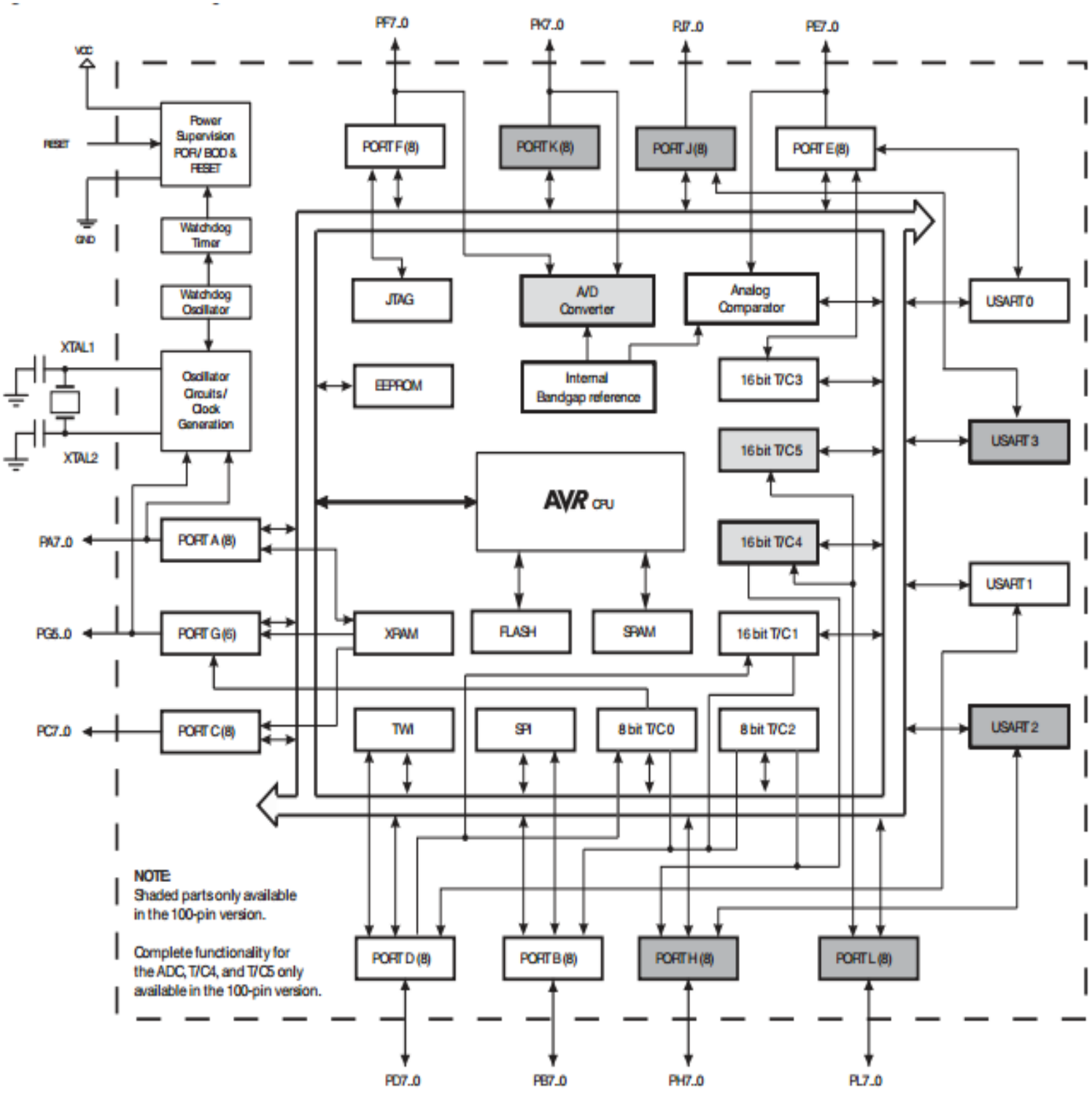


Figura 3.2. Diagrama de bloques del microcontrolador ATmega2560.

Respecto a las entradas analógicas en la tarjeta Arduino, comúnmente utilizadas en aplicaciones donde se utilizan sensores, éstas tienen una resolución de 10 bits, por lo que entregan valores decimales entre 0 y 1023. El rango de voltaje que aceptan está dado entre 0 y 5 [V].

Otro aspecto importante de mencionar es que la tarjeta Arduino, cuentan con terminales de alimentación para dispositivos periféricos que se conecten a ella. Estas terminales son capaces de suministrar 5 [V], 3.3 [V] y la respectiva tierra común en la tarjeta (GND).

En la figura 3.3 se muestra la distribución de las terminales del microcontrolador ATmega2560, así como los nombres asignados en la tarjeta Arduino, lo cual facilita al desarrollador la comprensión de la función que desempeña cada terminal de la misma.

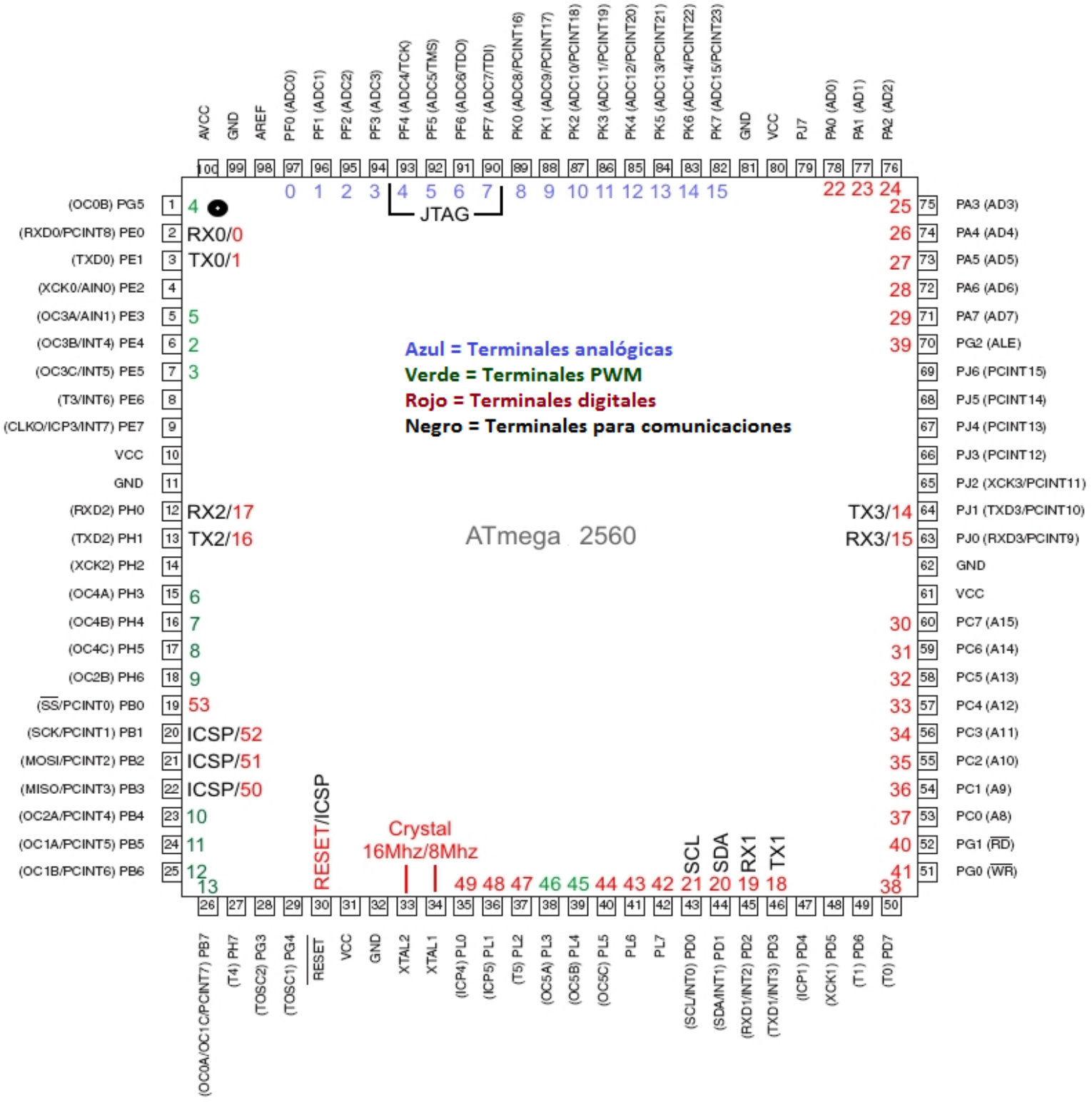


Figura 3.3. Relación entre Arduino y el microcontrolador ATmega2560.

3.1.2. Acelerómetro ADXL345

Una de las variables de suma importancia para la obtención de ciclos de manejo es la inclinación que el vehículo automotor o motocicleta experimenta en distintas condiciones sobre su trayectoria. Es por ello que, con el objetivo de recolectar el valor de dicha variable, el equipo *CYCLE-DAQ* hace uso de un acelerómetro digital de 3 ejes llamado ADXL345, fabricado por la marca *Analog Devices*.

El acelerómetro digital de 3 ejes ADXL345, figura 3.4, es un dispositivo pequeño, ligero y de bajo consumo, el cual realiza las mediciones tomando como referencia la aceleración de la gravedad en la tierra (9.81 [m/s^2]), llamada unidad [g]. Cuenta con rangos de medida ajustables de ±2, ±4, ±8 y ±16 [g]. Sus salidas son digitales en formato de 16 bits. Para la comunicación con microprocesadores cuenta con las interfaces de comunicación: SPI o I^{2}C. Es un dispositivo ideal para aplicaciones móviles, como es el caso del sistema *CYCLE-DAQ*.

El ADXL345 mide la aceleración dinámica resultante de movimientos o caídas, así como la aceleración estática de la gravedad para aplicaciones en donde es utilizado como sensor de inclinación. Debido a su alta resolución permite medir cambios de inclinación menores a 1.0° [36].

Figura 3.4. Acelerómetro ADXL345.

Este dispositivo es utilizado en aplicaciones como: instrumentación médica, dispositivos de juego, instrumentación industrial, dispositivos de navegación, protección en discos duros, accesorios deportivos.

En la tabla 3.4 se listan sus características más importantes y que sustentan la elección de este dispositivo para satisfacer las necesidades planteadas en el proyecto.

[36] ADX3L45 Triple Axis Digital Accelerometer DATASHEET, Analog Devices Inc., 2010-2012.

CARACTERÍSTICAS DEL ACELERÓMETRO ADXL345
Consumo de corriente en actividad: 40 [µA]
Resolución de medida seleccionable por el usuario mediante *software*
Detección de caída libre
Usado como sensor de inclinación
Monitoreo de actividad e inactividad
Voltaje de alimentación: 2.0 a 3.6 [V]
Temperatura de operación: -40 a 85 [°C]
Interfaces SPI e I^2C para comunicación con microcontroladores
Pequeño y ligero: 3 x 5 x 1 [mm]

Tabla 3.4. Características del acelerómetro ADXL345.

Otro aspecto importante a considerar es que el ADXL345 no necesita de etapas de acondicionamiento posteriores al sensado, como lo son la amplificación de la señal o la implementación de filtros analógicos, ya que éstas se encuentran integradas dentro del mismo encapsulado.

Para tener una mejor idea sobre el funcionamiento del acelerómetro, en la figura 3.5 se muestra el diagrama de bloques de las partes que integran al mismo.

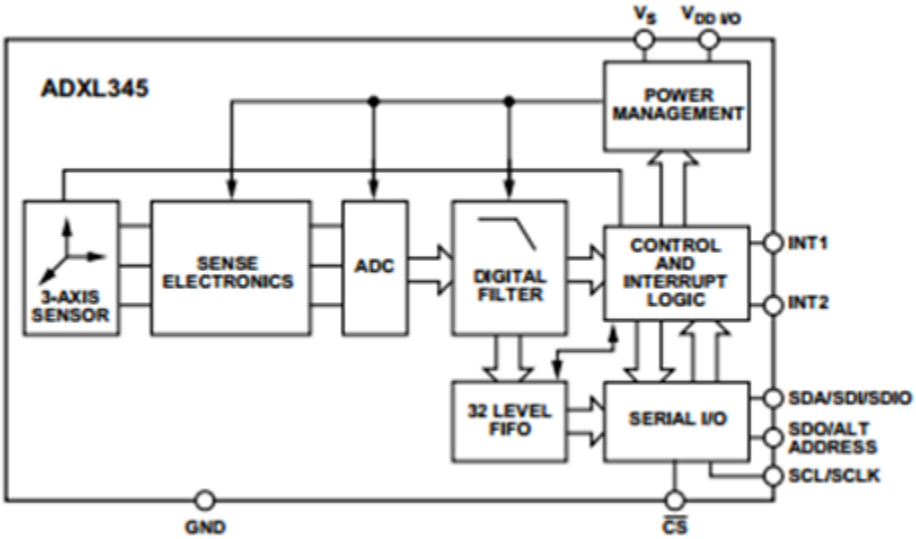


Figura 3.5. Diagrama de bloques del ADXL345.

***Bus* I^2C**

Para la adquisición de los datos provenientes del acelerómetro ADXL345, se realiza la conexión de este dispositivo con la tarjeta de desarrollo Arduino mediante el *bus* I^2C. Como se mencionó en la sección anterior, el ADXL345 permite al usuario establecer la comunicación con microprocesadores mediante el *bus* I^2C o el protocolo SPI. Sin embargo, para los fines de este desarrollo se utiliza el modo I^2C debido a que ocupa solo dos terminales para la transmisión de los datos, mientras que el módulo SPI requiere de cuatro terminales, lo cual reduce el tamaño del circuito a implementar; además de que el *bus* I^2C posee la capacidad de asegurar que la comunicación entre el acelerómetro y el microcontrolador se está efectuando adecuadamente.

La interfaz de comunicación I^2C es ideal para utilizarse en aplicaciones con microcontroladores, dado que con sólo dos líneas bidireccionales es posible comunicarse con distintos dispositivos conectados al *bus.* Una de estas líneas está dedicada para señal de sincronización o señal de reloj *Serial CLock*, (SCL) y la otra para la señal de transmisión de datos *Serial DAta* (SDA). El hardware externo necesario para que el módulo I^2C funcione correctamente son dos resistencias llamadas resistencias de *pull-up*, una en cada línea del *bus*, debido a que dichas salidas sonde tipo drenador abierto, esto indica que, en su estado de reposo, las líneas se encuentran en un estado lógico alto. Todos los dispositivos conectados al *bus* tienen direcciones independientes.

En la figura 3.6 se muestra el diagrama de conexión para el acelerómetro ADXL345 mediante el *bus* I^2C con un microcontrolador, para llevar a cabo la transmisión de datos entre ambos dispositivos.

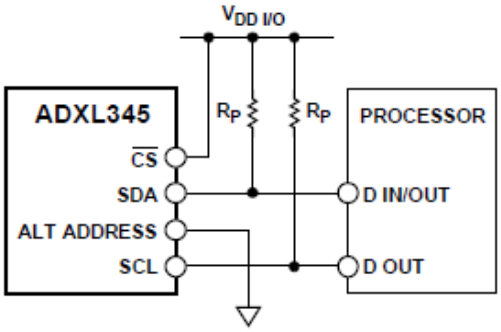


Figura 3.6. Diagrama de conexión del ADXL345 con un microcontrolador.

El valor de las resistencias de *pull-up* depende de la velocidad a la que se establece la comunicación I^2C. Para la velocidad estándar propuesta por el fabricante, igual a 100 [kHz], los valores de las resistencias están entre los 5 [kΩ] y los 10 [kΩ]. Para este caso las resistencias utilizadas en la implementación son de 4.7 [kΩ], por ser el valor comercial más cercano a los 5 [kΩ].

Para comunicaciones en donde se utiliza el *bus* I^2C, éste puede encontrarse en alguno de los estados que la norma define de la siguiente manera:

- ✓ **Libre:** Este estado se caracteriza por encontrase las líneas SDA y SCL en alto, es decir, en estado de reposo, sin que se esté realizando ningún tipo de transferencia.
- ✓ **Inicio**. Se produce una condición de inicio cuando el MAESTRO inicia una transacción. En concreto, consiste en un cambio de alto a bajo en la línea SDA mientras SCL permanece a alta. A partir de que se dé una condición de inicio se considerará que el bus está ocupado y ningún otro maestro deberá intentar generar su condición de inicio.
- ✓ **Cambio**. Se produce una condición de cambio cuando, estando en bajo la línea SCL, la línea SDA puede cambiar de estado. En la transferencia de datos por un *bus* I^2C éste es el único instante en el que el TRANSMISOR (que podrá ser tanto un maestro como un esclavo) podrá poner en la línea SDA cada bit del carácter a transmitir.
- ✓ **Dato**. Este estado es el que, una vez iniciada una transmisión, queda definido por el estado alto de la señal de sincronía SCL. Aquí se considera que el dato emitido es válido, y no se admite que pueda cambiar. Cabe recordar que, ya iniciada una transferencia, el único instante en que la línea de datos puede ser modificada es en el estado de cambio.
- ✓ **Parada**. Se produce una condición de fin o parada cuando, estando la línea SCL en alto, se origina un cambio de estado de bajo a alto en la línea SDA. Obsérvese que esto es una violación de la condición de dato, y es precisamente por esto por lo que se utiliza para que un maestro pueda indicar al esclavo que se finaliza la transmisión. Tras la condición de parada se entra automáticamente en el estado de *bus* libre.

En la figura 3.7 se muestra el diagrama de tiempos del *bus* I^2C para el acelerómetro ADXL345.

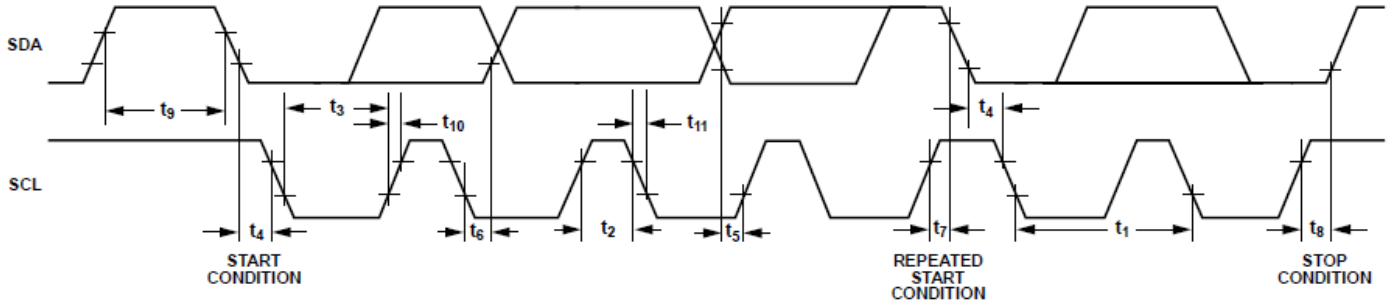


Figura 3.7. Diagrama de tiempos del ADXL345.

Una vez iniciada la transmisión por parte del dispositivo MAESTRO, mediante una señal de inicio, manda un paquete de direcciones que consta de nueve bits, de los cuales siete son de dirección, uno de lectura/escritura y uno de *acknowledge* (ACK). Si el bit de lectura/escritura se encuentra en alto, indica que el MAESTRO desea realizar una operación de lectura sobre el ESCLAVO direccionado, en caso contrario se realiza una operación de escritura sobre dicho dispositivo. Cuando el ESCLAVO reconoce que ha sido direccionado, éste debe poner en bajo la línea SDA en el bit nueve, si el

dispositivo ESCLAVO se encuentra ocupado o no puede atender la llamada del MAESTRO, éste debe dejar la línea SDA en alto durante el noveno ciclo de reloj. Una vez que el paquete de dirección es reconocido por el ESCLAVO, el TRANSMISOR envía el paquete de datos. Todos los paquetes de datos transmitidos por el *bus* I^2C tienen una longitud de nueve bits, siendo ocho de ellos de datos y uno de enterado o ACK. La transmisión comienza con el bit más significativo del byte transmitido. Durante la transmisión, el dispositivo MAESTRO es el encargado de generar la señal de inicio y paro, así como la señal de reloj; mientras el RECEPTOR manda las señales de enterado (ACK) al final de cada paquete, con lo que se asegura una correcta transmisión. En caso contrario, el RECEPTOR mantiene la señal SDA en alto, lo cual representa una señal de *no acknowledge* (NACK), la cual es enviada al transmisor al final del último byte. Cuando una señal NACK es recibida por el TRANSMISOR, en caso de ser éste un dispositivo ESCLAVO, dejará de transmitir información; en caso de que sea el MAESTRO, éste enviará una señal de inicio o paro. La señal de paro es transmitida por el MAESTRO cuando el intercambio de información ha terminado y éste desea que el *bus* quede libre.

La conexión del acelerómetro con la tarjeta Arduino MEGA se realiza de la siguiente manera: las líneas SDA y SCL del acelerómetro son conectadas a las terminales con los mismos nombres del microcontrolador, y a cada línea se le añade una resistencia de *pull-up* con valor de 4.7 [kΩ], esto último para que la comunicación pueda realizarse correctamente. El acelerómetro es polarizado mediante las salidas de 3.3 [V] y tierra (GND) provenientes de la tarjeta Arduino.

En la figura 3.8 se muestra el diagrama esquemático de las conexiones realizadas entre ambos dispositivos.

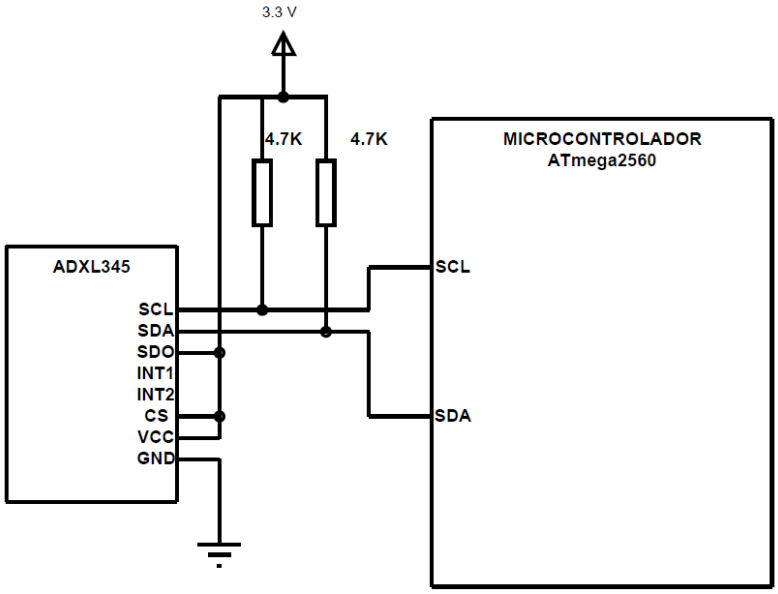


Figura 3.8. Diagrama de conexión del ADXL345.

3.1.3. RECEPTOR GPS EM-406A

Para conocer la distancia recorrida por el vehículo en el cual el sistema *CYCLE-DAQ* se encuentre instalado, así como la velocidad instantánea sin necesidad de utilizar sensores que intervengan en la parte de la mecánica del automóvil, se hace uso del receptor GPS EM-406A, mostrado en la figura 3.9.

Cabe mencionar que actualmente en diversos países de Europa y Asia, como Francia y Japón, respectivamente, la tecnología GPS es la base del proceso de recolección de datos para poder obtener ciclos de manejo representativos de las ciudades más importantes de dichos países, debido a que, como ocurre con el equipo *CYCLE-DAQ*, la tecnología GPS evita el uso de sensores externos al automóvil y ofrece una gran precisión al obtener los datos provenientes de los satélites.

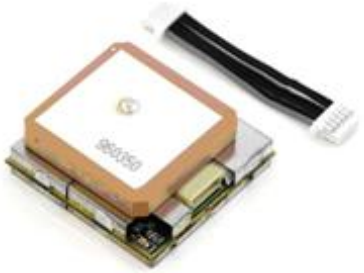


Figura 3.9. Receptor GPS EM-406A.

El receptor EM-406A es un módulo GPS de la marca *USGlobalSat.* Este dispositivo es ideal para aplicaciones embebidas o de radio control, en donde es necesario el uso de sistemas de posicionamiento global de manera eficaz, además que ofrece la característica de ser portable, es extremadamente pequeño ya que apenas mide unos 3x3 [cm] y tan sólo pesa 16 gramos.

Además de su bajo costo, este receptor ya cuenta con antena integrada, lo que reduce su espacio de trabajo y las conexiones necesarias para su interacción con el microcontrolador. Las características decisivas en la elección de éste dispositivo, además de tener una gran precisión y confiabilidad en los datos proporcionados se enumeran a continuación.

1. Procesador SiRF Star III de alto rendimiento
2. Soporta el protocolo de datos NMEA 0183.
3. Antena integrada.
4. Tamaño reducido.
5. Canales para comunicación: 20.
6. Rápida recuperación de la señal: 100 [ms].
7. Soporte WAAS para corrección de la señal.
8. Frecuencia de operación: 1575.42 [Mhz].

Precisión

Margen de error posicional de ± 10 [m] (5m con WAAS).
Error en la velocidad de ± 0.1 [m/s].
Error en tiempo de ± 1 [µs] de sincronización.

Condiciones dinámicas

Altitud: 18, 000 [m] máximo.
Velocidad: 515 [m/s] máximo.
Aceleración: menor de 4 [g].

Energía

Alimentación de entrada: 4.5 a 6.5 [V].
Consumo de corriente: 60 [mA] máximo.

Protocolo

Nivel eléctrico: TTL (***T****ransistor* ***T****ransistor* ***L****ogic*).
Nivel de voltaje a la salida: 0 a 2.85 [V].
Tasa de transferencia de 4800 [bps].
Mensaje de salida: NMEA 0138 GPGGA, GPGSA, GPGSV, GPRMC.
Protocolo RS-232.

Tiempo de adquisición

Readquisición de la señal: 0.1 [s] aproximadamente.
Arranque en *caliente*: 8 [s] aproximadamente.
Arranque *medio*: 38 [s] aproximadamente.
Arranque en *frio*: 42 [s] aproximadamente.

La distribución de las terminales del receptor EM406-A se muestra en la figura 3.10.

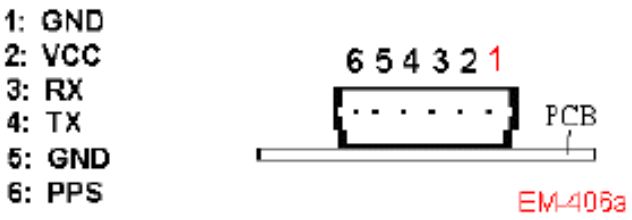


Figura 3.10. Distribución de las terminales del EM-406A.

La función de cada uno de las terminales es la siguiente:

VCC: (Entrada de alimentación): En esta terminal se conecta la fuente de alimentación necesaria para el funcionamiento del receptor.

TX: Es el canal de transmisión principal, a la salida se encuentran los datos de navegación y medición para que sean recibidos e interpretados por el usuario.

RX: Es el canal principal para la recepción de comandos enviados por el usuario al receptor. Normalmente esta terminal debe estar en alto, en caso de no ser usado deberá conectarse una resistencia de *pull-up* a 3.5 [V].

GND: Provee la señal de tierra al receptor. Las dos terminales marcadas con GND son comunes y deberán estar conectadas a la tierra de la fuente de alimentación.

PPS: Esta terminal provee un pulso por segundo, la cual indica que el receptor GPS se encuentra sincronizado con los satélites.

Estándar RS-232

Para la comunicación entre la tarjeta Arduino y el receptor EM-406A se hace uso del protocolo de datos NMEA 0183. Sin embargo, para poder establecer dicha comunicación es necesario definir la etapa física sobre la cual el protocolo de datos trabajará. Para ello y gracias a las características de salida que ofrece el receptor, se hace uso de uno de los cuatro módulos UART del microcontrolador ATmega2560. Debido a que el receptor cuenta con la característica de poder transmitir los datos provenientes de los satélites, bajo el estándar RS-232 con niveles TTL, no es necesario hacer uso de *hardware* externo para adaptar los niveles de voltaje a la salida y así poder ser conectado con el microcontrolador.

La norma RS-232 define un tipo de comunicación serial, lo cual establece que durante la transmisión de los datos éstos son enviados bit a bit, es decir, éstos se envían uno por uno desde el emisor al receptor o viceversa. Su principal característica radica en que se trata de un tipo de comunicación serial asíncrona, esto es, que no requiere ninguna señal de reloj para sincronizar la transmisión, en vez de esto, la duración de cada bit está determinada por la velocidad con la cual se realiza la transferencia de datos.

Normalmente cuando no se realiza ninguna transferencia de datos, la línea del transmisor se encuentra en *IDLE*, esto quiere decir que se encuentra en un nivel lógico alto. Para iniciar la transmisión de datos, el transmisor coloca está línea en bajo durante un cierto tiempo, lo cual se conoce como bit de arranque o *start bit* y a continuación empieza a transmitir con una intervalo de tiempo los bits correspondientes al dato, empezando siempre con el bit menos significativo (LSB) y terminando con el bit más significativo (MSB).[37]

La figura 3.11 muestra la estructura de un carácter que se transmite siguiendo la secuencia de una comunicación serial asíncrona.

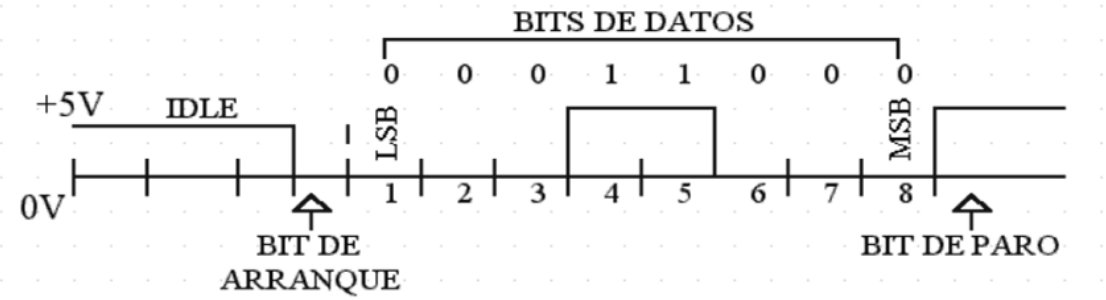


Figura 3.11. Transferencia serial asíncrona.

La organización *Electronics Industry Association* (EIA) elaboró la norma RS-232, la cual define la interfaz mecánica, la distribución de las terminales, las señales y los protocolos que debe cumplir la comunicación serial. La norma RS-232 en sus diferentes versiones cumple con los siguientes niveles de voltaje:

- Un “1” lógico es un voltaje comprendido entre -3[V] y -15[V].
- Un “0” lógico es un voltaje comprendido entre +3[V] y +15[V].

En los microcontroladores para representar un “0” lógico se trabaja con voltajes inferiores a 0.8 [V] y para un “1” lógico con voltajes mayores a 2.0 [V]. En general cuando se trabaja con familias TTL y CMOS se asume que un “0” lógico es igual a cero volts y un “1” lógico es igual a cinco volts. Además para poder establecer la comunicación mediante RS-232 entre un microcontrolador y otros

[37] **Tomasi Wayne** (2003). *Sistemas de Comunicaciones Electrónicas.* México: Prentice Hall.

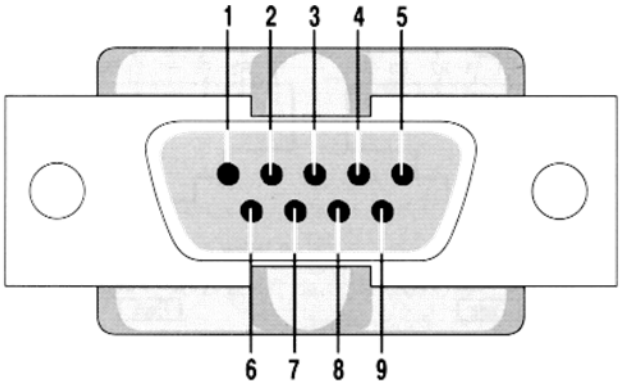
dispositivos periféricos, e incluso la PC, solo son necesarias al menos tres líneas, TXD, RXD y GND, para establecer la transmisión de los datos entre dispositivos.

Los parámetros que caracterizan a la comunicación serial son: velocidad, paridad, bits de datos y bits de parada.

- La velocidad de transmisión está estandarizada según la norma en baudios. Un baudio se define como el número de veces que cambia la señal portadora en un segundo. Los primeros dispositivos operaban a velocidades muy bajas, del orden de 110 a 1200 baudios. La velocidad más comúnmente utilizada actualmente es de 9600 baudios.
- Los bits de datos son aquellos enviados como caracteres ASCII, por lo que pueden utilizarse 7 u 8 bits, dependiendo de la longitud del carácter.
- Paridad se refiere al método detector de errores utilizado por el estándar RS232 y cuya función es agregar un bit a la trama de datos para verificar que la transmisión se ha realizado.
- Bit de parada es el que se envía después de un carácter, el cual tiene un valor lógico de "1", la duración de este bit puede ser de 1, 1.5 o 2 periodos.

Según el estándar del RS-232, la distancia máxima del cable de conexión utilizado en la comunicación serial es aquel cuya capacitancia es menos o igual a 2500 [pF]. Utilizando un cable estándar esta capacitancia se alcanza entre los 15 y 20 [m].

El conector originalmente definido para el estándar RS-232 es el conector DB-25, el cuál posee 25 terminales para realizar la conexión serial. Sin embargo, actualmente, con el fin de reducir el tamaño de dicho conector, la mayoría de los dispositivos utilizados en electrónica y comunicaciones cuentan con un conector tipo DB-9, el cual realiza la conexión con sólo 9 terminales. En la figura 3.12 se muestra la distribución de las terminales de un conector DB-9, así como sus nombres y la dirección para la transmisión de los datos.



Terminal	Señal
1	*Data Carrier Detect (DCD)*
2	*Received Data (RD)*
3	*Transmitted Data (TD)*
4	*Data Terminal Ready (DTR)*
5	*Ground*
6	*Data Set Ready (DSR)*
7	*Request to Send (RTS)*
8	*Clear to Send (CTS)*
9	*Ring Indicator (RI)*

Figura 3.12.Conector DB-9 y sus terminales.

Aunque el conector estándar es el anteriormente descrito, el receptor GPS tiene su conector propio, pero aún así se rige bajo el protocolo RS-232, en cuanto a especificaciones eléctricas y de comunicación se refiere.

Una vez comprendidos los aspectos generales sobre el receptor GPS y el protocolo que comunicación que utiliza, es necesario describir cómo es que dicho receptor interactúa con la tarjeta Arduino y su microcontrolador asociado.

El receptor EM-406A es alimentado mediante las terminales de 5 [V] y GND de la tarjeta de desarrollo. Como se mencionó anteriormente, para la conexión entre el receptor y el microcontrolador se usa uno de los cuatro módulos UART proporcionados por el microcontrolador, específicamente se usan las terminales TXD0 y RXD0, correspondientes al módulo UART0, las cuales se conectan a las terminales RX y TX del receptor, respectivamente.

La conexión entre el receptor GPS y el microcontrolador ATmega2560 de la tarjeta de desarrollo se muestra en la figura 3.13.

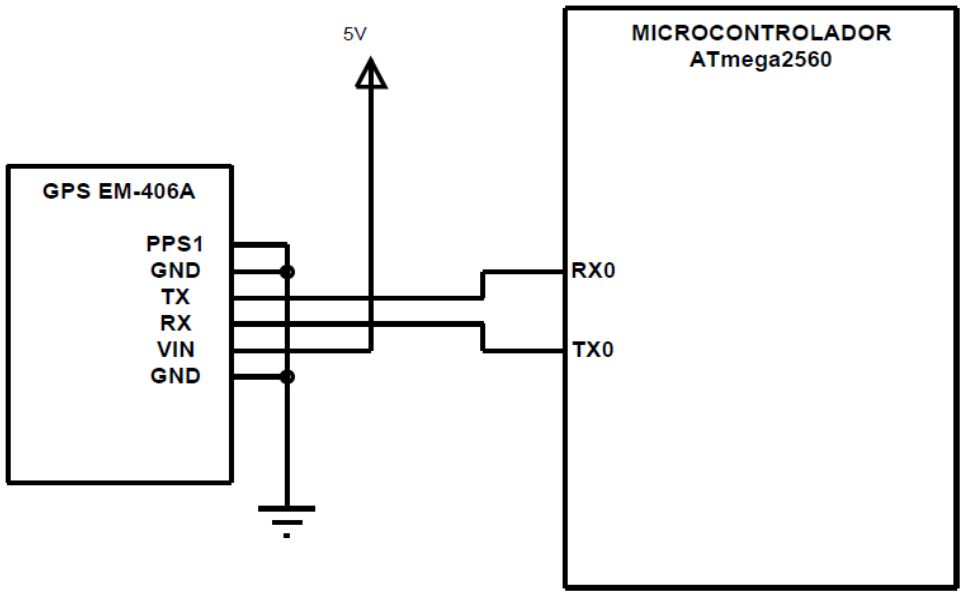


Figura 3.13. Diagrama de conexión del receptor EM-406A.

De acuerdo con las especificaciones del fabricante, el receptor funciona a una velocidad de transferencia de 4800 baudios, por lo que es necesario configurar al módulo UART0 del microcontrolador a la misma velocidad, para poder establecer la comunicación entre ambos dispositivos.

Además de las variables adquiridas provenientes del receptor GPS, éste también es utilizado para sincronizar el periodo de adquisición de datos igual a 1 segundo. Para fechar el registro de los datos adquiridos, se obtiene la hora y la fecha que el GPS provee.

La información adquirida por parte del microcontrolador, y que es necesaria para el desarrollo ciclos de manejo, sin la necesidad de utilizar otro tipo de dispositivos, se listan en la tabla 3.5.

PARÁMETROS	*UNIDAD DE MEDIDA*
Velocidad	[km/h]
Latitud	[rad]
Longitud	[rad]
Altitud	[m]
Distancia recorrida	[m]
Hora	HH:MM:SS
Fecha	DD/MM/AA

Tabla 3.5. Parámetros adquiridos del GPS.

Cabe mencionar que estos parámetros fueron seleccionados de acuerdo a las necesidades presentadas por parte del grupo de trabajo del LCE; sin embargo, no son las únicas que puede entregar el receptor EM-406A. Para tales fines sólo se hace uso de las sentencias $GPGGA y $GPRMC definidas, por el protocolo NMEA 0183; no obstante, éste no entrega los datos en el formato deseado, por lo que es necesario realizar operaciones mediante *software* para convertirlos a un formato deseado según las necesidades.

Como se puede observar, las ventajas que el receptor GPS seleccionado ofrece son muy grandes debido a la flexibilidad que provee para poder trabajar con él, además que brinda una gran precisión en cuanto a los datos que entrega, lo que garantiza que la obtención de ciclos de manejo se desarrolle de manera efectiva y confiable.

3.1.4. ANALIZADOR DE GASES ANDROS 6600

La banca analizadora de gases contaminantes, ANDROS 6600, es un dispositivo cuyo objetivo es medir los diversos contaminantes del aire en el gas del escape en vehículos automotores, para lo cual utiliza un conjunto de sensores y es controlada por un sistema central de procesamiento digital, que es capaz de comunicarse con la computadora personal o *PC*, a través de una interfaz serial asíncrona RS-

232. El análisis de la muestra de gas indicará si ésta contiene contaminantes en exceso o se encuentra dentro de los límites aceptables para un buen funcionamiento.

La integración de la banca analizadora de gases contaminantes fue realizada en el LCE, de acuerdo a una configuración propia, con el fin de contar con un instrumento acorde a las necesidades del laboratorio. En la figura 3.14 se puede observar una fotografía del analizador integrado en el LCE.

Figura 3.14. Analizador de gases ANDROS 6600.

El analizador de gases ANDROS 6600, de acuerdo con el manual de operación de Andros Inc., se basa en la segmentación de funciones, esto es, un proceso en el cual se someten todas las variables de muestra de gases a un análisis específico. Para poder efectuar dichas pruebas, el dispositivo cuenta con niveles de operación que rigen de manera ordenada la ejecución de cada proceso. Todo el conjunto de dispositivos que interactúan para el correcto funcionamiento de la banca puede gobernarse por medio de un *software* comercial, proporcionado por el fabricante y que puede adaptarse a las diversas necesidades del usuario, para decidir qué variable desea monitorear y en qué formato desea visualizarlas.

El funcionamiento del analizador de gases ANDROS 6600 incluye varias fases, desde la eliminación de partículas no deseadas en la muestra de gases, su distribución, hasta el análisis minucioso de la muestra por parte de los sensores para la obtención final de datos. Esto se realiza mediante el trabajo de sistemas independientes, los cuales conforman al analizador de gases. Dichos sistemas deben estar sincronizados y organizados para la correcta adquisición de los datos.

El analizador en cuestión, es capaz de medir las emisiones de contaminantes de los 5 principales gases (HC, CO_2, CO, O_2 y NOx), para dichas mediciones es necesario conocer la resolución y el rango establecido por el analizador de gases ante dicha situación.

Las características en cuanto a rango y resolución para la medición de los gases son presentadas en la tabla 3.6.

GAS	*RANGO*	*RESOLUCIÓN*
HC (hexano)	0 a 2000 ppm 2001 a 15000 ppm 15001 a 30000 ppm	1 ppm
HC (metano)	0 a 4000 ppm 4001 a 30000 ppm 30001 a 60000 ppm	1 ppm
CO	0% a 10% 10.01% a 15%	0.001 vol. %
CO_2	1% a 16% 16.01 % a 20.0%	0.01 vol. %
NOx	0 a 4000 ppm 4001 a 5000 ppm	1 ppm
O_2	0% a 25%	0.01 vol. %

Tabla 3.6. Rango y resolución de las concentraciones de gases.

En general se pueden nombrar cuatro sistemas en los cuales la banca ANDROS 6600 está basada:

- ✓ Sistema de Filtrado
- ✓ Sistema Neumático de transporte
- ✓ Sensores
- ✓ Módulo de alimentación

Sistema de Filtrado

En la primera fase o Sistema de Filtrado de la muestra de gases, se detectan los elementos sólidos y moléculas de agua que se encuentran dispersas en la muestra de gas. El objetivo de este proceso es limpiar los conductos del sistema completo para el correcto funcionamiento de los sensores, pues los residuos sólidos y moléculas dificultan el preciso análisis de la muestra y causa deficiencias en la obtención de datos. Este sistema está formado por dos componentes: la trampa de agua, figura 3.15, cuya función es separar las moléculas de agua suspendidas en la muestra de gases, ya que éstas pueden afectar de manera considerable las mediciones y el filtro de partículas o *trifilter*, figura 3.16, encargado de eliminar aquellos residuos sólidos que se encuentran en la muestra de gases y que no deben entrar en contacto con los sensores.

Figura 3.15. Trampa de agua.

Figura 3.16. *Trifilter*.

Sistema Neumático

La segunda etapa ó Sistema Neumático de transporte, se encarga de distribuir la muestra de gas por los conductos de todo el sistema. Este sistema a su vez se divide en varios módulos, utilizando dispositivos como bombas, válvulas, mangueras, etc. Dichos módulos se describen a continuación.

- ✓ **Módulo de desvío de gases**. Controla la distribución de gases mediante una válvula solenoide, figura 3.17, la cual funge como una compuerta para permitir o no el paso de la muestra de gas.

Figura 3.17. Válvula solenoide.

- ✓ **Módulo de bombeo**. Se encarga de transportar la muestra de gases que provienen del *trifilter*.

- ✓ **Módulo antiretorno.** Dispone de una válvula *check*, figura 3.18, la cual permite que los gases circulen en una sola dirección, evitando su regreso al sistema.

Figura 3.18. Válvula *check*.

- ✓ **Módulo de regulación de flujo**. Se trata de un orificio mediante el cual se regula el flujo de la muestra de gases, de tal forma que éste no exceda el tamaño requerido por las cámaras de análisis.

- ✓ **Módulo detector de fugas.** Sirve para detectar de forma oportuna cuando existe una fuga en el sistema neumático, o bien, si éste está obstruido.

- ✓ **Sonda de muestreo**. Es una manguera de fabricación especial, figura 3.19, por donde los gases de la muestra viajan del escape del automóvil hasta la banca. Mide 6 [m] de largo y su diámetro es de 1/4 “.

Figura 3.19. Sonda de muestreo.

Sensores

En la tercera etapa de operación del analizador de gases es donde, mediante sensores electrónicos, se realiza la medición de características particulares de los gases, para poder finalmente tener los resultados de la evaluación. Este conjunto de sensores es llamado "*Módulo de Análisis*" y es aquel que resguarda los sensores que participan activamente en la detección de gases específicos dentro de la muestra. Los elementos que lo conforman se muestran en la tabla 3.7.

ELEMENTO	*FUNCIÓN*
Celdas electroquímicas	Detecta la concentración de O y NOx en la muestra.
Celda infrarroja	Detecta las concentraciones de CO, CO_2 y HC.
Microbomba	Transporta cantidades pequeñas de cada muestra dentro de las cámaras para su respectivo análisis.
Sistema inteligente	Interpreta las lecturas de los sensores como concentraciones de gases, sirve como interfaz para la transmisión de los datos.
CAD	Convierte las señales analógicas a formato digital.
Ocho terminales salidas TTL	Controlan las válvulas y bombas del sistema neumático de la banca.

Tabla 3.7. Elementos del Módulo de Análisis.

Para conocer la distribución de los elementos anteriormente descritos, en la figura 3.20 se muestra un diagrama del módulo de análisis.

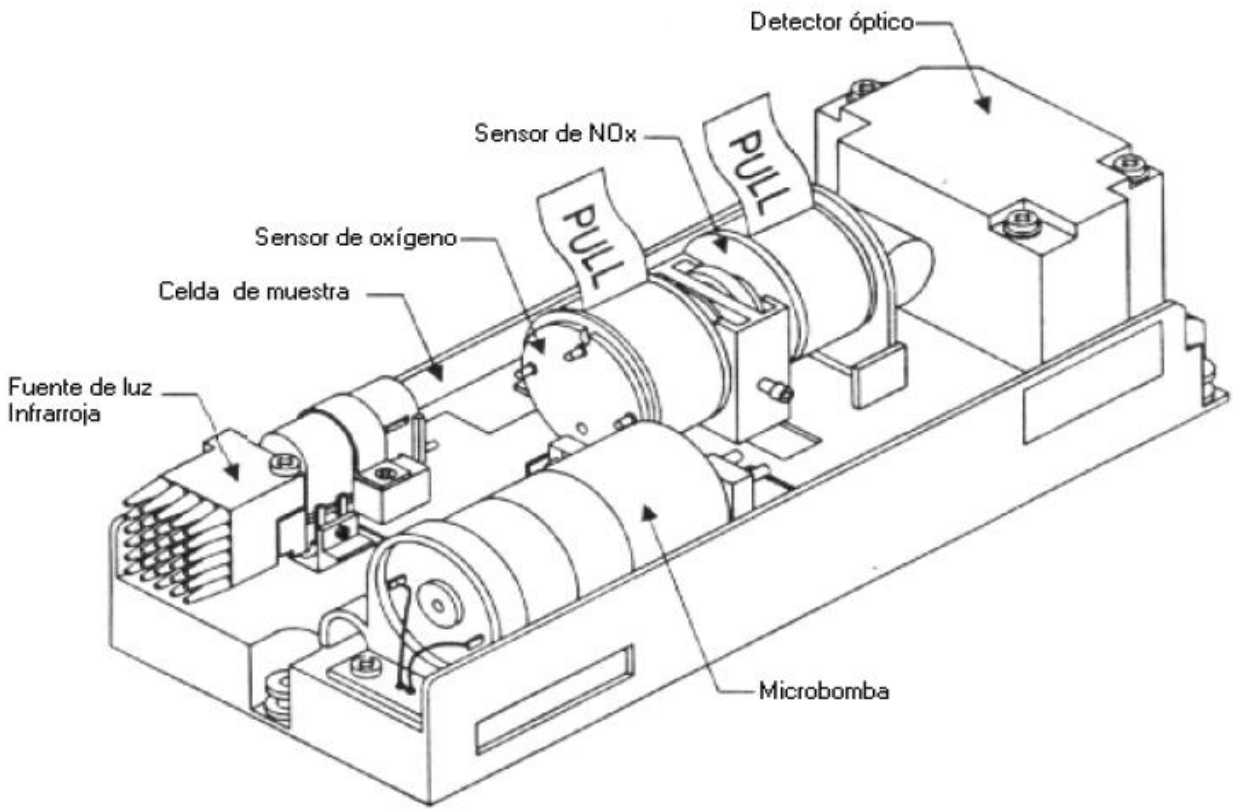


Figura 3.20. Módulo de Análisis.

Módulo de alimentación

Para que la banca analizadora de gases y sus elementos periféricos puedan funcionar, ésta cuenta con 3 fuentes de corriente directa, las cuales entregan voltajes de 5, 12 y -12 [V] y que para ser alimentadas deben ser conectadas a la red eléctrica, esto es 127 [V] en corriente alterna.

Bajo este contexto y con las especificaciones planteadas para el desarrollo del sistema, el analizador de gases deberá ser instalado dentro del vehículo en el cual se deseen realizar las mediciones de los gases contaminantes. Es por ello que para la instalación e integración de las pruebas con el equipo CYCLE-DAQ, es necesario adecuar el voltaje de alimentación del vehículo mediante un inversor de corriente directa a corriente alterna (CD-CA), 12 [VDC] a 127 [VAC] y con una potencia de 600[W], figura 3.21, suficiente para energizar a la banca analizadora de gases y sus dispositivos periféricos (microbomba, bombas de aire, válvula solenoide), los cuales requieren de 300 [W]. La fuente de CD será el voltaje de una batería para automóvil, la cual proporciona 12 [V] en corriente directa mientras el vehículo está apagado y 14 [V] cuando éste se encuentra en marcha.

Figura 3.21. Inversor CD-CA de 600 [W].

En la tabla 3.8 se muestran las características generales del inversor CD-CA.

CARACTERÍSTICAS DEL INVERSOR DE 600 [W]
Alimentación: 10-14 [VDC]
Salida: 100-120 [VAC] @ 60 [Hz]
Potencia: 600[W] máximo
Contacto de 2 y 3 polos
Ventilador
Forma de onda de salida: senoidal modificada
Dimensiones: 13 x 24 x 7 [cm]
Peso: 603[g]

Tabla 3.8. Características del inversor de 600 [W].

Descripción funcional del analizador de gases

El proceso para la adquisición de los datos de emisiones contaminantes es iniciado cuando una muestra de gas proveniente directamente del escape de un vehículo automotor, es conducida al analizador de gases ANDROS 6600 mediante la sonda de muestreo.

En la primera fase se retienen partículas sólidas de mayor tamaño inmersas en la muestra, para ello se ocupan algunos sistemas de filtrado. Primero pasa por la trampa de agua, la cual utiliza un filtro de grado seis para partículas de 0.3 [µm], posteriormente la muestra pasa al *trifilter* que permite el paso

a partículas no mayores a 1.2 [µm], además elimina la humedad, pues ésta puede acarrear problemas en la medición con los sensores.

De manera simultánea, el sistema neumático de transporte se encarga de la distribución desde la entrada del gas a la banca hasta su salida, en general este sistema es controlado por medio de ocho terminales TTL**,** y transductores de flujo**.** El módulo de bombeo trabaja con una taza aproximada de 750[ml/min] de la muestra de gases.

Una vez, tras la adquisición de un muestra libre, el gas se entrega a la celda de prueba para realizar la medición de la concentración del HC, CO y CO_2, específicamente con la celda infrarroja y la absorción de luz. La presión y temperatura de la muestra de gases se miden por medio de sus respectivos sensores con el objetivo de garantizar que la banca se encuentra operando correctamente; en caso de que la operación no sea correcta, el usuario será notificado para que aborte el proceso de análisis. La concentración de O_2 y de NOx es medida por los sensores químicos, posteriormente la muestra de gases es expulsada del equipo.

En la figura 3.22 se muestra un diagrama en donde se puede apreciar el procedimiento funcional del analizador de gases.

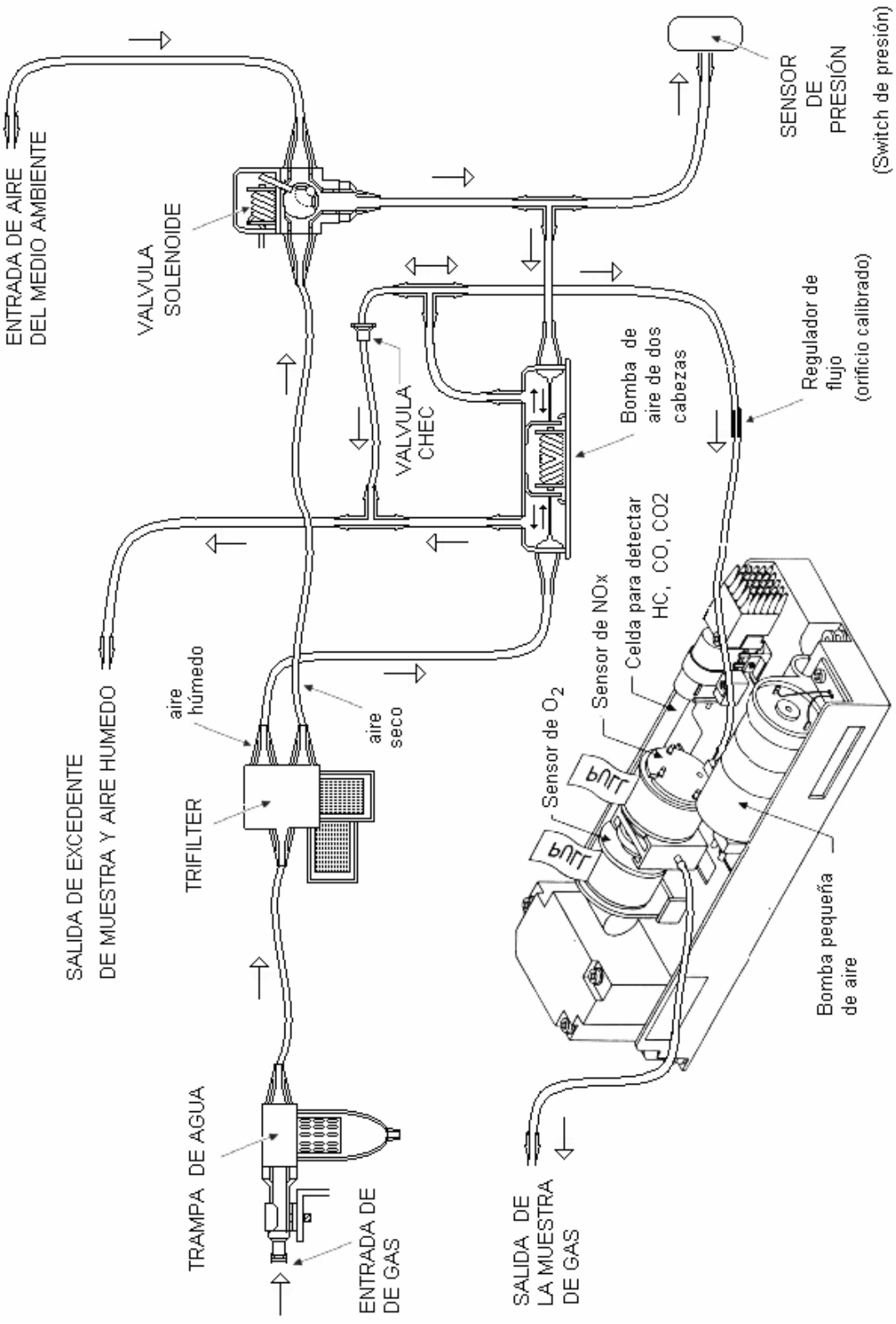


Figura 3.22. Diagrama funcional del ANDROS 6600.

Protocolo de comunicación

El analizador ANDROS 6600 cuenta con la interfaz RS232 asíncrona para establecer comunicación con otros dispositivos y admite dos velocidades de comunicación: 19,200 y 9,600 baudios por segundo. El dato tiene un formato de 1 bit de inicio, 8 bits de datos, sin bit de paridad, 1 bit de fin (*STOP*). Las señales eléctricas que maneja son: una línea para recepción de datos, una línea para transmisión y una señal de tierra.

El analizador de gases cuenta con un software comercial para establecer la comunicación entre el dispositivo y una PC, el cual fue provisto por el fabricante durante su adquisición; sin embargo, se habían desarrollado ya anteriormente diversas versiones de *software* para realizar la adquisición de los datos bajo la plataforma LabVIEW.

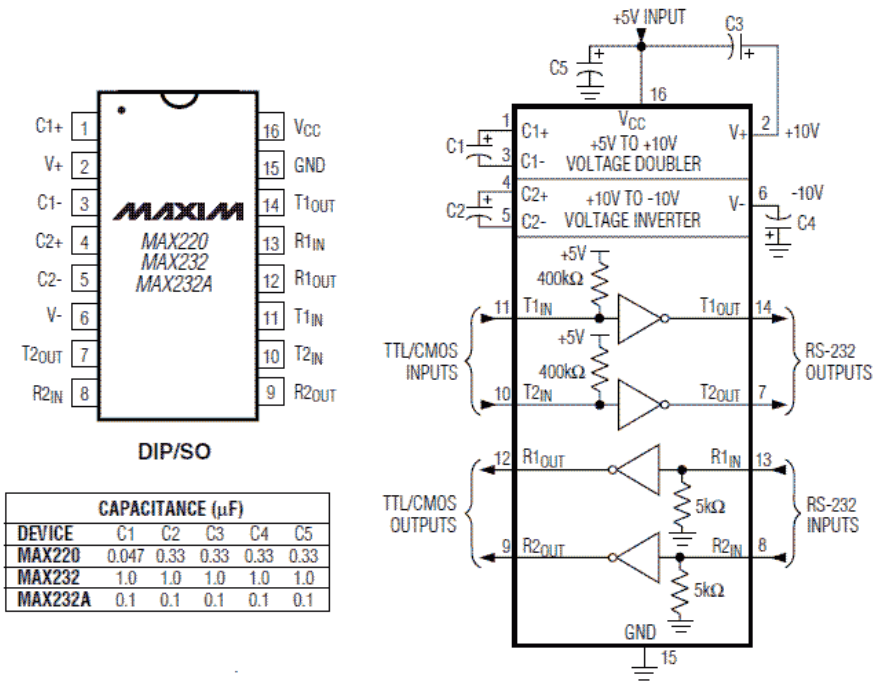
El ANDROS 6600 cuenta con una terminal DB-9 hembra, para poder ser conectado con dispositivos externos, como computadoras personales, *laptops*, microprocesadores y microcontroladores.

3.1.5. Conexión microcontrolador – analizador de gases

La integración del ANDROS 6600 con el sistema se realiza a través de la tarjeta de desarrollo basada en microcontrolador, aprovechando las ventajas que ésta ofrece. En específico la comunicación se realiza utilizando un puerto UART del microcontrolador ATmega2560, UART2; sin embargo, los niveles de voltaje que maneja la banca ANDROS 6600 no son los mismos aceptados por el microcontrolador, es por ello que se hace uso de un transceptor RS-232, llamado MAX 232.

El MAX232 es un circuito integrado que convierte los niveles de las líneas de un puerto serie RS232 a niveles TTL y viceversa. Lo interesante es que sólo necesita una alimentación de 5 [V], ya que genera internamente algunas tensiones que son necesarias para el estándar RS232.

En la figura 3.23 se muestra el transceptor MAX 232, así como la configuración propuesta en la hoja de especificaciones de dicho circuito integrado.



CAPACITANCE (µF)					
DEVICE	C1	C2	C3	C4	C5
MAX220	0.047	0.33	0.33	0.33	0.33
MAX232	1.0	1.0	1.0	1.0	1.0
MAX232A	0.1	0.1	0.1	0.1	0.1

Figura 3.23. Transceptor MAX 232.

Para su funcionamiento, el transceptor, requiere únicamente de cuatro capacitores electrolíticos de 1 [µF] y es polarizado con las líneas de 5 [V] y la terminal GND de la tarjeta de desarrollo; además, se hace uso de un conector DB-9 macho para poder conectar el equipo CYCLE-DAQ con el ANDROS 6600, utilizando cable *null modem*. La conexión de estos elementos se puede observar en la figura 3.24.

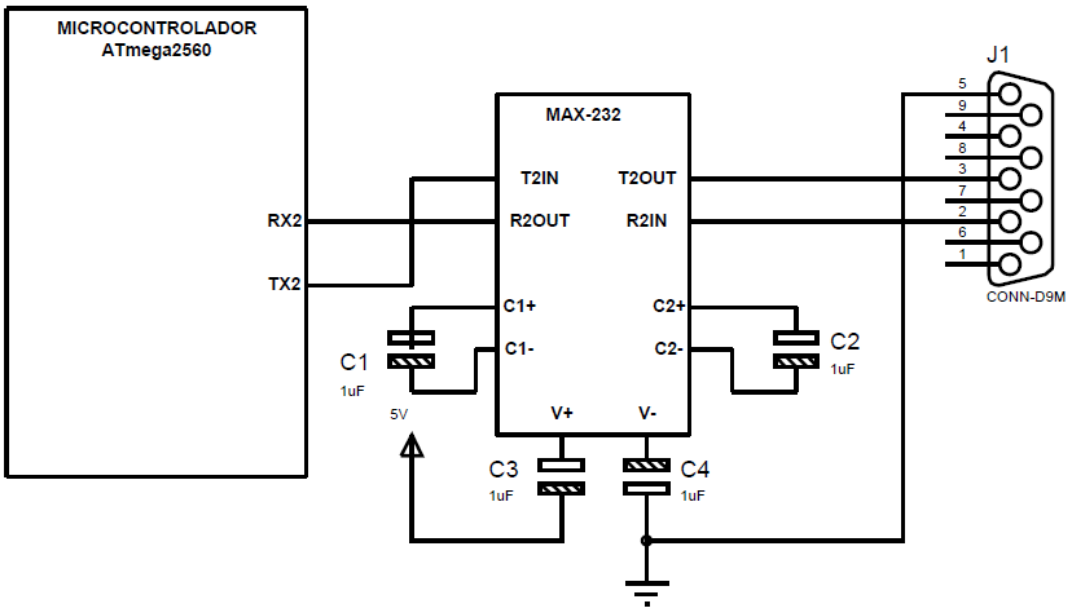


Figura 3.24. Diagrama de conexión del ANDROS 6600.

3.1.6. Sensor de voltaje de la batería del automóvil

Una variable más que debe ser monitoreada y registrada por el equipo *CYCLE-DAQ* es el voltaje de la batería del automóvil, dicho voltaje es tomado del encendedor electrónico que se encuentra en la mayoría de los automóviles. El objetivo de dicha medición es comprobar que cuando el vehículo es puesto en marcha, la concentración de los 5 principales gases contaminantes es mayor, y en dicho momento el voltaje de la batería del automóvil presenta una caída, disminuyendo su valor a aproximadamente 10 [V], en caso contrario, el voltaje máximo de ésta es de 14 [V].

La medición de dicho voltaje se efectúa a través de uno de los 16 canales para conversión A/D que posee el microcontrolador ATmega2560, la cual corresponde a la terminal analógica 15 de la tarjeta Arduino.

Como la tarjeta Arduino acepta máximo 5 [V] de entrada en sus terminales y el voltaje de la batería supera en gran medida dicho valor, fue necesario disminuirlo a un nivel aceptado por la tarjeta. Para realizar la medición se utilizó un circuito divisor de voltaje, implementado con dos resistencias conectadas en serie (R_1 = 4.7 [kΩ] y R_2= 1.5 [kΩ]. Siguiendo la ecuación 3.1 se observa que el voltaje a la salida del circuito divisor de tensión es de 2.4 [V], cuando el voltaje de la batería es de 10 [V], mientras en la ecuación 3.2, el valor a la salida es de 3.3 [V] cuando la batería entrega 14 [V].

$$Vsalida_{10V} = \frac{R_2 * Vbatería}{R_2 + R_1} = \frac{1500\ [\Omega] * 10\ [V]}{1500\ [\Omega] + 4700\ [\Omega]} = 2.4\ [V] \ldots\ldots\ldots\ldots\ldots (3.1)$$

$$Vsalida_{14V} = \frac{R_2 * Vbatería}{R_2 + R_1} = \frac{1500\ [\Omega] * 14\ [V]}{1500\ [\Omega] + 4700\ [\Omega]} = 3.3\ [V] \ldots\ldots\ldots\ldots\ldots (3.2)$$

Además, con el objetivo de proteger al microcontrolador se colocó un diodo zener de 5.1 [V] @ 1 [W] de potencia, en paralelo a la segunda resistencia para que funja como regulador de voltaje y así evitar valores mayores a los 5.1 [V]. Para que el diodo zener funcione correctamente, debe trabajar en la zona de ruptura, dicha condición se cumple si: el voltaje de la batería del automóvil es mayor al voltaje zener (5.1 [V]) y la corriente suministrada por la batería se encuentre dentro del rango establecido por el fabricante del diodo, Izk = 1 [mA] e Izm = 49 [mA]. Para comprobar lo anterior, las ecuaciones 3.3 y 3.4 muestran los cálculos de la corriente mínima y máxima en el circuito divisor de tensión, para 10 y 14 [V] respectivamente.

$$Imínima = \frac{Vbatería - Vzener}{R_2} = \frac{10\ [V] - 5.1\ [V]}{1500\ [\Omega]} = 3.2\ [mA] \ldots\ldots\ldots\ldots\ldots (3.3)$$

$$Imáxima = \frac{Vbatería - Vzener}{R_2} = \frac{14[V] - 5.1\ [V]}{1500\ [\Omega]} = 5.9\ [mA] \ldots\ldots\ldots\ldots\ldots (3.4)$$

Finalmente, es necesario corroborar que la potencia en el diodo zener no supere el límite establecido por el fabricante (1 [W]). Las ecuaciones 3.5 y 3.6 muestran dicha comprobación.

$$Pmínima = Vsalida_{10V} * Imínima = 2.4[V] * 3.2\ [mA] = 7.68\ [mW] \ldots\ldots\ldots\ldots\ldots (3.5)$$

$$Pmáxima = Vsalida_{14V} * Imáxima = 3.3[V] * 5.9\ [mA] = 19.74\ [mW] \ldots\ldots\ldots\ldots\ldots (3.6)$$

En la figura 3.25 se muestra el diagrama electrónico del divisor de voltaje y su conexión con el microcontrolador, a través de la terminal ADC 15.

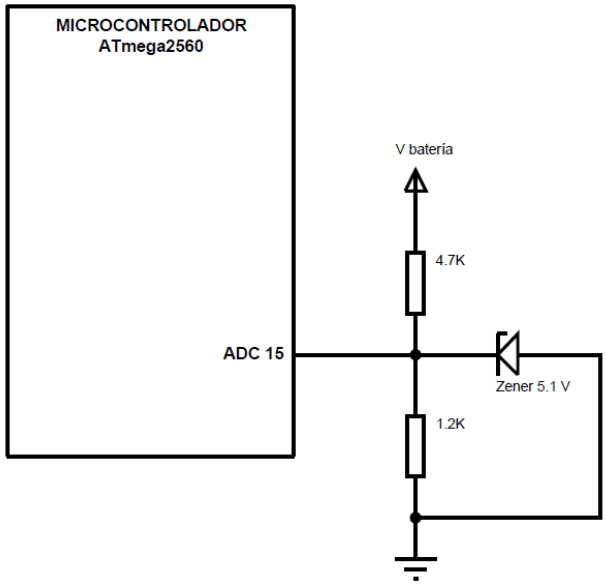


Figura 3.25. Diagrama de conexión del sensor de voltaje de la batería.

3.1.7. Tarjeta de memoria micro SD Card

Al ser el equipo *CYCLE-DAQ* un SARD, es imprescindible efectuar el registro de las variables medidas. Para ello se hace uso de una memoria SD, en una de sus más recientes variantes, el tipo microSD Card. Entre las características de este tipo de memorias están: bajo costo, tamaño reducido y facilidad para adquirirla.

Para interactuar con las tarjetas microSD es necesario definir sus particulares principales: voltaje de operación, disposición de terminales y protocolos de comunicación, necesarios para realizar las operaciones básicas sobre la memoria (lectura y escritura).

Protocolo SPI

SPI (*Serial Periphal Interface*), es un estándar de cuatro líneas, establecido por Motorola, sobre el cual se transmiten paquetes de información de 8 bits. Cada una de estas líneas porta la información

entre los diferentes dispositivos conectados al bus. Cada dispositivo conectado al bus puede actuar como transmisor y receptor al mismo tiempo, por lo que este tipo de comunicación serial es *full duplex*. Dos de estas líneas trasfieren los datos (una en cada dirección), llamadas: *Master Output Slave Input* (MOSI) y *Master Input Slave Output* (MISO), la tercer línea es la del reloj (SCLK), mientras que la cuarta línea es para seleccionar a cada dispositivo esclavo, llamada *Slave Select* (SS*).*

Los dispositivos conectados al bus son definidos como maestro y esclavos. Un maestro es aquel que inicia la transferencia de información sobre el bus y genera las señales de reloj y control. Un esclavo es un dispositivo controlado por el maestro. Cada esclavo es controlado sobre el bus a través de la línea *Select Slave*, por lo tanto el esclavo es activado sólo cuando esta línea es seleccionada.

Un diagrama de bloques que permite comprender mejor el funcionamiento del protocolo SPI se muestra en la figura 3.26.

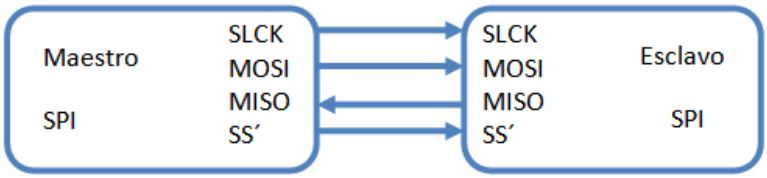


Figura 3.26. Protocolo SPI.

Para la comunicación entre maestro y esclavo se definen dos parámetros: CPOL (*Clock Polarity)* y CPHA *(Clock Phase)*, con los cuales se determina con respecto al pulso de reloj, el momento en el cual se considera válido un dato de entrada o se genera un dato de salida.

De manera breve, a continuación se describe la serie de pasos completa en una comunicación SPI:

1. Para iniciar la comunicación, el maestro configura el reloj usando una frecuencia menor o igual a la frecuencia máxima que soporta el esclavo. Estas frecuencias suelen estar en el rango de 1 a 70 [MHz].

2. El maestro a continuación pone a nivel bajo la señal *Slave Select* del esclavo para indicarle que se va a comunicar con él. Si es necesario, esperará un tiempo antes de iniciar la comunicación. Por ejemplo para permitir una conversión analógico/digital, el maestro esperará al menos ese tiempo antes de proseguir con el intercambio de información.

3. Durante cada ciclo de reloj se produce una comunicación en los dos sentidos, ya que por una parte el maestro transmite la información al esclavo a través de la línea MOSI y el esclavo responde al maestro a través de la línea MISO. Los datos son transmitidos comenzando por el bit más significativo (MSB) hasta el bit menos significativo (LSB) como se muestra en la figura 3.27.

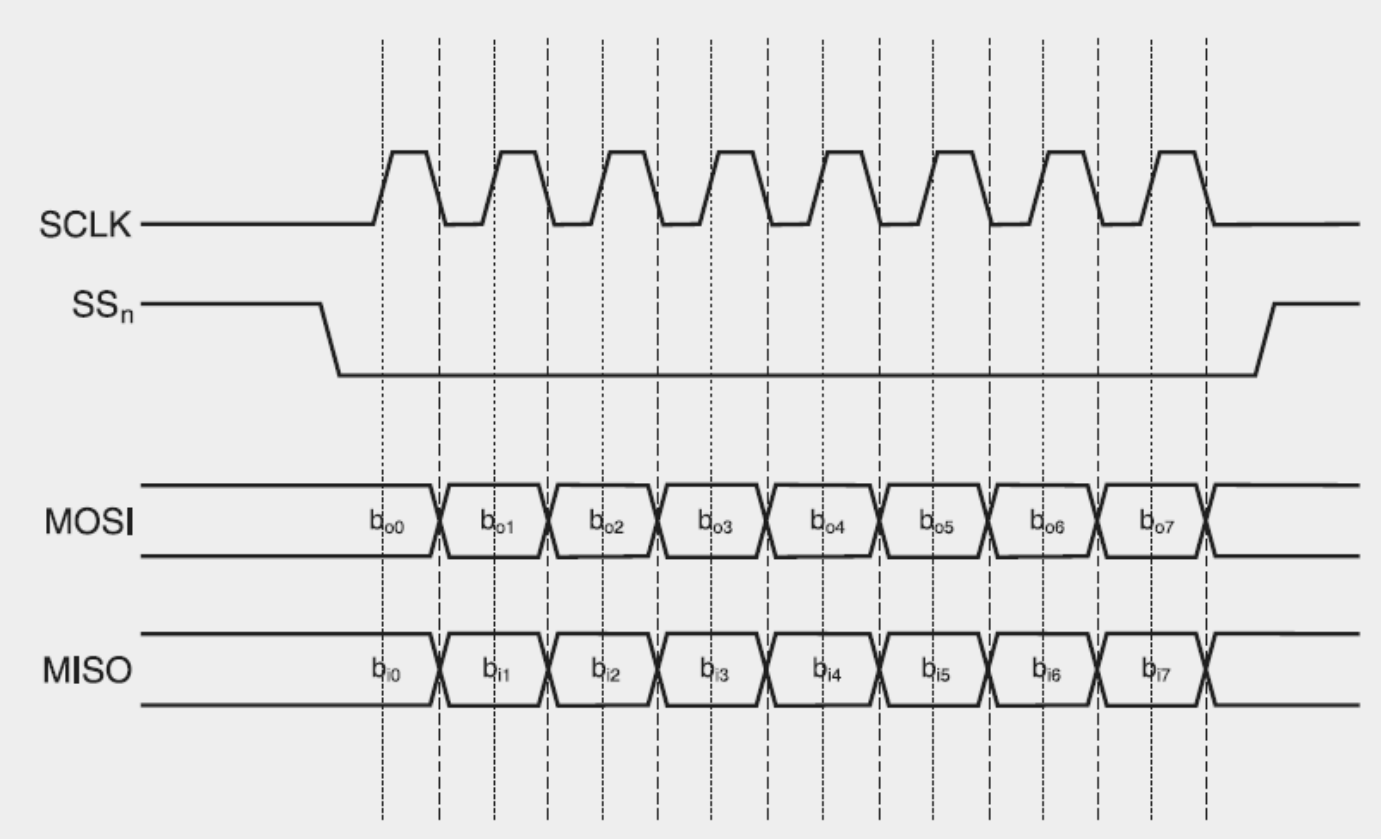


Figura 3.27. Diagrama de tiempos del protocolo SPI.

4. Cuando ya no quedan datos que transmitir, el maestro deja de accionar la señal de reloj y normalmente vuelve a colocar a nivel alto la señal de *Slave Select*, para así deseleccionar al dispositivo.

5. Cualquier dispositivo que no tenga a nivel bajo su señal de selección, ignorará los movimientos que haya en las líneas MISO y MOSI, con lo que podemos tener distintos dispositivos conectados a esas mismas líneas, sin que éstas interfieran en la comunicación.

6. Además de la frecuencia de reloj, el maestro también puede configurar la polaridad y la fase de la señal de reloj con respecto a los datos, mediante las líneas CPOL y CPHA. De acuerdo al valor de dichas líneas se tiene 4 modos de configuración del SPI, éstas son:

- Si CPOL = 0, el valor base de la señal de reloj es el nivel bajo.
  - Si CPHA = 0, los datos se capturan en el flanco de subida de la señal de reloj, y se propagan en el flanco de bajada. Este modo se denomina **modo 0**.
  - Si CPHA = 1, los datos de capturan en el flanco de bajada de la señal de reloj, y se propagan en el flanco de subida. Este modo se denomina **modo 1**.

- Si CPOL = 1, el valor base de la señal de reloj es el nivel alto.
  - Si CPHA = 0, los datos de capturan en el flanco de bajada de la señal de reloj, y se propagan en el flanco de subida. Este modo se denomina **modo 2.**
  - Si CPHA = 1, los datos se capturan en el flanco de subida de la señal de reloj, y se propagan en el flanco de bajada. Este modo se denomina **modo 3.**

La ventaja de utilizar el modo SPI en las memorias SD radica en la capacidad que se tiene de implementar a un microcontrolador como dispositivo maestro, facilitando así el diseño y la programación de la comunicación entre ambos dispositivos. La desventaja que se tiene es que en dicho protocolo no se tiene una señal para saber si la transferencia de los datos ha sido correcta, como en el caso del bus I^2C.

Conexión de la tarjeta microSD

Una vez comprendidos los aspectos necesarios sobre el protocolo de comunicación SPI, es conveniente conocer a grandes rasgos la estructura interna de la tarjeta microSD. En la figura 3.28 se muestra el diagrama de bloques de los elementos funcionales que componen a la tarjeta de memoria mencionada.[38]

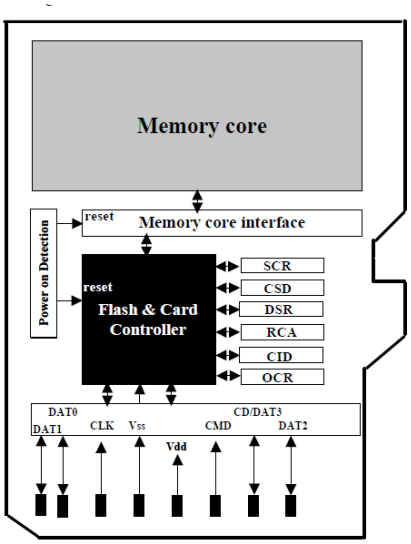
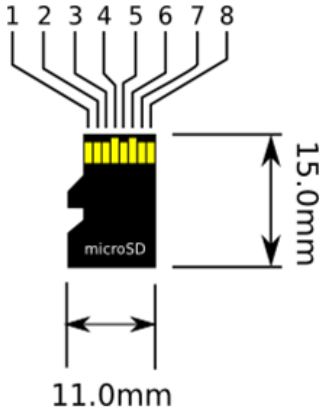


Figura 3.28. Diagrama de bloques de la tarjeta microSD.

[38] MicroSD Card™ Product SpecificationVersion 1.0. TwinMOS Techonlogy Inc. 2001

La distribución de las terminales de la memoria microSD, así como la función de cada una de ellas, se muestra en la figura 3.29.



No. De terminal	Nombre	Descripción
1	RSV	*Reserved*
2	CS	*Chip Select*
3	DI	*Data In*
4	Vdd	*Supply Voltage*
5	SCKL	*Clock*
6	Vss	*Ground*
7	DO	*Data out*
8	RSV	*Reserved*

Figura 3.29. Distribución de terminales de la tarjeta micro SD.

Para poder realizar la conexión de la tarjeta de memoria microSD con la tarjeta Arduino MEGA es necesario mencionar que, de acuerdo a la hoja de especificaciones de la tarjeta microSD, ésta deber ser alimentada con un voltaje de entre 2.7 a 3.3 [V]. Dicha alimentación será proporcionada mediante la terminal de 3.3 [V] de la tarjeta de de desarrollo. Aunado a esto, las señales del bus SPI también deberán ser de 0 [V] para una cero lógico y de 3.3 [V] para el uno lógico; sin embargo, las terminales de la tarjeta Arduino a las cuales las línea de la microSD son conectadas, trabajan en un rango de 0 a 5 [V]. Para adecuar los niveles de voltaje necesarios y asegurar el correcto funcionamiento de la memoria microSD se coloca un divisor por cada línea de transmisión de datos (DO, DI) y la de reloj (SCLK), lo cual permitirá reducir los 5 [V] provenientes de las terminales de Arduino a un valor de 3.3 [V].

En la figura 3.30 se muestra el diagrama esquemático de conexión entre la memoria microSD y el microcontrolador ATmega2560.

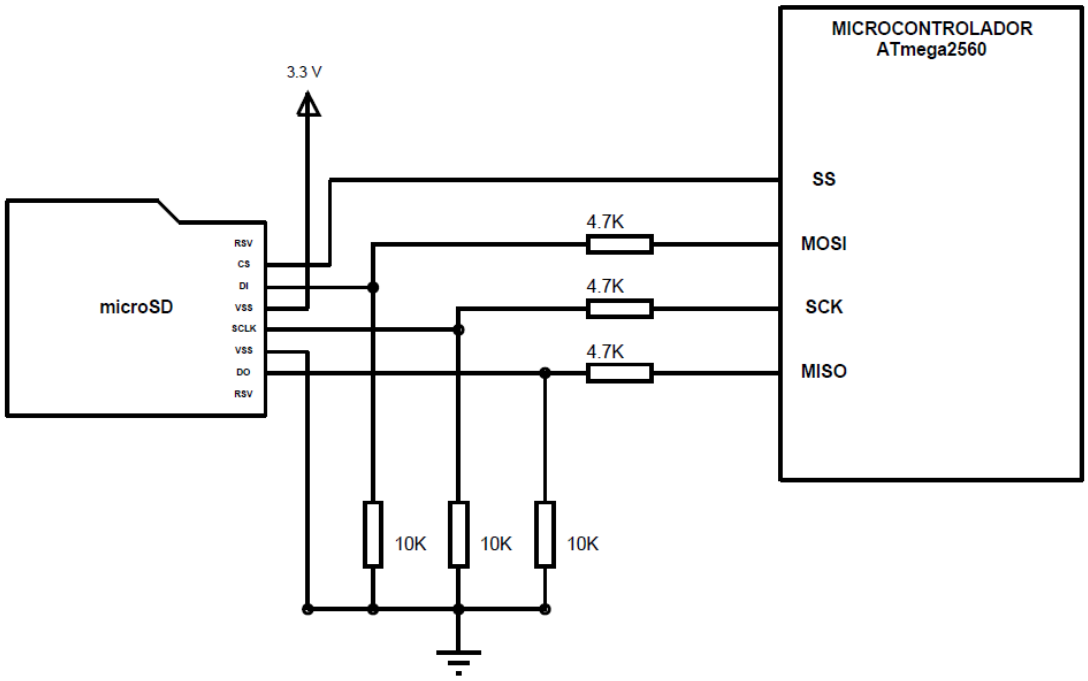


Figura 3.30. Diagrama de conexión de la memoria microSD.

3.1.8. *Display* gráfico

Debido a las características del equipo *CYCLE-DAQ*, es necesario que el usuario del equipo pueda visualizar las variables medidas de mayor interés, así como el estado de conexión de los dispositivos periféricos y sensores descritos anteriormente. Por tal motivo, se hace uso de un *display* de cristal líquido gráfico (*Grpahic Liquid Crystal Display*: *GLCD*), cuya resolución es medida en pixeles.

El *display* seleccionado es el GDM12864H de 128x64 pixeles. Contiene dos controladores KS0108 de Samsung. Cada controlador KS0108 es independiente, es decir, no transmiten información entre ellos. Para elegir a qué controlador hablarle, se usan las dos líneas de control llamadas CS1 y CS2 (CS = *Chip Select*).

En la figura 3.31 se muestra la apariencia del *display* utilizado, mientras que en la tabla 3.9 se hace referencia a la distribución de las terminales de dicho dispositivo y su conexión asociada con la tarjeta de desarrollo Arduino MEGA.

Figura 3.31. *Display* gráfico GDM12864H.

No. Terminal	*Nombre*	*Descripción*	*Terminal de Arduino MEGA*
1	Vdd	Voltaje de operación positivo	5 [V]
2	Vss	Tierra	GND
3	V0	Voltaje para contraste (variable)	--
4 -11	DB0-DB7	Bits de datos 0-8	*Analog0- Analog7*
12	CS2	*Chipselect* controlador 1	*Analog8*
13	CS1	*Chipselect* controlador 2	*Analog9*
14	/RES	Reset	*Analog10*
15	R/W	Lectura/escritura	*Analog11*
16	D/I	Control de comando	*Analog12*
17	E	Señal de habilitación	*Analog13*
18	Vee	Voltaje interno del *display*	--
19	A	Voltaje positivo para el LED interno	--
20	K	Tierra del para el LED interno	GND

Tabla 3.9. Terminales del *display* GDM12864H.

El *display* es alimentado mediante las terminales 1 y 2, Vdd y Vss respectivamente, con los 5[V] y la tierra proporcionados por la tarjeta Arduino.

La terminal V0 es la correspondiente al contraste del *display*, la cual permite ajustar el brillo del mismo y es controlada mediante un potenciómetro de 10 [kΩ], conectado entre Vss y la terminal 18 del

dispositivo llamada Vee. El *display* genera un voltaje interno negativo mediante esta línea. El diagrama esquemático de la conexión del potenciómetro es mostrado en la figura 3.32.

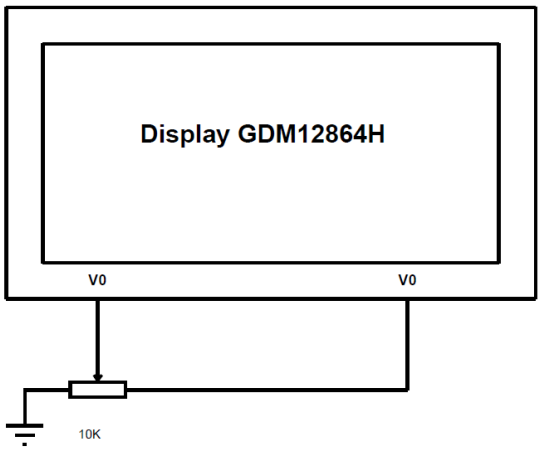


Figura 3.32. Diagrama de conexión para el contraste del display.

El bus de datos está conformado por las terminales DB0-DB7, mediante éstas, se leen y escriben los datos a transferir. Por diseño, estas líneas son conectadas a las terminales analógicas de la tarjeta de desarrollo, debido a que también pueden ser usadas como E/S digitales.

La terminal RES corresponde a la señal de *reset* del controlador KS0108, éste funciona mediante lógica negada.

Mediante la terminal R/W se elige cuál de las dos operaciones básicas sobre el *display* se desea efectuar; uno lógico corresponde a lectura y cero lógico a escritura.

La terminal de control D/I funciona para indicar al *display* si se le envía un dato o un comando. Si D/I se encuentra en alto significa que hay envío de datos, mientras que, si está en bajo, lo que se envía es un comando.

Para la habilitación del dispositivo se utiliza la terminal E, enviando a ésta un pulso de aproximadamente 500 [ns]. En caso de encontrarse en estado bajo, el *display* estará deshabilitado.

El *display* cuenta con luz de fondo, para dicho fin es necesario encender un LED integrado en el mismo dispositivo. Para ello se utilizan las terminales K y A del mismo, en la terminal K se proporciona la señal del voltaje de alimentación y en A para la de tierra.

3.1.9. Indicadores

Con la finalidad de que el usuario pueda conocer el estado de la operación del equipo CYCLE-DAQ, se colocaron tres LEDs. El primer LED, de color rojo, indica que el sistema se encuentra energizado. Un segundo LED, verde, indica que la memoria microSD ha sido reconocida y se encuentra en el proceso de almacenamiento de datos. Finalmente, un LED azul, indica que el receptor GPS se ha sincronizado con la constelación de satélites al parpadear cada segundo. En la tabla 3.10 se muestra la relación entre dichos LED, el proceso que indican y las terminales de la tarjeta de desarrollo en donde éstos son conectados.

LED	*Proceso indicado*	*Terminal de Arduino MEGA*
ROJO	Energía	5 [V]
VERDE	microSD Card	*Digital 32*
AZUL	GPS	*Digital 54*

Tabla 3.10. LEDs indicadores.

3.1.10. Módulo de alimentación

Debido a que el equipo CYCLE-DAQ debe ser autónomo, éste es alimentado con una batería recargable sellada de plomo que entrega 6 [V], suficiente tensión para alimentar a la tarjeta de desarrollo y además cuenta una capacidad de 4 [Ah], lo cual garantiza autonomía al equipo por varios días, sin necesidad de recargar la batería constantemente. Para asegurar que la capacidad de la batería es suficiente, se midió la corriente del circuito, obteniendo un valor de 157.0 [mA]. En la ecuación 3.7 se muestra el cálculo del tiempo de duración de la batería en relación a la capacidad de la misma y el consumo de corriente del circuito.

$$tiempo_{terórico}\ [h] = \frac{capacidad\ batería\ [Ah]}{consumo\ de\ corriente\ [A]} = \frac{4\ [Ah]}{157.0\ [mA]} = 25.47\ [h] \ldots\ldots\ldots\ldots (3.7)$$

El tiempo calculado con la ecuación anterior es teórico, en la práctica, se presentan pérdidas en la capacidad de la batería, por lo que el valor real aproximado corresponde al 90% del tiempo teórico y se obtiene con la ecuación 3.8.

$$tiempo_{real}\ [h] = 0.9 * tiempo_{terórico} = 0.9 * 25.47[h] = 22.92\ [h] \ldots\ldots\ldots\ldots (3.8)$$

De acuerdo a los cálculos anteriores, se muestra que el equipo CYCLE-DAQ puede operar durante 22.92 horas continuas, tiempo suficiente para realizar por varios días los muestreos necesarios

para desarrollar ciclos de manejo, considerando que dichos muestreos se efectúan en el Valle de México y duran aproximadamente 4 horas al día.

Una vez explicados cada uno de los elementos de *hardware* que conforma al equipo CYCLE-DAQ, podemos entonces definir un diagrama de bloques general del sistema, el cual se muestra en la figura 3.33.

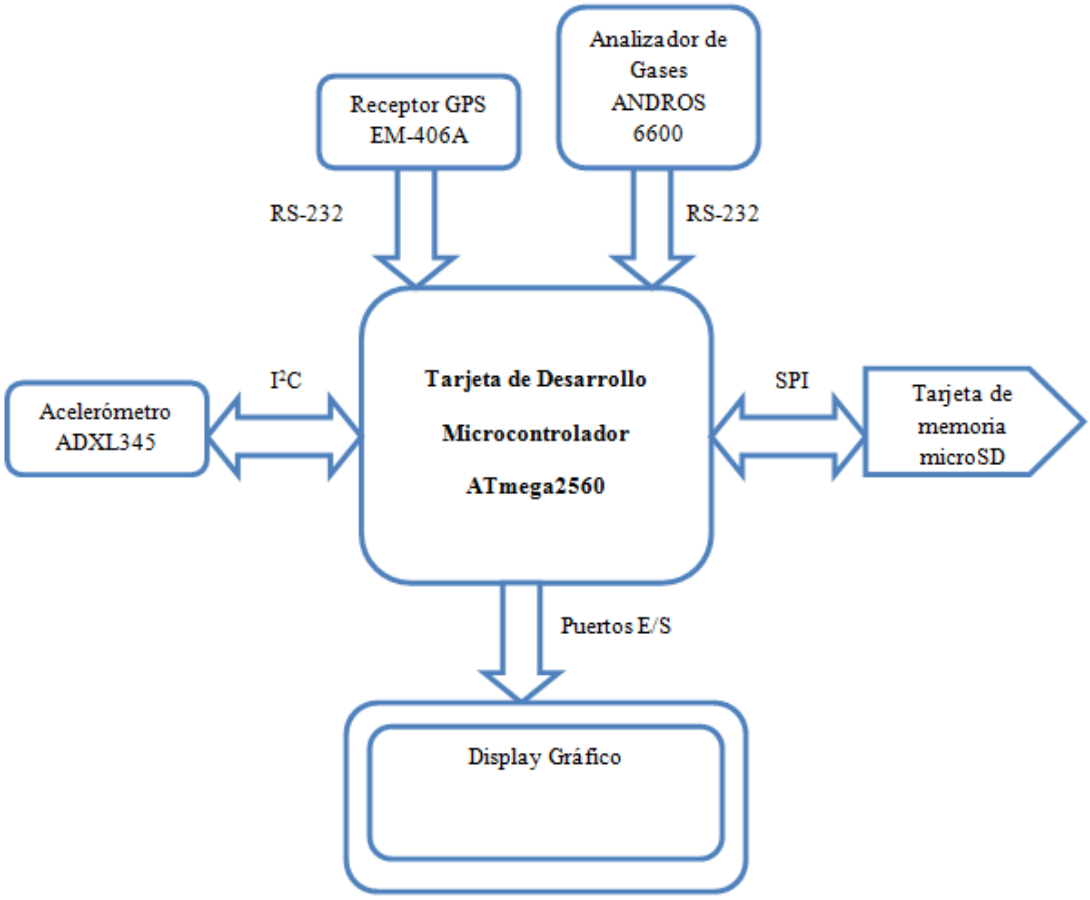


Figura 3.33. Diagrama de bloques del equipo CYCLE-DAQ.

3.2. Desarrollo del programa del microcontrolador

Para desarrollar proyectos con la tarjeta Arduino, en cualquiera de sus versiones, es necesario programar el microcontrolador asociado a ella. Para este fin los creadores de Arduino definieron un lenguaje de programación propio, el cual se basa en los lenguajes *C/C++*, conocidos mundialmente en el ámbito de la programación de sistemas. En este contexto se dice que la tarjeta Arduino es programada bajo la plataforma *Wiring*, la cual se concibe como el conjunto de funciones, variables y constantes que encapsulan el funcionamiento del hardware facilitando el uso del mismo.

Adicionalmente se pueden utilizar las características propias del lenguaje C/*C++* dentro del ambiente de desarrollo, lo que permite crear prototipos de funciones, punteros, clases y objetos e incluso utilizar lenguaje ensamblador y otras características propias de los microcontroladores AVR, como es el caso de la manipulación de registros.

Lo descrito anteriormente proporciona una gran flexibilidad al momento de crear proyectos complejos y, gracias al ambiente de desarrollo, permite crear aplicaciones rápidamente. También es posible desarrollar librerías que pueden ser instaladas dentro del ambiente de desarrollo y existe un gran número de ellas en Internet, que permiten el manejo de sensores, actuadores, comunicación serial, pantallas LCD, GPS y muchos otros componentes. Debido al gran éxito de las tarjetas Arduino a nivel mundial, el cual en gran medida tiene su base en su lenguaje de programación, se ha creado una cita que a la letra dice*: "LA INTELIGENCIA DE ARDUINO RADICA EN SU LENGUAJE DE PROGRAMACIÓN".*

En cuanto a la programación de Arduino, se conoce como *Sketch* a los programas diseñados por los desarrolladores y que radican en la memoria del microcontrolador AVR, o bien, es la unidad de código que se carga y ejecuta en la tarjeta Arduino. Para las personas que han trabajado con otros microcontroladores puede ser confuso utilizar el término *Sketch*, ya que generalmente se conoce como programa, *firmware* o código. Para el caso de este trabajo se denominará programa al código implementado en el equipo CYCLE-DAQ.

En Arduino, un programa básicamente se divide en dos partes:

1. *void* ***setup****():* Es la primera parte del programa, en ella se realiza la declaración de variables y en su caso, se inicializan, también se utiliza para definir las terminales de la tarjeta que serán utilizadas y declararlas como entradas o salidas, así como la configuración de los puertos de comunicación. Esta parte debe considerarse como un conjunto de código que solo se ejecuta una vez cuando la tarjeta es conectada a una fuente de alimentación, o cuando se presiona el botón de *Reset.* En resumen, esta parte corresponde a la configuración inicial de la tarjeta.

2. *void* ***loop****():* Es la parte principal del programa, pues en ésta, es necesario escribir el código que el microcontrolador ejecutará de manera iterativa, es decir, esta parte se repetirá una y otra vez mientras la tarjeta Arduino se encuentre energizada. Este segmento es el encargado de realizar las operaciones propias de la aplicación desarrollada, esto es: lectura de señales, activación y control de dispositivos, implementación de algoritmos, etc.

3.2.1. Entorno de desarrollo integrado

El entorno de desarrollo integrado de Arduino o *Integrated Development Enviroment (IDE)*, es de código abierto, éste permite escribir fácil e intuitivamente el código y cargarlo a la tarjeta. El *IDE* está escrito en lenguaje Java y permite realizar funciones como: compilar el código, ejecutar el código sobre la tarjeta, lectura y escritura al puerto serial de Arduino, visualización de ejemplos, configuración de la versión de la tarjeta a utilizar, entre otras. En la figura 3.34 se muestra la apariencia del entorno de desarrollo bajo Windows y sus principales partes.

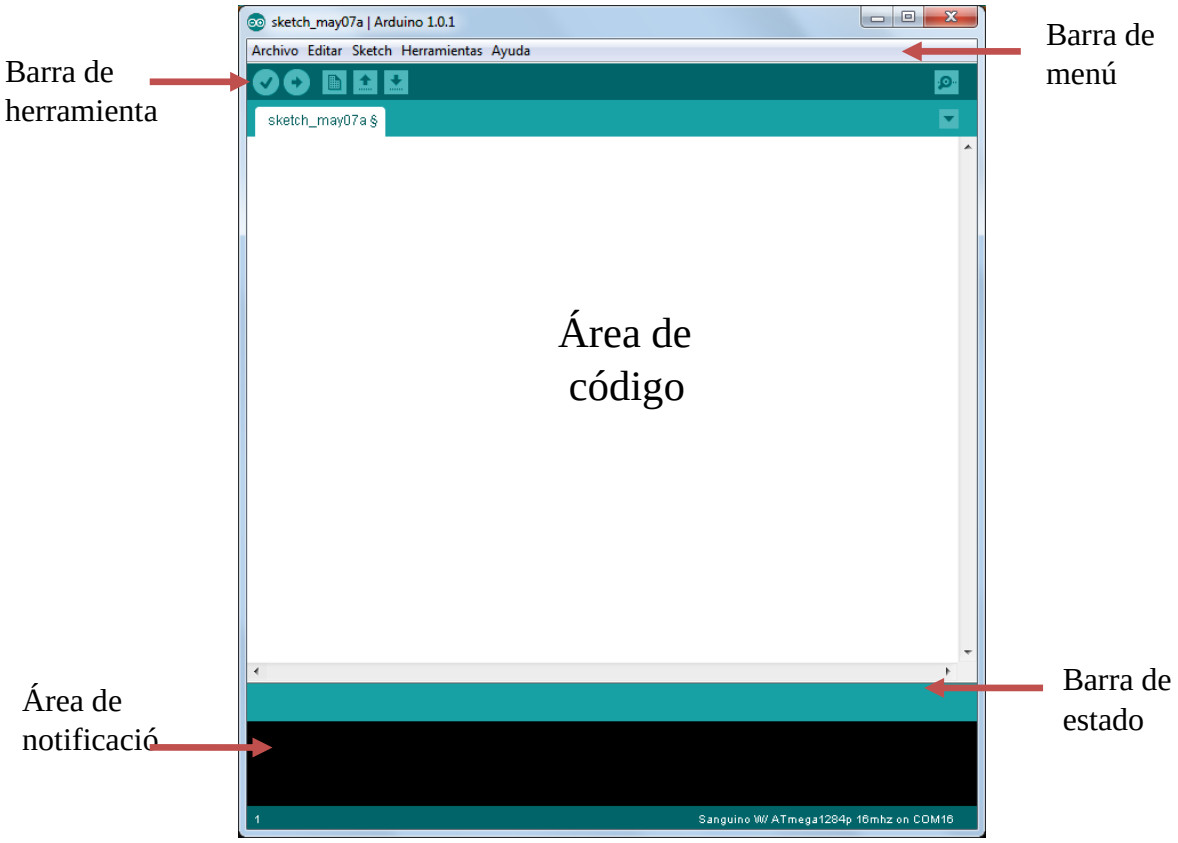


Figura 3.34. Entorno de Desarrollo Integrado de Arduino.

Para comprender mejor la función de cada una de las partes del *IDE*, a continuación se describe paso a paso el procedimiento para desarrollar un programa y cargarlo en la tarjeta Arduino.

1) **Selección la tarjeta Arduino:** Este proceso se lleva acabo de la siguiente manera: se hace *click* sobre la barra de menú, en la opción "Herramientas" y posteriormente se selecciona "Tarjeta". Se desplegará una lista con las versiones de Arduino soportadas por el *IDE*, el usuario deberá seleccionar la tarjeta que programará. En la figura 3.35 se muestra el proceso de selección de la tarjeta Arduino.

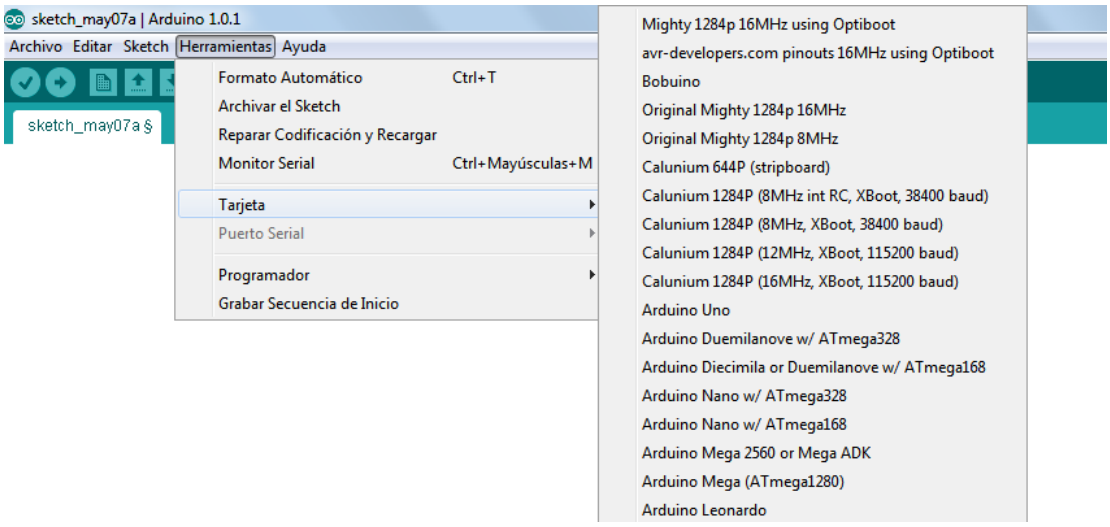


Figura 3.35. Selección de la tarjeta Arduino en el *IDE*.

2) **Puerto serial:** Cuando se conecta la tarjeta Arduino a una computadora personal mediante el cable USB, se instalan los controladores necesarios y el SO asigna a la tarjeta un puerto de comunicaciones virtual RS-232, llamado COM. Por tal motivo es importante seleccionar el puerto COM adecuado, en caso contrario, la tarjeta no podrá ser reconocida por el *IDE* y será imposible cargar el programa en ella. Dicho proceso se lleva acabo seleccionado la opción "Puerto Serial" del menú "Herramientas", como se muestra en la figura 3.36.

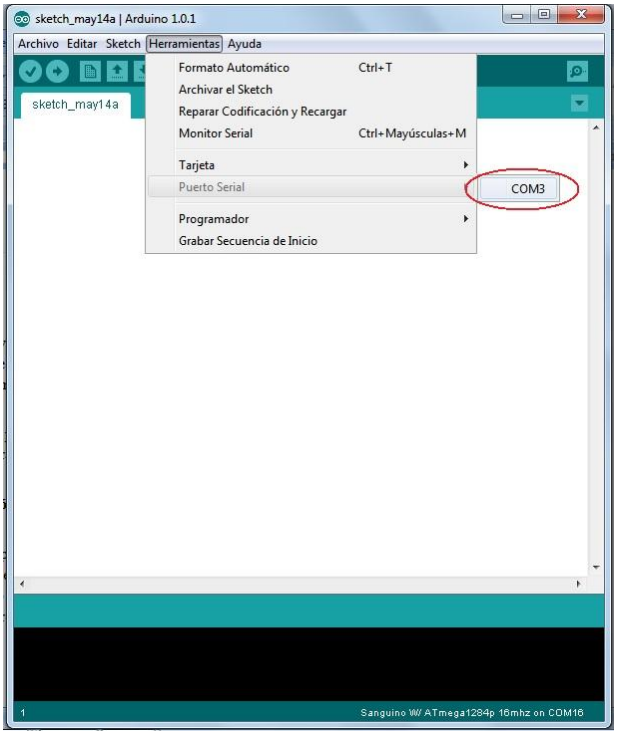


Figura 3.36. Selección del puerto serial

3) **Código:** El siguiente paso consiste en escribir el código a implementar en la tarjeta de desarrollo. Dicha operación se realiza en el área de código del *IDE*, como se muestra en la figura 3.37.

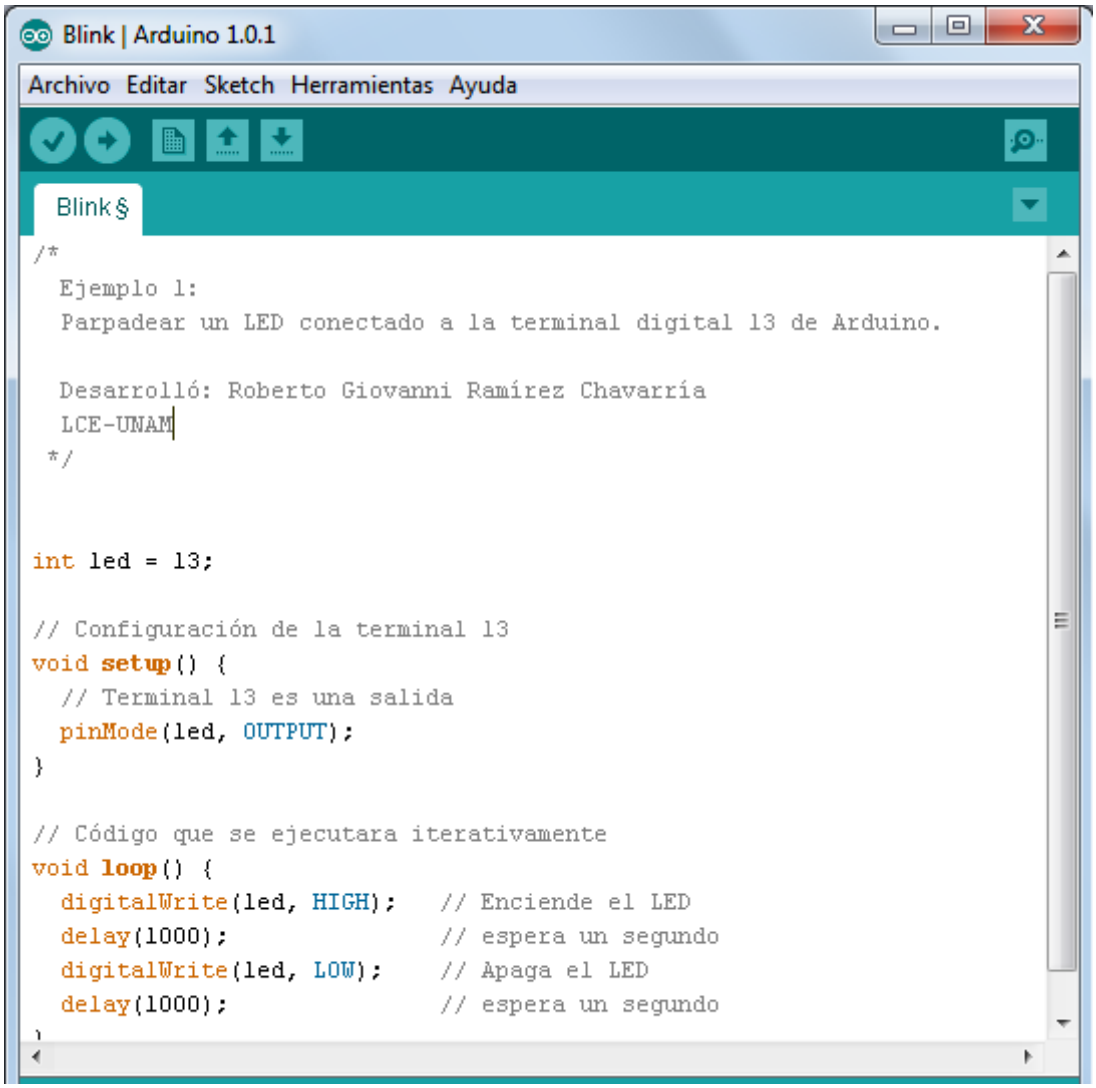


Figura 3.37. Escritura del código en el *IDE*.

4) **Compilación:** Una vez que el código ha sido escrito, es necesario verificar que no contenga errores y que la programación cumpla con las reglas establecidas por el lenguaje. Para ello, es necesario hacer *click* en el ícono  llamado "Verificar", de la barra de herramientas. En caso de contener errores, el IDE notificará al desarrollador sobre la posible causa de éstos y una breve descripción de los mismos como se muestra en la figura 3.38. Dicha descripción de errores se realiza en el área de notificación.

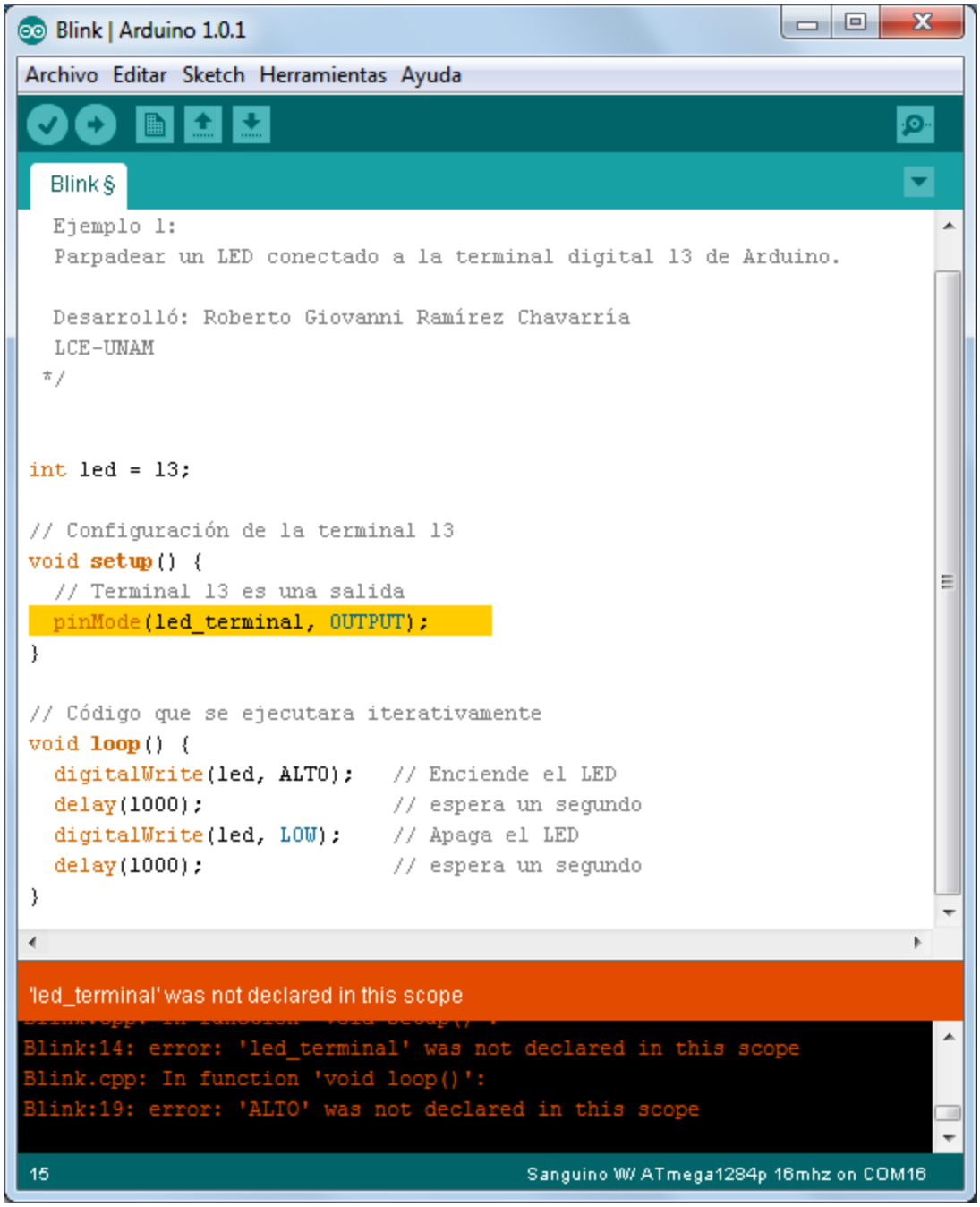


Figura 3.38. Notificación de errores en el *IDE*.

5) **Carga del programa:** Una vez que se ha verificado que el código es correcto, es decir, ha sido compilado, el código ahora será llamado programa y éste deberá ser descargado a la tarjeta a través del puerto serial (COM), en este momento es cuando el programa se graba en el microcontrolador asociado a la tarjeta de desarrollo. Para realizar el proceso de carga, basta con hacer click sobre el ícono "Cargar"  de la barra de herramientas.

Cabe recordar que, el microcontrolador de la tarjeta Arduino, es capaz de "auto-programarse", es decir, no necesita de un programador externo u otro medio para grabar en su memoria flash el programa, en vez de ello, el código compilado es transferido al microcontrolador a través de su puerto serial. Dicho proceso se lleva a cabo gracias a una pequeña parte de código llamado *Bootloader*, el cual reside en localidades reservadas de la memoria *flash* del microcontrolador AVR.

6) **Ejecución del programa:** Finalmente, una vez que el programa se ha grabado en el microcontrolador, éste comenzará su ejecución iterativa y se detendrá hasta que la tarjeta sea desconectada de la computadora o reiniciará la ejecución cuando se presione el botón de *Reset.*

3.2.2. Estructura del programa

Una vez propuestos y descritos los componentes de hardware implementados en el equipo CYCLE-DAQ, es necesario describir cómo es que éstos interactúan entre sí, dicha interacción se realiza mediante la programación del microcontrolador bajo el ambiente del *IDE* de Arduino. La estructura del programa principal se compone de funciones que ejecutan una tarea determinada para cada dispositivo. A continuación se listan las tareas programadas para el correcto funcionamiento de cada elemento y que posteriormente serán explicadas a detalle.

- **Receptor GPS**

  - ✓ Comunicación mediante RS-232
  - ✓ Adquisición de las sentencias NMEA
  - ✓ Decodificación de la sentencias NMEA
  - ✓ Procesamiento de los datos
  - ✓ Cálculo de la distancia recorrida

- **Acelerómetro ADXL345**

  - ✓ Comunicación mediante I^2C
  - ✓ Inicialización del dispositivo
  - ✓ Verificación de la conexión
  - ✓ Parámetros de medición
  - ✓ Conversión de unidades [g] a grados.

- **Analizador de Gases ANDROS 6600**

  - ✓ Comunicación RS-232

- ✓ Verificación de la conexión
- ✓ Adquisición de los datos provenientes del analizador
- ✓ Decodificación de los datos
- ✓ Manipulación de los datos para obtener los resultados

- **Sensor de voltaje de la batería**

  - ✓ Lectura del convertidor A/D
  - ✓ Escalamiento del valor de voltaje

- **Memoria microSD**

  - ✓ Comunicación por SPI
  - ✓ Inicialización de la memoria
  - ✓ Verificación continua del estado de conexión
  - ✓ Registro de las variables

- **Display Gráfico**

  - ✓ Pantalla de bienvenida
  - ✓ Pantalla de configuración de dispositivos
  - ✓ Mostrar estado inicial de conexión de los dispositivos
  - ✓ Visualización de variables segundo a segundo

- **Indicadores**

  - ✓ LED indicador del estado del receptor GPS
  - ✓ LED indicador de reconocimiento de la memoria microSD y registro de datos

Cabe mencionar que las tareas citadas anteriormente se pueden encontrar en el *void setup()* o en el *void loop()*, debido a que algunas de ellas solo se ejecutan una vez, cuando el equipo es energizado; mientras que otras se ejecutan iterativamente segundo a segundo.

3.2.3. Manejo del receptor GPS

El receptor GPS EM-406A es el principal dispositivo periférico del equipo CYCLE-DAQ, ya que es el encargado de entregar las variables más importantes para el desarrollo de ciclos de manejo, además de que sincroniza 1[s] el periodo de muestreo de la adquisición y registro de datos.

Para establecer la comunicación entre el receptor GPS y la tarjeta de desarrollo, se hace uso de las funciones para el manejo del puerto serial UART, dichas funciones permiten: iniciar al puerto serial, verificar que se encuentre disponible, así como leer al puerto y escribir en él.

La primer tarea es establecer la comunicación serial consiste en configurar la velocidad de transmisión a 4800 baudios, mediante la función *Serial.begin(4800)*. Como segunda tarea, se realiza la adquisición de los datos mediante la función *readline_gps()*, la cual se encarga de leer los datos disponibles en el puerto serial y de esta manera obtener todas las sentencias NMEA provenientes del receptor GPS . Durante la ejecución de dicha función, primero, es necesario verificar si existen datos disponibles en el puerto serial mediante la función *Serial.available()*, la cual devuelve "1" en caso de existir datos y "0" en caso contrario. Una vez verificado el estado del puerto, se lee la cadena proveniente del receptor con la función *Serial.read()*, la cual permitirá almacenar la sentencia disponible en un arreglo de caracteres.

En la figura 3.39 se muestra el diagrama de flujo con el procedimiento que ejecuta la función *readline_gps ( )*.

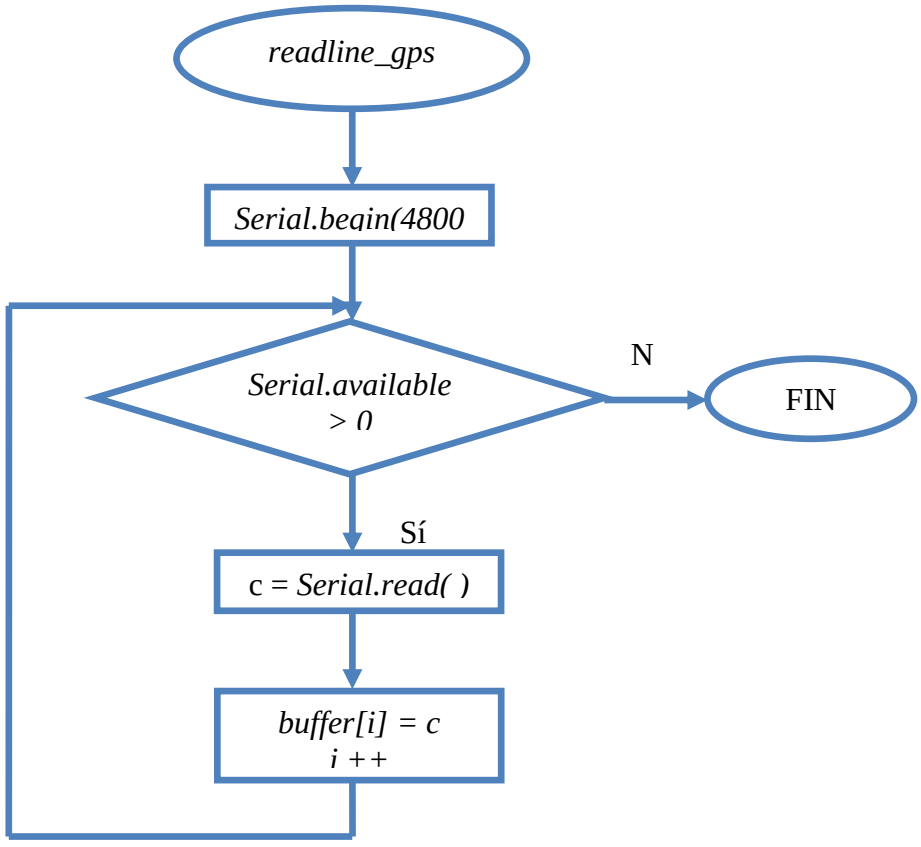


Figura 3.39. Diagrama de flujo de la función *readline_gps ( )*.

La tercera tarea es la decodificación de las sentencias NMEA. La decodificación es un proceso necesario, el cual se divide en dos partes, la primera corresponde a la conversión de tipo de dato y la segunda a la separación de los datos.

Las sentencias arrojadas por el receptor GPS en su estado inicial son sólo cadenas de caracteres y deben ser convertidos a formato numérico para poder realizar operaciones con ellos, como es el caso de la conversión de unidades de velocidad, de [nudos] a [km/h] y para el cálculo de la distancia recorrida por el vehículo, ésta es la primera parte de la decodificación de los datos.

Aunado a lo anterior, los datos se encuentran delimitados por una coma, por lo que es necesario separarlos para que éstos puedan ser almacenados en variables y de esta manera será posible manipularlos individualmente.

En la figura 3.40 se muestran las sentencias NMEA enviadas por el GPS antes de ser decodificadas.

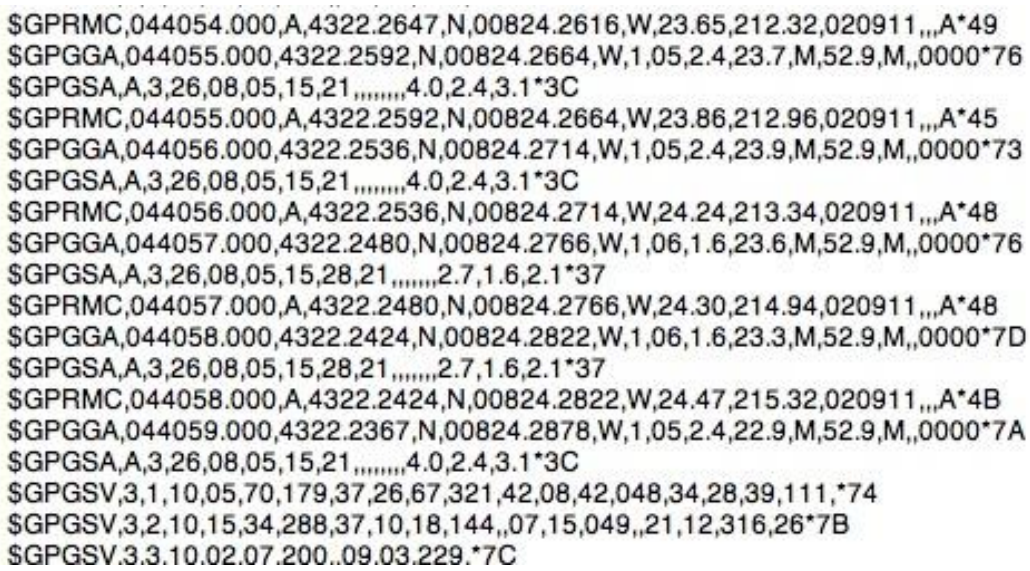


Figura 3.40. Sentencias NMEA arrojadas por el receptor GPS.

Para los fines de decodificación, se programaron funciones bajo el estándar ANSI C, mediante el manejo de cadenas y caracteres con las librerías *stdlib.h y ctype.h.*

La conversión de tipo carácter a tipo de dato entero se realiza con la función *uint32_t parsedecimal (char *str).* Dicha función toma como argumento a un elemento de un arreglo de caracteres, mediante el uso de un apuntador, el cual es comparado para verificar que sea distinto del carácter nulo, además de verificar que se trate de un solo digito entre 0 y 9, para posteriormente hacer

operaciones aritméticas entre él y número entero de 32 bits, de ésta manera, el elemento será tratado como un número y no como un carácter. Finalmente, después de dichas operaciones, la función devuelve al arreglo ya convertido en un número entero de 32 bits.

En la figura 3.41 se muestra el diagrama de flujo que permite comprender mejor el uso de la función *parsedecimal ( ).*

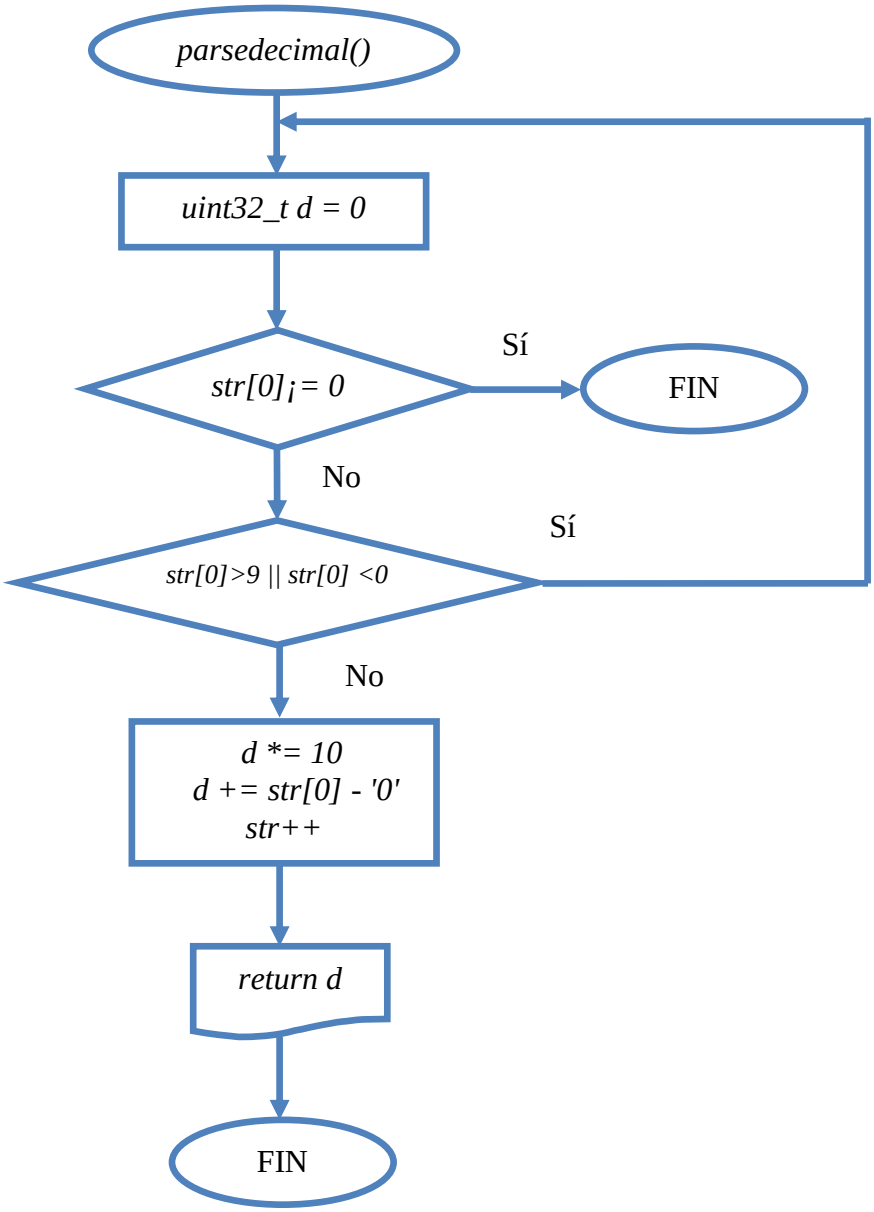


Figura 3.41. Diagrama de flujo de la función *parsedecimal( ).*

Cabe recordar que, según los requerimientos para el desarrollo del equipo CYCLE-DAQ, solo se hace uso de dos sentencias NMEA: $GPGGA y $GPRMC.

La sentencia $GPGGA proporciona el dato referente a la altitud medida en metros, para obtener dicho dato es necesario leer las sentencias disponibles en el puerto serial mediante la función *redaline_gps( )*, posteriormente debe verificarse que la sentencia $GPGGA se encuentre en el arreglo de caracteres y en caso de ser cierto, el dato de altitud deberá ser convertido a decimal y almacenado en una variable de tipo entero, con la función *parsedecimal( ).* Finalmente, el valor de la altitud debe ser dividido entre 10 para obtener el valor real, debido a que en su estado inicial el punto decimal que contiene se encuentra recorrido una posición a la derecha.

En la figura 3.42 se muestra el proceso de decodificación de los datos provenientes del receptor EM-406A para la sentencia $GPGGA.

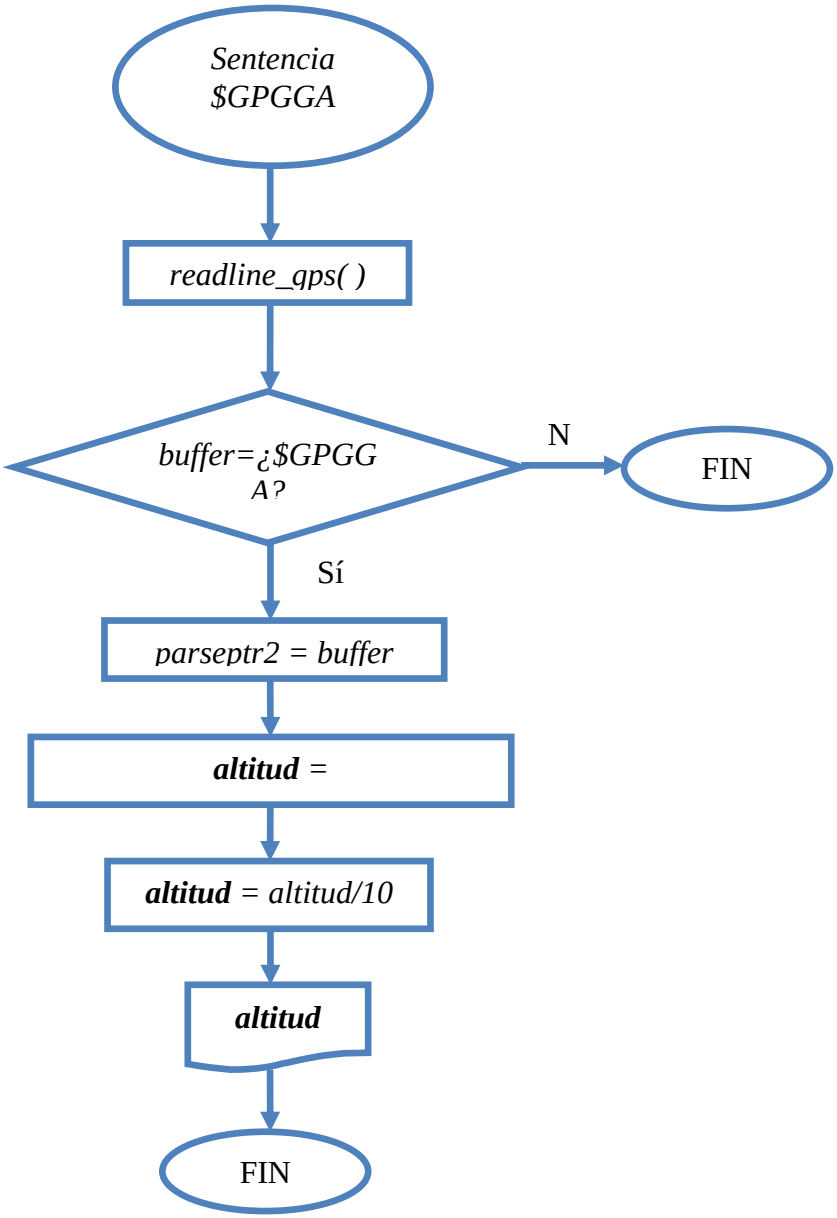


Figura 3.42. Diagrama de flujo de la sentencia $GPGGA.

La sentencia $GPRMC es la encargada de proporcionar los datos restantes, estos son: fecha, hora, latitud, longitud y velocidad. El procedimiento para obtener los valores de dichos datos es idéntico al descrito en la sentencia $GPGGA. En la figura 3.43 se muestra el diagrama de flujo correspondiente a la obtención de dichos parámetros.

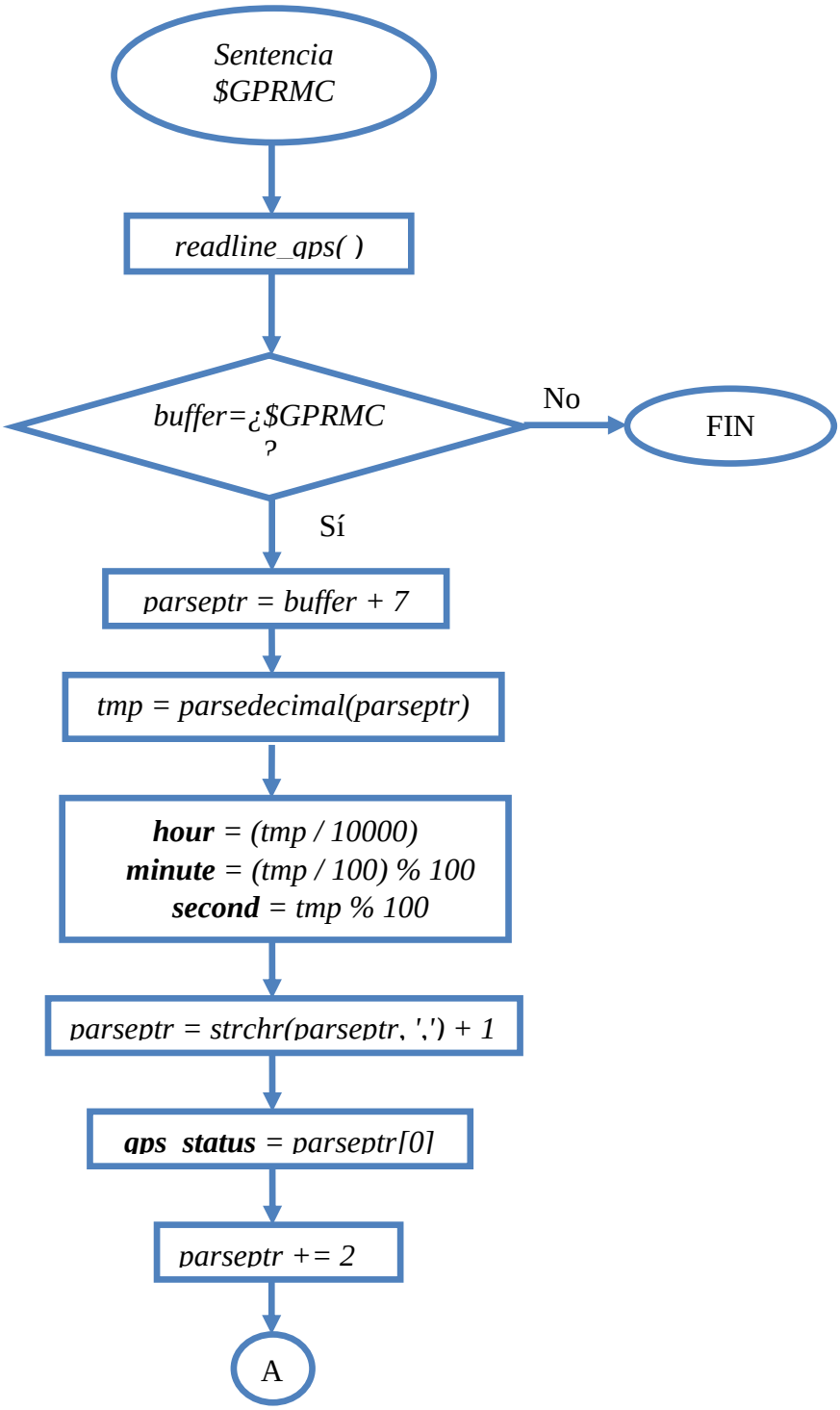


Figura 3.43. Diagrama de flujo de la sentencia $GPRMC…….. (Continúa).

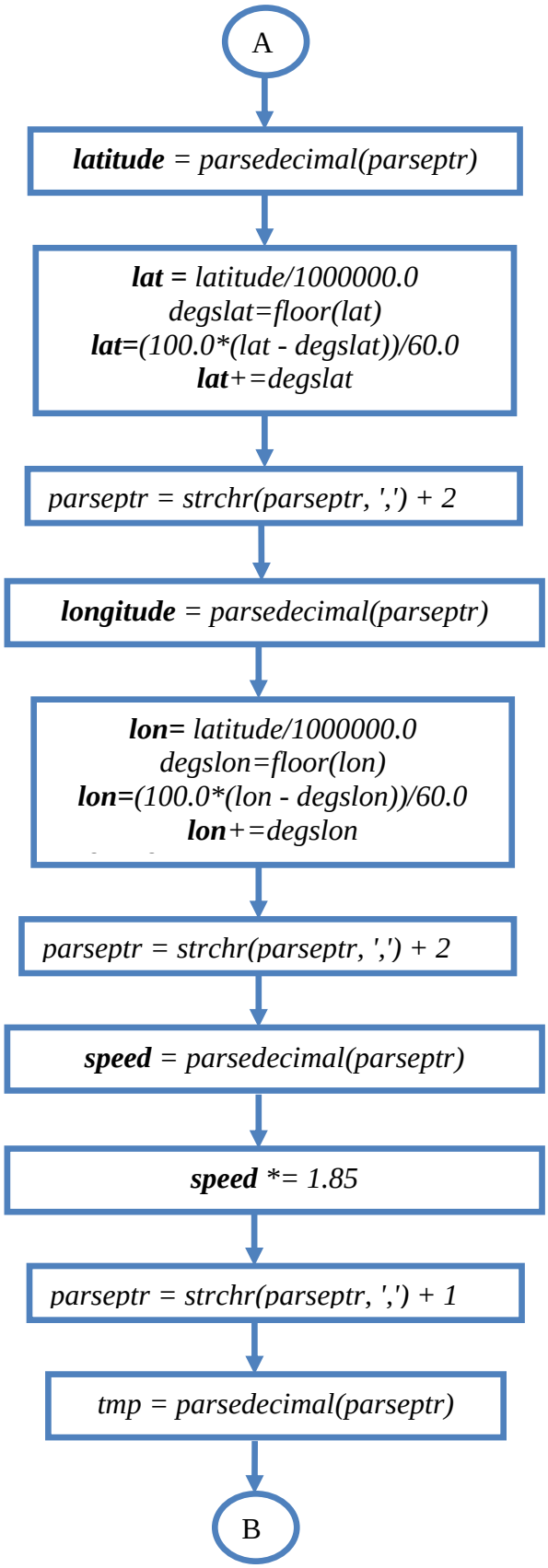


Figura 3.43. Diagrama de flujo de la sentencia $GPRMC…….. (Continúa).

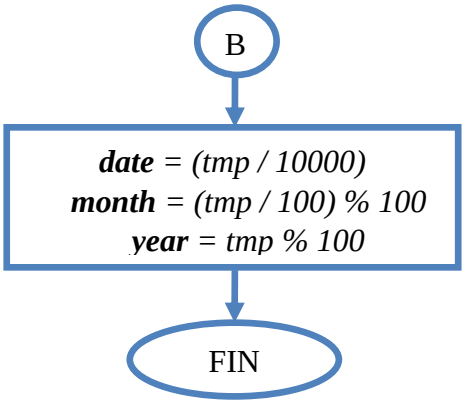


Figura 3.43. Diagrama de flujo de la sentencia $GPRMC.

Para el cálculo de la distancia recorrida por el vehículo se emplea la fórmula de Haversine, la cual hace uso de ecuaciones matemáticas complejas y funciones trigonométricas, por lo que para su implementación utiliza la librería *math.h* de ANSI C.

La fórmula de Haversine es implementada mediante la función: *calc_dist(float latitud1, float longitud1, float latitud2, float longitud2*. Dicha función recibe cuatro parámetros a saber: las coordenadas latitud y longitud del punto de partida y las coordenadas del siguiente punto. Este procedimiento se ejecuta cada segundo cuando las variables latitud y longitud son actualizadas. De esta manera, las distancias entre cada par de puntos son sumadas para así ir acumulándose y obtener la distancia recorrida total.

En la figura 3.44 se muestra el diagrama de flujo que describe el procedimiento de operación de la función implementada en la programación de la tarjeta Arduino.

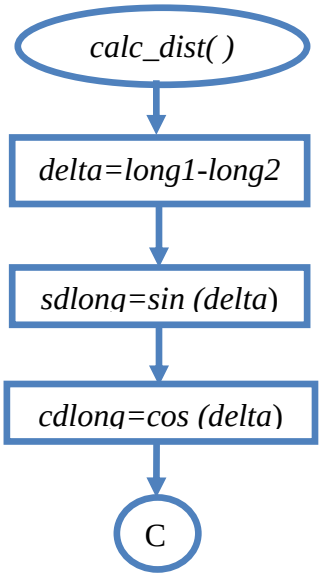


Figura 3.44. Diagrama de flujo de la función *calc_dist ( )*……. (Continúa).

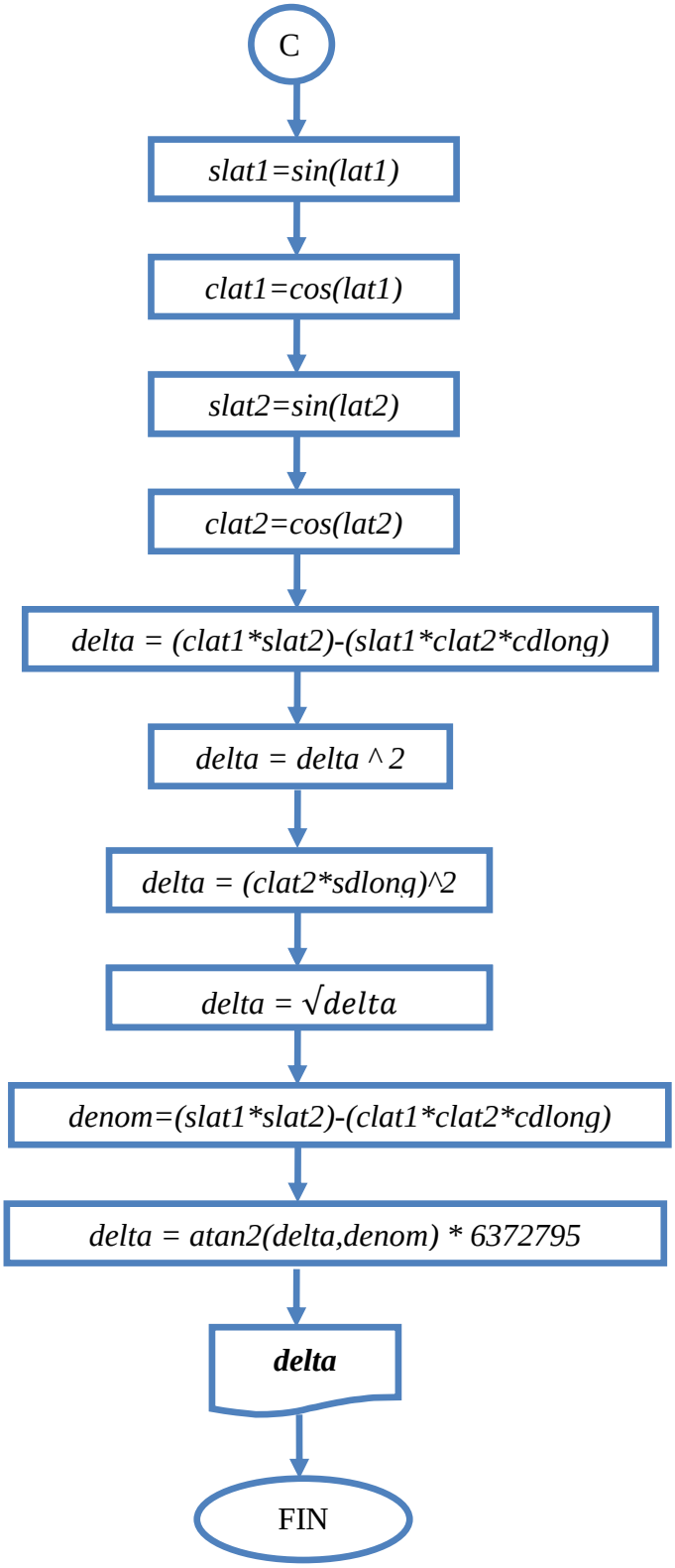


Figura 3.44. Diagrama de flujo de la función *calc_dist ( )*.

3.2.4. Manejo del acelerómetro ADXL345

La comunicación entre la tarjeta Arduino y el acelerómetro ADXL345 se realiza mediante la interfaz I^2C, para ello se dispone de una librería llamada *Wire.h* en el ambiente de desarrollo de Arduino. Dicha librería es la encargada de establecer la comunicación entre el maestro y el esclavo, mediante la configuración de las direcciones físicas de ambos dispositivos, así como la transmisión de bytes entre ambos. Para iniciar dicha comunicación se usa la función *Wire.begin( )*, que al no recibir ningún parámetro define al dispositivo como maestro, siendo para este desarrollo la situación de la tarjeta Arduino.

Además de *Wire.h*, también se utiliza la librería ADX345.h, la cual se encuentra disponible en la red para el uso de este dispositivo con Arduino; sin embargo, dicha librería fue modificada para poder ser implementada en el desarrollo del *CYCLE-DAQ*. Esto debido a la resolución manejada por default en dicha librería, con valor de $\pm$16[g]. La resolución adecuada para poder medir ángulos de inclinación debe ser menor ($\pm$2[g]), ya que con esta resolución se reduce notablemente la sensibilidad del acelerómetro y de esta manera las medidas no se ven afectadas por las vibraciones que experimenta el dispositivo.

La librería ADXL345.h está desarrollada bajo el lenguaje C++, por lo que es posible manejar características de la Programación Orientada a Objetos (POO) durante la integración del acelerómetro al *sketch* del *CYCLE-DAQ*.

La primer tarea a implementar consiste en verificar que la comunicación entre el ADXL345 y Arduino se ha iniciado con la función *Wire.begin( )*, para ello es necesario asignar un objeto a la función *ADXL345( )*, lo cual asigna la dirección 0X1D al acelerómetro. Realizado esto, la segunda tarea es corroborar que la señal de ACK ha llegado correctamente mediante la función *accel.EnsureConnected( )*. Si esta condición se cumple es necesario realizar la tercer tarea, que consiste en llamar a la función *accel.EnableMeasurements( )* e indica al dispositivo que todo está listo para comenzar con las mediciones.

La descripción anterior corresponde a la configuración del ADXL345 y solo se ejecuta una vez en el programa. Posteriormente, como cuarta tarea asignada al acelerómetro, es necesario comenzar a registrar las medidas arrojadas por el acelerómetro en variables globales. Para ello, se busca verificar que, en cada corrida del programa, el acelerómetro está conectado mediante la función *accel.IsConnected( )* la cual devuelve un valor verdadero en caso de que la corrección sea correcto y falso en caso contrario. Finalmente, si el valor es verdadero, se leen los valores en unidades [g], mediante la asignación de las variables globales a los métodos *scaled.XAxis* y *scaled.YAxis*.

Los datos obtenidos directamente del acelerómetro son en unidades de gravedad [g], por lo que es necesario realizar algunas conversiones para que el dispositivo funja como un sensor de inclinación. Para convertir la aceleración medida en [g] a un ángulo de inclinación en radianes, se calcula el seno inverso del eje X y el coseno inverso del eje Y, como se muestran en las ecuaciones 3.9 y 3.10:

$$\theta = \sin^{-1}\left(\frac{A_{xOUT}\,[g]}{1[g]}\right) \quad \ldots\ldots\ldots\ldots\ldots\ldots \quad (3.9)$$

$$\theta = \cos^{-1}\left(\frac{A_{yOUT}\,[g]}{1[g]}\right) \quad \ldots\ldots\ldots\ldots\ldots\ldots \quad (3.10)$$

Para comprender mejor el procedimiento anteriormente descrito, en la figura 3.45 se muestra el diagrama de flujo correspondiente.

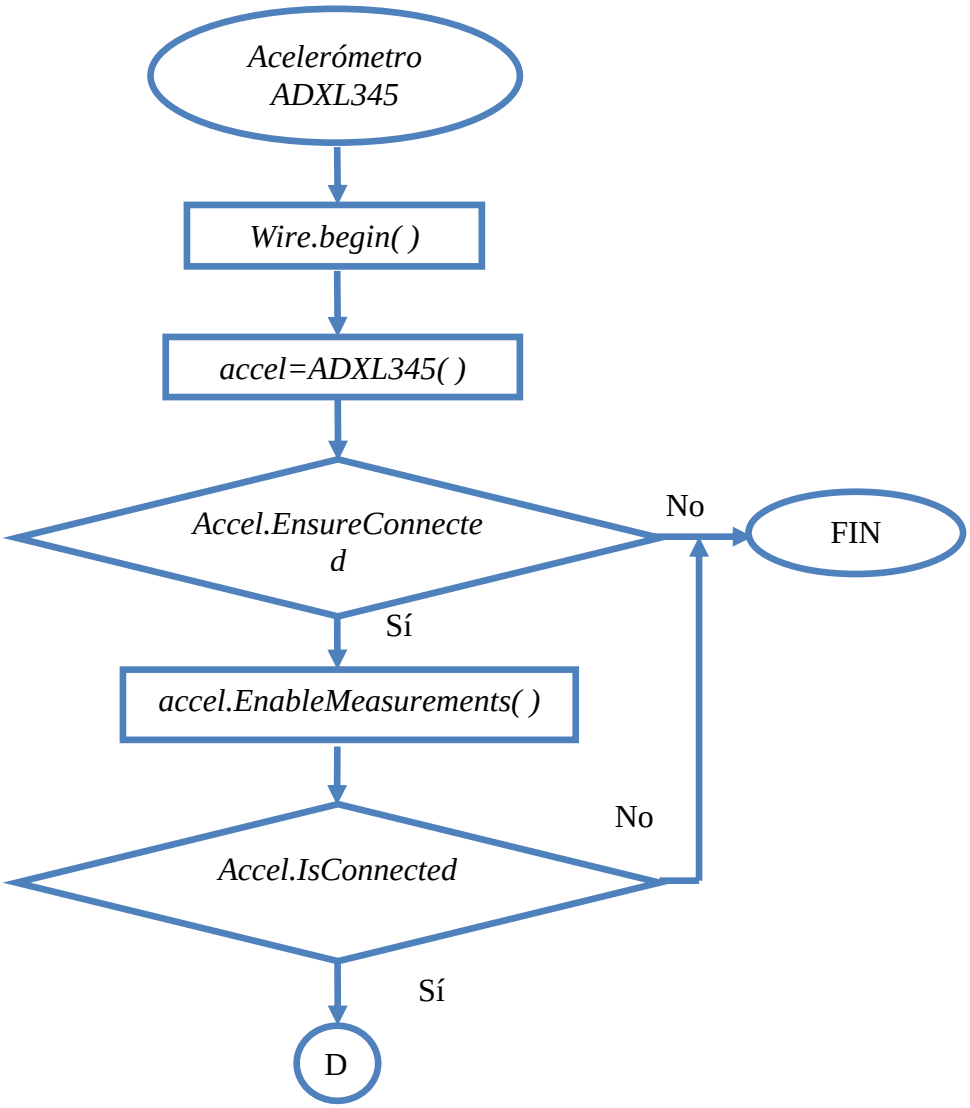


Figura 3.45. Diagrama de flujo del acelerómetro ADXL345…….. (Continúa).

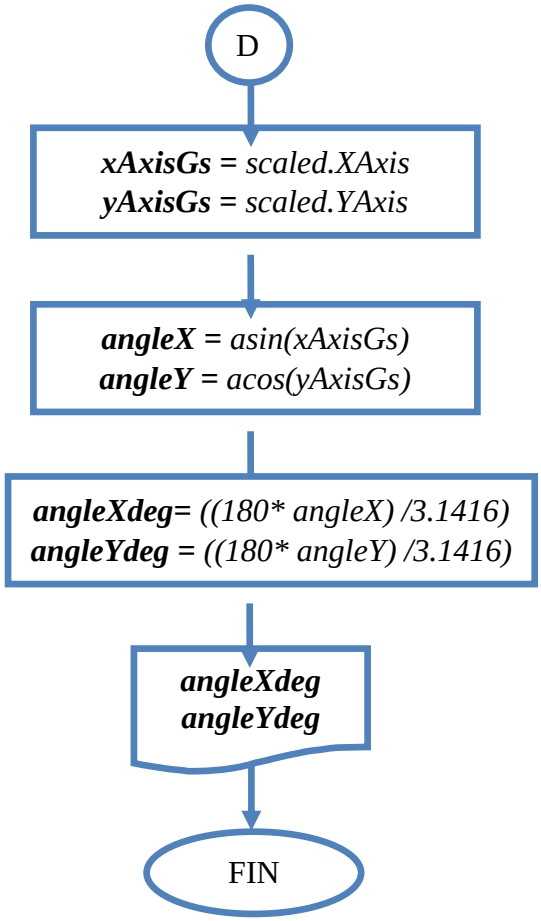


Figura 3.45. Diagrama de flujo del acelerómetro ADXL345.

3.2.5. Interacción con ANDROS 6600

El analizador de gases ANDROS 6600, al igual que el receptor GPS, interactúa con la tarjeta Arduino mediante la conexión RS-232.

La configuración y control de la operación del analizador de gases se basa en un sistema que utiliza comandos para su operación. Dichos comandos son formados mediante una cadena compuesta de bytes definidos por un formato predeterminado. El sistema de comandos permite al usuario saber si la cadena de bytes enviada al analizador tiene el formato establecido y ha sido reconocida por éste, ya que responderá con un comando de aceptación (ACK) en caso de detectar que los datos sean válidos o con un comando de rechazo (NAK) en caso contrario.

El formato de cada comando consta de 4 o más bytes, dependiendo del comando en cuestión, y tiene la siguiente forma: **DID-LD-CMD-DF-CS**

Donde:

DID: Número de identificación del comando.

LD: Especifica la longitud de la cadena de los bytes

CMD: Transmite el código del comando

DF: Datos del comando, cuya longitud varía de acuerdo a cada comando

CS: *CheckSum*, mediante este byte se determina si hay o no error en la transmisión de la información. Su valor está representado en complemento a dos y se calcula mediante la ecuación 3.11:

$$CS=NOT\ (DID\ LD\ CMD\ DF)\ +1 \quad (3.11)$$

Aunque el analizador de gases cuenta con 16 comandos para su completo funcionamiento, en el equipo *CYLCE-DAQ* sólo son implementados 5 de éstos. Dicha selección se debe a la autonomía del equipo y a que son los comandos necesarios para controlar los dispositivos más importantes de éste y para adquirir datos del mismo. En la tabla 3.10 se muestran los comandos utilizados y su descripción funcional.

COMANDO PROGRAMADO	*IDENTIFICADOR DEL COMANDO*	*FUNCIÓN*
System-ID	*$04*	Muestra los datos técnicos de la banca analizadora de gases.
Data Status	*$01*	Determina el modo de operación actual de la banca analizadora de gases se encuentra. Además, arroja las mediciones de gases que la banca analizadora ha procesado.
Auto Zero	*$02*	Permite ajustar los canales de medición que integran el sistema de análisis, ejecutando un tipo de calibración para el O_2.
V_valve	*$08*	El control indica y realiza una inspección de la presión que presenta el sistema neumático, así como el flujo de gas que corre a través del sistema, por medio de la válvula solenoide.
Pump On/Off	*$07*	Controla el funcionamiento de la microbomba del sistema neumático.

Tabla 3.11. Comandos programados en el equipo CYCLE-DAQ.

Comando *System-ID*

Este comando proporciona la información general del ANDROS 6600, es decir: modelo, número de serie, versión de *hardware* y *software*. En el equipo CYCLE-DAQ, la información anterior no es registrada, ya que la única función de este comando es verificar que existe comunicación entre el equipo y el analizador de gases y de esta manera es posible asegurar al usuario su correcta conexión.

El comando *System-ID* está conformado de la siguiente manera: **0x02-0x01-0x04-0xF9** y es enviado sólo una vez en la parte de configuración de Arduino a través del puerto serial de éste. Una vez enviado, el analizador responderá con una serie de bytes que indicarán si el comando fue válido o no. En la figura 3.46 se muestra el diagrama de flujo de cómo interactúa el equipo CYCLE-DAQ con el analizador de gases.

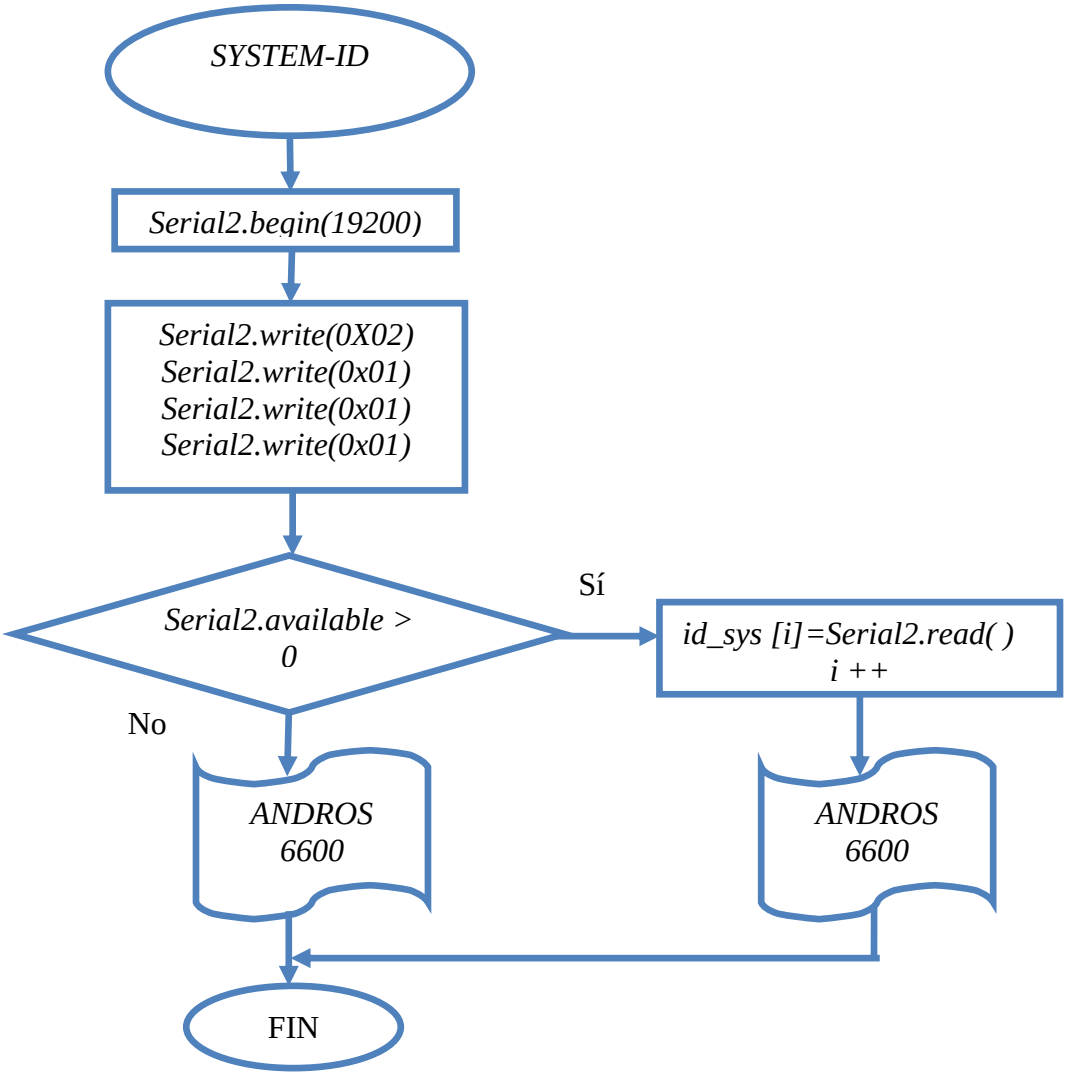


Figura 3.46. Diagrama de flujo del comando *System-ID*.

Comando *Data-Status*

Es el comando más importante ya que de él, se obtienen las variables relativas a las concentraciones de los cinco gases contaminantes. Este comando es enviado cada segundo con el objetivo de registrar las mediciones de las concentraciones en dicho intervalo de tiempo. El formato del comando es el siguiente: **0x02-0x03-0x01-0x01-0x00-0xF9.**

Al enviar el comando *Data-Status*, el analizador de gases responde con 20 bytes, de los cuales, sólo 14 son utilizados por el equipo CYCLE-DAQ. El tercer byte recibido por el puerto serial del microcontrolador y brinda la información del estado del analizador de gases. Para este desarrollo, sólo es importante verificar el estado de los bits 4 y 5 de dicho byte. El bit 5 indica si es necesario o no realizar una calibración o ajuste de cero, mientras que el bit 4 informa sobre el estado de dicha calibración, es decir, si es que está en progreso o ya ha finalizado. El sexto byte brinda la información sobre el estado de la microbomba en su bit número 7, el estado de este bit es importante ya que en caso de que ésta falle deberá ser apagada inmediatamente. Los últimos doce bytes del comando *Data-Status* corresponden a las concentraciones de los gases contaminantes y se interpretan de la siguiente manera: los bytes siete y ocho proporcionan la información sobre la concentración de CO_2 en la muestra, los bytes nueve y diez la de CO, los bytes once y doce corresponden a los HC reportados como hexano mientras que el trece y catorce reportan a HC como propano, los bytes quince y dieciséis informan sobre el O_2 y finalmente el byte diecisiete y el byte reportan a los NOx.

El diagrama de flujo para el procedimiento de envío, recepción y decodificación del comando *Data-Status* se muestra en la figura 3.47.

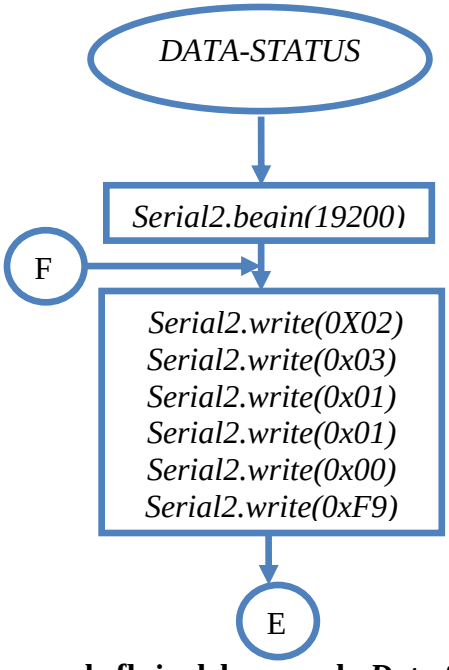


Figura 3.47. Diagrama de flujo del comando *Data-Status*…….. (Continúa).

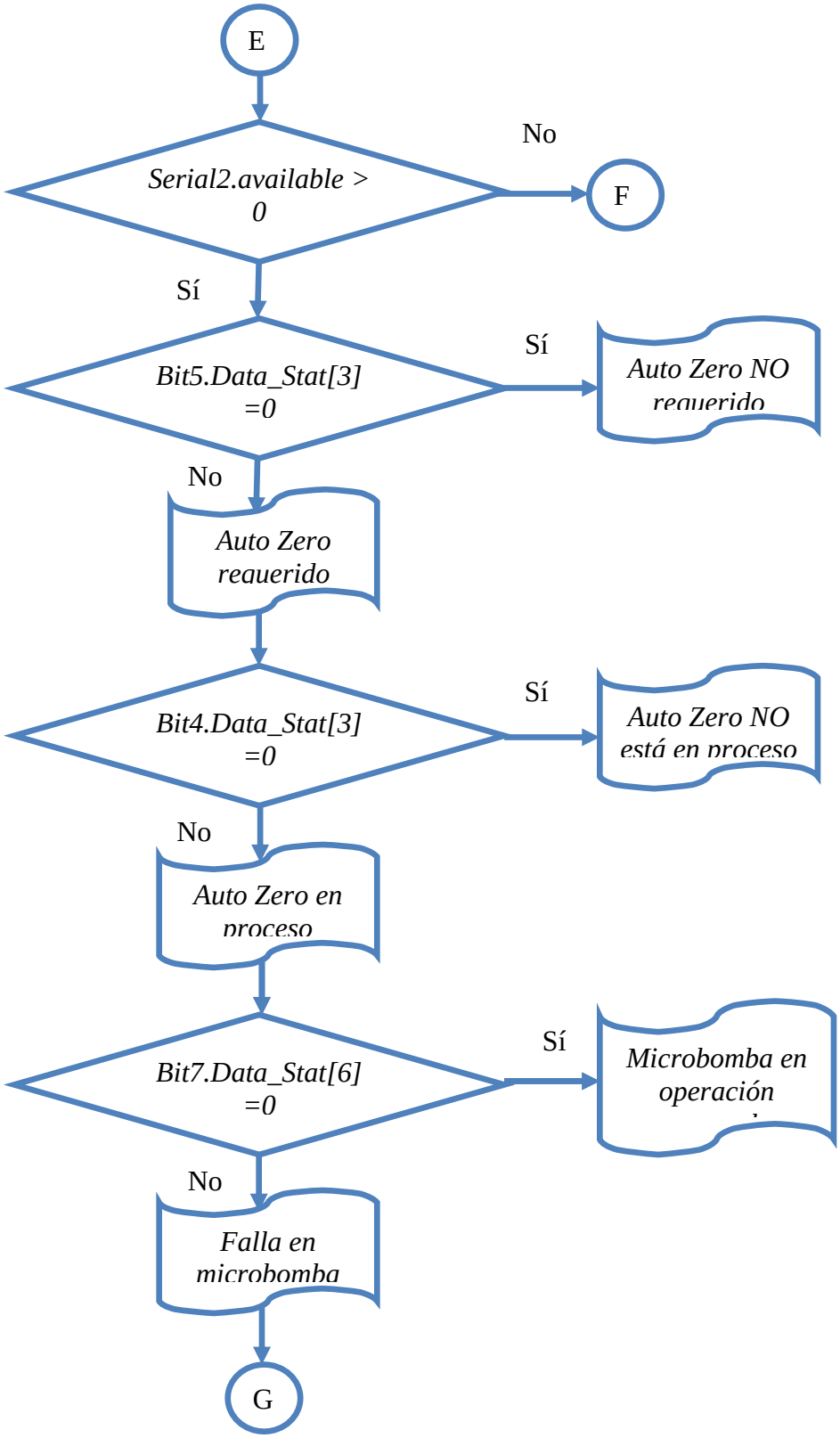


Figura 3.47. Diagrama de flujo del comando *Data-Status*…….. (Continúa).

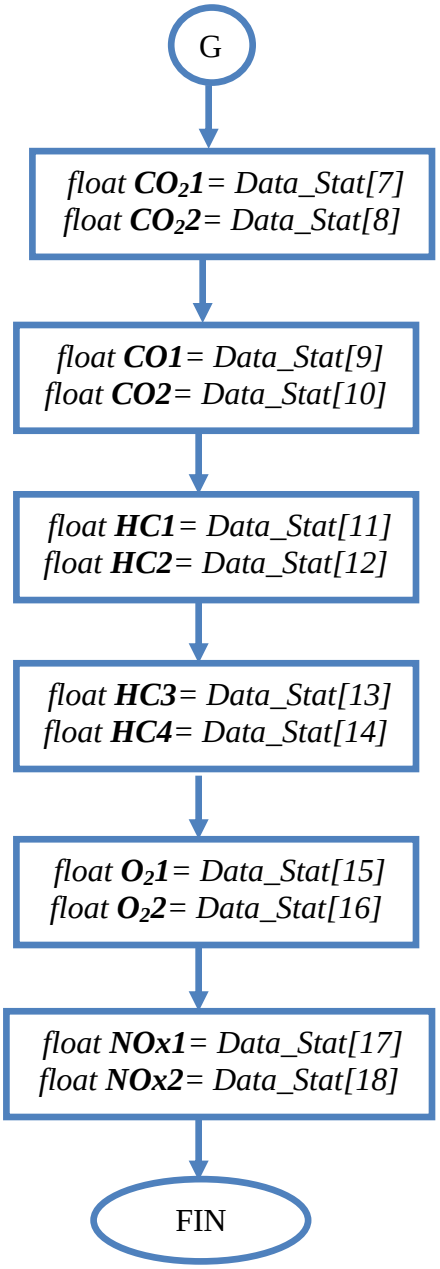


Figura 3.47. Diagrama de flujo del comando *Data-Status*.

Comando *Auto Zero*

Este comando es enviado al ANDROS 6600 cuando el comando *Data-Status* informa que el proceso de calibración a cero es necesario. Sirve para ajustar las mediciones de los gases contaminantes con el aire del medio ambiente en el que éste se encuentra y es fundamental para que la información arrojada sobre las concentraciones no sea errónea. El formato del comando es el siguiente: **0x02-0x02-0x02-0x02-0x02-0xF.**

El proceso de calibración a cero o también llamado Auto Zero solo dura alrededor de 15 segundos y se ejecuta al inicio del proceso de adquisición de datos y posteriormente deber realizarse cada 30 minutos según el comando *Data-Status* lo requiera.

Comando *V_Valve*

Este comando tiene el formato: **0x02-0x03-0x08-0x80-0XF8** y es enviado al analizador de gases para que active o desactive la válvula solenoide que se encuentra conectada a una de las 8 terminales TTL del analizador. Dicha válvula entra en operación cuando se realiza el ajuste a cero (*Auto Zero*) y solo se encontrará activa mientras dicho ajuste dure, después de ello, la válvula deberá ser apagada.

En la figura 3.48 se muestra el proceso que involucra a los comando *Auto Zero* y *V_Valve* durante el proceso de adquisición de datos.

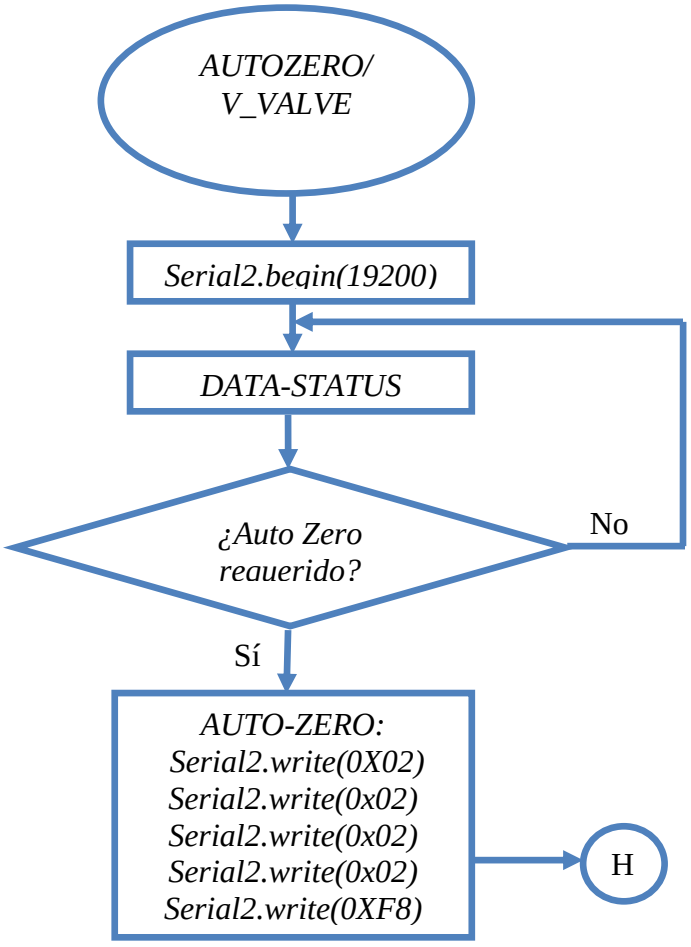


Figura 3.48. Diagrama de flujo de los comandos *Auto Zero y V_Valve*…….. (Continúa).

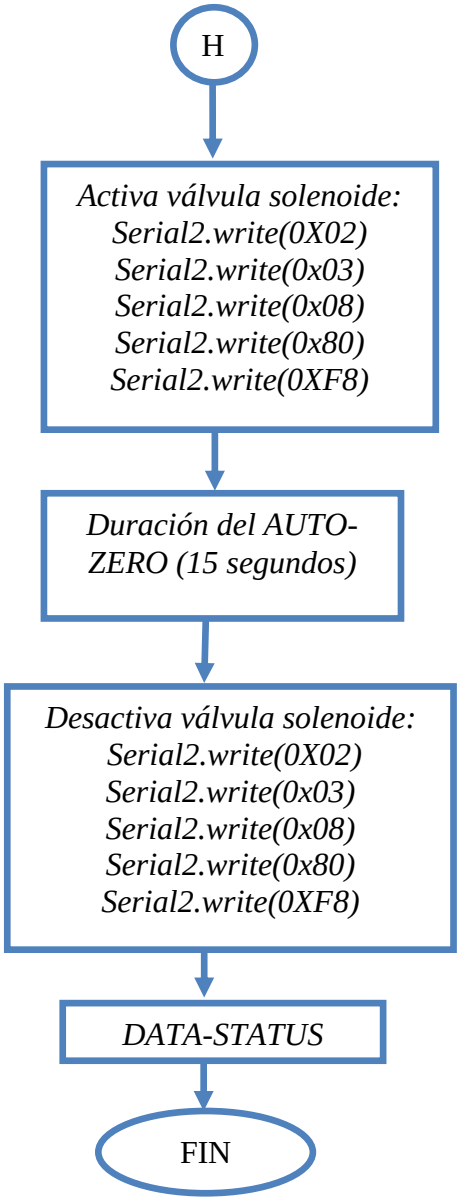


Figura 3.48. Comandos *Auto Zero y V_Valve.*

Comando *Pump On/Off*

El comando *Pump On/Off* es bastante simple, ya que su única función es controlar el encendido y apagado de la microbomba del ANDROS 6600, dicha bomba es la encargada de llevar la muestra de gases al interior de la banca para que ésta pueda ser analizada. Cuando el comando *Data-Status* informe sobre el malfuncionamiento de la microbomba, ésta deberá ser desactivada inmediatamente; en caso contrario, no deberá ejecutarse acción alguna y la microbomba se encontrará activa. Éste comando tiene el siguiente formato: **0x02-0x03-0x07-0x00-0XF5**.

En la figura 3.49 se muestra el diagrama de flujo que representa al algoritmo implementado en el equipo CYCLE-DAQ para el funcionamiento de la microbomba.

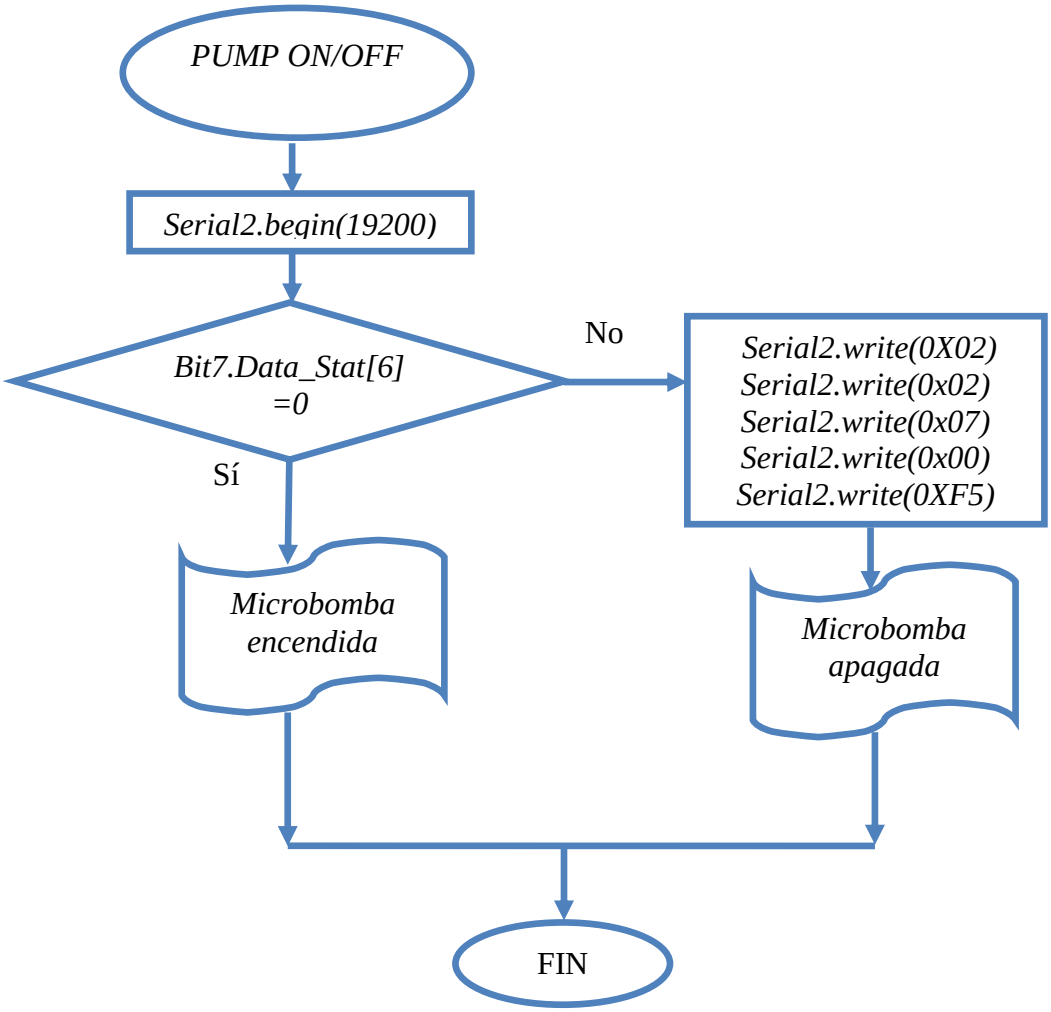


Figura 3.49. Diagrama de flujo del comando *Pump On/Off*.

3.2.6. Sensor de voltaje de la batería

La lectura del sensor de voltaje de la batería, se realiza definiendo la terminal analógica 15 como entrada mediante la función *pinMode(A15, INPUT)*, en la etapa de configuración de la tarjeta de desarrollo. En la ejecución iterativa del microcontrolador es necesario hacer uso de la función *analogRead( )*, la cual entrega un valor de 0 a 1023 y corresponde al formato digital del voltaje leído. Posteriormente, este valor deberá ser almacenado en una variable global para poder obtener su equivalente analógico presente en la salida del sensor, como lo muestra la ecuación 3.12.

$$señal_{analógica} = señal\ _{digital} * \left(\frac{5.0}{1023}\right) \ldots\ldots\ldots\ldots\ldots (3.12)$$

Una vez que ya se tiene el valor de la señal analógica, es necesario escalarlo para poder conocer con exactitud el voltaje que se encuentra en la entrada del sensor y que corresponde al voltaje real de la batería del automóvil. El procedimiento de escalamiento se muestra en la ecuación 3.13.

$$voltaje_{entrada} = señal_{analógica} * \left(\frac{R1 + R2}{R2}\right) \ldots\ldots\ldots\ldots\ldots (3.13)$$

Donde, R1=4700 y R2= 1500 y corresponden a los valores en Ohm de las resistencias que forman al circuito divisor de voltaje.

En la figura 3.50 se muestra el diagrama de flujo correspondiente a la medición del voltaje de la batería del automóvil.

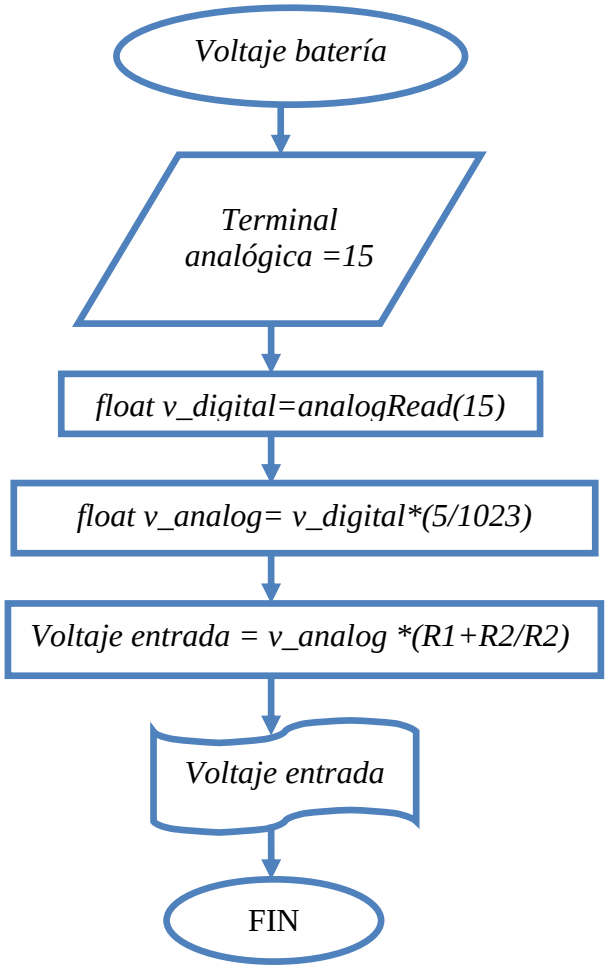


Figura 3.50. Diagrama de flujo del sensor de voltaje.

3.2.7. Registro de datos en memoria microSD Card

La comunicación entre la memoria microSD Card y la tarjeta Arduino se lleva a cabo mediante el protocolo SPI. Para este fin, se hace uso de la librería *SdFat.h*, la cual cuenta con las siguientes características:

- Soporta los sistemas de archivos FAT16 y FAT32 en memorias microSD.
- Solo es posible implementar nombres de hasta 12 caracteres.
- Permite operaciones de creación, borrado, lectura, escritura y modificación de archivos.
- Permite la creación y/o modificación de subdirectorios.
- Es compatible con la mayoría de las tarjetas Arduino.
- Está basada en el lenguaje C++.

Al implementar el código para el buen funcionamiento de la librería *SdFat.h,* es necesario crear un objeto por cada una de las tres principales clases que de ella se desprenden, estas son: *Sd2Card, SdVolume* y *SdFile.*

La clase *Sd2Card* es la encargada de iniciar a la memoria microSD mediante la llamada al método *Sd2Card::Init( ).*

La clase *SdVolume* se encarga de manejar el sistema de archivos a implementar, FAT32 en nuestro caso, dicha clase es puesta en marcha con el método *SdVolume::Init( ),* si el método *Init( )* no recibe parámetro alguno, entonces maneja FAT32.

Para la implementación de las operaciones con el archivo de registro de datos, se hace uso de la clase *SdFile,* la cual proporciona los métodos: *open ( ),* para abrir al archivo, *read( )* para leer el archivo, *remove( )* para eliminarlo, *write( )* para escribir datos y *close( )* para cerrar el archivo.

El procedimiento de registro de datos en el equipo CYCLE-DAQ se realiza en dos etapas:

1. **Configuración de la memoria microSD:** Esta parte se ejecuta solo una vez en el microcontrolador y corresponde a la inicialización de la tarjeta y su sistema de archivos, así como a la creación del archivo de registro. La configuración consta de los siguientes pasos:

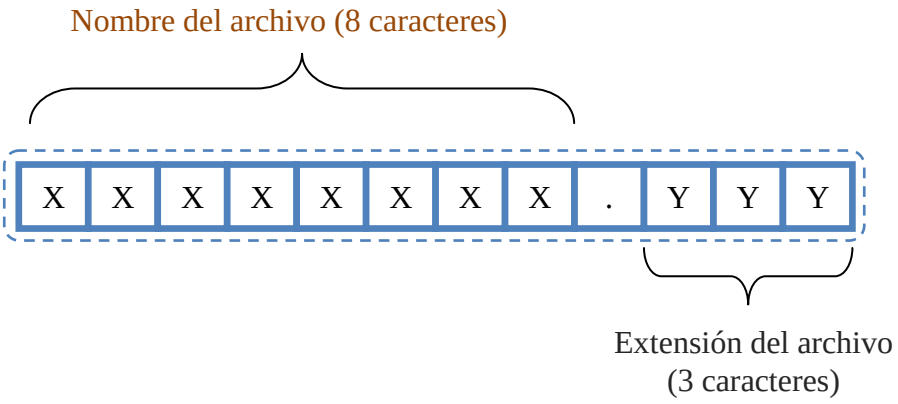
    1.1. Creación del objeto archivo. El archivo es creado mediante una instancia de la clase *SdFile,* el nombre del objeto instanciado es indistinto. Por ejemplo: *SdFile* ***MiArchivo.***

    1.2. Creación del objeto tarjeta. El objeto que hace referencia a la tarjeta de memoria se desprende de la clase Sd2Card: Por ejemplo: *Sd2Card* ***MiTarjeta.***

1.3. Creación del objeto volumen. Debido a que la tarjeta deberá tener asignado un sistema de archivos, debe instanciarse un objeto de la clase SdVolume, el cual hace referencia al formato de archivos a implementar. Por ejemplo SdVolume ***MiVolumen.***

1.4. Verificación de la memoria: En este punto se verifica que la tarjeta se encuentre física y lógicamente presente, así como que los sectores y clústeres no estén dañados y se pueda tener acceso a ella para posteriores operaciones de lectura y escritura. La verificación de la memoria se basa en la llamada a los métodos descritos anteriormente, los cuales responden con un "1" lógico, cuando no se detecten problemas, y con un "0" lógico, en caso contrario; lo cual permite activar una bandera para indicar si es que existe algún problema con la memoria y el proceso de registro de datos no será llevado a cabo.

1.5. Creación del archivo y su nombre: En esta parte se crea físicamente el archivo en la memoria microSD, para su creación se hace uso del objeto creado en el punto 1.1, acompañado por el método *open ( )*. Por ejemplo ***MiArchivo.open()***, de igual manera que ocurre con la verificación de la memoria, este método devuelve "1" si se ha creado correctamente el archivo y "0" en caso de no haberse creado. Una vez se realiza esta comprobación, es necesario asignarle un nombre, el cual, debido a las características de la librería, tendrá una longitud máxima de 12 caracteres, distribuidos como se muestra en la figura 3.51.



.

Figura 3.51. Estructura del nombre del archivo en memoria microSD.

En el equipo CYCLE-DAQ se programó un algoritmo para implementar el nombre del archivo de registro, el cual numera consecutivamente al archivo en los últimos dos caracteres de su nombre. El archivo creado es de texto plano y el nombre asignado es: "LOGGERxx.txt", donde "xx" representa un número entre 00 y 99, dicha numeración permite crear hasta 100 archivos, equivalente a realizar 100 procesos de adquisición y registro de datos. Cabe mencionar que los archivos se crean en el directorio raíz de la tarjeta de memoria y no en subdirectorios o subcarpetas, además que, el tamaño máximo del archivo es el establecido por el sistema de archivos FAT32.

2. **Registro de datos:** Este proceso se ejecuta cada segundo y contiene las funciones de apertura del archivo creado previamente, permite la escritura en el archivo y lo cierra cuando la escritura ha finalizado.

La etapa de escritura es la más importante debido a que, en este momento las 12 variables entregadas por los sensores, son registradas en el archivo de texto plano, separadas por una coma e incluyen fin de línea y retorno de carro cada segundo para tener un orden en el registro de las mismas.

El método que permite escribir en el archivo es asignado al objeto ***MiArchivo***, de la siguiente manera: ***MiArchivo.write(variable)***, donde variable, es el argumento del método ***write ( )*** y que corresponde al valor que se desea escribir, es decir, éste método se repite para cada una de las 12 variables adquiridas por el equipo CYCLE-DAQ.

Para conceptualizar de manera gráfica los procesos anteriormente descritos, en la figura 3.52 se muestra el diagrama de flujo que corresponde al proceso de registro de datos en la memoria microSD.

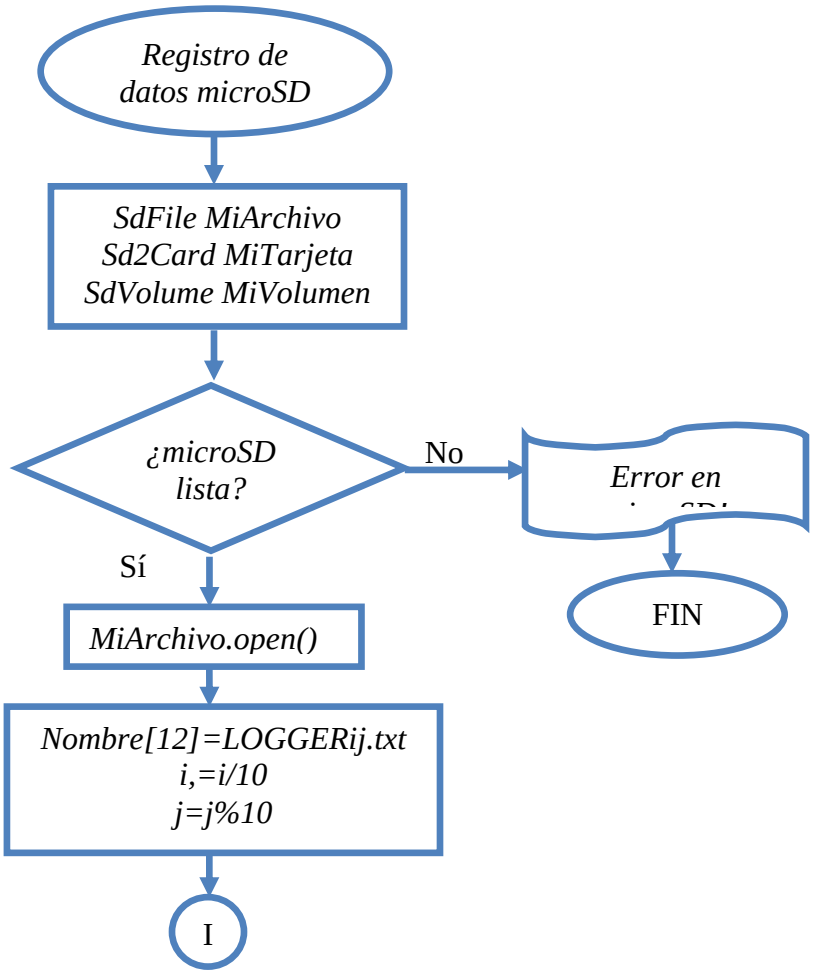


Figura 3.52. Diagrama de flujo del registro de datos en la tarjeta microSD…… (Continúa).

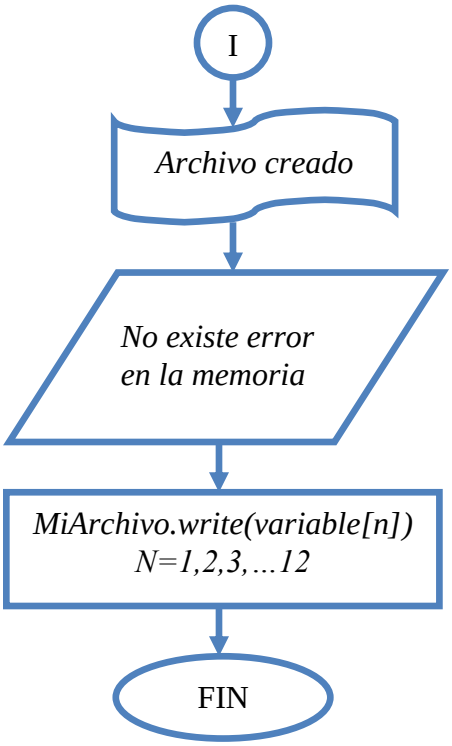


Figura 3.52. Diagrama de flujo del registro de datos en la tarjeta microSD.

3.2.8. Manejo del *display* gráfico

Para controlar al *display* gráfico se hace uso de la librería *glcd.h*, útil para manejar dispositivos de despliegue basados en el controlador KS0108. Esta librería está basada en el lenguaje C++, por lo que es necesario trabajar con las características de dicho lenguaje para lograr un correcto funcionamiento del dispositivo.

El procedimiento de operación del GLCD se basa en los métodos proporcionados por la librería y que a continuación se describen:

- Inicialización del *display*. El primer paso para poner en marcha el *display* consiste en inicializar a éste mediante la llamada al método *GLCD.Init()* durante la etapa de configuración de la tarjeta de desarrollo.

- Selección del tipo y tamaño de letra. La librería posee una variedad de tipografías para la impresión de caracteres. Por tal motivo, es necesario seleccionar la fuente que se utilizará en el desarrollo mediante el método *GLCD.SelectFont()*.

- Colocación del cursor. Para ubicar al cursor de la pantalla en un lugar específico y escribir caracteres es necesario llamar al método *GLCD.GotoXY(x,y)*, recordando que el valor máximo de "x" es 128 y el de "y" es 64 (128 x 64 pixeles).

- Escritura en el *display*. En el *display* es posible escribir un caracter, cadenas de caracteres o variables de tipo numérico, para ello existe el método *GLCD.print(argumento)*, en donde el argumento puede ser cualquiera de los tipos de dato mencionados.

- Borrado del display. Esta función es demasiado útil en situaciones en donde es necesario actualizar la pantalla y se debe borrar todo el contenido previo del display, esto es posible con la llamada al método *GLCD.ClearScreen()*.

En el equipo CYCLE-DAQ se presentan 4 pantallas de notificación al usuario, las cuales se muestran en la figura 3.53, indicando con un número su orden de aparición durante la ejecución del equipo.

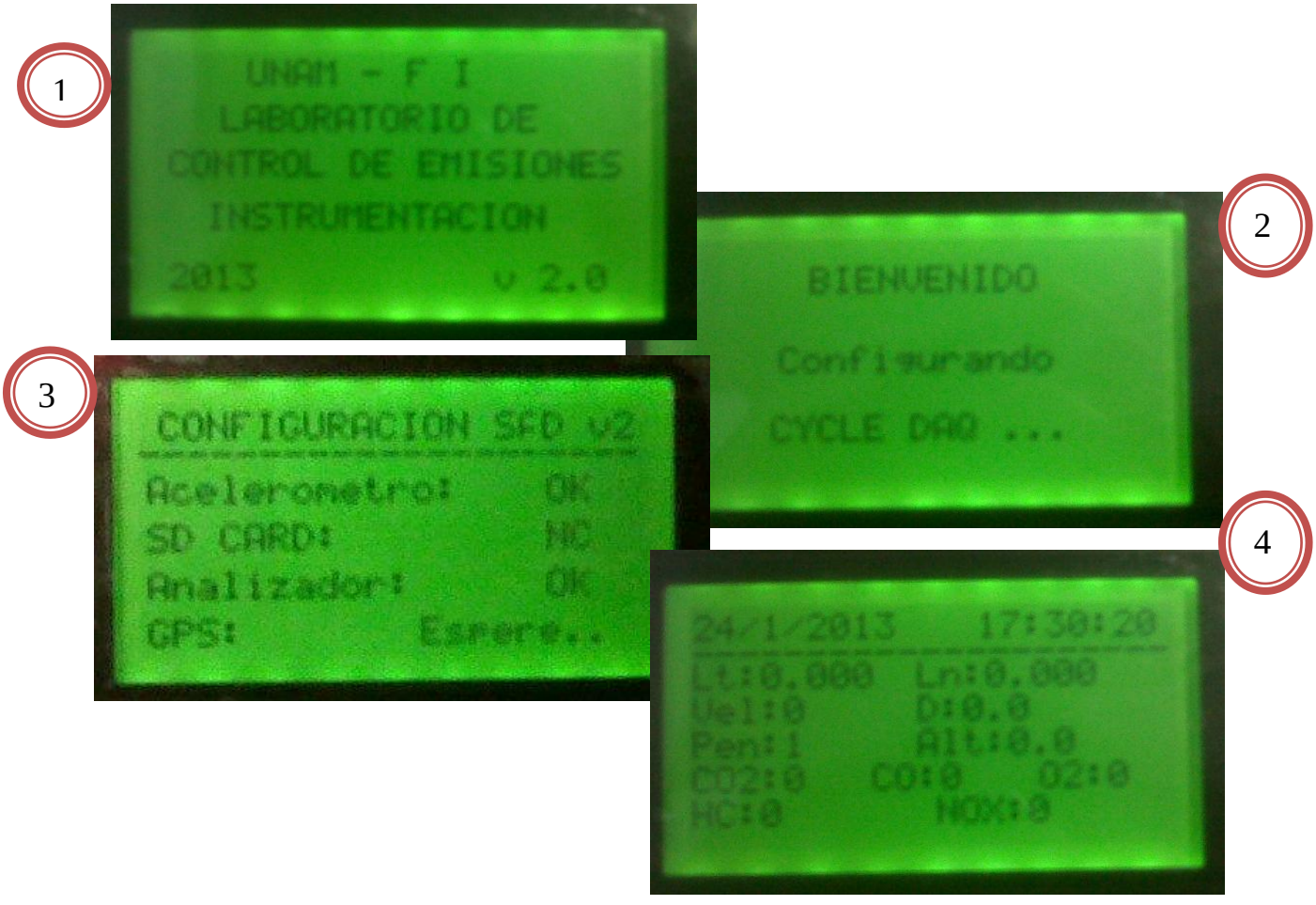


Figura 3.53. Pantallas del equipo CYCLE-DAQ.

3.2.9. Manejo de indicadores

El equipo CYCLE-DAQ cuenta con tres LEDs para indicar su correcto funcionamiento; sin embargo solo dos de éstos son activados mediante *software*. El LED verde, que corresponde a la memoria microSD, se activa y permanece encendido cuando la memoria ha sido reconocida y el proceso de registros se encuentra en actividad; mientras que el LED azul parpadea con una frecuencia de 1[Hz], es decir, cada segundo, cuando el receptor GPS se ha sincronizado con la constelación de satélites necesarios, lo que implica que los datos enviados por éste, son válidos. La funcionalidad de estos LEDs fue propuesta por el personal del LCE, ya que permite visualizar fácilmente si el equipo está o no en operación normal.

El procedimiento para encender y apagar un LED con la tarjeta Arduino es trivial. Basta con declarar la terminal a la que el LED se conecta como salida y deberá ser activado o desactivado con la función *digitalWrite(terminal, HIGH/LOW)*, donde el argumento *terminal* corresponde al número de salida de la tarjeta; mientras que, *HIGH/LOW* indica el estado en el que se desea poner a la *terminal*, esto es, alto o bajo.

En la figura 3.54 se muestra el diagrama de flujo que describe el funcionamiento de los indicadores visuales

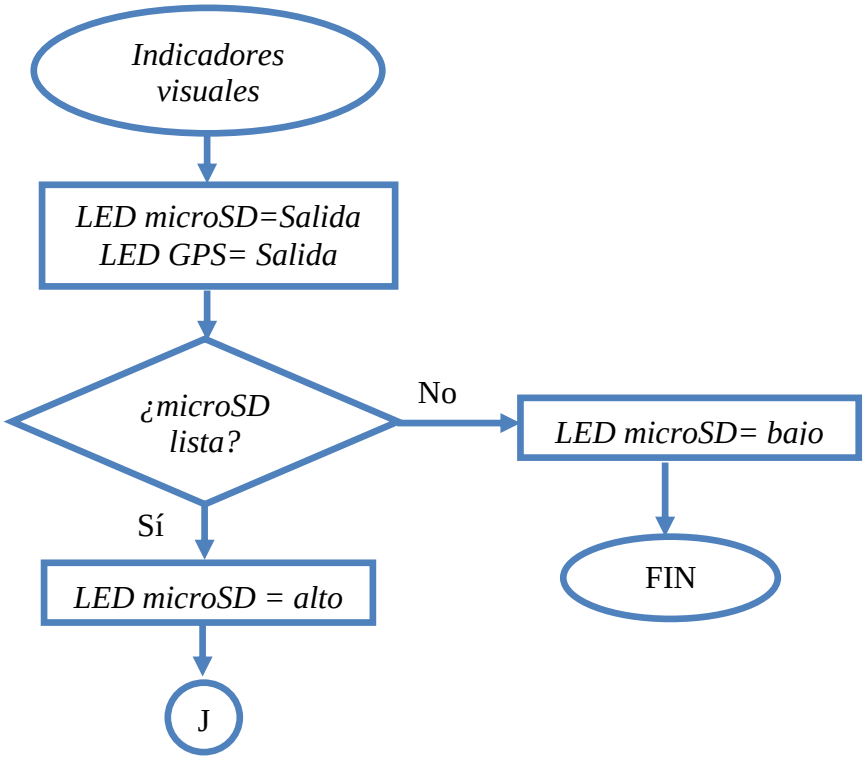


Figura 3.54. Diagrama de flujo de los indicadores visuales…….. (Continúa).

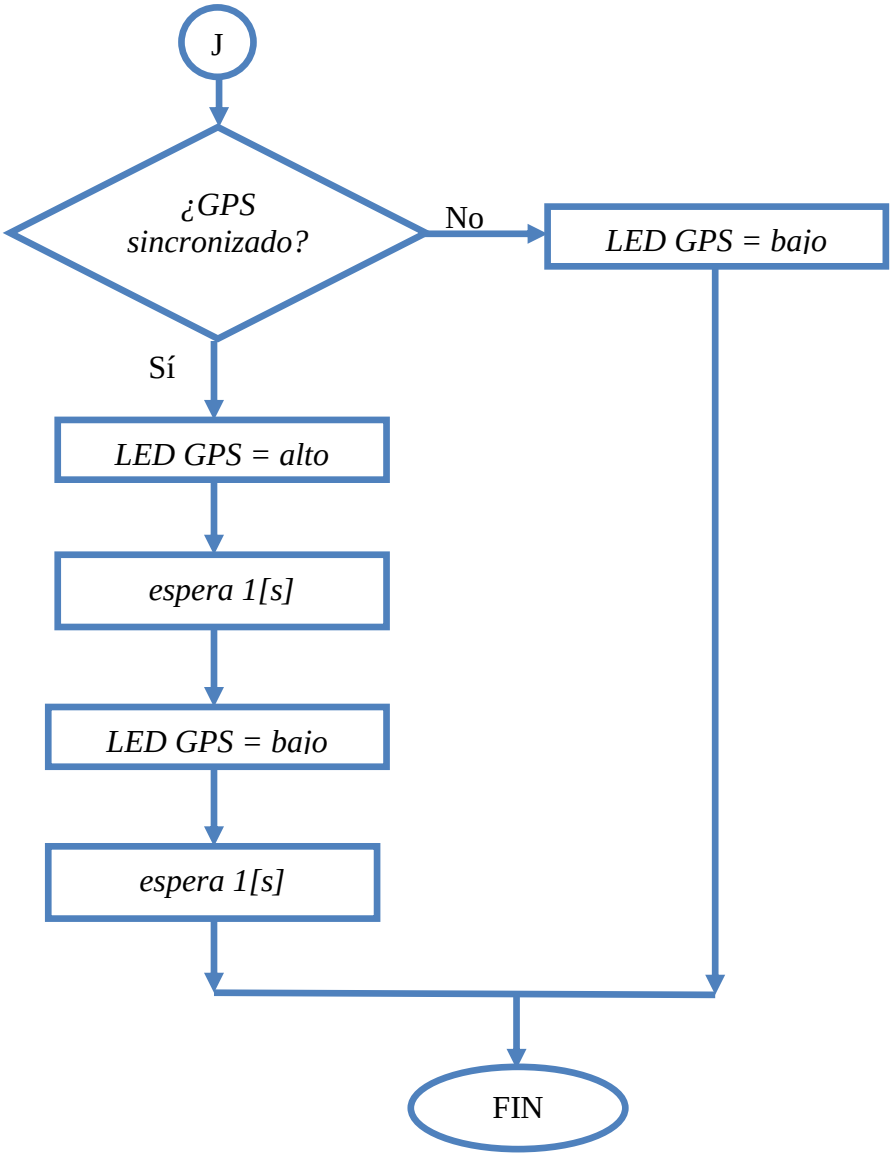


Figura 3.54. Diagrama de flujo de los indicadores visuales.

3.3. Desarrollo del *software* de procesamiento de datos

Una vez que los datos son almacenados en el archivo de texto plano, LOGGERxx.txt, éstos deben ser procesados para que puedan ser manipulados estadísticamente por parte del grupo de trabajo del LCE. Para ello se desarrolló un software de procesamiento de datos bajo la plataforma LabVIEW, utilizando las características de programación que ésta ofrece. El *software* desarrollado es llamado DRIVE-SOFT, por lo que a partir de éste momento, nos referiremos a éste, con dicho nombre.

3.3.1 Diagrama de bloques

Básicamente, el software desarrollado consta de 3 etapas, las cuales a continuación se describen:

1. **Lectura del archivo de origen:** El archivo de origen es el generado por el equipo CYCLE-DAQ y que se encuentra en el directorio raíz de la tarjeta de memoria. Con el bloque *Open/Create/ Replace File*, en modo *"open"*, se abre el archivo de texto plano y posteriormente con el bloque *Read From Text File*, se lee el contenido del archivo. En la figura 3.55 se muestra la parte de la programación del diagrama de bloques correspondiente a esta etapa.

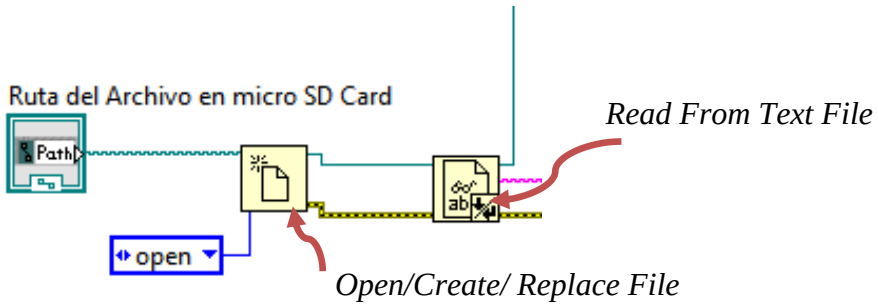


Figura 3.55. Lectura del archivo de origen.

2. **Procesamiento de la cadena de datos:** Cada segundo los datos son almacenados en la memoria microSD en una cadena compuesta de 100 caracteres por línea incluyendo las comas que separan a los datos. El propósito de esta etapa es separar a éstos, con el objetivo de quitar las comas que se encuentran entre ellos. Para este propósito se hace uso de dos bloques para manipulación de caracteres en LabVIEW, el primero es el llamado *Pick Line*, cuya función es tomar una línea del archivo de texto en cada ciclo de ejecución del programa, mientras que el segundo bloque, llamado *Spreadsheet String to Array,* separa los datos y los coloca en un arreglo, quitando de la cadena a las comas separadoras. En la figura 3.56 se muestra la programación de esta etapa.

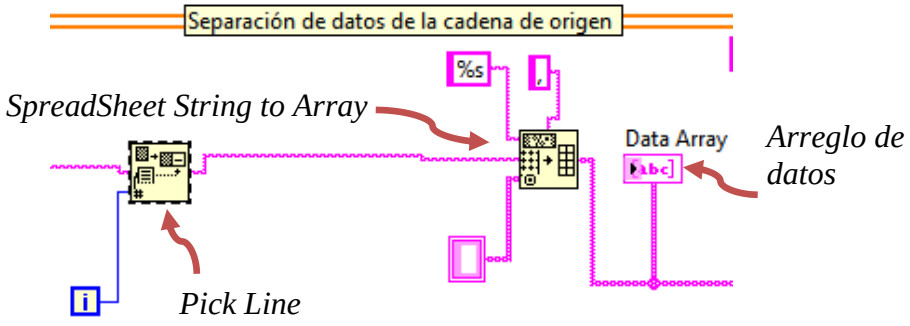


Figura 3.56. Separación de la cadena de datos.

Una vez que los datos han sido separados, éstos son manipulados numéricamente para ser representados en indicadores gráficos y numéricos. Para dichos fines, se hace uso de múltiples bloques de programación como son indicadores y controles numéricos, arreglos, gráficas, clusteres, etcétera; los cuales no serán explicados a detalle y pueden ser consultados en la bibliografía citada en este trabajo. En la figura 3.57 se muestra la parte del diagrama de bloques correspondiente al procesamiento de los datos.

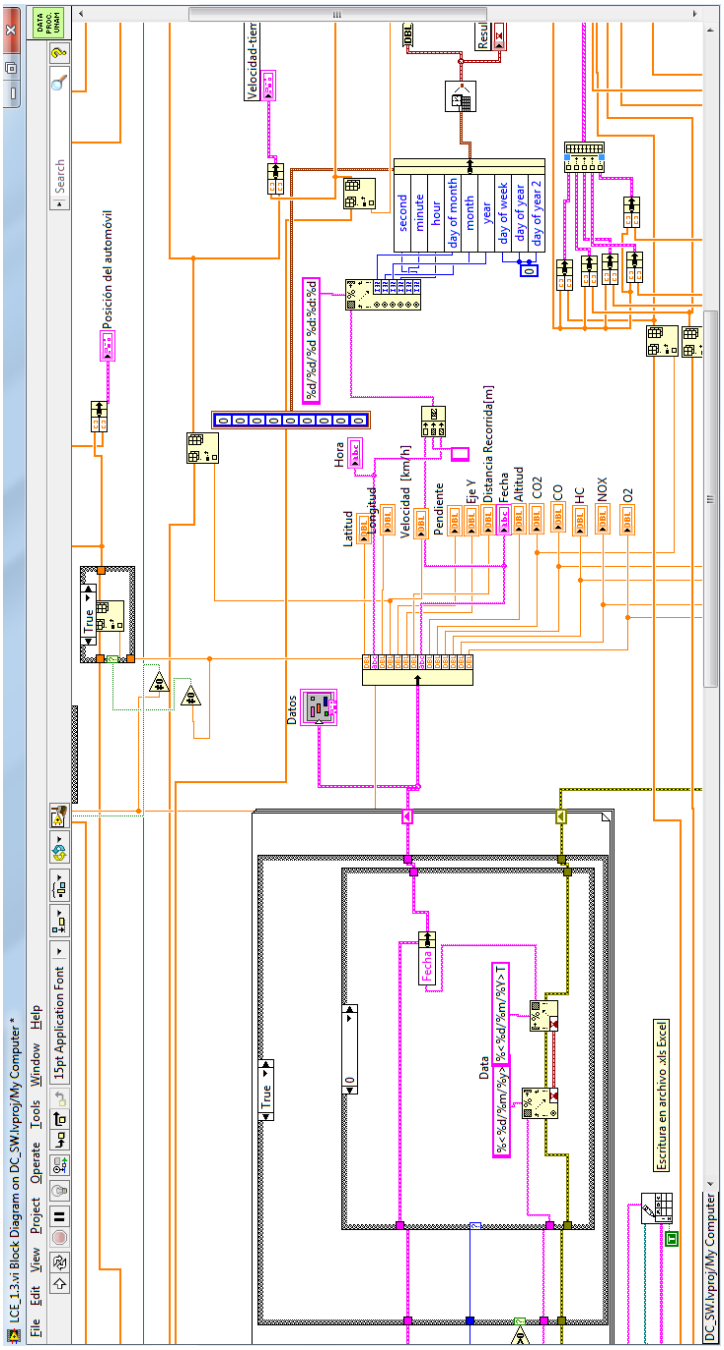


Figura 3.57. Procesamiento de la cadena de datos.

3. **Escritura en un archivo destino**. Finalmente, con el objetivo de tener un registro histórico de los datos y poder analizarlos estadísticamente, los datos son nuevamente almacenados, pero esta vez, en un archivo de hoja de cálculo. Para ello, se usan tres bloques principales; en primera instancia se usa el bloque *Open/Create/ Replace File*, para crear el archivo, seguido del bloque llamado *Write to SpreadSheet File* y que se encarga de la escritura de los datos en archivo de hoja de cálculo y que previamente han sido separados. Finalmente se usa el bloque *Close File* para cerrar el archivo. En la figura 3.58 se muestra el proceso de escritura en la hoja de cálculo.

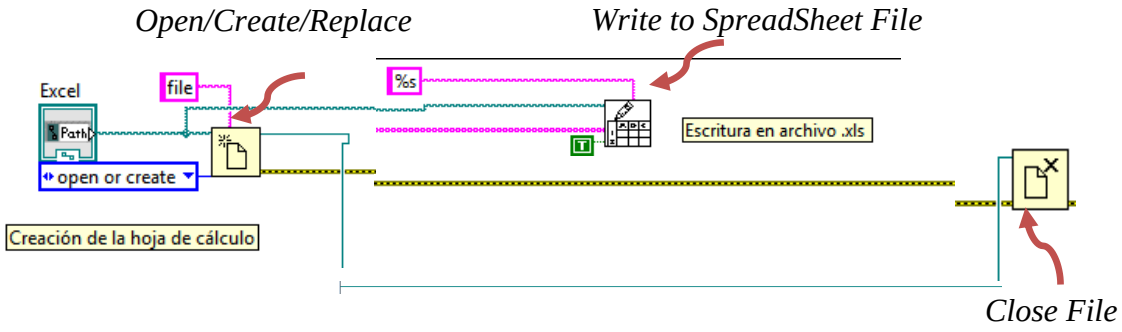


Figura 3.58. Escritura de datos en hoja de cálculo.

3.3.2. Panel Frontal

Una vez explicado brevemente el diagrama de bloques del DRIVE-SOFT, ahora es momento de explicar cada uno de los indicadores que componen el panel frontal del instrumento virtual, mostrado en la figura 3.59.

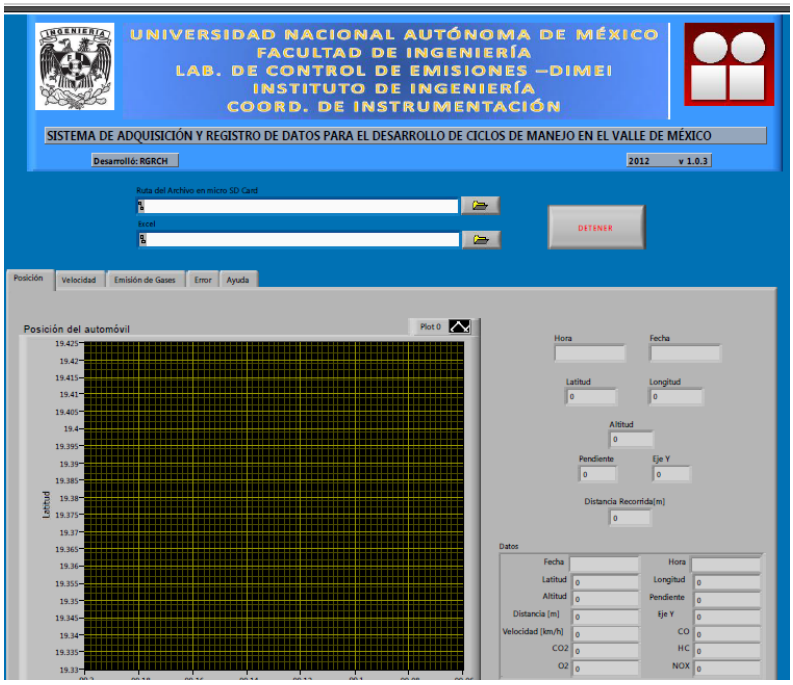


Figura 3.59. Panel frontal del DRIVE-SOFT.

Como se mencionó anteriormente, además de registrar los datos en una hoja de cálculo, el uso de DIRVE-SOFT permite al usuario tener una herramienta visual de los datos que se obtuvieron durante un muestreo.

Una vez que se ha explicado de manera general la estructura del programa, a continuación se describen detalladamente las partes más importantes que componen al DRIVE-SOFT.

3.3.3. Ingreso de archivos

En la parte superior del panel frontal hay dos barras de texto, la primera permite seleccionar el archivo origen con extensión *.txt* almacenado en la memoria microSD, mientras la segunda se utiliza elegir el archivo destino, esto es, la hoja de cálculo en la que se registraran los datos después de ser procesados. En la figura 3.60 se muestran las barras de texto para el ingreso de los archivos.

Figura 3.60. Barras de texto para el ingreso de los archivos.

3.3.4. Indicadores y gráficas

El *software* DRIVE-SOFT cuenta con 3 principales gráficas, referentes a la posición, velocidad y emisión de gases contaminantes, las cuales están distribuidas para mayor comodidad visual en diferentes pestañas, además de una pestaña para mensajes de error y ayuda al usuario, como se muestra en la figura 3.61

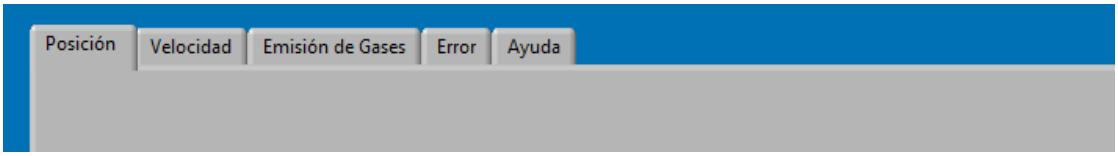


Figura 3.61. Pestañas de selección de gráficas y mensajes.

La primera pestaña, figura 3.62, corresponde a la posición del automóvil; en ella, se podrá observar del lado izquierdo, el gráfico correspondiente a la trayectoria que siguió el automóvil durante el muestreo; mientras del lado derecho se observarán las variables adquiridas durante la ejecución de la prueba de muestreo, datos que estarán siendo procesados.

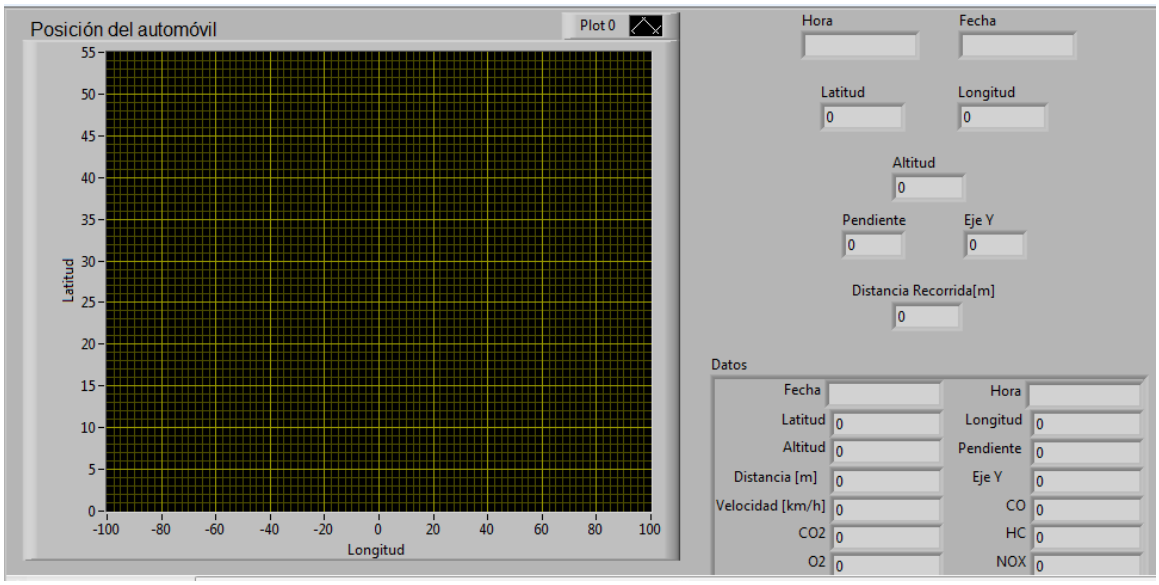


Figura 3.62. Gráfica de la posición del automóvil.

En la segunda pestaña, figura 3.63, se observará el perfil velocidad-tiempo (ciclo de manejo), con el objetivo de que el usuario pueda realizar observaciones sobre el comportamiento dinámico de la velocidad registrada durante el muestreo.

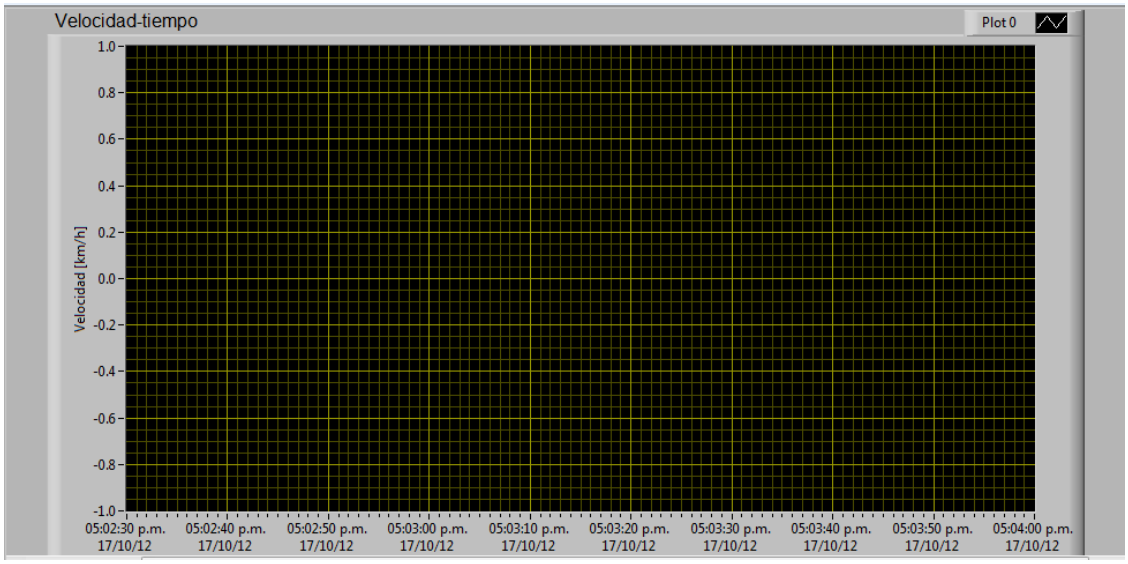


Figura 3.63. Gráfica de la velocidad del automóvil.

La tercera pestaña, mostrada en la figura 3.64, corresponde a la gráfica de las concentraciones de las emisiones contaminantes, obtenidas del analizador ANDROS 6600, respecto al tiempo. Al lado derecho de la gráfica se observa el color asignado a la gráfica de cada uno de los cinco gases, para que éstos puedan ser distinguidos por el usuario.

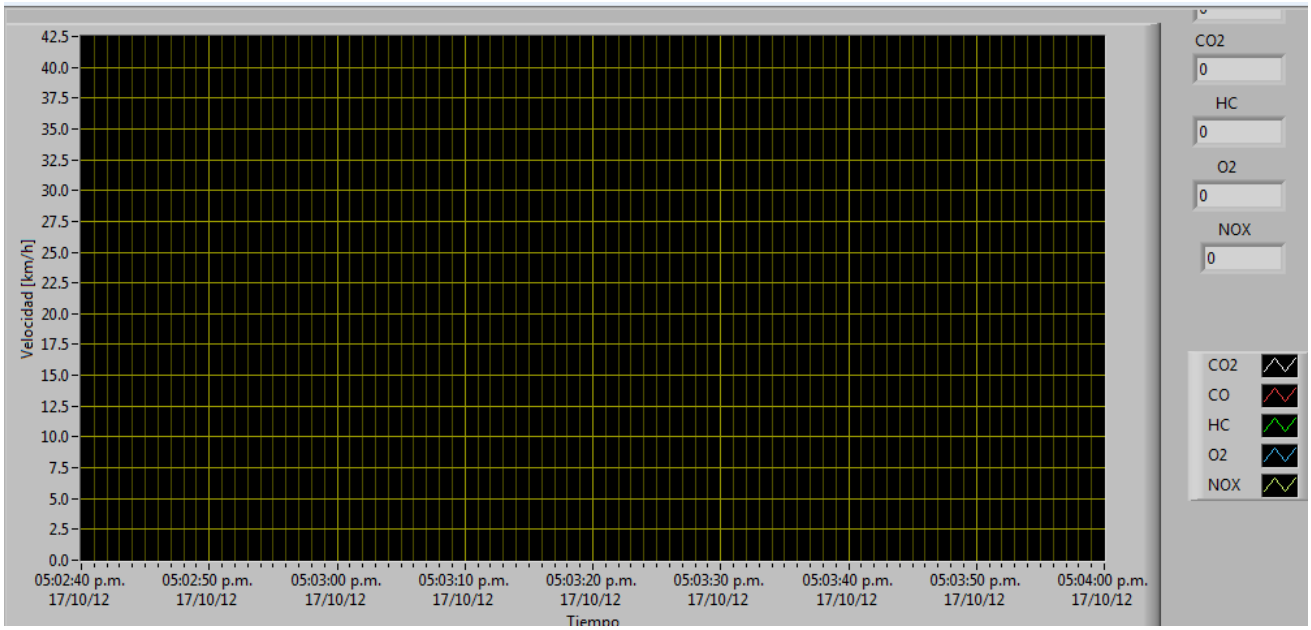


Figura 3.64. Gráfica de las emisiones contaminantes.

En la pestaña “Error”, figura 3.65, se muestran los posibles errores en la lectura y/o escritura de los archivos necesarios para el procesamiento de los datos. En dichos indicadores se notifican también las posibles causas del error, lo cual permite al usuario corregir la selección de los registros y repetir el proceso de ingreso de los archivos.

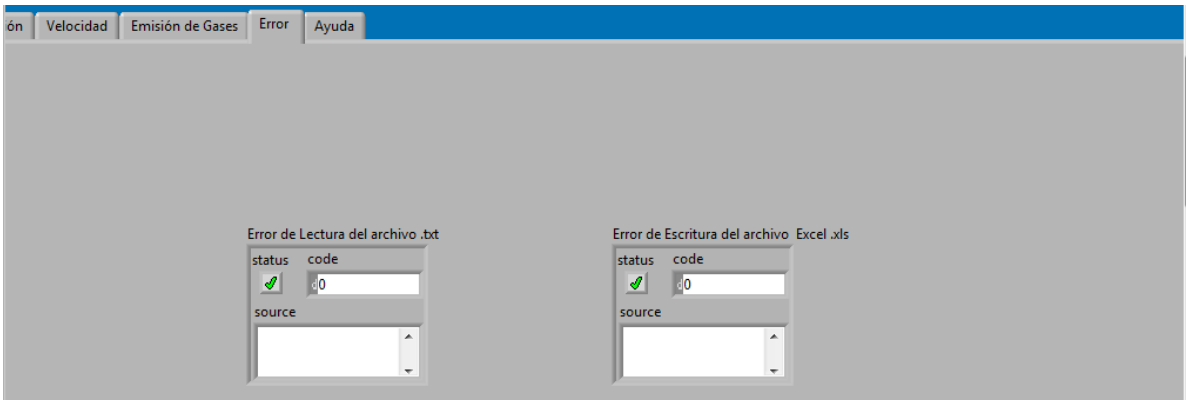


Figura 3.65. Pestaña de error en el ingreso de archivos.

Finalmente, la última pestaña, figura 3.66, corresponde a una breve descripción de los pasos a seguir en la puesta en marcha del programa DRIVE-SOFT.

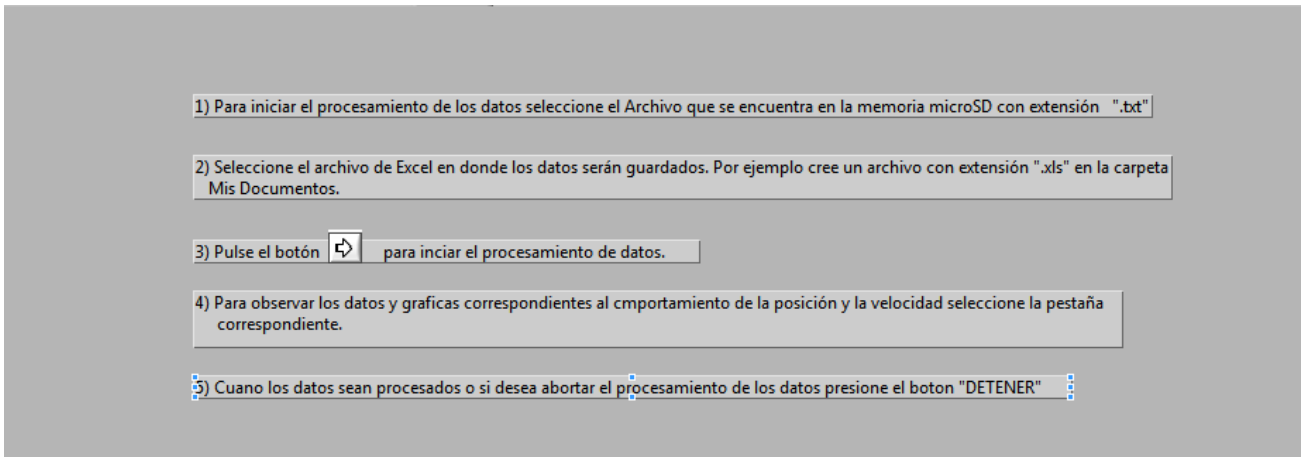


Figura 3.66. Pestaña de ayuda al usuario.

Una vez que se han explicado todos los elementos que intervienen en la funcionalidad del equipo CYCLE-DAQ y el software DRIVE-SOFT, en el siguiente capítulo se describirá la integración, evaluación y puesta a punto de ambos, para que el lector comprenda su funcionamiento en el proceso de desarrollar ciclos de manejo.

C A P Í T U L O 4

INTEGRACIÓN, EVALUACIÓN Y PUESTA A PUNTO

En este capítulo se comenta la integración final del equipo CYCLE-DAQ, la evaluación de sus dispositivos más importantes y se abordan las pruebas finales realizadas al hardware y al software, necesarias para asegurar el correcto desarrollo de ciclos de manejo en el Valle de México.

4.1. Integración del equipo CYCLE-DAQ

Una vez que se analizó por separado el funcionamiento de cada uno de los dispositivos que conforman al equipo CYCLE-DAQ, éstos se integraron como primer prototipo en una tablilla perforada para circuitos electrónicos, lo cual permitió realizar distintas pruebas, hacer cambios e integrar más componentes al sistema de acuerdo a los requerimientos que fueron surgiendo conforme el desarrollo avanzó. En la figura 4.1 se muestra el primer prototipo del equipo CYCLE-DAQ.

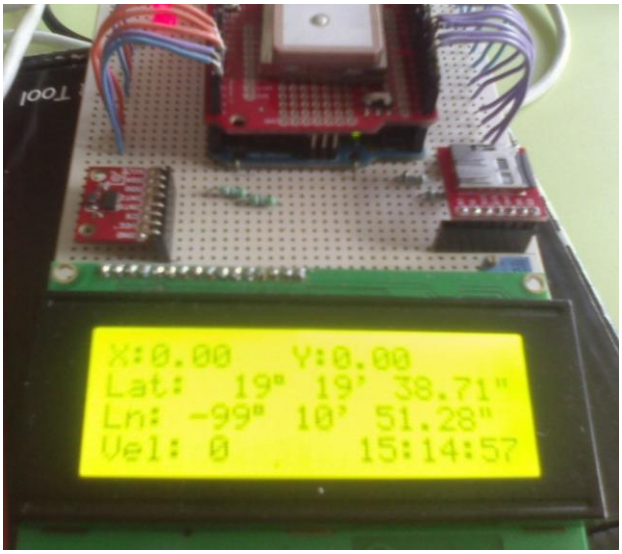


Figura 4.1. Primer prototipo del equipo CYCLE-DAQ.

Una vez que se integraron todos los elementos al equipo, se realizó el diagrama eléctrico del mismo, mostrado en la figura 4.2.

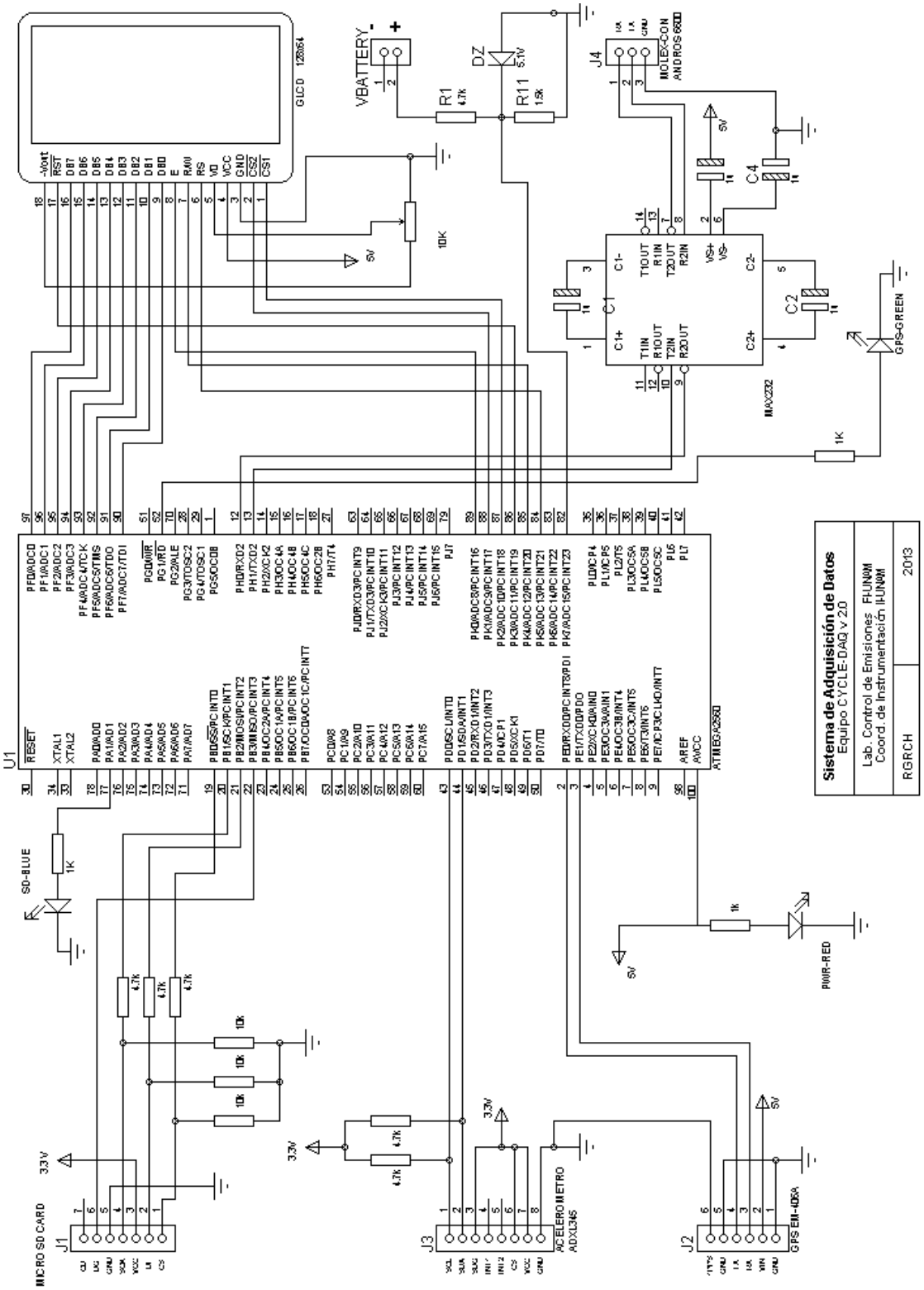


Figura 4.2. Diagrama eléctrico del equipo CYCLE-DAQ.

Posteriormente, ya que los componentes de hardware quedaron perfectamente definidos, se siguió el diagrama eléctrico de la figura 4.2 para la elaboración del circuito impreso o *Printed Circuit Board (PCB).* El *PCB* se diseñó teniendo en cuenta dos principales objetivos: tener un orden en los dispositivos que conforman al equipo CYCLE-DAQ y evitar posibles fallas eléctricas como cortocircuitos o falsos contactos, producidos por el uso de cables, como era el caso del primer prototipo desarrollado. Tanto el diseño, como la fabricación del circuito impreso, se hicieron mediante herramientas de manufactura y diseño asistido por computadora. En la figura 4.3 se muestra el original mecánico del PCB.

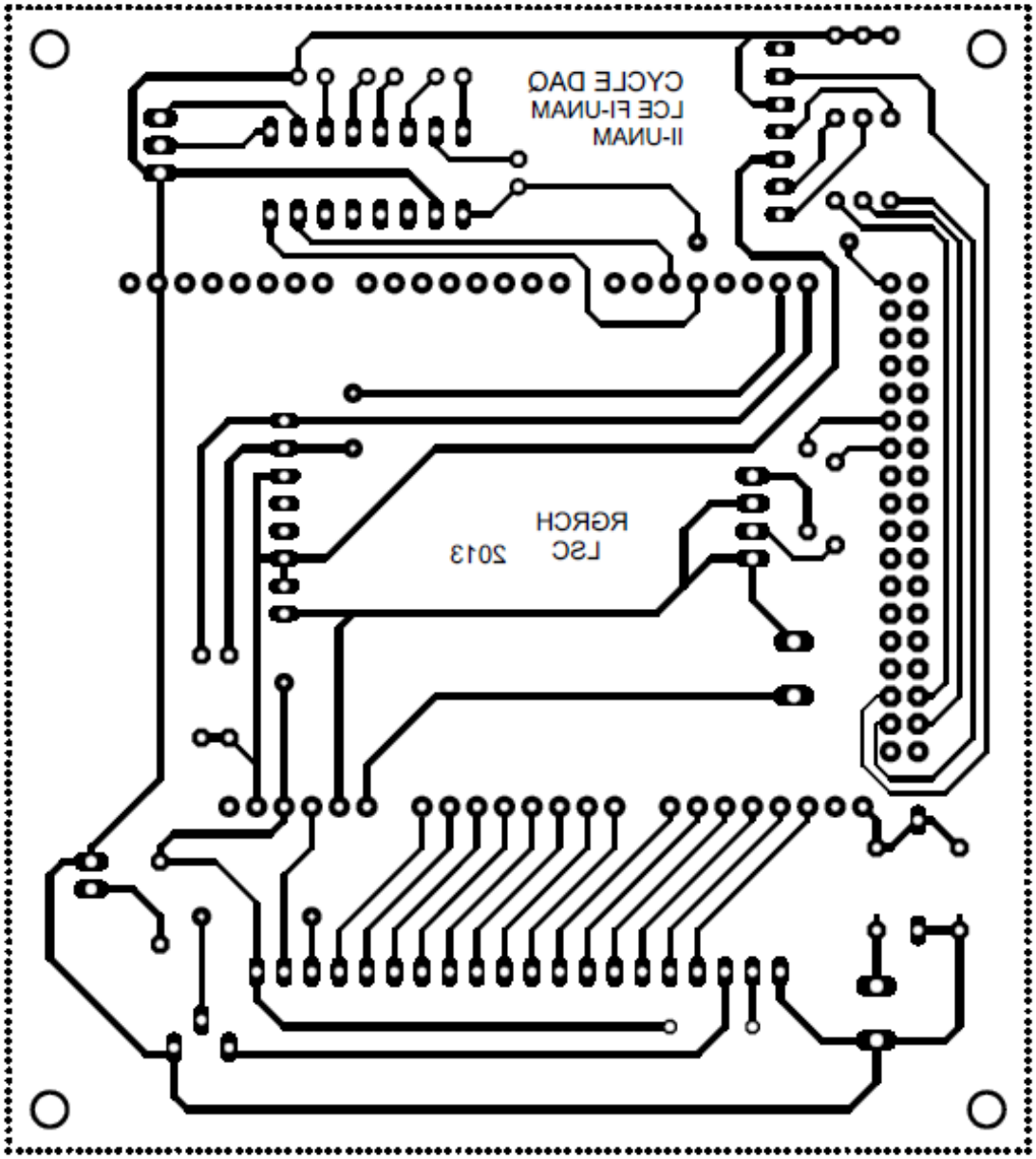


Figura 4.3. Original mecánico del circuito impreso.

Finalmente, se procedió a realizar el montaje de los componentes de *hardware* en el circuito impreso. En las figuras 4.4 y 4.5 se muestra la distribución de dichos elementos, los cuales n numerados para su identificación.

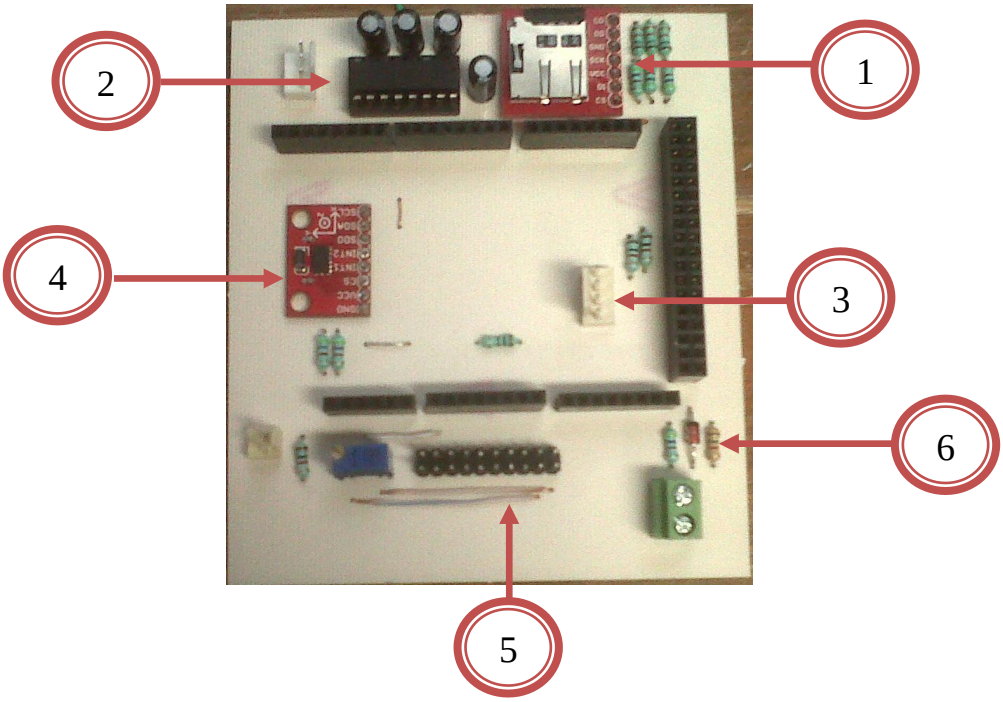


Figura 4.4. Vista superior de los componentes del circuito impreso.

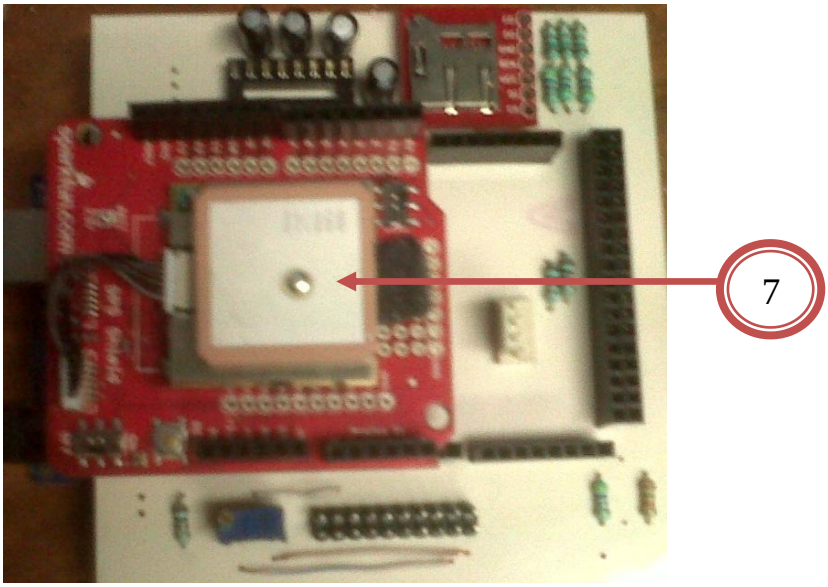


Figura 4.5. Vista superior del receptor GPS en el circuito impreso.

A continuación se nombran los componentes mostrados en las figuras anteriores:

1. Ranura para la inserción de la memoria microSD
2. Transceptor para comunicación RS-232 con el analizador de gases ANDROS 6600
3. Conector para indicadores visuales, LEDs.
4. Acelerómetro ADXL345
5. Conector para el display gráfico
6. Sensor de voltaje de la batería del automóvil
7. Receptor GPS EM-406A

Debido a que el diseño del equipo se basó en el apilamiento de tarjetas, esto es, el *PCB* se colocó encima de la tarjeta de desarrollo, en la figura 4.6, se muestra una vista en isométrico de la integración de la tarjetas electrónicas apiladas de abajo hacia arriba en el siguiente orden: tarjeta de desarrollo, circuito impreso y tarjeta del receptor GPS.

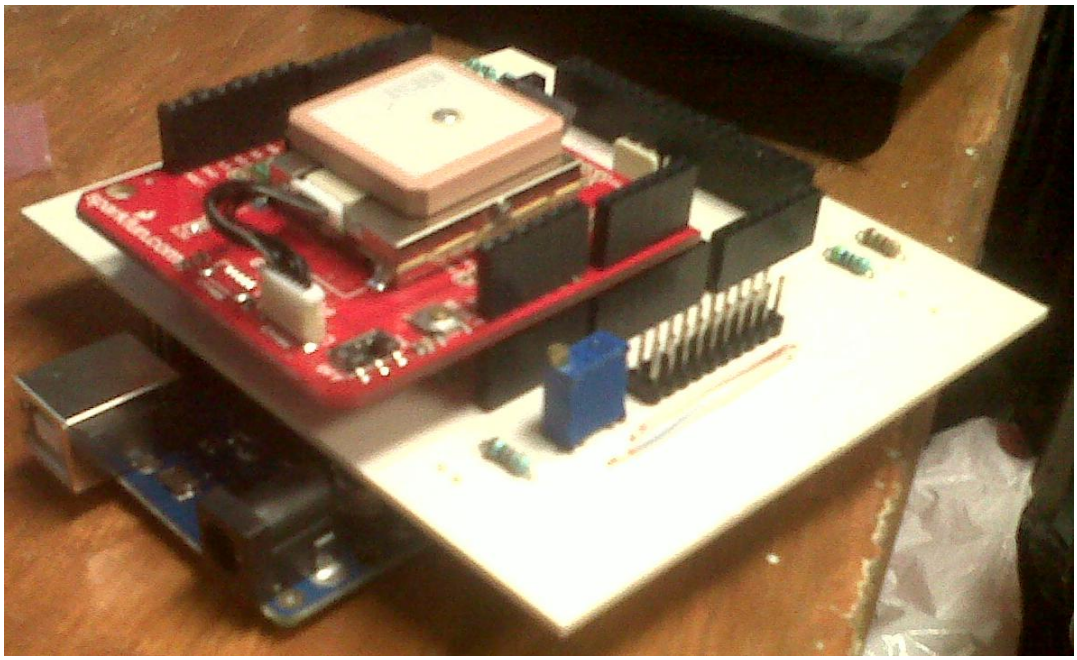


Figura 4.6. Vista en isométrico de las tarjetas electrónicas.

4.2. Evaluación del sistema

Cuando se desarrolló el primer prototipo del equipo CYCLE-DAQ se realizaron las pruebas necesarias para verificar el funcionamiento de cada uno de los dispositivos periféricos como son: conexión física, comunicación con el microcontrolador y registro de datos. Una vez que el equipo fue implementando en el circuito impreso se repitieron dichas pruebas con el objetivo de asegurar que cada dispositivo es capaz de responder de manera adecuada a las necesidades planteadas por el grupo de trabajo del LCE. A continuación se describe la evaluación y pruebas realizadas a los elementos más importantes del equipo y al *software* de procesamiento de datos.

4.2.1. Evaluación del receptor GPS

El objetivo de esta evaluación fue corroborar que el receptor GPS pudiera establecer comunicación con la tarjeta de desarrollo, una vez que se implementó el programa necesario en éste último, además de, corroborar que efectivamente los datos arrojados por el receptor son lo más exactos posibles, considerando las fuentes de error en el sistema GPS.

Para la evaluación del receptor GPS, fue necesario monitorear las variables de interés, estas son: hora, fecha, latitud, longitud, altitud y velocidad. Dichas variables son visualizadas a través del monitor del puerto serial (COM), del entorno de desarrollo de Arduino, el cual permite imprimir cadenas de texto y variables de tipo numérico en la pantalla de la computadora personal en donde la tarjeta de desarrollo es conectada, con el objetivo de monitorear su actividad con los dispositivos que interactúan con ella, como es el caso del receptor EM406-A. En la figura 4.7, se muestra la captura de pantalla del monitor serial en donde las variables que corresponden al receptor GPS son visualizadas.

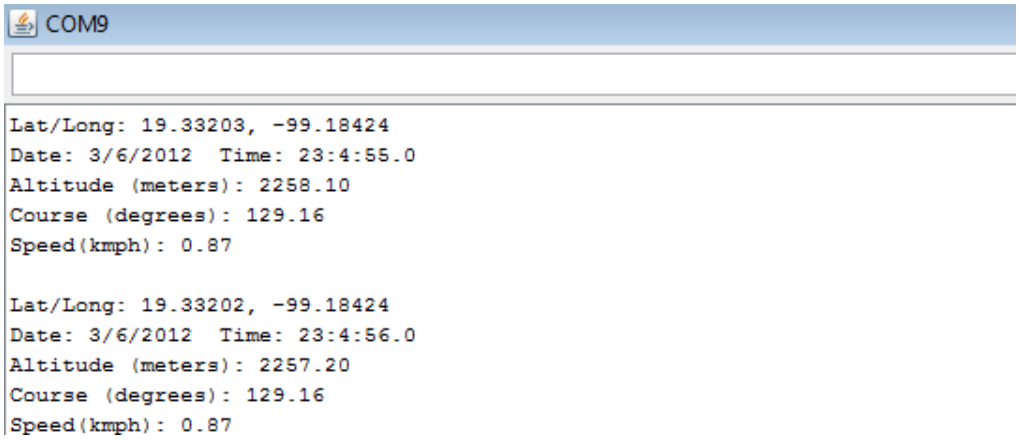


Figura 4.7. Monitoreo de las variables provenientes del receptor GPS.

Los datos mostrados en la figura 4.7, corresponden a dos lecturas consecutivas realizadas por la tarjeta Arduino, esto es, en dos segundos. Se puede observar que, en ese instante, lo que los datos no varían significativamente entre una lectura y otra ya que el receptor GPS se encontraba fijo.

Ahora bien, con el objetivo de constatar que las lecturas tomadas del receptor GPS coinciden con el lugar físico en el que éste se encuentra, los datos de la figura anterior fueron introducidos al *software Google Earth*, éste permite ubicar virtualmente al usuario en cierto lugar, con sólo introducir las coordenadas de latitud y longitud conocidas. En la figura 4.8 se muestra la captura de pantalla del programa, en la que se muestra que el receptor GPS se encontraba ubicado en la Facultad de Ingeniería, en Ciudad Universitaria.

Figura 4.8. Comprobación de la ubicación del receptor GPS.

Para corroborar el dato correspondiente a la altitud, se usó el *software* anteriormente mencionado, obteniendo los mismos resultados en Google Earth y el receptor GPS

Los datos de la fecha y la hora se verificaron en el momento de la evaluación del receptor GPS, comparándolos con los datos de la computadora personal en donde la prueba se realizó.

El dato correspondiente a la velocidad, mostrado en la figura 4.7, tiene un valor muy próximo a cero, debido a que el receptor se encontraba en reposo. De manera dinámica, este valor se comprobó al introducir el receptor GPS en un vehículo, comparando el dato de velocidad con el valor marcado por

el odómetro del automóvil y un navegador GPS comercial de la marca *Garmin*. Como resultado de lo anterior, se obtuvo una diferencia de aproximadamente 2 [km/h] entre los tres dispositivos, considerando como referencia válida al dato proporcionado por el navegador comercial. La diferencia entre los valores registrados, se ajusta perfectamente a la precisión requerida por el grupo de trabajo del LCE. La comparación de las velocidades obtenidas es mostrada en la tabla 4.1.

DISPOSITIVO	*VELOCIDAD REGISTRADA [KM/H]*
Odómetro del vehículo	38
Navegador GPS Garmin	39.5
Receptor GPS EM-406A	40

Tabla 4.1. Comparación de velocidades.

Finalmente, se corroboró el dato correspondiente a la distancia recorrida por el vehículo, la cual es calculada con la fórmula de Haversine. Para ello, se realizó una prueba con el receptor GPS a bordo en una motocicleta marca *Honda* con motor de 120 c.c., siguiendo una trayectoria alrededor del circuito principal de Ciudad Universitaria. La verificación de la distancia obtenida por el receptor, se hizo con respecto a la distancia registrada por el navegador *Garmin*, obteniendo como resultado, una distancia de 3.9 [km] por parte del receptor EM-406A y de 4 [km] en el navegador GPS. Con los datos obtenidos, observamos que la diferencia entre ambas distancias es aceptable.

Las pruebas anteriormente mencionadas confirman que el receptor GPS EM406-A cumple con los requisitos establecidos y que los datos obtenidos de él, pueden ser considerados como confiables.

4.2.2. Evaluación y calibración del acelerómetro

Una vez que se verificó que el acelerómetro puede establecer comunicación con la tarjeta de desarrollo mediante el *bus* I^2C, y que se implementaron las funciones necesarias para obtener como resultado un ángulo de inclinación, fue necesario corroborar que el dato correspondiente a dicho ángulo fuera correcto. El proceso de verificación se realizó siguiendo los tres pasos a continuación se describen:

1. **Fijar una referencia:** Este paso consiste en ubicar al acelerómetro en una superficie lisa y que no tenga inclinación alguna, esto último podemos saberlo colocando un “nivel” sobre la superficie. Una vez colocado el acelerómetro en este punto, el dato correspondiente a la inclinación del eje X deberá ser cero.

2. **Calibración por *software*:** Como segundo paso, fue necesario observar el dato del eje X en el monitor serial de Arduino, el cual no fue exactamente de cero, si no de 0.9 [°], por tal motivo se realizó un simple ajuste a la programación de la tarjeta de desarrollo, esto es, se restó 0.9 a la variable correspondiente al eje X.

3. **Comprobación del dato con otro instrumento:** Una vez realizado el ajuste anterior, se procedió a realizar mediciones con el acelerómetro, inclinándolo respecto a la referencia fijada anteriormente y enviando los datos al monitor serial. Para comprobar que las mediciones mostradas en la pantalla de la computadora personal fueran las indicadas, se colocó un transportador para medición de ángulos en conjunto con el acelerómetro, de esta manera, cuando el acelerómetro era inclinado, el dato mostrado en pantalla debía coincidir con el observado en el transportador. Cabe mencionar que dicho proceso de medición puede contener dos errores: el primero es inherente al observador y el segundo es debido a la resolución del transportador.

Al finalizar el procedimiento anterior, se pudo observar que los ángulos medidos por el acelerómetro no exceden los límites de precisión establecidos en la tabla de requerimientos proporcionada al inicio del proyecto. Por tal motivo, el acelerómetro ADXL345 puede señalarse como un instrumento adecuado para medir ángulos de inclinación y, específicamente en el caso del desarrollo de ciclos de manejo, para medir la inclinación de la trayectoria que un vehículo sigue.

4.2.3. Evaluación y calibración del analizador de gases

El tercer dispositivo a evaluar es el analizador de gases ANDROS 6600. Es de suma importancia verificar el correcto funcionamiento del analizador debido a que, en caso de presentar anomalías, las mediciones de las concentraciones de gases serán incorrectas y por tanto el análisis posterior también lo será.

El primer paso para poner en marcha el analizador de gases, es calibrarlo. Éste procedimiento se realiza con ayuda del *software* proporcionado por el fabricante de la banca y un tanque de gases de calibración, ubicado en el LCE y que, en la parte superior, cuenta con una etiqueta en donde se indican las concentraciones de los gases dentro del tanque. En la figura 4.9 se muestra una imagen del tanque y la etiqueta de información.

Figura 4.9. Tanque de gases de calibración para el ANDROS 6600.

El objetivo de la calibración es lograr que el analizador mida las concentraciones de los gases del tanque: CO_2, CO, O_2, HC y NOx. Para ello, en el software de la banca, deben introducirse los valores mostrados en la figura anterior, este procedimiento permitirá tomar a dichas concentraciones como referencia para las mediciones que posteriormente se realicen. En la figura 4.10 se muestra la captura de pantalla en donde las concentraciones del tanque son enviados al analizador de gases para su calibración.

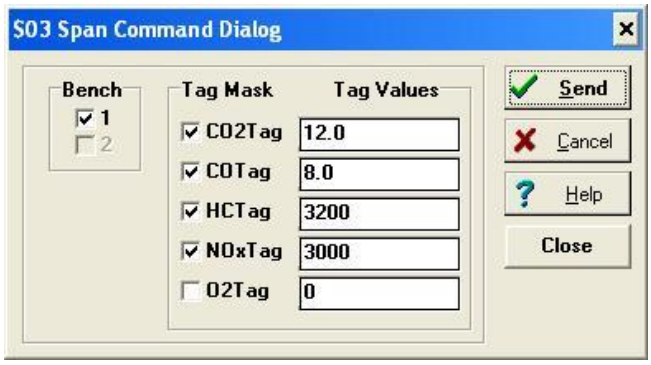


Figura 4.10. Valores de calibración para el analizador de gases.

Una vez que la calibración fue realizada, se corroboró que el analizador de gases midiera las concentraciones correctas, pero esta vez con el equipo CYCLE-DAQ. Este proceso de medición, al igual que para los dispositivos anteriormente evaluados, se hizo con ayuda del monitor serial del entorno de desarrollo, figura 4.11, en el que se observan las concentraciones de los gases, producto de la correcta comunicación entre el analizador y el equipo.

```
----------------------------CONCENTRACIONES DE GASES -----------------------
CO2: 12
CO: 7
HC_HEX: 1492
O2: 0
NOX: 3054
```

Figura 4.11. Concentraciones de gases medidas por el equipo CYCLE-DAQ.

Cabe mencionar que, debido a la resolución manejada por el analizador de gases en cuanto a las concentraciones se refiere, los datos adquiridos por el CYCLE-DAQ no son idénticos a los señalados por el tanque de gases de calibración, además que, la concentración de HC es reportada en hexano, equivalente a la mitad de la concentración en propano, el cual es proporcionado por el tanque de gases. En la tabla 4.2 se muestra la comparación entre los valores de las concentraciones del tanque y los adquiridos por el equipo CYCLE-DAQ.

GAS	*Concentración en el tanque*	*Concentración adquirida por el CYCLE-DAQ*
NOX	3000 p.p.m.	3054 p.p.m.
CO2	12 %	12 %
CO	8 %	7 %
HC	1500 p.p.m.	1492 p.p.m.

Tabla 4.2. Comparación de las concentraciones de gases.

Después de corroborar los valores de las concentraciones, fue necesario realizar una prueba real con el analizador de gases, conectando la sonda de muestreo del analizador al escape de un vehículo. Dicha prueba, fue realizada de manera estática, es decir, el vehículo únicamente se encendió y fue sometido a la acción del acelerador, pero sin avanzar.

La prueba estática se realizó con el *software* original y tuvo dos principales objetivos, el primero, verificar que todos los sistemas que componen al analizador de gases funcionaran correctamente; mientras que el segundo consistió en comprobar que el comportamiento de éste después de 30 minutos operación, fuera el indicado por el fabricante del mismo, esto es, que se realizará el ajuste de cero de forma correcta para evitar mediciones erróneas en las concentraciones de gases.

En la figura 4.12, se muestran una imagen del proceso de la prueba estática.

Figura 4.12. Prueba estática al analizador de gases.

4.2.4. Evaluación del sensor de voltaje

Para verificar el funcionamiento del sensor de voltaje de la batería del automóvil, se utilizó un multímetro, con el objetivo de comparar las mediciones arrojadas por éste y las registradas por el equipo CYCLE-DAQ. Además, esta prueba fue necesaria para determinar el comportamiento del voltaje de la batería en momentos clave de la operación del automóvil. El primer paso, consistió en medir y registrar el voltaje cuando el vehículo se encuentra únicamente con el interruptor cerrado, es decir, sin marcha. En dicho momento, se obtuvo un valor de 13.11 [V] en el multímetro y de 13.01 [V] en el sensor de voltaje. Como segundo paso, se le dio marcha al vehículo y el voltaje obtenido fue de 10.01 [V] para el caso del multímetro y 10.09 para el sensor. Finalmente, una vez que el automóvil se encendió el multimetro arrojo 14.61 [V] y el sensor 14.85 [V], sin variaciones significativas durante su trayecto. Los tres pasos anteriores, corresponden a los 3 momentos más importantes en la puesta en marcha de un automóvil y con los cuales se pudieron obtener las siguientes conjeturas:

1. El valor del voltaje de la batería sin el motor del automóvil en funcionamiento, se encuentra por debajo del voltaje medido cuando el motor está operando.
2. Cuando se enciende el vehículo, el voltaje de la batería cae en aproximadamente 3 unidades.

3. El voltaje de la batería cuando el vehículo se encuentra circulando no varía significativamente, por lo que no se ve afectado por cargas presentes en el motor, como es el caso de una aceleración súbita.
4. El sensor del voltaje de la batería únicamente sirve para comprobar que, cuando el automóvil es puesto en marcha, las concentraciones de los cincos gases emitidos por éste, son mayores que en cualquier otra situación.

En la figura 4.13 se muestran las imágenes captadas durante los 3 sucesos anteriormente mencionados.

Figura 4.13. Mediciones del voltaje de la batería.

Al finalizar la evaluación del sensor del voltaje de la batería se tuvo certeza de que las mediciones realizadas por el sensor de voltaje son válidas y que éste ayudará al análisis estadístico de las emisiones contaminantes.

4.2.5. Evaluación del proceso de registro de datos en la memoria microSD

Hasta el momento se ha comprobado y verificado el correcto funcionamiento de los dispositivos más importantes que integran al equipo CYCLE-DAQ; sin embargo, la adquisición de las variables provenientes de dichos dispositivos no es el único objetivo del equipo. El registro de dichas variables es un proceso imprescindible para poder desarrollar los ciclos de manejo por parte del personal del LCE, ya que sin el registro de los datos, los objetivos del proyecto no se cumplen. El propósito de este apartado es confirmar que el proceso de registro se realiza e integra adecuadamente, mediante la descripción de las pruebas que se realizaron a la memoria microSD y al archivo de registro.

El manejo de la memoria microSD fue detallado en el capítulo anterior, en donde se describió cómo es que la tarjeta de desarrollo realiza el reconocimiento de la memoria y asigna un nombre al archivo creado. Para ilustrar cómo es que la tarjeta de memoria es reconocida y se notifica al usuario sobre dicha situación, en la figura 4.14 se muestra la pantalla de configuración de dispositivos del equipo CYCLE-DAQ, en donde se marca con "OK" a los elementos que se han conectado y comunicado correctamente con el equipo.

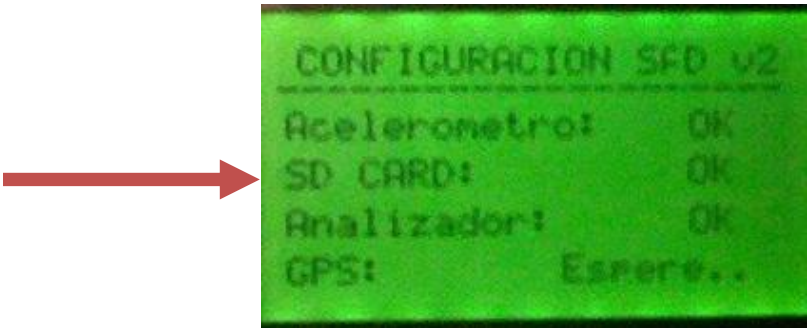


Figura 4.14. Reconocimiento de la memoria microSD.

Un aspecto importante de mencionar, es que, durante las pruebas se insertaron 3 diferentes memorias microSD en el equipo CYCLE-DAQ de diferente fabricante y capacidad. En la tabla 4.3 se muestra la relación de tarjetas microSD probadas.

CAPACIDAD DE LA MEMORIA	*FABRICANTE*
2 [GB]	*SanDisk*
4 [GB]	*Samsung*
8 [GB]	*Sony*

Tabla 4.3. Memorias microSD probadas en el equipo CYCLE-DAQ.

Al insertar las memorias mencionadas en la tabla anterior, no se presentó problema en el reconocimiento de éstas. Por tal motivo, la memoria elegida para el registro de los datos fue la de 8 [GB] marca *Sony*, debido a que es la de mayor capacidad y a la facilidad para ser adquirida.

Una vez que la memoria ha sido reconocida, fue necesario verificar que el archivo *LOGGERxx.txt* haya sido creado, para lo cual, se retiró la tarjeta microSD de la ranura correspondiente en el equipo CYCLE-DAQ. Posteriormente, se introdujo en una computadora personal, para acceder a su contenido. Cuando la memoria microSD fue reconocida por el SO de la PC, dentro de ella se puede encontrar el archivo o archivos generados. Cabe mencionar que cada que el equipo CYCLE-DAQ es energizado, se genera un nuevo archivo en el directorio raíz de la memoria.

En la figura 4.15 se muestra la captura de pantalla en donde, se aprecian siete archivos generados por el equipo CYCLE-DAQ.

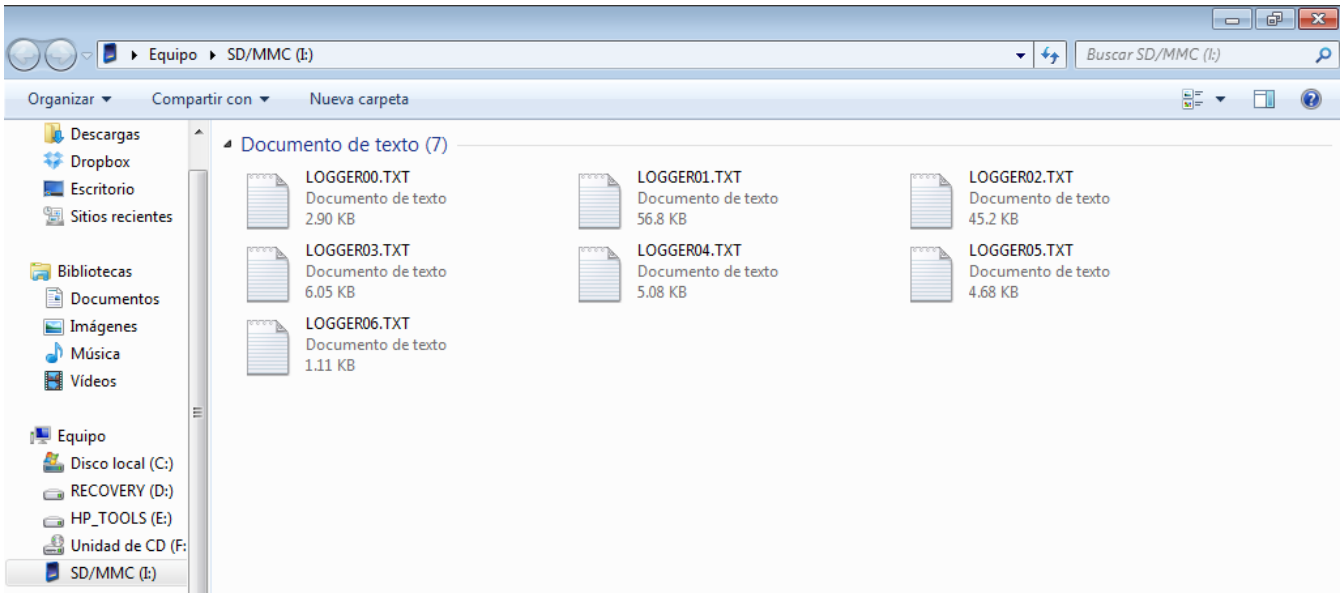


Figura 4.15. Archivos generados por el equipo CYCLE-DAQ.

Para comprobar que el registro de los datos se realizó correctamente, se abrió uno de los archivos generados y se analizó cuidadosamente, observando que efectivamente, los datos fueran escritos en el archivo siguiendo el formato deseado, esto es, una línea por segundo y los datos separados por una coma (ver figura 4.16).

LOGGER00: Bloc de notas

Archivo Edición Formato Ver Ayuda

```
1/3/13,20:36:35,19.616832,-99.297981,1.10,0.000,0,8.9,-89.8,0,0,0,0,0,6.5
1/3/13,20:36:36,19.616260,-99.297569,2301.30,76.885,5,8.9,-89.8,0,0,0,0,0,6.5
1/3/13,20:36:37,19.616327,-99.297569,2304.70,84.292,1,8.7,-90.1,0,0,0,0,0,6.5
1/3/13,20:36:38,19.616355,-99.297561,2309.60,87.618,0,15.5,-90.1,0,0,0,0,0,6.5
1/3/13,20:36:39,19.616355,-99.297546,2308.30,89.228,0,14.1,-90.3,0,0,0,0,0,6.5
1/3/13,20:36:40,19.616355,-99.297546,2307.90,89.228,0,15.3,-90.1,0,0,0,0,0,6.5
1/3/13,20:36:41,19.616374,-99.297584,2313.50,93.829,0,10.5,-89.6,0,0,0,0,0,6.5
1/3/13,20:36:42,19.616413,-99.297645,2322.70,101.266,0,15.5,-89.8,0,0,0,0,0,6.5
1/3/13,20:36:43,19.616451,-99.297683,2324.70,107.329,0,11.6,-90.1,0,0,0,0,0,6.5
1/3/13,20:36:44,19.616512,-99.297737,2330.80,116.017,0,8.7,-89.8,0,0,0,0,0,6.5
1/3/13,20:36:45,19.616516,-99.297760,2336.50,118.445,1,12.8,-89.6,0,0,0,0,0,6.5
1/3/13,20:36:46,19.616502,-99.297760,2337.10,119.965,0,14.1,-90.5,0,0,0,0,0,6.5
1/3/13,20:36:47,19.616493,-99.297752,2335.10,121.206,1,10.5,-89.8,0,0,0,0,0,6.5
1/3/13,20:36:48,19.616493,-99.297760,2337.10,122.005,1,13.2,-89.4,0,0,0,0,0,6.5
1/3/13,20:36:49,19.616487,-99.297760,2338.00,122.765,1,15.3,-90.3,0,0,0,0,0,6.5
1/3/13,20:36:50,19.616479,-99.297760,2338.80,123.715,1,10.0,-90.5,0,0,0,0,0,6.5
1/3/13,20:36:51,19.616479,-99.297760,2341.10,123.715,0,9.6,-88.0,0,0,0,0,0,6.5
1/3/13,20:36:52,19.616479,-99.297760,2342.00,123.715,0,9.8,-91.0,0,0,0,0,0,6.5
1/3/13,20:36:53,19.616479,-99.297760,2342.10,123.715,0,10.7,-90.5,0,0,0,0,0,6.5
1/3/13,20:36:54,19.616479,-99.297760,2340.90,123.715,1,18.1,-89.2,0,0,0,0,0,6.5
1/3/13,20:36:55,19.616479,-99.297775,2341.20,125.324,1,11.2,-88.7,0,0,0,0,0,6.5
1/3/13,20:36:56,19.616487,-99.297760,2339.40,127.094,0,11.4,-88.7,0,0,0,0,0,6.5
1/3/13,20:36:57,19.616489,-99.297760,2336.90,127.474,0,12.8,-86.0,0,0,0,0,0,6.5
1/3/13,20:36:58,19.616489,-99.297760,2335.70,127.474,1,11.2,-86.7,0,0,0,0,0,6.5
1/3/13,20:36:59,19.616493,-99.297760,2335.20,127.854,0,8.0,-86.7,0,0,0,0,0,6.5
1/3/13,20:37:0,19.616497,-99.297760,2335.30,128.044,0,10.5,-87.1,0,0,0,0,0,6.5
1/3/13,20:37:1,19.616498,-99.297760,2335.00,128.234,0,8.7,-87.1,0,0,0,0,0,6.5
1/3/13,20:37:2,19.616498,-99.297760,2335.10,128.234,0,8.9,-87.3,0,0,0,0,0,6.5
1/3/13,20:37:3,19.616498,-99.297760,2334.20,128.234,0,6.9,-87.8,0,0,0,0,0,6.5
1/3/13,20:37:4,19.616498,-99.297760,2333.70,128.234,0,8.9,-86.4,0,0,0,0,0,6.5
1/3/13,20:37:5,19.616497,-99.297760,2333.40,128.424,0,2.9,-89.4,0,0,0,0,0,6.5
1/3/13,20:37:6,19.616502,-99.297760,2334.50,129.184,0,9.8,-85.1,0,0,0,0,0,6.5
1/3/13,20:37:7,19.616506,-99.297752,2334.40,130.069,0,5.8,-88.9,0,0,0,0,0,6.5
1/3/13,20:37:8,19.616502,-99.297760,2334.70,130.954,1,12.1,-88.0,0,0,0,0,0,6.5
1/3/13,20:37:9,19.616498,-99.297760,2335.20,131.523,0,17.1,-88.9,0,0,0,0,0,6.5
1/3/13,20:37:10,19.616506,-99.297752,2335.50,132.765,0,10.0,-92.1,0,0,0,0,0,6.5
1/3/13,20:37:11,19.616512,-99.297752,2335.60,133.334,1,6.7,-93.0,0,0,0,0,0,6.5
1/3/13,20:37:12,19.616521,-99.297752,2335.20,134.474,1,10.3,-96.3,0,0,0,0,0,6.5
1/3/13,20:37:13,19.616535,-99.297752,2332.90,135.993,5,4.4,-88.0,0,0,0,0,0,6.5
1/3/13,20:37:14,19.616546,-99.297760,2332.30,137.385,7,7.8,-88.7,0,0,0,0,0,6.5
1/3/13,20:37:15,19.616559,-99.297775,2331.20,139.465,7,10.5,-93.0,0,0,0,0,0,6.5
1/3/13,20:37:16,19.616573,-99.297790,2330.10,141.428,5,6.7,-87.6,0,0,0,0,0,6.5
1/3/13,20:37:17,19.616584,-99.297798,2329.20,142.979,7,12.1,-86.0,0,0,0,0,0,6.5
1/3/13,20:37:18,19.616601,-99.297813,2328.50,145.462,7,8.5,-93.0,0,0,0,0,0,6.5
1/3/13,20:37:19,19.616613,-99.297828,2327.00,147.321,7,5.3,-91.0,0,0,0,0,0,6.5
1/3/13,20:37:20,19.616641,-99.297851,2327.00,151.045,11,10.5,-91.2,0,0,0,0,0,6.5
1/3/13,20:37:21,19.616674,-99.297866,2327.80,154.819,12,5.3,-88.9,0,0,0,0,0,6.5
1/3/13,20:37:22,19.616699,-99.297904,2328.30,159.727,11,16.4,-92.3,0,0,0,0,0,6.5
1/3/13,20:37:23,19.616727,-99.297927,2330.10,163.598,11,14.4,-91.4,0,0,0,0,0,6.5
1/3/13,20:37:24,19.616764,-99.297966,2332.10,169.244,12,14.4,-89.6,0,0,0,0,0,6.5
1/3/13,20:37:25,19.616802,-99.297988,2333.80,173.577,14,7.1,-91.4,0,0,0,0,0,6.5
```

Figura 4.16. Datos almacenados en el archivo LOGGERxx.txt.

La última prueba realizada al proceso de registro de datos, consistió en dejar al equipo CYCLE-DAQ operando por 20 horas continuas, bajo ésta condición se pudo determinar el tamaño del archivo generado y de ésta manera, comprobar que la capacidad de la tarjeta de memoria es suficiente para registrar los 100 archivos que el equipo puede generar. En la figura 4.17 se muestra la captura de pantalla con la información referente al archivo generado por esta prueba.

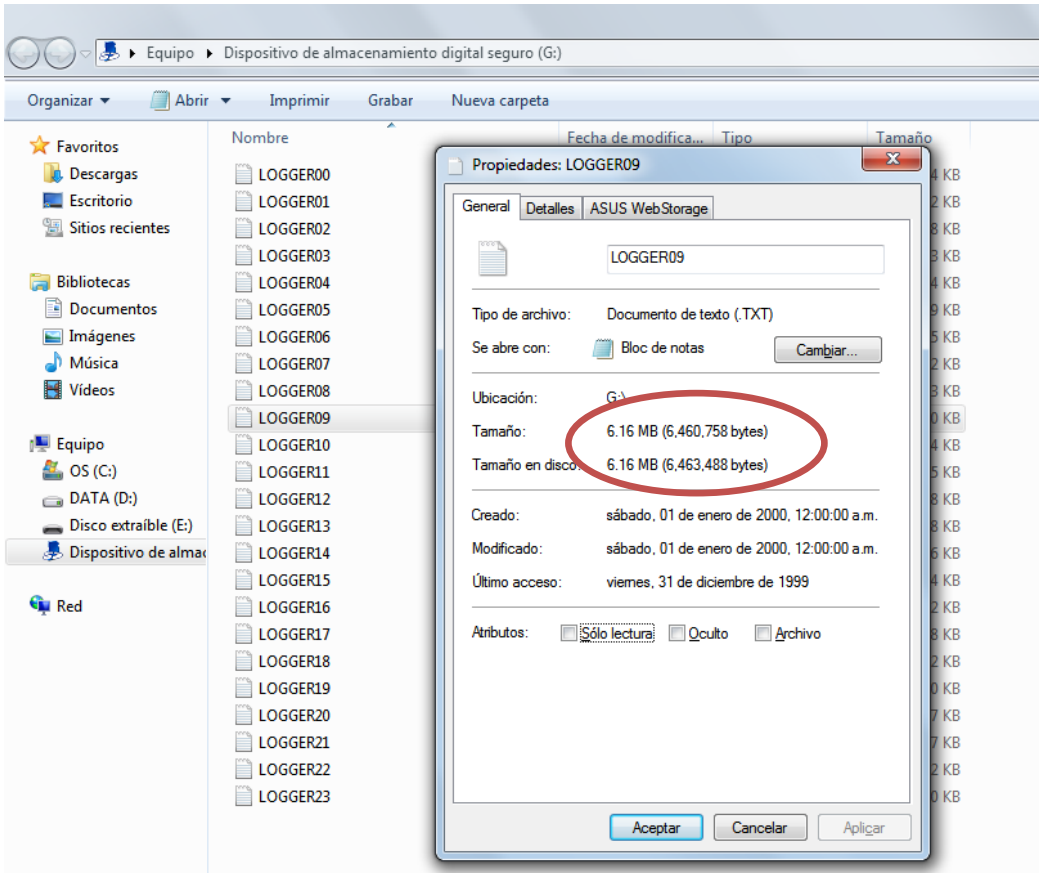


Figura 4.17. Archivo resultante de una prueba de 20 horas.

El archivo registrado en la tarjeta de memoria tuvo 6.16 [MB] de tamaño; considerando que se generaran 100 archivos de dicha longitud, el espacio ocupado en la memoria microSD sería de 616 [MB], esta cantidad representa únicamente el 33% de su capacidad total.

Con las pruebas realizadas a la tarjeta de memoria, se comprueba que el registro de datos se realiza de manera ordena y sin tener grandes volúmenes de información.

4.2.6. Evaluación del módulo de alimentación

En el capítulo anterior, se presentaron los cálculos teóricos para obtener la duración de la batería que alimenta al equipo CYCLE-DAQ, la cual es de 22.92 [h]. El objetivo de este apartado es evaluar el comportamiento de la batería y corroborar que el tiempo de duración de ésta, es lo más cercano posible al valor teórico reportado.

Para realizar el proceso de evaluación a la batería se siguieron los pasos que a continuación se describen:

1. **Carga total de la batería:** Como primer paso, la batería fue totalmente cargada después de ser adquirida con el proveedor.

2. **Descarga de la batería:** El segundo pasó consistió en conectar el equipo CYCLE-DAQ a la batería, con lo cual, el equipo y todos sus elementos fueron energizados y puestos en marcha. El voltaje de la batería fue monitoreado en intervalos de 1 hora, de esta manera, se pudo obtener una relación de voltaje-tiempo, para determinar la razón de descarga de la batería. Es importante mencionar que, cuando la batería está totalmente cargada, el voltaje medido fue de 6.9 [V], el cual, difiere del valor teórico dado por el fabricante de la batería, 6[V]. En la tabla 4.4 se muestran los valores de voltaje y el tiempo registrado.

Voltaje medido[V]	Tiempo transcurrido [h]	% de carga de la batería
6.9	1	100
6.9	2	100
6.8	3	99
6.6	4	95
6.5	5	94
6.5	6	94
6.4	7	92
6.3	8	91
6.2	9	89
6.2	10	89
6.1	11	88
6.1	12	88
6.1	13	88
6.0	14	87
6.0	15	87
6.0	16	87
5.9	17	86
5.8	18	85
5.8	19	85
5.7	20	84

Tabla 4.4. Razón de descarga de la batería de alimentación.

La tabla anterior muestra la razón de descarga hasta 20 horas, esto es debido a que en voltajes por debajo de los 5.6 [V], el funcionamiento del equipo CYCLE-DAQ no es estable y eventualmente pueden presentarse fallas en los dispositivos periféricos. Finalmente, se puede concluir que, la batería de alimentación tiene una duración de 20 horas, tiempo suficiente para poder operar al equipo CYCLE-DAQ por varios días, teniendo en cuenta que su tiempo diario de uso es de 4 horas.

3. **Carga de la batería:** Una vez que la batería fue descarga al 84% de su valor máximo, se verificó el tiempo de carga de la batería. El objetivo de este proceso fue, informar al personal del LCE cuanto tiempo deberán dejar la batería cargando antes de volver a utilizar el equipo CYCLE-DAQ. Para cargar la batería únicamente fue necesario conectar el cargador de la batería y esperar a que llegue a su valor de voltaje máximo, la forma de verificar cuando la batería ha sido cargada, se realiza de dos formas: la primera, observar cuando el LED del cargador se encienda, y la segunda, medir con el multímetro el voltaje en los bornes de la batería hasta que éste llegue a 6.9 [V].

4.2.7. Evaluación del *software* de procesamiento de datos

El proceso de evaluación del *software* DRIVE-SOFT, tuvo como objetivo verificar que las gráficas del recorrido del vehículo y del perfil velocidad-tiempo, así como el registro de los datos en una hoja de cálculo, sean de ayuda para el análisis estadístico de los datos registrados por el equipo CYCLE-DAQ.

Para verificar la gráfica correspondiente al recorrido del vehículo, es decir, la trayectoria que éste siguió durante el muestreo, se recurrió a un recurso disponible en Internet, llamado *Google Maps*. En éste, solo es necesario indicar las coordenadas de latitud y longitud del punto de origen y las del punto de destino, con lo cual el programa grafica una trayectoria sugerida entre ambos puntos. El procedimiento de evaluación se realizó instalando al equipo CYCLE-DAQ dentro de un vehículo, el cual, recorrió 11.3 [km] en el suroeste del Valle de México. Una vez terminado el recorrido, se introdujo el archivo registrado en la memoria microSD al *software* DRIVE-SOFT para que éste procesara los datos obtenidos y así obtener la gráfica de la trayectoria. Posteriormente, con ayuda de Google Maps, se graficó la trayectoria del vehículo indicando las coordenadas del punto de partida al punto de llegada. La idea de comparar ambas gráficas consistió en observar cómo es que el programa DRIVE-SOFT brinda una buena representación gráfica del recorrido de los vehículos.

En la figura 4.18 se muestran las gráficas obtenidas en el proceso anteriormente descritas, en la cual, se puede observar claramente que el resultado es muy similar.

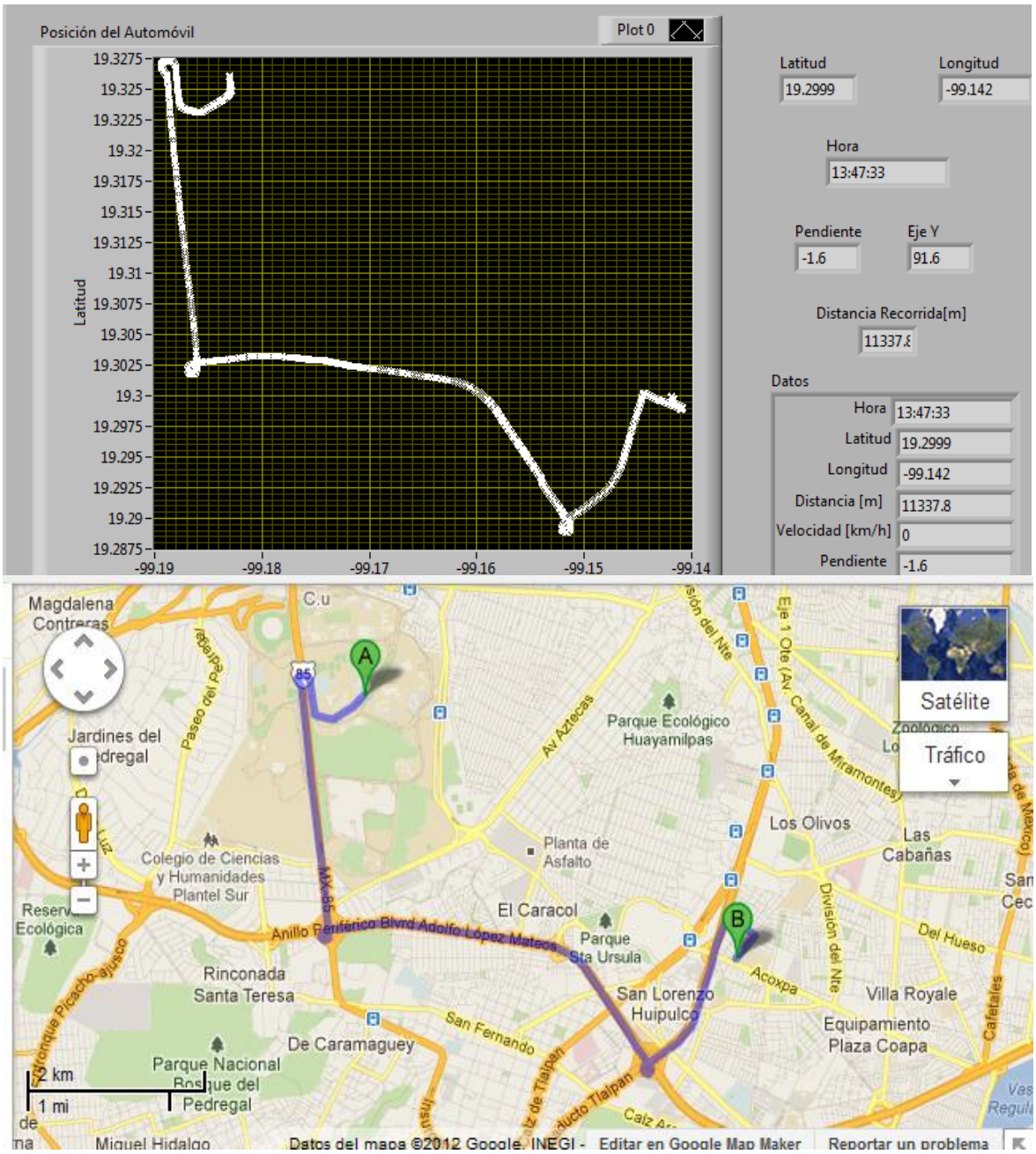


Figura 4.18. Comparación de las gráficas del recorrido de un automóvil.

Como segundo punto, se evalúo la gráfica de velocidad-tiempo, para ello, se instaló el equipo CYCLE-DAQ en un vehículo cuyo recorrido abarcó distintos tipos de calles y avenidas, lo que trajo como consecuencia que, la distancia recorrida fuera mucho mayor a la de la prueba anterior. El vehículo circulo en total 60.2 [km] partiendo del norte del Valle de México hasta llegar al sur de éste y tuvo una duración de 1 hora con 44 minutos. En la figura 4.19 se muestra la gráfica de velocidad contra tiempo obtenida como resultado del procesamiento de datos en el *software* DRIVE-SOFT.

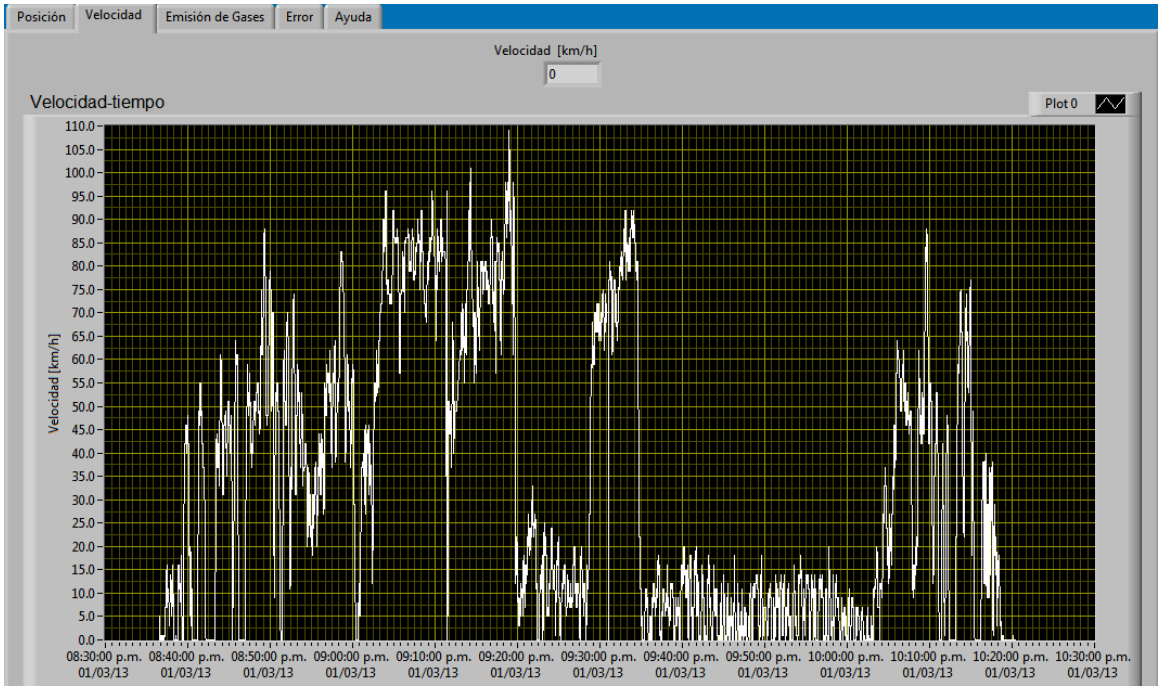


Figura 4.19. Verificación de la gráfica velocidad-tiempo.

Cabe mencionar que, así como la gráfica anterior representa al ciclo de manejo que el vehículo siguió, en todos los procesos de adquisición y registro de datos que se hagan, deberán obtenerse las gráficas velocidad-tiempo asociadas, con el objetivo de identificar gráficamente los patrones de velocidad requeridos en el desarrollo de ciclos de maneo.

Finalmente, se comprobó que los datos de la prueba anteriormente citada, fueran registrados en una hoja de cálculo de forma satisfactoria, pues éste, es el principal objetivo del *software* DRIVE-SOFT. En la figura 4.20 se muestra la captura de pantalla de dicho proceso.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	FECHA	HORA	LATITUD	LONGITUD	ALTITUD	DISTANCIA	VELOCIDAD	PENDIENTE	EJE Y	CO2	CO	HC	O2	NOX	V_BAT	
2	01/03/2013	20:36:35	19.616832	-99.297981	1.1	0	0	8.9	-89.8	0	0	0	0	0	6.5	
3	01/03/2013	20:36:36	19.61626	-99.297569	2301.3	76.885	5	8.9	-89.8	0	0	0	0	0	6.5	
4	01/03/2013	20:36:37	19.616327	-99.297569	2304.7	84.292	1	8.7	-90.1	0	0	0	0	0	6.5	
5	01/03/2013	20:36:38	19.616355	-99.297561	2309.6	87.618	0	15.5	-90.1	0	0	0	0	0	6.5	
6	01/03/2013	20:36:39	19.616355	-99.297546	2308.3	89.228	0	14.1	-90.3	0	0	0	0	0	6.5	
7	01/03/2013	20:36:40	19.616355	-99.297546	2307.9	89.228	0	15.3	-90.1	0	0	0	0	0	6.5	
8	01/03/2013	20:36:41	19.616374	-99.297584	2313.5	93.829	0	10.5	-89.6	0	0	0	0	0	6.5	
9	01/03/2013	20:36:42	19.616413	-99.297645	2322.7	101.266	0	15.5	-89.8	0	0	0	0	0	6.5	
10	01/03/2013	20:36:43	19.616451	-99.297683	2324.7	107.329	0	11.6	-90.1	0	0	0	0	0	6.5	
11	01/03/2013	20:36:44	19.616512	-99.297737	2330.8	116.017	0	8.7	-89.8	0	0	0	0	0	6.5	
12	01/03/2013	20:36:45	19.616516	-99.29776	2336.5	118.445	1	12.8	-89.6	0	0	0	0	0	6.5	
13	01/03/2013	20:36:46	19.616502	-99.29776	2337.1	119.965	0	14.1	-90.5	0	0	0	0	0	6.5	
14	01/03/2013	20:36:47	19.616493	-99.297752	2335.1	121.206	1	10.5	-89.8	0	0	0	0	0	6.5	
15	01/03/2013	20:36:48	19.616493	-99.29776	2337.1	122.005	1	13.2	-89.4	0	0	0	0	0	6.5	
16	01/03/2013	20:36:49	19.616487	-99.29776	2338	122.765	1	15.3	-90.3	0	0	0	0	0	6.5	
17	01/03/2013	20:36:50	19.616479	-99.29776	2338.8	123.715	1	10	-90.5	0	0	0	0	0	6.5	
18	01/03/2013	20:36:51	19.616479	-99.29776	2341.1	123.715	0	9.6	-88	0	0	0	0	0	6.5	
19	01/03/2013	20:36:52	19.616479	-99.29776	2342	123.715	0	9.8	-91	0	0	0	0	0	6.5	
20	01/03/2013	20:36:53	19.616479	-99.29776	2342.1	123.715	0	10.7	-90.5	0	0	0	0	0	6.5	

Figura 4.20. Datos registrados en hoja de cálculo.

4.3 Puesta a punto

Una vez que se evalúo y verificó el funcionamiento del equipo CYCLE-DAQ, éste fue colocado dentro de un gabinete plástico, con el objetivo de cumplir con las características de portabilidad y modularidad. El gabinete seleccionado cuenta con el grado de protección IP65, especificado para su uso en equipo eléctrico y electrónico. Aunque el gabinete tuvo que ser sometido a algunas modificaciones con el objetivo de fijar a él, los LEDs indicadores, el *display* gráfico, el interruptor de encendido y el conector DB-9, éste no sufrió alteraciones significativas, conservando así las características del IP65, protección contra polvo y agua. En la figura 4.21 se muestran al equipo CYCLE-DAQ instalado dentro del gabinete.

Figura 4.21. Equipo CYCLE-DAQ dentro del gabinete.

Finalmente, el equipo CYCLE-DAQ fue sometido a distintas y numerosas pruebas, colocándolo en vehículos compactos, camiones de transporte de carga y motocicletas. Para fines prácticos de éste trabajo, se presentan los resultados obtenidos durante un muestreo realizado en cada tipo de vehículo mencionado. A continuación se menciona una breve descripción de cada recorrido, incluyendo la ruta trazada por *Google Maps*, para hacer una comparación aproximada de esta última con la trayectoria registrada por el equipo. Se podrá observar que existe una variación entre ambas, debido a que el recorrido sugerido por el *software* considera el trayecto más corto y no el que se siguió durante la prueba.

4.3.1 Vehículo ligero

El muestreo realizado en el vehículo ligero, tuvo como objetivo integrar al analizador de gases con el equipo CYCLE-DAQ. Para ello, se instalaron todos los elementos necesarios dentro de un automóvil tipo sedan, como se muestra en la figura 4.22.

Figura 4.22. Instalación del equipo en un vehículo ligero.

Para tener una idea más clara sobre el recorrido que se realizó a bordo del vehículo, en la tabla 4.5 se muestra la ficha técnica que describe al muestreo.

VEHÍCULO LIGERO	
Marca	Nissan
Modelo	Tsuru 2002
Ruta	Circuito exterior, Ciudad Universitaria
Distancia recorrida	5.2 [km]
Duración	10 minutos
Analizador de gases	Sí

Tabla 4.5. Ficha técnica del muestreo realizado en el vehículo ligero.

Una vez finalizado el recorrido, los datos fueron procesados con el *software* DRIVE-SOFT obteniendo como resultado las gráficas que se muestran de la figura 4.23 a la figura 4.25.

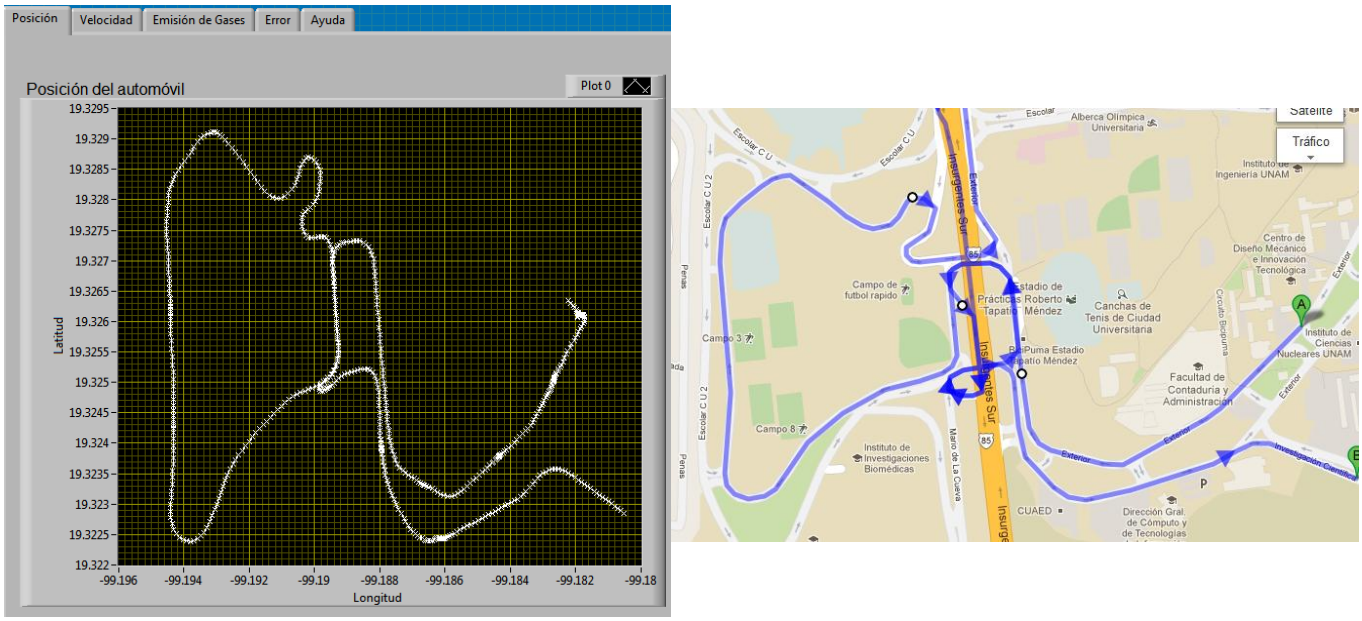


Figura 4.23. Gráfica del recorrido de un vehículo ligero.

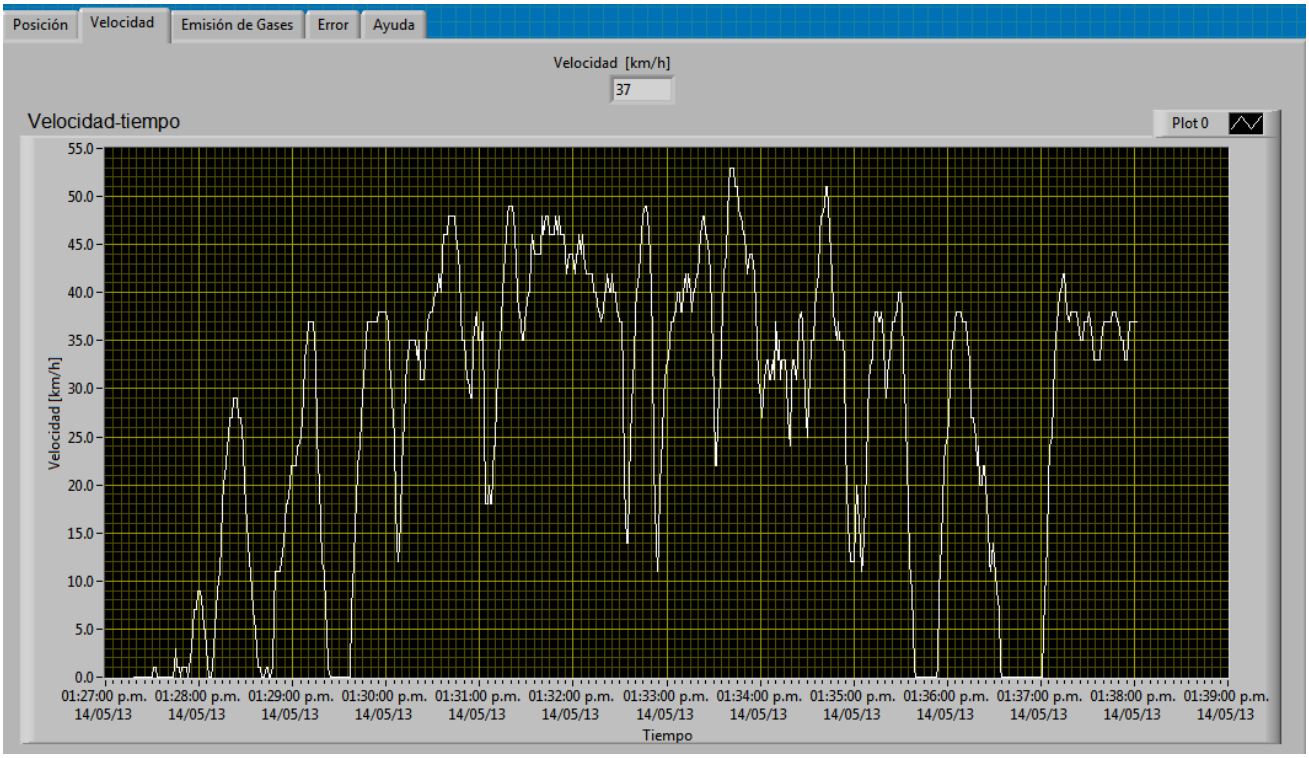


Figura 4.24. Gráfica velocidad-tiempo de un vehículo ligero.

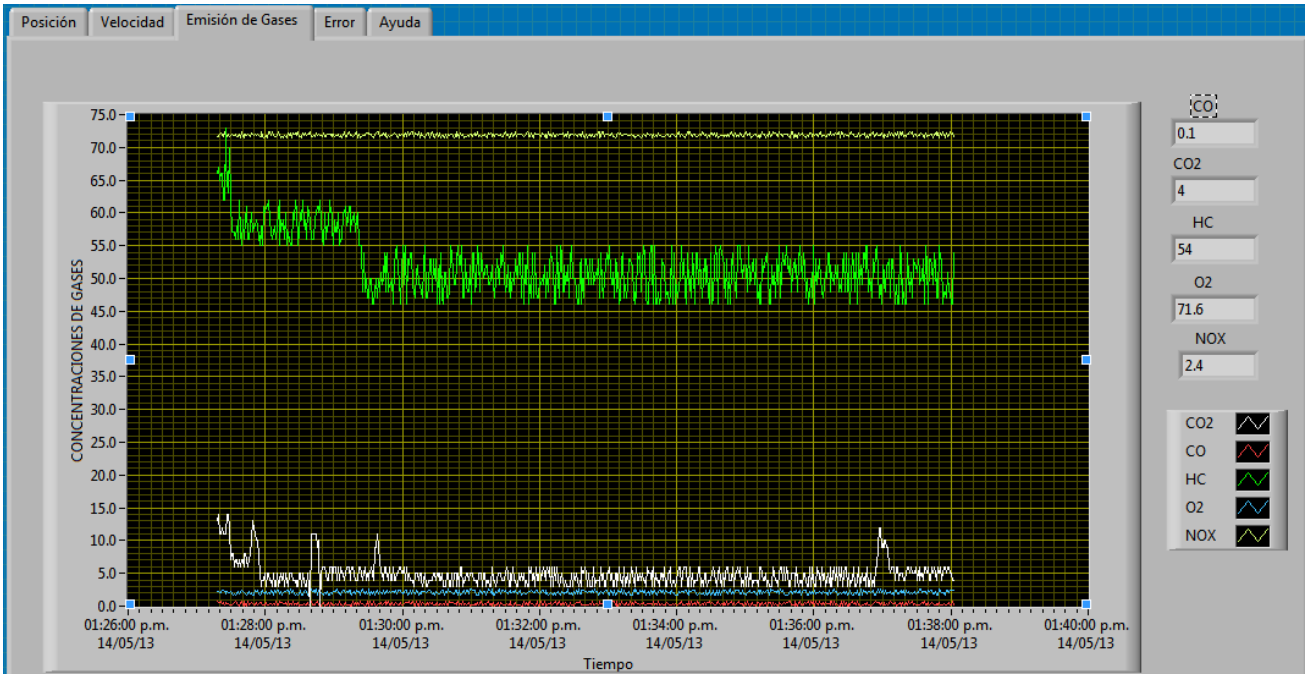


Figura 4.25. Gráfica de las emisiones contaminantes de un vehículo ligero.

4.3.2. Vehículo de transporte de carga

Este muestreo se llevó a cabo colocando únicamente al equipo CYCLE-DAQ, sin el analizador de gases, a bordo de un camión de 2 ejes. En la tabla 4.6 se mencionan los detalles del recorrido.

VEHÍCULO DE TRANSPORTE DE CARGA	
Marca	Ford
Modelo	Camión de 2 ejes 1997
Ruta	Irapuato, Guanajuato - Ciudad de México
Distancia recorrida	305 [km]
Duración	3 horas 17 minutos
Analizador de gases	No

Tabla 4.6. Ficha técnica del muestreo realizado en el vehículo de transporte de carga.

En las figuras 4.26 y 4.27 se pueden observar las gráficas, resultado del procesamiento de los datos adquiridos en la prueba del vehículo de transporte de carga.

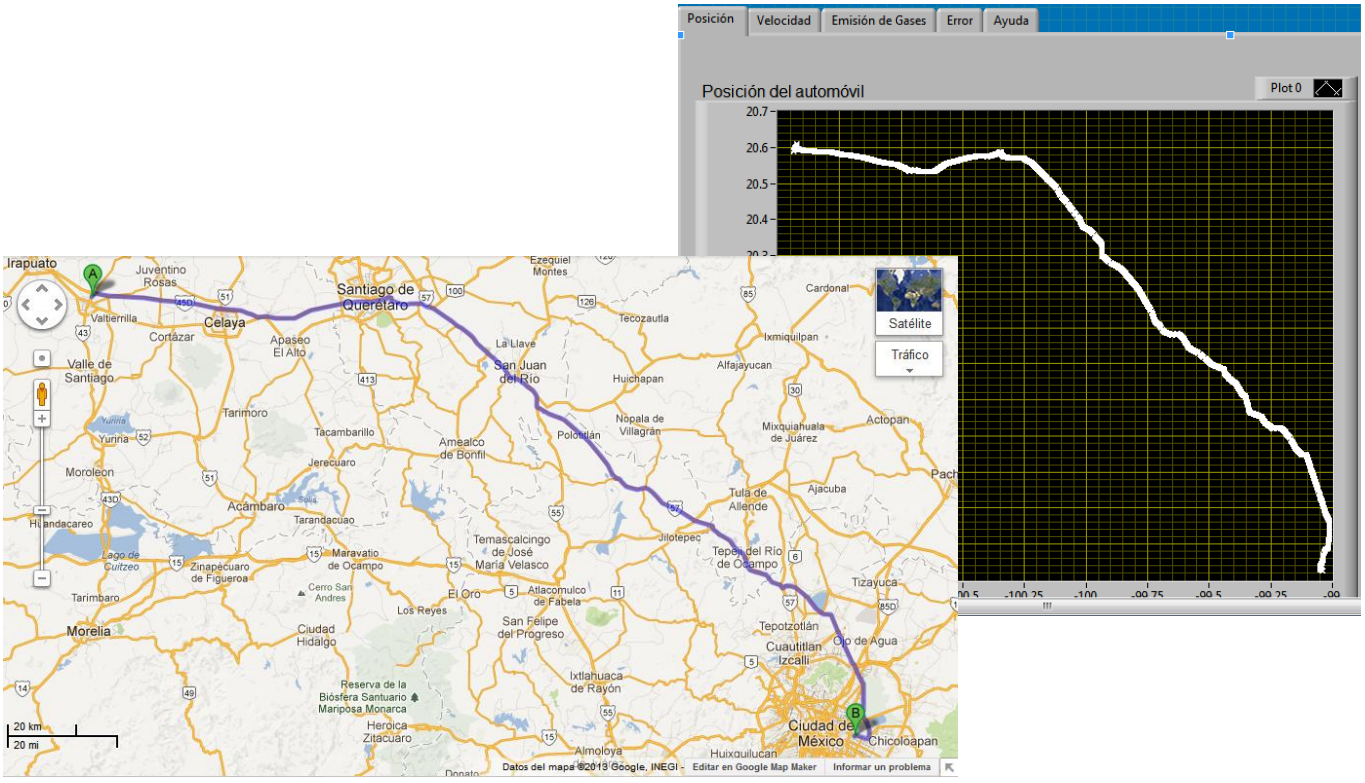


Figura 4.26. Gráfica del recorrido del vehículo de transporte de carga.

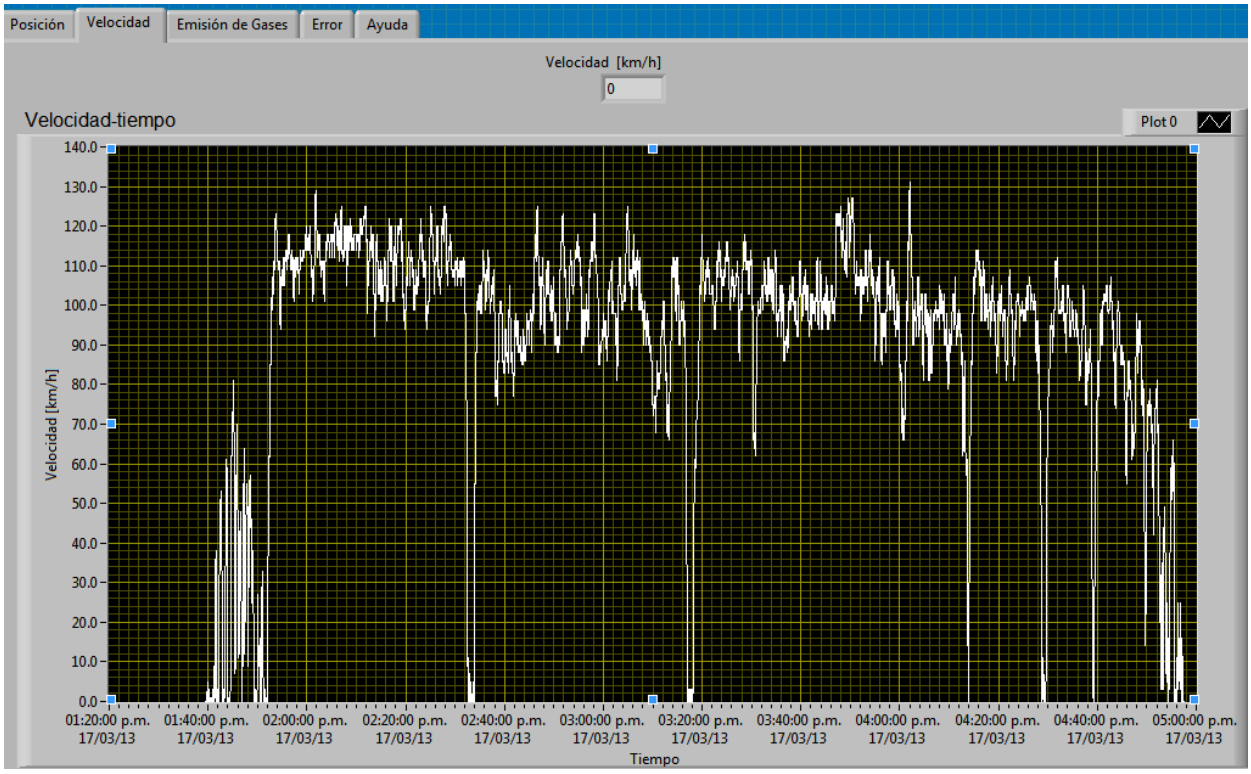


Figura 4.27. Gráfica velocidad-tiempo del vehículo de transporte de carga.

4.3.3. Motocicleta

Finalmente, el equipo se montó en una motocicleta, para seguir una ruta corta; con ésta se comprobó la robustez, portabilidad y estabilidad del equipo CYCLE-DAQ. En la tabla 4.7 se muestran los detalles técnicos del recorrido.

VEHÍCULO LIGERO	
Marca	Suzuki
Modelo	Motor de 600 c.c. Año 2003
Ruta	Santa Úrsula Coapa - salida a Cuernavaca
Distancia recorrida	7.4 [km]
Duración	10 minutos
Analizador de gases	No

Tabla 4.7. Ficha técnica del muestreo realizado en la motocicleta.

En la figura 4.28 se ilustra la ruta seguida por la motocicleta, mientras en la figura 4.29 se observa la gráfica velocidad-tiempo correspondiente.

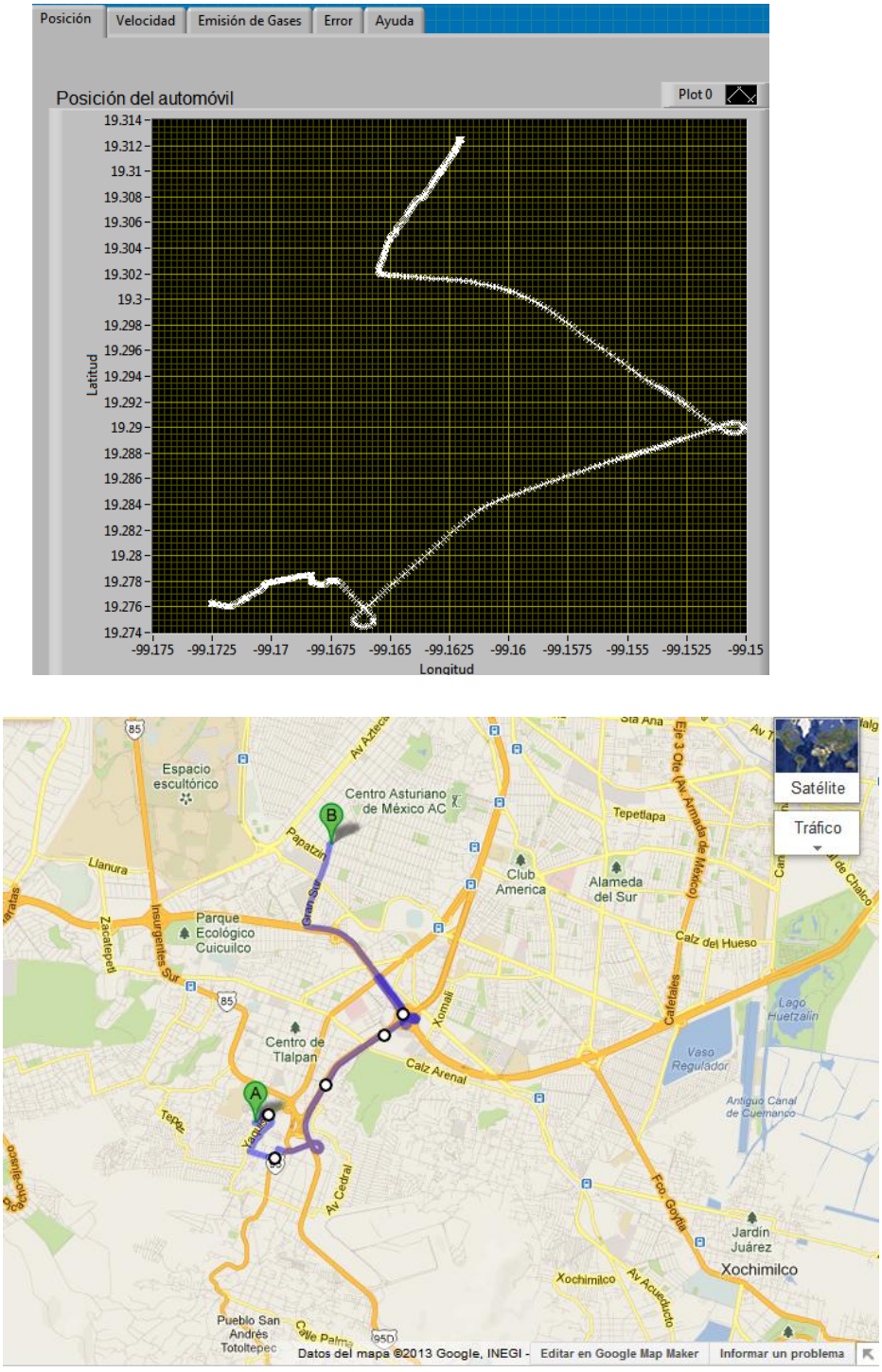


Figura 4.28. Gráfica del recorrido de la motocicleta.

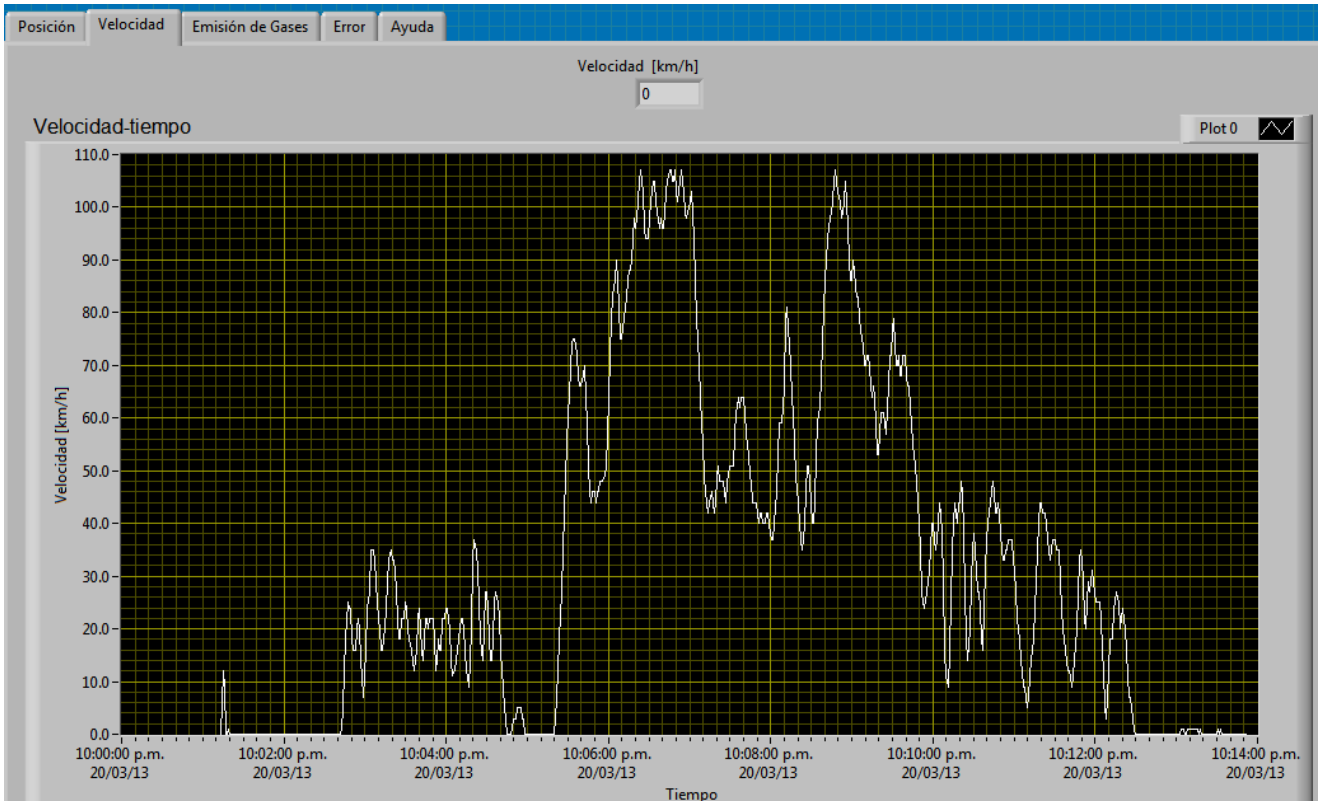


Figura 4.29. Gráfica velocidad-tiempo de la motocicleta.

Una vez que se realizaron las pruebas pertinentes para la verificación del funcionamiento del equipo CYCLE-DAQ y fue posible procesar, analizar y comparar los datos resultantes del proceso de adquisición. En el capítulo siguiente se resumen los resultados generales del proyecto y se concluye al respecto, con la oportunidad de enlistar recomendaciones a cerca del trabajo.

C A P Í T U L O 5

RESULTADOS Y CONCLUSIONES

En este capítulo se informan los resultados obtenidos al finalizar el desarrollo del equipo CYCLE-DAQ y el software DRIVE-SOFT. De igual manera, se presentan las conclusiones finales y las recomendaciones que permitirán mejorar al sistema en un futuro.

5.1. Resultados

Al finalizar el desarrollo del sistema de adquisición y registro de parámetros vehiculares para el desarrollo de ciclos de manejo en el Valle de México, el cual está integrado por: el equipo CYCLE-DAQ y el *software* de procesamiento de datos DRIVE-SOFT, se obtuvieron resultados que pueden ser divididos en tres partes: integración del *hardware*, programación de la tarjeta de desarrollo e implementación del *software* de procesamiento de datos.

En cuanto a la integración *hardware* del equipo CYCLE-DAQ, se consiguió implementar SARD basado en un microcontrolador integrado en una tarjeta de desarrollo, además, se diseño un circuito impreso a la medida de dicha tarjeta, cabe mencionar que el circuito impreso fue sujeto a diversas modificaciones hasta llegar a su versión final. El receptor GPS cumplió su objetivo principal: evitar el uso de sensores externos para determinar la velocidad del vehículo y la distancia que recorre; además permitió fechar el proceso de adquisición y registro de datos. El uso de indicadores visuales y un dispositivo de despliegue de información para el usuario, permiten conocer el estado de operación del equipo y de esta manera evitar errores en las mediciones realizadas.

Aunque al inicio del proyecto no se tenía contemplado incorporar el módulo analizador de gases contaminantes, finalmente éste ayudará a tener un proceso de análisis de datos más profundo, obteniendo de esta forma, mejores resultados para el proyecto.

Se logró que el equipo fuera totalmente autónomo, es decir, que el usuario no interactúa con él; lo único que debe hacer es encenderlo por medio de un interruptor. Al haber colocado las tarjetas electrónicas dentro de un gabinete plástico, se facilitó el proceso de operación del equipo, al mismo tiempo que le proporcionó robustez. La batería seleccionada garantiza al equipo operar continuamente

hasta por 20 horas seguidas, haciendo énfasis en que el proceso de carga de ésta es una ejecución sencilla y que puede ser efectuada por el usuario. Por último, es importante a resaltar el almacenamiento de las 12 variables adquiridas, en distintos archivos dentro de una memoria microSD. Estos registros son de tamaño reducido, por lo que no requieren de gran espacio de almacenamiento y son compatibles con los formatos FAT16 y FAT32.

Referente a la programación de la tarjeta de desarrollo, se efectúo integrando paulatinamente las tareas correspondientes a cada dispositivo, de esta manera, el programa pudo ser evaluado módulo por módulo, evitando así, fallas en la implementación. Dentro de esta parte, es necesario mencionar que se implementaron:

- Tres protocolos de comunicación diferentes
- Una conversión analógica digital y
- Manejo de las terminales de E/S para los LEDs y el *display* gráfico.

La implementación del *software* DRIVE-SOFT, para el procesamiento de los datos registrados en la memoria microSD, resultó ser una herramienta sumamente útil, con la cual es posible tener una idea gráfica y concreta del comportamiento del vehículo durante su trayectoria, situación que no hubiera sido fácil lograr únicamente con el equipo CYCLE-DAQ. Además, el *software* permite registrar los datos en una hoja de cálculo, facilitando así su manipulación y análisis estadístico e incluso se pueden generar bases de datos para tener un registro histórico y ordenado de los mismos.

Cabe mencionar que, debido a los buenos resultados obtenidos por el equipo CYCLE-DAQ, se tuvo la necesidad de reproducirlo en seis unidades. Con el objetivo de generar la mayor cantidad de datos, el grupo de trabajo del LCE, conformó tres brigadas: una para vehículos ligeros, la segunda para vehículos pesados y la tercera para motocicletas. De esta manera, cada brigada cuenta con dos ejemplares del equipo y se encuentran trabajando con ellos. A la fecha se cuenta con un registro de aproximadamente 120 muestreos realizados por las diferentes brigadas.

Adicionalmente, se logró la participación en el XXVII Congreso de Instrumentación, organizado por la Sociedad Mexicana de Instrumentación (SOMI), realizado en Culiacán, Sinaloa, durante el año 2012, publicándose así un artículo de investigación en las memorias de dicho congreso.

5.2. Conclusiones

El objetivo de este trabajo fue el diseño, desarrollo, evaluación, integración y puesta a punto de un sistema de adquisición y registro de parámetros vehiculares. Dichos parámetros son necesarios para

actualizar y desarrollar ciclos de manejo, principal propósito del LCE, durante el año 2012 y el presente.

Al ser el equipo CYCLE-DAQ un sistema autónomo e independiente del funcionamiento mecánico del automóvil, la problemática de ya no contar con los instrumentos necesarios para desarrollar los ciclos en el laboratorio, fue resuelta. Adicionalmente, un aspecto importante a considerar, es que, la integración del módulo analizador de gases es un proceso más complicado e invasivo al automóvil, lo cual implica que parte de la portabilidad del equipo sea perdida.

Con base a lo mencionado anteriormente, se concluye que el equipo CYCLE-DAQ cumple con el objetivo inicial y con los requerimientos establecidos al inicio del proyecto por parte del personal del LCE.

Personalmente, puedo concluir que con la realización de este trabajo, pude poner en práctica los conocimientos adquiridos en la Universidad durante mis estudios como ingeniero. Además, mi objetivo personal se cumplió, esto fue, aplicar mis conocimientos y habilidades en beneficio de la sociedad mexicana. Finalmente, puedo concluir que mi formación humana se enriqueció al asumir nuevos retos, vivir nuevas experiencias y convivir con profesionales de la ingeniería.

5.3. Recomendaciones

El desarrollo del equipo CYCLE-DAQ y el *software* de procesamiento de datos, fue un proceso que involucró distintas pruebas, sugerencias y observaciones. Con base en ello, se llegó al prototipo final, presentado en este trabajo. Sin embargo, se pueden resaltar algunas recomendaciones con las cuales, en un futuro, se podrán realizar mejoras al sistema y así optimizar cada vez más su desempeño. A continuación se listan las recomendaciones propuestas:

1. Integrar todo el sistema en una sola tarjeta electrónica, lo cual requiere un rediseño total del circuito impreso y no utilizar una tarjeta de desarrollo.
2. Verificar otros medios para la evaluación de emisiones contaminantes debido a la poca practicidad del analizador de gases ANDROS 6600 para ser instalado a bordo de vehículos.
3. En cuanto la programación del microcontrolador, se recomienda integrar a las funciones programadas para el receptor GPS en una sola librería.
4. Realizar un monitoreo constante de la batería de alimentación del equipo y mostrar el porcentaje de carga de la misma, con el objetivo de que, el usuario sepa exactamente cuándo recargar la batería.

5. Mejorar el modulo de alimentación, esto es, realizar el diseño de una fuente que conmute entre la batería recargable y el encendedor del automóvil, cuando uno u otro falle o no se encuentre presente en el equipo.

A P É N D I C E S

APÉNDICE A. Listado de funciones programadas más relevantes

Función *calc_dist*: Cálcula la distancia entre dos puntos con la fórmula de Haversine.

```
// Función que calcula la distancia entre dos puntos con la fórmula de Haversine
// Recibe 4 parámetros: latitud1, longitud1, latitud2 y longitud2
// Regresa la distancia calculada en metros
float calc_dist (float lat1, float long1, float lat2, float long2) {
  float delta = radians(long1-long2);
  float sdlong = sin(delta);
  float cdlong = cos(delta);
  lat1 = radians(lat1);
  lat2 = radians(lat2);
  float slat1 = sin(lat1);
  float clat1 = cos(lat1);
  float slat2 = sin(lat2);
  float clat2 = cos(lat2);
  delta = (clat1 * slat2) - (slat1 * clat2 * cdlong);
  delta = sq(delta);
  delta += sq(clat2 * sdlong);
  delta = sqrt(delta);
  float denom = (slat1 * slat2) + (clat1 * clat2 * cdlong);
  delta = atan2(delta, denom);
  return delta * 6372795 * 1.0;
}
```

Función *getBit*: Obtiene el estado de un determinado bit de un byte.

```
// Función que determina el estado de un bit de un determinado byte
// a través de un corrimiento lógico de bits

boolean getBit(byte myByte, byte whatBit) {
  boolean bitState;
  bitState = myByte & (1 << whatBit);
  return bitState;
}
```

Función *readline_gps*: Obtiene las sentencias NMEA provenientes del receptor GPS.

```
//Función que obtiene las sentencias NMEA del receptor GPS
//EM-406A através del puerto serial de Arduino
//el resultado es almacenado en un arreglo llamado buffer[]
void readline_gps(void) {
  char c;

  buffidx = 0;
  while (1) {
        c=Serial3.read();
        if (c == -1)
          continue;
        //Serial.print(c);
        if (c == '\n')
          continue;
        if ((buffidx == BUFFSIZ-1) || (c == '\r')) {
          buffer[buffidx] = 0;
          return;
        }
        buffer[buffidx++]= c;


   }

}
```

Función *parsedecimal*: Convierte una cadena de caracteres a entero de 32 bits.

```
//Función que convierte una cadena de caracteres a entero
//de 32 bits. Recibe como parámtero un apuntador a caracter
//y devuelve un entero de 32 bits

uint32_t parsedecimal(char *str) {
  uint32_t d = 0;

  while (str[0] != 0) {
   if ((str[0] > '9') || (str[0] < '0'))
     return d;
   d *= 10;
   d += str[0] - '0';
   str++;
  }
  return d;
}
```

APÉNDICE B. Hojas de especificaciones

Features

- High Performance, Low Power Atmel® AVR® 8-Bit Microcontroller
- Advanced RISC Architecture
  - 135 Powerful Instructions – Most Single Clock Cycle Execution
  - 32 x 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 16 MIPS Throughput at 16MHz
  - On-Chip 2-cycle Multiplier
- High Endurance Non-volatile Memory Segments
  - 64K/128K/256KBytes of In-System Self-Programmable Flash
  - 4Kbytes EEPROM
  - 8Kbytes Internal SRAM
  - Write/Erase Cycles:10,000 Flash/100,000 EEPROM
  - Data retention: 20 years at 85°C/ 100 years at 25°C
  - Optional Boot Code Section with Independent Lock Bits
    - In-System Programming by On-chip Boot Program
    - True Read-While-Write Operation
  - Programming Lock for Software Security
    - Endurance: Up to 64Kbytes Optional External Memory Space
- Atmel® QTouch® library support
  - Capacitive touch buttons, sliders and wheels
  - QTouch and QMatrix® acquisition
  - Up to 64 sense channels
- JTAG (IEEE std. 1149.1 compliant) Interface
  - Boundary-scan Capabilities According to the JTAG Standard
  - Extensive On-chip Debug Support
  - Programming of Flash, EEPROM, Fuses, and Lock Bits through the JTAG Interface
- Peripheral Features
  - Two 8-bit Timer/Counters with Separate Prescaler and Compare Mode
  - Four 16-bit Timer/Counter with Separate Prescaler, Compare- and Capture Mode
  - Real Time Counter with Separate Oscillator
  - Four 8-bit PWM Channels
  - Six/Twelve PWM Channels with Programmable Resolution from 2 to 16 Bits (ATmega1281/2561, ATmega640/1280/2560)
  - Output Compare Modulator
  - 8/16-channel, 10-bit ADC (ATmega1281/2561, ATmega640/1280/2560)
  - Two/Four Programmable Serial USART (ATmega1281/2561, ATmega640/1280/2560)
  - Master/Slave SPI Serial Interface
  - Byte Oriented 2-wire Serial Interface
  - Programmable Watchdog Timer with Separate On-chip Oscillator
  - On-chip Analog Comparator
  - Interrupt and Wake-up on Pin Change
- Special Microcontroller Features
  - Power-on Reset and Programmable Brown-out Detection
  - Internal Calibrated Oscillator
  - External and Internal Interrupt Sources
  - Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby, and Extended Standby
- I/O and Packages
  - 54/86 Programmable I/O Lines (ATmega1281/2561, ATmega640/1280/2560)
  - 64-pad QFN/MLF, 64-lead TQFP (ATmega1281/2561)
  - 100-lead TQFP, 100-ball CBGA (ATmega640/1280/2560)
  - RoHS/Fully Green
- Temperature Range:
  - -40°C to 85°C Industrial
- Ultra-Low Power Consumption
  - Active Mode: 1MHz, 1.8V: 500µA
  - Power-down Mode: 0.1µA at 1.8V
- Speed Grade:
  - ATmega640V/ATmega1280V/ATmega1281V:
    - 0 - 4MHz @ 1.8V - 5.5V, 0 - 8MHz @ 2.7V - 5.5V
  - ATmega2560V/ATmega2561V:
    - 0 - 2MHz @ 1.8V - 5.5V, 0 - 8MHz @ 2.7V - 5.5V
  - ATmega640/ATmega1280/ATmega1281:
    - 0 - 8MHz @ 2.7V - 5.5V, 0 - 16MHz @ 4.5V - 5.5V
  - ATmega2560/ATmega2561:
    - 0 - 16MHz @ 4.5V - 5.5V

8-bit Atmel Microcontroller with 64K/128K/256K Bytes In-System Programmable Flash

ATmega640/V
ATmega1280/V
ATmega1281/V
ATmega2560/V
ATmega2561/V

2549P–AVR–10/2012

ANALOG DEVICES

3-Axis, ±2 *g*/±4 *g*/±8 *g*/±16 *g* Digital Accelerometer

ADXL345

FEATURES

- Ultralow power: as low as 23 µA in measurement mode and 0.1 µA in standby mode at V_S = 2.5 V (typical)
- Power consumption scales automatically with bandwidth
- User-selectable resolution
  - Fixed 10-bit resolution
  - Full resolution, where resolution increases with *g* range, up to 13-bit resolution at ±16 *g* (maintaining 4 mg/LSB scale factor in all *g* ranges)
- Patent pending, embedded memory management system with FIFO technology minimizes host processor load
- Single tap/double tap detection
- Activity/inactivity monitoring
- Free-fall detection
- Supply voltage range: 2.0 V to 3.6 V
- I/O voltage range: 1.7 V to V_S
- SPI (3- and 4-wire) and I²C digital interfaces
- Flexible interrupt modes mappable to either interrupt pin
- Measurement ranges selectable via serial command
- Bandwidth selectable via serial command
- Wide temperature range (−40°C to +85°C)
- 10,000 *g* shock survival
- Pb free/RoHS compliant
- Small and thin: 3 mm × 5 mm × 1 mm LGA package

APPLICATIONS

- Handsets
- Medical instrumentation
- Gaming and pointing devices
- Industrial instrumentation
- Personal navigation devices
- Hard disk drive (HDD) protection

GENERAL DESCRIPTION

The ADXL345 is a small, thin, ultralow power, 3-axis accelerometer with high resolution (13-bit) measurement at up to ±16 *g*. Digital output data is formatted as 16-bit twos complement and is accessible through either a SPI (3- or 4-wire) or I²C digital interface.

The ADXL345 is well suited for mobile device applications. It measures the static acceleration of gravity in tilt-sensing applications, as well as dynamic acceleration resulting from motion or shock. Its high resolution (3.9 mg/LSB) enables measurement of inclination changes less than 1.0°.

Several special sensing functions are provided. Activity and inactivity sensing detect the presence or lack of motion by comparing the acceleration on any axis with user-set thresholds. Tap sensing detects single and double taps in any direction. Free-fall sensing detects if the device is falling. These functions can be mapped individually to either of two interrupt output pins. An integrated, patent pending memory management system with a 32-level first in, first out (FIFO) buffer can be used to store data to minimize host processor activity and lower overall system power consumption.

Low power modes enable intelligent motion-based power management with threshold sensing and active acceleration measurement at extremely low power dissipation.

The ADXL345 is supplied in a small, thin, 3 mm × 5 mm × 1 mm, 14-lead, plastic package.

FUNCTIONAL BLOCK DIAGRAM

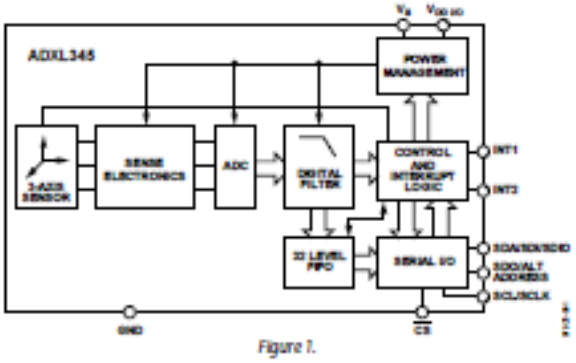
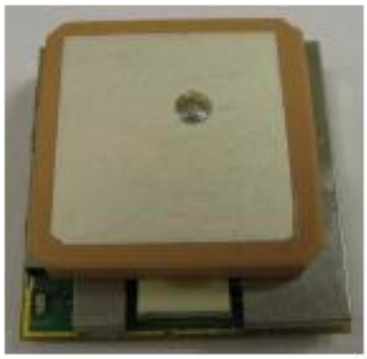


Figure 1.

Rev. C

Information furnished by Analog Devices is believed to be accurate and reliable. However, no responsibility is assumed by Analog Devices for its use, nor for any infringements of patents or other rights of third parties that may result from its use. Specifications subject to change without notice. No license is granted by implication or otherwise under any patent or patent rights of Analog Devices. Trademarks and registered trademarks are the property of their respective owners. See the last page for disclaimers.

One Technology Way, P.O. Box 9106, Norwood, MA 02062-9106, U.S.A.
Tel: 781.329.4700 www.analog.com
Fax: 781.461.3113 ©2009–2011 Analog Devices, Inc. All rights reserved.



EM-406a GPS BOARD OVERVIEW

The EM-406a GPS engine board is low cost but maintains high reliability and accuracy making it an ideal choice for integration with OEM/ODM systems. The EM-406a features an integrated patch antenna for complete implementation.

FEATURES:

1. SIRF Star III high performance GPS chipset
2. Very high sensitivity (Tracking Sensitivity: -159dBm)
3. Extremely fast TTFF (Time To First Fix) at low signal levels
4. Supports the NMEA 0183 data protocol
5. Built-in SuperCap to maintain system data for rapid satellite acquisition
6. Built-in patch antenna
7. Foliage Lock for weak signal tracking
8. Compact in size
9. All-in-view 20-channel parallel processing
10. Snap Lock 100ms re-acquisition time
11. Superior urban canyon performance
12. WAAS / EGNOS /MSAS support
13. RoHS compliant

Differences between the EM-406 and EM-406a :

a.) RoHS lead-free

b.) 1 PPS added to pin #6

SiRF star III GPS Module
SJ-301
Specification

2007.2

勁永資通股份有限公司
TwinMOS Technologies Inc.

Micro SD Card™

Product Specification

Version 1.0

Information in this document is provided in connection with TwinMOS products. No license, express or implied, by estoppels or otherwise, to any intellectual property rights is granted by this document. Except as provided in TwinMOS's Terms and Conditions of Sale for such products, TwinMOS assumes no liability whatsoever, and TwinMOS disclaims any express or implied warranty, relating to sale and/or use of TwinMOS products including liability or warranties relating to fitness for a particular purpose, merchantability, or infringement of any patent, copyright or other intellectual property right. TwinMOS may make changes to specifications and product descriptions at any time, without notice.

Copyright © TwinMOS Technology Inc. 2001

本文件為勁永資通(股)有限公司專有之財產，非經書面許可，不准透露或使用本文件，亦不准複製或轉變成任何其他形式使用。
The information contained herein is the exclusive property of TwinMOS Technologies Inc. and shall not be distributed, reproduced, or disclosed in whole or in part without prior written permission of TwinMOS Technologies Inc. http://www.twinmos.com.tw

19-4323; Rev 11; 2/03

MAXIM

+5V-Powered, Multichannel RS-232 Drivers/Receivers

General Description

The MAX220–MAX249 family of line drivers/receivers is intended for all EIA/TIA-232E and V.28/V.24 communications interfaces, particularly applications where ±12V is not available.

These parts are especially useful in battery-powered systems, since their low-power shutdown mode reduces power dissipation to less than 5µW. The MAX225, MAX233, MAX235, and MAX245/MAX246/MAX247 use no external components and are recommended for applications where printed circuit board space is critical.

Applications

Portable Computers

Low-Power Modems

Interface Translation

Battery-Powered RS-232 Systems

Multidrop RS-232 Networks

Features

Superior to Bipolar

- Operate from Single +5V Power Supply (+5V and +12V—MAX231/MAX239)
- Low-Power Receive Mode in Shutdown (MAX223/MAX242)
- Meet All EIA/TIA-232E and V.28 Specifications
- Multiple Drivers and Receivers
- 3-State Driver and Receiver Outputs
- Open-Line Detection (MAX243)

Ordering Information

PART	TEMP RANGE	PIN-PACKAGE
MAX220CPE	0°C to +70°C	16 Plastic DIP
MAX220CSE	0°C to +70°C	16 Narrow SO
MAX220CWE	0°C to +70°C	16 Wide SO
MAX220C/D	0°C to +70°C	Dice*
MAX220EPE	-40°C to +85°C	16 Plastic DIP
MAX220ESE	-40°C to +85°C	16 Narrow SO
MAX220EWE	-40°C to +85°C	16 Wide SO
MAX220EJE	-40°C to +85°C	16 CERDIP
MAX220MJE	-55°C to +125°C	16 CERDIP

Ordering information continued at end of data sheet.

*Contact factory for dice specifications.

Selection Table

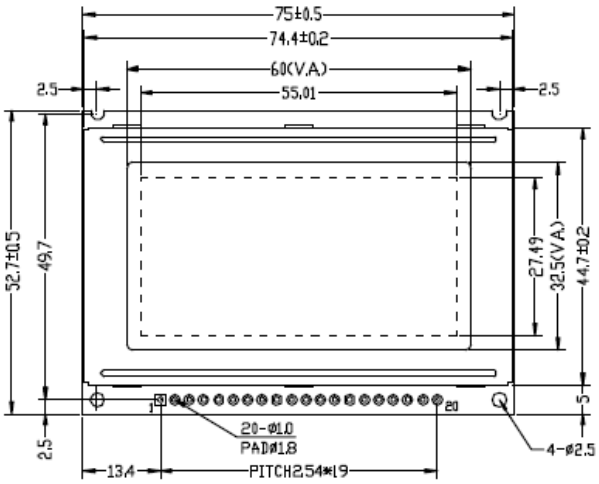
Part Number	Power Supply (V)	No. of RS-232 Drivers/Rx	No. of Ext. Caps	Nominal Cap. Value (µF)	SHDN & Three-State	Rx Active in SHDN	Data Rate (kbps)	Features
MAX220	+5	2/2	4	0.1	No	—	120	Ultra-low-power, industry-standard pinout
MAX222	+5	2/2	4	0.1	Yes	—	200	Low-power shutdown
MAX223 (MAX213)	+5	4/5	4	1.0 (0.1)	Yes	✔	120	MAX241 and receivers active in shutdown
MAX225	+5	5/5	0	—	Yes	✔	120	Available in SO
MAX230 (MAX200)	+5	5/0	4	1.0 (0.1)	Yes	—	120	5 drivers with shutdown
MAX231 (MAX201)	+5 and +7.5 to +13.2	2/2	2	1.0 (0.1)	No	—	120	Standard +5/+ 12V or battery supplies; same functions as MAX232
MAX232 (MAX202)	+5	2/2	4	1.0 (0.1)	No	—	120 (64)	Industry standard
MAX232A	+5	2/2	4	0.1	No	—	200	Higher slew rate, small caps
MAX233 (MAX203)	+5	2/2	0	—	No	—	120	No external caps
MAX233A	+5	2/2	0	—	No	—	200	No external caps, high slew rate
MAX234 (MAX204)	+5	4/0	4	1.0 (0.1)	No	—	120	Replaces 1488
MAX235 (MAX205)	+5	5/5	0	—	Yes	—	120	No external caps
MAX236 (MAX206)	+5	4/3	4	1.0 (0.1)	Yes	—	120	Shutdown, three state
MAX237 (MAX207)	+5	5/3	4	1.0 (0.1)	No	—	120	Complements IBM PC serial port
MAX238 (MAX208)	+5	4/4	4	1.0 (0.1)	No	—	120	Replaces 1488 and 1489
MAX239 (MAX209)	+5 and +7.5 to +13.2	3/5	2	1.0 (0.1)	No	—	120	Standard +5/+ 12V or battery supplies; single-package solution for IBM PC serial port
MAX240	+5	5/5	4	1.0	Yes	—	120	DIP or flatpack package
MAX241 (MAX211)	+5	4/5	4	1.0 (0.1)	Yes	—	120	Complete IBM PC serial port
MAX242	+5	2/2	4	0.1	Yes	✔	200	Separate shutdown and enable
MAX243	+5	2/2	4	0.1	No	—	200	Open-line detection simplifies cabling
MAX244	+5	8/10	4	1.0	No	—	120	High slew rate
MAX245	+5	8/10	0	—	Yes	✔	120	High slew rate, int. caps, two shutdown modes
MAX246	+5	8/10	0	—	Yes	✔	120	High slew rate, int. caps, three shutdown modes
MAX247	+5	8/9	0	—	Yes	✔	120	High slew rate, int. caps, nine operating modes
MAX248	+5	8/8	4	1.0	Yes	✔	120	High slew rate, selective half-chip enables
MAX249	+5	6/10	4	1.0	Yes	✔	120	Available in quad flatpack package

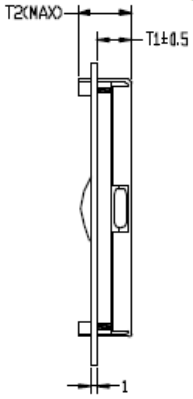
For pricing, delivery, and ordering information, please contact Maxim/Dallas Direct! at 1-888-629-4642, or visit Maxim's website at www.maxim-ic.com.

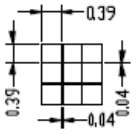
➢ Mechanical diagram

GDM12864HLCM

(Liquid Crystal Display Module)







VERSION	T1	T2	UNIT
EL&NO BKL	6.2	10.0	mm
SIDE BKL	6.2	10.0	mm
LED BKL	8.8	12.5	mm

➢ Absolute maximum ratings

Item	Symbol	Min.	Max.	Unit
Supply voltage for logic	Vdd - Vss	0	6.5	V
Input voltage	Vin	0	Vdd	
Operating temperature range	T0p	-20	70	℃
Storage temperature range	Tst	-25	75	

➢ Interface pin connections

Pin No.	Symbol	Level	Description
1	Vdd	5.0V	Supply voltage for logic and LCD (+)
2	Vss	0V	Ground
3	V0	-	Operating voltage for LCD (variable)
4~11	DB0~DB7	H/L	Data bit 0~7
12	CS2	L	Chip select signal for IC2
13	CS1	L	Chip select signal for IC1
14	/RES	L	Reset signal
15	R/W	H/L	H: read (MUP<－module),L: write (MPU－>module)
16	D/I	H/L	H: data, L: instruction code
17	E	H, H L	Chip enable signal
18	VEE	-	Operating voltage for LCD (variable)
19	A	4.2V	Backlight power supply
20	K	0V	Backlight power supply

B I B L I O G R A F Í A

Libros

[1] Banzi M., *Getting Started with Arduino*, First Edition, O´Reilly, 2008.

[2] Boylestad R., Nashelsky L., *Electrónica: Teoría de circuitos y dispositivos electrónicos*, Octava edición, Pearson Prentice Hall, México, 2003.

[3] Deitel H., Deitel P., *Como programar en C/C++ y Java*, Cuarta Edición, Pearson Prentice Hall, México, 2004.

[4] El-Rabbany Ahmed, *Engineer's Guide to GPS: The Global Positioning System*, First Edition, Artech House Inc, U.S.A., 2002.

[4] Kernighan W., Ritchie M., *El lenguaje de programación C*, Segunda edición, Pearson Prentice Hall, México, 1991.

[5] Manuel A., *Instrumentación Virtual: Adquisición, procesado y análisis de señales*, Primera edición, Grupo Alfa Omega, España, 2002.

[6] Margolis M., *Arduino Cookbook*, First Edition, O'Reilly, U.S.A, 2011

[7] Riu P., Rosell J., Ramos J., *Sistemas de Instrumentación*, Primera Edición, Ediciones de la Universidad Nacional de Cataluny, España, 1995.

[8] Xu Guochang, GPS Theory, *Algorithms and Applications*, First Edition, Springer, Germany, 2003.

Artículos y Tesis

[1] André M., *The ARTEMIS European Driving Cycles for Measuring Car Pollutant Emissions, Science of the Total Enviroment*, pp. 74-84, U.SA., 2004.

[2] Armenta C., *Integración de un sistema autónomo para el análisis y monitoreo de gases contaminantes automotrices, Tesis de Licenciatura,* UNAM, 2009.

[3] Jie Lin, Debbie A. Niemeier, *Estimating Regional Air Quality Vehicle Emission Inventories: Constructing Robust Driving Cycles*, Center for the Environment, Harvard University, 2003.

[4] Riemersama I., Hendriksen P., Gense N., Smokers, R., *Methodology for the Development of Representative Driving Cycles*, Technical University of Graz, 1997.

[5] Szauter F., Székely J. Á., Horváth E., *Development of Vehicle-Dynamic Measurement and Information System*, Hungarian Journal of Industrial Chemistry Veszprém, Vol 39(2) pp. 305 - 308, 2011.

[6] Santiago Cruz L., Rincón Gómez P., *Instrumento virtual para el análisis de gases de combustión, Laboratorio de Control de Emisiones*, Facultad de Ingeniería UNAM, 2002.

[7] Zi Kun, Huang Yung Qing, Tu Xian Ku, Yang Ren Fa, *GPS and its Application in the investigation of a Vehicle Driving Cycle in Ningbo City*. The 8th. International Conference of Chinese Logistics and Transportation Professionals, pp. 1819 -1825, China, 2008.

Hojas de datos y manuales

[1] *+5V-Powered, Multichannel RS-232 Drivers/Receivers datasheet.*

[2] *8-bit AVR Atmel Microcontroller with 256K Bytes In-System Programmable Flash, ATmega2650 datasheet,* 2012.

[3] *ADX3L45 Triple Axis Digital Accelerometer datasheet, Analog Devices Inc.,* 2010 – 2012.

[4] *AN- 1057 Application Note, "Using an Accelerometer for Inclination Sensing", Analog Devices Inc., 2010.*

[5] Display Gráfico de 128 x 64 puntos GDM12864H, hoja de especificaciones.

[6] *GPS Reciever Engine Board EM - 406A User Manual, GlobalSat Technology Corporation.*

[7] Manual Oficial de Operación del Analizador de Gases, Laboratorio de Control de Emisiones, Facultad de Ingeniería UNAM, 2008.

[8] *MicroSD Card™ Product Specification, Version 1.0., TwinMOS Techonlogy Inc.*, 2001.

[9] *Model 6600 Miniature Automotive Gas Analyzer , Product Manual, ANDROS Inc.*

[10] *SD Specifications Part 1: Physical Layer Simplified Specification Version 2.00, SD Group*, 2006.

Mesografía

[1] http:// www.adafruit.com/datasheets/arduino_hole_dimensions.pdf

[2] http://www.bot-thoughts.com/2010/02/logging-data-to-sd-cards.html

[3] http://www.buenastareas.com/ensayos/Manejo-Dinamico-Con-Archivos-Csv/2950459.html

[4] http://www.embedded-lab.com/blog/?p=3096

[5] https://www.fhwa.dot.gov/environment/air_quality/conformity/research/modeling_procedures

[6] http://www.nmea.org

.