

UNIVERSIDAD TECNOLÓGICA ISRAEL
CARRERA DE ELECTRÓNICA

Diseño e Implementación de un Sistema de Control de Ruta
Vehicular Mediante Comparación con el uso de GPS

Estudiante

Diana Natividad Pazmiño Pineda

Tutor

Ing. Fabrizio Villasís

Quito-Ecuador

Septiembre 2012

UNIVERSIDAD TECNOLÓGICA ISRAEL

CARRERA DE ELECTRÓNICA

CERTIFICADO DE RESPONSABILIDAD

Yo Ing. Fabrizio Villasís, certifico que la señorita Diana Natividad Pazmiño Pineda con C.C No 0602905820 realizo la presente tesis con titulo **“Diseño e implementación de un sistema de control de ruta vehicular mediante comparación con el uso de GPS”**, y que es autor intelectual del mismo, que es original, autentico y personal.

Ing. Fabrizio Villasís

UNIVERSIDAD TECNOLÓGICA ISRAEL

CARRERA DE ELECTRÓNICA

CERTIFICADO DE AUTORÍA

El documento de tesis con título **“Diseño e implementación de un sistema de control de ruta vehicular mediante comparación con el uso de GPS”** ha sido desarrollada por Diana Natividad Pazmiño Pineda con C.C No 0602905820 persona que posee los derechos de autoría y responsabilidad, restringiéndose la copia o utilización de cada uno de los productos de esta tesis sin previa autorización.

Diana Natividad Pazmiño Pineda

DEDICATORIA

A mis queridos padres los cuales me supieron dar su cariño y comprensión, los que estuvieron a mi lado en logros, derrotas, en fin en todas las etapas de mi vida, gracias por su amor incondicional.

A pesar de no contar físicamente con ellos les dedico donde quiera que estén a mi querido Papa Lucho, Mi abuelita María y mi ñaño Mateo, que siempre estuvieron dándome fortaleza y de cierta manera me motivaban a terminar estas metas, siempre estarán en mi corazón.

A todos mis familiares y amigos gracias por su paciencia, fortaleza, apoyo, alones de orejas, consejos y su dedicación de todo corazón este triunfo es nuestro.

AGRADECIMIENTO

A un ser Supremo que me ha dado fuerzas para seguir adelante a pesar de tantas dificultades, por que cuando quise claudicar no me lo permitía dándome su paz.

A mis Padres por apoyarme siempre, por la paciencia que tuvieron, la dedicación, los ánimos y fortaleza que me brindaron, cuando ya quise rendirme siempre estaban a mi lado con un abrazo, una palabra de aliento gracias José Luis Pazmiño, gracias Nelly Ligia Pineda mis padres queridos.

A mi ñaño Sebas que supo tolerar mi mal genio, a pesar de todo se que estaba ahí pendiente de mi, gracias por regalarme un hermoso sobrino.

Como olvidarme de mis queridos amigos los que me acompañaron a lo largo de mi vida estudiantil, con los que compartí alegrías, locuras, risas, amanecidas en fin muchos buenos y malos momentos, en una lucha constante llamada Universidad, de los que estoy segura siempre me acompañaran tanto personal como profesionalmente.

A mi tutor Fabrízio Villasís el que por ninguna razón me dejo rendirme, el que siempre estuvo pendiente dándome ánimos, por los alones de orejas y por ser más que mi tutor un gran amigo.

A cada uno de mis profesores que supieron impartir su cátedra con toda la dedicación, demostrando buenos valores y más que nada paciencia para educar.

Diana

RESUMEN

Se trata de un sistema llamado GPS LOCARD el cual funciona con software y hardware libre denominado Arduino UNO, el mismo que consta de una interfaz de usuario sumamente amigable.

El GPS LOCARD adicionalmente tiene un GPS que capta las señales emitidas por el satélite dando una longitud, latitud, tiempo, velocidad y altura de los vehículos que porten el dispositivo, posteriormente se almacenan los datos en una MicroSD en formato de archivo .CV.

Para poder visualizar la ruta establecida por los vehículos es necesario introducir la MicroSD en un computador, descargar el archivo obtenido y subirlo en Google Earth.

Pudiendo comparar la ruta establecida originalmente con la ruta realizada por los vehículos obteniendo un control adecuado de los vehículos.

SUMMARY

This is a GPS system called Locard which works with free software and hardware called Arduino UNO, the same consisting of a very friendly user interface.

The GPS has a GPS Locard additionally captures signals from the satellite giving a longitude, latitude, time, speed and height of the vehicles that carry the device, then the data is stored in a file format MicroSD. CV.

To view the route established by the vehicles need to enter the MicroSD into a computer, download and upload the resulting file in Google Earth. Being able to compare the beaten path originally by vehicles getting proper control of the vehicle.

TABLA DE CONTENIDO

CAPITULO 1 PROBLEMATIZACIÓN

1.1 Introducción	1
1.2 Antecedentes	2
1.3 Problema Investigado	5
1.4 Formulación del Problema	5
1.5 Justificación	5
1.5.1 Justificación Teórica	5
1.5.2 Justificación Metodológica	6
1.5.3 Justificación Práctica	6
1.6 Sistematización	7
1.6.1 Diagnostico	7
1.6.2 Pronostico	8
1.6.3 Control de Pronóstico	8
1.7 Objetivos	8
1.7.1 Objetivo General	8
1.7.2 Objetivos Específicos.....	8
1.8 Alcance y Limitaciones.....	9
1.8.1 Alcance	9
1.8.2 Limitaciones	10
1.9 Estudios de Factibilidad	10
1.9.1 Técnica	10

1.9.2 Operativa	11
1.9.3 Económica	11
1.10 Metodología científica	11

CAPITULO 2 MARCO TEÓRICO

2.1 Introducción	12
2.2 El GPS	12
2.2.1 Descripción de GPS.....	12
2.2.2 Funcionamiento del GPS	13
2.2.3 Nombres y Descripción de las funciones de un GPS.....	15
2.3 Microcontroladores.....	16
2.3.1 Familias de microcontroladores	17
2.4 Puerto de Comunicación USB.....	17
2.4.1 Velocidades de transmisión	18
2.4.2 Tipos de conectores.....	19
2.4.3 Funcionamiento del USB	20
2.5 Definición de Software Libre	22
2.6 Hardware Libre.....	23
2.7 Arduino.....	24
2.7.1 Hardware de Arduino	25
2.7.2 Arduino Uno.....	28
2.7.2.1 Características Generales	29
2.7.3 Software para Arduino	34

2.7.3.1 Entorno Arduino	34
2.7.4.1 Estructura	39
2.7.4.2 Sintaxis	40
2.7.4.3 Variables.....	42
2.7.4.4 Tipos de datos	44
2.7.4.5 Operadores Booleanos	48
2.7.4.6 Control de Flujo.....	48

CAPITULO 3 DISEÑO E IMPLEMENTACIÓN

3.1 Introducción	52
3.2 Representación del comportamiento.....	53
3.2.1 SISTEMA MICROCONTROLADO	57
3.2.2 DISPOSITIVO DE RECEPCIÓN GPS	72
3.2.3 Dispositivo de Almacenamiento MicroSD	75
3.2.4 Etapa de Comparación	76
3.2.5 Etapa de Alimentación	79

CAPITULO 4 ANÁLISIS FINANCIERO

4.1. Matriz FODA	80
4.2. Análisis de costo - beneficio.....	81

CAPITULO 5 CONCLUSIONES Y RECOMENDACIONES

5.1 CONCLUSIONES	83
5.2 RECOMENDACIONES	84

LISTA DE CUADROS Y GRAFICOS

Gráficos

Fig.1.1 Diseño de modelo GPS-GSM-SMS	3
Fig.1.2 Sistema GPS con WIFI	4
Fig.1.3 Sistema de posicionamiento para vehículos autónomos	4
Fig.2.1 Satélite GPS en órbita. Representación gráfica: NASA	13
Fig.2.2 Satélites GPS en órbita	14
Fig.2.3 Esquema de un microcontrolador	16
Fig.2.4 Tipos de conectores USB	19
Fig.2.5 Definición gráfica de Software Libre	23
Fig.2.6 Placa Arduino Uno	25
Fig.2.7 Placa Arduino Duemilanove	26
Fig.2.8 Arduino GPS Shield	27
Fig.2.9 Arduino Shield Micro SD	28
Fig.2.10 Componentes de la placa Arduino Uno	29
Fig.2.11 IDE para Arduino	35
Fig. 3.1 Funcionamiento de Sistema GPS LOCARD	53
Fig. 3.2 Diagrama General Sistema GPS LOCARD	53
Fig. 3.3 Diagrama de Etapas y Funciones Sistema GPS LOCARD	54
Fig. 3.4 Diagrama de Bloques Sistema GPS LOCARD	55
Fig. 3.5 Diagrama de etapas con elementos utilizados en sistema GPS LOCARD	57
Fig. 3.6 Descripción grafica de distribución de elementos en placa Arduino UNO	58
Fig. 3.7 Programa para Arduino UNO	59

Fig. 3.8 Asignación de puerto Serial en Arduino UNO	60
Fig. 3.9 Librerías necesarias para ejecutar el programa	61
Fig. 3.10 Despliegue de datos con GPS LOCARD conectado al computador	63
Fig. 3.11 Modo de conexión placa Arduino UNO con modulo GPS EM-406A	64
Fig. 3.12 Datos captados por el GPS	74
Fig. 3.13 Distribución de pines GPS modelo EM-406A	74
Fig. 3.14 Modulo GPS modelo EM-406A	75
Fig. 3.15 Modulo Shield MicroSD para Arduino	75
Fig. 3.16 Adaptador para MicroSD	76
Fig. 3.17 Archivos .CV guardados en la MicroSD	76
Fig. 3.18 Datos en archivo .CV	77
Fig. 3.19 Configuración para despliegue de datos en Google Earth	77
Fig. 3.20 Visualización en Google Earth realizada por el vehículo	78
Fig. 3.21 Baterías recargables	79

Tablas

Tabla 2.1. Familia de Microcontroladores	17
Tabla 2.2 Características generales de la placa Arduino	30
Tabla 4.1 Matriz FODA	80
Tabla 4.2 Costo de materiales utilizados en el módulo GPS LOCARD	81
Tabla 4.3 Costo de la implementación del sistema de control de Ruta GPS LOCARD	82

LISTA DE ANEXOS

Anexo 1 DATA SHEET GPS MODELO EM-406A	92
Anexo 2 FOTOGRAFÍAS DE SISTEMA GPS LOCARD	94
Anexo 3 MANEJO DEL USB	95

CAPÍTULO 1

PROBLEMATIZACION

1.1 Introducción

En el Ecuador como en el resto del mundo existen empresas grandes y pequeñas que realizan diferentes actividades como por ejemplo repartición de productos, prestación de servicios, transportación, entre otras, todas ellas deben cumplir con una ruta y cronogramas establecidos para satisfacer a sus clientes.

En muchos de los casos la planificación de trabajo y cronogramas no se cumplen por el desvío de ruta por parte de los empleados, debido a que, en lugar de cumplir con las diligencias programadas, estos realizan trámites personales u otras funciones ajenas a sus obligaciones y como las empresas generalmente no cuentan con algún medio de control, esto provoca problemas con los clientes, retrasos en las entregas, y a futuro que los usuarios escojan a otras empresas para dar solución a sus necesidades.

Es por ello que el presente proyecto entrega una solución práctica para el control de ruta en vehículos mediante el dispositivo llamado GPS LOCARD, el cual está basado en software y hardware libre de última tecnología, llamado Arduino y de su complemento (Shield) GPS.

El dispositivo GPS LOCARD permite tener un control eficiente de la ruta de vehículos, mostrando si el mismo tuvo algún desvío de la ruta trazada originalmente, guardando los datos en el dispositivo para posteriormente ser descargados en un computador como un archivo cualquiera. Previamente se han

realizado las respectivas pruebas del dispositivo en un vehículo de transporte público de la ciudad de Quito, demostrando que el equipo es confiable, seguro y de fácil operación.

1.2 Antecedentes

En diversas empresas se cuenta con servicio de entrega de productos, prestación de servicios, transportación, etc., para cada uno de estos casos es necesario seguir una ruta establecida así como un cronograma para cumplir con los clientes. Para ello es necesario contar con personal capacitado así como eficiente para realizar el trabajo, pero en muchos de los casos los empleados realizan otras actividades ajenas a sus obligaciones y no cumplen con las necesidades de los clientes, provocando a futuro la pérdida de los mismos y perjudicando a la empresa.

De ahí la importancia de proponer un proyecto que controle de una forma eficaz a los vehículos mediante un sistema de control de ruta para tener la localización que tuvieron los vehículos en todo momento, teniendo un registro computarizado y para posteriormente compararlo con la ruta original.

En el Ecuador desde el 2010 ya se implementó un sistema de monitoreo GPS en 2000 unidades de transporte urbano, pero la plataforma tecnológica está incompleta por lo que no está totalmente operativo el sistema mencionado.

Este sistema envía una señal que es receptada en el Centro de Gestión de Flotas de la Empresa Pública Metropolitana de Movilidad y Obras Públicas de la ciudad de Quito EPMMOP, ahí los datos sobre número de buses, horarios, rutas y

frecuencias son monitoreados desde hace un año, la información en este centro de monitoreo se actualiza cada 30 minutos.

De acuerdo a la página <http://dspace.ups.edu.ec/handle/123456789/70> en la Universidad Politécnica Salesiana de la ciudad de Cuenca se realizó el Localizador vehicular de un móvil mediante GPS-GSM-SMS. Se trata de un sistema de monitoreo de un solo bloque utilizando GPS-tecnología GSM y SMS, de esta forma el software realiza una llamada utilizando un modem que no es más que un teléfono celular GSM, el equipo es un Nokia 6230, al GPS-GSM el cual devuelve los datos de latitud y longitud mediante el sistema SMS al modem, procesando la información en el software. Se inicializa el software antes de empezar a ubicar al móvil, ingresando los datos de la persona encargada del vehículo, se carga y el sistema empieza a funcionar en conjunto. Una vez que el modem recibe los datos; el software lo ubica en el mapa de Cuenca. Su localización, puede aparecer en distintos colores según el operador lo elija y puede localizar al móvil en testeos de minutos según la aplicación y el requerimiento del usuario.

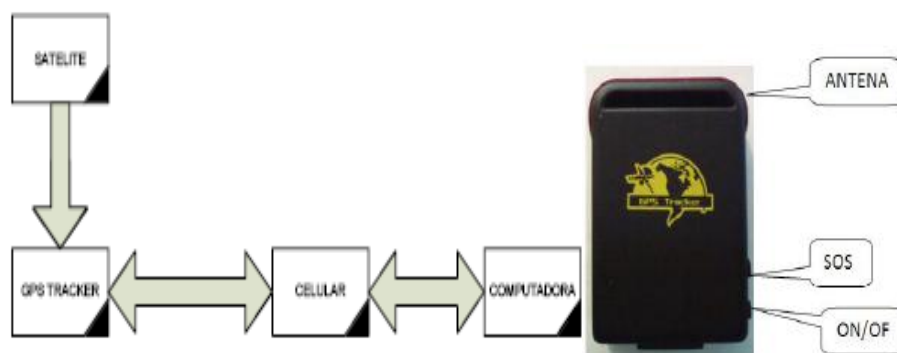


Fig. 1.1 Diseño de modelo GPS-GSM-SMS

Adicionalmente fuera del país se ha realizado las siguientes implementaciones:

- En la página

http://www.robSAFE.com/personal/mocana/papers/waf08_fernando.pdf se puede encontrar que se ha realizado investigaciones con GPS y WIFI aplicado en evacuaciones de emergencia de la Universidad de Alcalá mostrando las posibles rutas de evacuación en caso de emergencia.



Fig.1.2 Sistema GPS con WIFI

- Sistema de posicionamiento para vehículos autónomos realizado por la Universidad Politécnica de Madrid es un sistema de posicionamiento formado por la combinación de un GPS con una unidad de medida inercial ayudada por los sensores embarcados en el coche para realizar el guiado.



Fig. 1.3 Sistema de posicionamiento para vehículos autónomos

Los principales problemas que presentan todos estos sistemas mencionados son los elevados costos, complejo funcionamiento, así también están sujetos a suscripciones para tener acceso al servicio de localización GPS.

Por lo anteriormente expuesto se propone un sistema sencillo con software y hardware libre, para que empresas pequeñas que necesiten un monitoreo de sus vehículos tengan un correcto control de sus rutas a bajos costos, detectando posibles desvíos en las mismas y permitiendo tomar acciones correctivas en el desvío de rutas.

1.3 Problema Investigado

En el Ecuador no se cuenta con un sistema de Control de Ruta vehicular con GPS y software libre.

1.4 Formulación del Problema

¿Se puede diseñar un sistema de control de ruta mediante GPS que permita la comparación de localización vehicular con hardware y software libre?

1.5 Justificación

1.5.1 Justificación Teórica

La justificación teórica se la realiza mediante el conocimiento de los elementos a ser empleados en el presente trabajo de titulación como son el GPS, módulo Arduino y la utilización de Software Libre para Arduino, las diversas shields para almacenar los datos en una tarjeta MicroSD; los mismos que cumplen con todos los requerimientos del Control de Ruta.

1.5.2 Justificación Metodológica

Se utilizaron los siguientes métodos de investigación de acuerdo al desarrollo del proyecto:

- Método de análisis que permite determinar la realidad actual en la que se encuentra el control de ruta mediante comparación con GPS empleando software libre.
- Métodos deductivo e inductivo se aplicaron con el fin de determinar los elementos y software que más se acople para cumplir con las necesidades del presente trabajo de titulación.
- Con el método experimental se demostró el funcionamiento del Sistema de Control de Ruta.

1.5.3 Justificación Práctica

En las diversas empresas existen vehículos que tienen que cumplir con visitas establecidas para llegar a sus clientes estas deben seguir un cronograma determinado, en muchos de los casos las empresas realizan los controles de ruta de forma manual o no cuentan con un control de ruta para sus vehículos, esto hace que dichas visitas se realicen de una forma ineficiente. Razón por la cual se ve la necesidad de llevar un control de ruta en los vehículos de las empresas de una manera automatizada.

A continuación se detallan las razones que justifican el proyecto:

1. Permite tener un control de los vehículos de las empresas pudiendo detectar si hubieron desvíos y posteriormente corregirlos, para evitar pérdidas de recursos en las empresas.

2. Permite tener un registro permanente de todos los lugares visitados por el vehículo mediante un sistema de base de datos, el cual sirve para comparar la ruta realizada con la original o esperada, optimizando el tiempo y mejorando las rutas a realizarse.
3. Con el control de ruta se optimiza el tiempo de llegada a los clientes teniendo a futuro mayor ganancias para la empresa y que el cliente este satisfecho.
4. Debido a que el sistema de control de ruta cuenta con software y hardware libre no está sujeto a suscripciones representando un ahorro para la empresa.

1.6 Sistematización

1.6.1 Diagnostico

Actualmente en el Ecuador se cuenta con localizadores vehiculares basados en GPS como por ejemplo GPS Tracker, Chevystar, entre otros; el problema de estos equipos es que están sujetos a suscripciones para su funcionamiento, son extremadamente caros y no en ninguno de los casos cuenta con software libre.

Entre las principales problemáticas que existen con los usuarios si no se implementa el control de ruta son:

- Desvió de la ruta programada para que los vehículos realicen actividades ajenas a sus funciones.
- Aplazamientos en el cronograma establecido.
- Desperdicio de recursos económicos en los vehículos.

1.6.2 Pronostico

En el caso de no contar con un control de ruta vehicular se producirán los siguientes problemas:

- No se tendrá un control adecuado de las rutas de los vehículos.
- Aplazamientos y modificación del cronograma de trabajo.

1.6.3 Control de Pronóstico

La solución que se recomienda aplicar para evitar que sucedan las falencias detectadas por la falta de un Control de Ruta es implementar el presente trabajo de titulación.

1.7 Objetivos

1.7.1 Objetivo General

Investigar, diseñar e implementar un sistema prototipo para el control de ruta de vehículos, con el uso de GPS, software y hardware libre, que almacene las coordenadas de la ruta recorrida por el automotor y permita compararlas con la ruta establecida.

1.7.2 Objetivos Específicos

- Realizar un estudio y diseño del funcionamiento y disponibilidad de los elementos para la implementación del Sistema, adicionalmente encontrar la manera adecuada para comparar los resultados entre los datos obtenidos y la ruta establecida.
- Implementar un circuito electrónico basándose en software y hardware libre que permita la captura de información del GPS para obtener el momento y

las posiciones de ruta seguidas por el vehículo y posteriormente almacenar la información obtenida.

- Registrar en archivos de datos los momentos y las ubicaciones de la ruta seguida por el vehículo durante el día para poder compararlo con la ruta establecida.

1.8 Alcance y Limitaciones

1.8.1 Alcance

El dispositivo posee una interfaz fácil de manejar por el usuario dotado de un lector de memorias microSD para descargar los archivos de datos de las coordenadas almacenadas en el dispositivo, para posteriormente comparar los datos con la ruta original visualizando tiempos y ubicaciones exactas del vehículo.

El dispositivo debe ser capaz de soportar las condiciones ambientales y de transporte en un vehículo por lo cual para demostrar que el dispositivo funciona adecuadamente se realizó las respectivas pruebas implementando el equipo en un bus de transporte urbano de la ciudad de Quito. El equipo irá almacenando las coordenadas por donde el vehículo circule y posteriormente estos datos podrán ser descargados en el computador, pudiendo comparar los resultados con los de la ruta original.

Los beneficios obtenidos al implementar este dispositivo en los vehículos de las empresas se ve reflejados en la eficiencia el momento de llegar al cliente, así como contar con una base de datos confiable, pudiendo rectificar desvíos en la ruta si los hubiera reduciendo los gastos económicos excesivos en los vehículos

siendo también una manera ecológica de preservar recursos y de controlar al personal.

1.8.2 Limitaciones

1. Solo se diseñará un sistema que obtenga la posición del vehículo y la almacene en el mismo, guardando estos datos en la memoria del dispositivo cada segundo.
2. El software se encargara únicamente de desplegar la ruta seguida si se cumple por el vehículo en un mapa por lo que una persona deberá determinar si se siguió la ruta propuesta o si hubo desvíos no planificados o faltas en la ruta.
3. Se pueden tener limitantes por razones de espacio en la memoria RAM del dispositivo aunque podrá almacenar ## días seguidos con 8 horas de trabajo diario.
4. Solo se implementa el dispositivo en un vehículo de transporte urbano por motivo de validación del sistema para verificar que funciona adecuadamente.

1.9 Estudios de Factibilidad

El estudio de factibilidad requerido para efectos del diseño del proyecto, se basa en 3 aspectos o niveles: Técnico, Económico y Operativo. A continuación, se evalúa cada una de estas factibilidades por separado:

1.9.1 Técnica

Técnicamente, existe la tecnología necesaria para satisfacer los requerimientos que se plantean en este proyecto, ya que los equipos y dispositivos electrónicos necesarios están disponibles en el mercado. Además, como parte del desarrollo del proyecto se adquieren las competencias necesarias para trabajar personalmente con todos los equipos y dispositivos electrónicos. Este hecho

implica que no será necesaria la contratación de servicios o personal externo, lo que evitaría un gasto adicional.

1.9.2 Operativa

En el área de aplicación del sistema los usuarios no necesitan un conocimiento especial para poder utilizar los equipos del sistema de una forma adecuada. Además que se tratara de realizar un sistema que sea amigable con el usuario. La entrega de manuales de usuario evitará los posibles daños causados por el desconocimiento de los equipos disponibles.

1.9.3 Económica

El costo que genera el diseño del sistema que se propone es bajo, ya que la tecnología que se utilizará es bajo la filosofía de hardware y software libre, que se caracteriza por ser económica, a comparación con otras tecnologías. En función de ello, y de los beneficios que aportaría este sistema, se considera que el proyecto es económicamente factible.

1.10 Metodología científica

El sistema metodológico usado para la recopilación de información fue el método de investigación bibliográfico; debido a que fue necesario conseguir la mayor cantidad de información acerca de toda la tecnología a utilizarse, como se trata de manejar software y hardware libre y no está muy desarrollado en el medio es decir no existe una amplia documentación acerca del tema, fue por ello que la investigación necesaria se basó en páginas web e información de diferentes portales de internet, en este caso se debió aplicar el método lógico deductivo y el experimental para a través de pruebas y deducción obtener la información más certera acerca del Control de Ruta mediante GPS.

CAPITULO 2

MARCO TEÓRICO

2.1 Introducción

En este capítulo se hace una descripción detallada de los componentes relevantes del sistema GPS LOCARD, así como las metodologías y normas a utilizar, sus características, la forma en que son utilizadas, como ayudan en el proceso de desarrollo y los beneficios que conllevan.

2.2 El GPS

2.2.1 Descripción de GPS

El GPS o sistema de posicionamiento Global, es un sofisticado sistema de orientación y navegación cuyo funcionamiento está basado en la recepción y procesamiento de las informaciones emitidas por una constelación de 24 satélites conocida como NAVSTAR, orbitando en diferentes alturas a unos 20.000 km. por encima de la superficie terrestre.

Cada satélite, da dos vueltas diarias al planeta. Las trayectorias y la velocidad orbital han sido calculadas para que formen una especie de red alrededor de la tierra (debe haber en todo momento cinco satélites a la vista en cualquier zona), de manera que un receptor GPS a cualquier hora del día o de la noche, en cualquier lugar, con independencia de las condiciones meteorológicas, pueda facilitar la posición que ocupa al captar y procesar las señales emitidas por un mínimo de tres satélites.

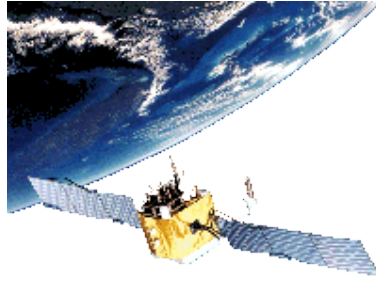


Fig. 2.1 Satélite GPS en órbita. Representación gráfica: NASA

El GPS fue desarrollado por el departamento de defensa de Estados Unidos al final del período de la "Guerra Fría" con fines militares. Superada esta fase, se extendió su uso a aplicaciones civiles comenzando a utilizarse en náutica y aviación.

En sus comienzos la cobertura no era total pues faltaba situar en órbita varios satélites, además su elevado precio los ponía fuera de alcance de la mayoría de los usuarios potenciales. Actualmente la red es totalmente operativa, incluyendo satélites de reserva y hay disponibles en el mercado receptores GPS a precio asequible.

2.2.2 Funcionamiento del GPS

Cada satélite de la constelación GPS emite continuamente dos códigos de datos diferentes en formato digital. Estos datos son transmitidos por medio de señales de radio.

Uno de los códigos está reservado para uso exclusivamente militar y no puede ser captado por los receptores GPS civiles. El otro código, (de uso civil) transmite dos series de datos conocidas como Almanaque y Evento. Los datos ofrecidos por el almanaque y los eventos informan sobre el estado operativo de funcionamiento del satélite, su situación orbital, la fecha y la hora.

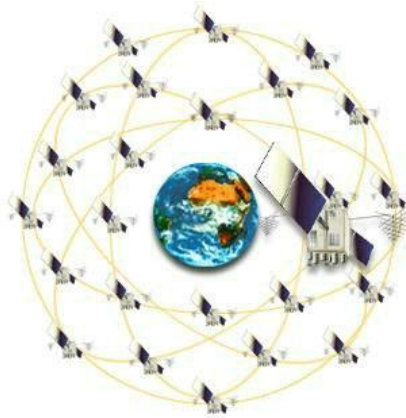


Fig. 2.2 Satélites GPS en órbita.

Obviamente cada satélite emite sus propios eventos y almanaques que incluyen un código de identificación específico para cada satélite. Los satélites están equipados con relojes atómicos que garantizan una precisión casi total, ofreciendo un error estimado en un segundo cada 70.000 años.

Un receptor GPS debe disponer en su memoria del almanaque y los eventos actualizadas (si no lo están se actualizarán automáticamente en poco tiempo, cuando el receptor sintonice las señales emitidas por un mínimo de tres satélites), de esta manera sabrá dónde buscar los satélites en el firmamento.

Los satélites transmiten continuamente su situación orbital y la hora exacta. El tiempo transcurrido entre la emisión de los satélites y la recepción de la señal por parte del receptor GPS, se convierte en distancia mediante una simple fórmula aritmética (el tiempo es medido en nanosegundos).

Al captar las señales de un mínimo de tres satélites, por triangulación el receptor GPS determina la posición que ocupa sobre la superficie de la tierra mediante el valor de las coordenadas de longitud y latitud (dos dimensiones). Dichas coordenadas pueden venir expresadas en grados, minutos y/o segundos o en las unidades de medición utilizadas en otros sistemas geodésicos. La captación de

cuatro o más satélites facilita, además, la altura del receptor con respecto al nivel del mar (tres dimensiones). Las coordenadas de posición y otras informaciones que puede facilitar el receptor, se actualizan cada segundo o cada dos segundos.

2.2.3 Nombres y Descripción de las funciones de un GPS

POSICIÓN: Indicar la posición del GPS. Facilita la localización casi exacta del receptor. Para ello el GPS tiene que haber captado las señales emitidas al menos por tres satélites.

ALTURA: al captar 4 o más satélites el GPS indica la altura sobre el nivel del mar. (Sensible a Disponibilidad Selectiva)

TIEMPO: el GPS una vez inicializado, aunque no reciba señales satelitales indica la hora y fecha, si recibe señales indica la hora exacta.

PUNTO DE PASO O PUNTO DE REFERENCIA: El waypoint es la posición de un único lugar sobre la superficie de la tierra expresada por sus coordenadas.

Un waypoint puede ser un punto de inicio, de destino o un punto de paso intermedio en una ruta. Todos los GPS pueden almacenar en memoria varios Waypoints, los cuales se pueden borrar, editar, e identificar mediante caracteres alfa numérico.

Algunos GPS permiten agrupar una sucesión de waypoints representando un recorrido, a esto se le llama ruta.

DISTANCIA: introduciendo las coordenadas de dos puntos, la función distancia del GPS informa la separación de ambos y el rumbo en grados que hay que seguir desde el marcado como inicio al de destino.

2.3 Microcontroladores

Un microcontrolador es un circuito integrado programable, capaz de ejecutar las órdenes grabadas en su memoria. Está compuesto de varios bloques funcionales, los cuales cumplen una tarea específica. Un microcontrolador incluye en su interior las tres principales unidades funcionales de una computadora: unidad central de procesamiento, memoria y periféricos de entrada/salida.

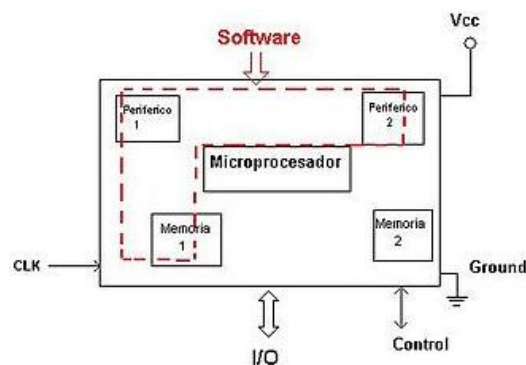


Fig. 2.3. Esquema de un microcontrolador

Algunos microcontroladores pueden utilizar palabras de cuatro bits y, funcionan a velocidad de reloj con frecuencias tan bajas como 4 kHz, con un consumo de baja potencia (mW o microvatios). Por lo general, tendrá la capacidad para mantener la funcionalidad a la espera de un evento como pulsar un botón o de otra interrupción, el consumo de energía durante el sueño (reloj de la CPU y los periférico de la mayoría) puede ser sólo nanovatios, lo que hace que muchos de ellos muy adecuados para aplicaciones con batería de larga duración. Otros microcontroladores pueden servir para roles de rendimiento crítico, donde sea necesario actuar más como un procesador digital de señal (DSP), con velocidades de reloj y consumo de energía más altos.

Al ser fabricados, la memoria ROM del microcontrolador no posee datos. Para que pueda controlar algún proceso es necesario generar o crear y luego grabar en

la memoria EEPROM o equivalente del microcontrolador algún programa, el cual puede ser escrito en lenguaje ensamblador u otro lenguaje para microcontroladores; sin embargo, para que el programa pueda ser grabado en la memoria del microcontrolador, debe ser codificado en sistema numérico hexadecimal que es finalmente el sistema que hace trabajar al microcontrolador cuando éste es alimentado con el voltaje adecuado y asociado a dispositivos analógicos y discretos para su funcionamiento.

2.3.1 Familias de microcontroladores

Los microcontroladores más comunes en uso son:

Empresa	8 bits	16 bits	32 bits
Atmel	AVR (mega y tinv), 89Sxxxx familia similar 8051		SAM7 (ARM7TDMI), SAM3 (AEM Cortex-M3), SAM9 (ARM926)
Freescale (antes Motorola)	68HC05, 68HC08, 68HC11, HCS08	68HC12, 68HCS12, 68HCSX12, 68HC16	683XX, PowerPC, ColdFire
Intel	MCS-48 (familia 8048) MCS51 (familia 8051)	MCS96, MXS96	X
Microchip	Familia 10f2xx, 12Cxx, 12Fxx, 16Cxx, 16Fxx, 18Cxx y 18Fxx	PIC24F, PIC24H, dsPIC30Fxx, dsPIC33F con motor dsp integrado	PIC32

Tabla 2.1. Familia de Microcontroladores

2.4 Puerto de Comunicación USB

El Universal Serial Bus (Bus Universal Serie), abreviado comúnmente USB, es un puerto que sirve para conectar periféricos a un ordenador.

El diseño del USB tenía en mente eliminar la necesidad de adquirir tarjetas separadas para poner en los puertos bus ISA o PCI, y mejorar las capacidades Plug-and-Play permitiendo a esos dispositivos ser conectados o desconectados al sistema sin necesidad de reiniciar, y también fue pensado como un reemplazo para conexiones en serie y paralelo de un PC y sus periféricos.

2.4.1 Velocidades de transmisión

Los dispositivos USB se clasifican en cuatro tipos según su velocidad de transferencia de datos:

- **Baja velocidad (1.0):** Tasa de transferencia de hasta 1,5Mbps (192kB/s). Utilizado en su mayor parte por dispositivos de interfaz humana (Human Interface Device, en inglés) como los teclados, los ratones (mouse), las cámaras web, etc.
- **Velocidad completa (1.1):** Tasa de transferencia de hasta 12Mbps (1,5MB/s) según este estándar, pero se dice en fuentes independientes que habría que realizar nuevamente las mediciones. Ésta fue la más rápida antes de la especificación USB 2.0, y muchos dispositivos fabricados en la actualidad trabajan a esta velocidad. Estos dispositivos dividen el ancho de banda de la conexión USB entre ellos, basados en un algoritmo de impedancias LIFO (Last In FirstOut - último en entrar, primero en salir).
- **Alta velocidad (2.0):** Tasa de transferencia de hasta 480Mbps (60MB/s) pero por lo general de hasta 125Mbps (16MB/s). Está presente casi en el 99% de los PC actuales. El cable USB 2.0 dispone de cuatro líneas, un par para datos, una de corriente y un cuarto que es el negativo o retorno.

- Súper alta velocidad (3.0): Tiene una tasa de transferencia de hasta 4.8Gbps (600MB/s). La velocidad del bus es diez veces más rápida que la del USB 2.0, debido a que han incluido 5 conectores extra, desechando el conector de fibra óptica propuesto inicialmente, y será compatible con los estándares anteriores, usa un cable de 9 hilos. En Octubre de 2009 la compañía taiwanesa ASUS lanzó la primera placa base que incluía puertos USB, tras ella muchas otras le han seguido y se espera que en 2012 ya sea el estándar de facto.

Las señales del USB se transmiten en un cable de par trenzado con impedancia característica de $90 \Omega \pm 15\%$, cuyos hilos se denominan D+ y D-. Estos, colectivamente, utilizan señalización diferencial en half dúplex excepto el USB 3.0 que utiliza un segundo par de hilos para realizar una comunicación en full dúplex. La razón por la cual se realiza la comunicación en modo diferencial es simple, reduce el efecto del ruido electromagnético en enlaces largos.

2.4.2 Tipos de conectores

Existen dos tipos de conectores USB:

- ⤴ Los conectores conocidos como **tipo A** (figura 2.4), cuya forma es rectangular y se utilizan, generalmente, para dispositivos que no requieren demasiado ancho de banda (como el teclado, el ratón, las cámaras Web, etc.);
- ⤴ Los conectores conocidos como **tipo B** (figura 2.4) poseen una forma cuadrada y se utilizan principalmente para dispositivos de alta velocidad (discos duros externos, etc.).



Fig. 2.4. Tipos de conectores USB

1. Fuente de alimentación de +5 V (*VBUS*) máximo 100 mA
2. Datos (*D-*)
3. Datos (*D+*)
4. Conexión a tierra (*GND*)

2.4.3 Funcionamiento del USB

Una característica de la arquitectura USB es que puede proporcionar fuente de alimentación a los dispositivos con los que se conecta, con un límite máximo de 15 V por dispositivo. Para poder hacerlo, utiliza un cable que consta de cuatro hilos (la conexión a tierra *GND*, la alimentación del *BUS* y dos hilos de datos llamados *D-* y *D+*).

El estándar USB permite que los dispositivos se encadenen mediante el uso de una topología en bus o de estrella. Por lo tanto, los dispositivos pueden conectarse entre ellos tanto en forma de cadena como en forma ramificada. La ramificación se realiza mediante el uso de cajas llamadas "concentradores" que constan de una sola entrada y varias salidas. Algunos son activos (es decir, suministran energía) y otros pasivos (la energía es suministrada por el ordenador).

La comunicación entre el host (equipo) y los dispositivos se lleva a cabo según un protocolo (lenguaje de comunicación) basado en el principio de red en anillo. Esto significa que el ancho de banda se comparte temporalmente entre todos los dispositivos conectados. El host (equipo) emite una señal para comenzar la secuencia cada un milisegundo (ms), el intervalo de tiempo durante el cual le ofrecerá simultáneamente a cada dispositivo la oportunidad de "hablar". Cuando

el host desea comunicarse con un dispositivo, transmite una red (un paquete de datos que contiene la dirección del dispositivo cifrada en 7 bits) que designa un dispositivo, de manera tal que es el host el que decide "hablar" con los dispositivos. Si el dispositivo reconoce su dirección en la red, envía un paquete de datos (entre 8 y 255 bytes) como respuesta. De lo contrario, le pasa el paquete a los otros dispositivos conectados. Los datos que se intercambian de esta manera están cifrados conforme a la codificación NRZI.

Como la dirección está cifrada en 7 bits, 128 dispositivos (2^7) pueden estar conectados simultáneamente a un puerto de este tipo. En realidad, es recomendable reducir esta cantidad a 127 porque la dirección 0 es una dirección reservada.

Debido a la longitud máxima de 5 metros del cable entre los dos dispositivos y a la cantidad máxima de 5 concentradores (a los que se les suministra energía), es posible crear una cadena de 25 metros de longitud.

Los puertos USB admiten dispositivos Plug and play de conexión en caliente. Por lo tanto, los dispositivos pueden conectarse sin apagar el equipo (conexión en caliente). Cuando un dispositivo está conectado al host, detecta cuando se está agregando un nuevo elemento gracias a un cambio de tensión entre los hilos D+ y D-. En ese momento, el equipo envía una señal de inicialización al dispositivo durante 10 ms para después suministrarle la corriente eléctrica mediante los hilos *GND* y *VBUS* (hasta 100 mA). A continuación, se le suministra corriente eléctrica al dispositivo y temporalmente se apodera de la dirección predeterminada (dirección 0). La siguiente etapa consiste en brindarle la dirección definitiva. Para

hacerlo, el equipo interroga a los dispositivos ya conectados para poder conocer sus direcciones y asigna una nueva, que lo identifica por retorno. Una vez que cuenta con todos los requisitos necesarios, el host puede cargar el driver adecuado.

2.5 Definición de Software Libre

Software libre significa que el software respeta la libertad de los usuarios y la comunidad. En términos generales, los usuarios tienen la libertad de copiar, distribuir, estudiar, modificar y mejorar el software. Con estas libertades, los usuarios (tanto individualmente como en forma colectiva) controlan el programa y lo que hace.

Un programa es software libre si los usuarios tienen las cuatro libertades esenciales:

- La libertad de ejecutar el programa para cualquier propósito (libertad 0).
- La libertad de estudiar cómo funciona el programa, y cambiarlo para que haga lo que usted quiera (libertad 1). El acceso al código fuente es una condición necesaria.
- La libertad de redistribuir copias para ayudar a su prójimo (libertad 2).
- La libertad de distribuir copias de sus versiones modificadas a terceros (libertad 3). Esto le permite ofrecer a toda la comunidad la oportunidad de beneficiarse de las modificaciones. El acceso al código fuente es una condición necesaria para ello.

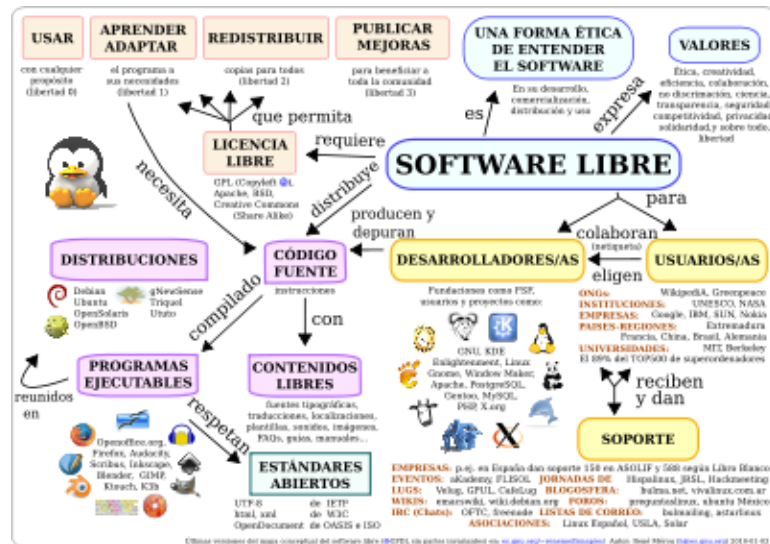


Fig. 2.5 Definición gráfica de Software Libre

2.6 Hardware Libre

Se llama hardware libre a los dispositivos de hardware cuyas especificaciones y diagramas esquemáticos son de acceso público, ya sea bajo algún tipo de pago o de forma gratuita. La filosofía del software libre (las ideas sobre la libertad del conocimiento) es aplicable a la del hardware libre. Se debe recordar en todo momento que libre no es sinónimo de gratis. El hardware libre forma parte de la cultura libre. Algo que tiene en común el hardware con el software es que ambos corresponden a las partes tangibles de un sistema informático sus componentes son; eléctricos electromecánicos y mecánicos son cables gabinetes o cajas.

Un ejemplo de hardware libre es la arquitectura UltraSparc cuyas especificaciones están disponibles bajo una licencia libre.

Dado que el hardware tiene asociados a él costos variables directos, ninguna definición de software libre se puede aplicar directamente sin modificación. En cambio, el término hardware libre se ha usado principalmente para reflejar el uso

del software libre con el hardware y el lanzamiento libre de la información con respecto al hardware, a menudo incluyendo el lanzamiento de los diagramas esquemáticos, diseños, tamaños y otra información acerca del hardware. De todos modos, incluye el diseño del hardware y la distribución de los elementos en la tarjeta madre.

Con el auge de los dispositivos de lógica programable reconfigurables, el compartir los diseños lógicos es también una forma de hardware libre. En vez de compartir los diagramas esquemáticos, el código HDL es compartido. Esto difiere del software libre. Las descripciones HDL son usadas comúnmente para instalar sistemas SoC en FPGA o directamente en diseños ASIC. Los módulos HDL, cuando se distribuyen, son llamados semiconductor intelectual property cores, o núcleos IP.

Existen muchas comunidades que trabajan en el diseño, desarrollo y pruebas de hardware libre, y que además brindan soporte. Algunas de ellas son Open Collector, OpenCores y el Proyecto gEDA.

2.7 Arduino

Arduino es una plataforma de electrónica abierta para la creación de prototipos basada en software y hardware flexibles y fáciles de usar. Se creó para artistas, diseñadores, aficionados y cualquiera interesado en crear entornos u objetos interactivos.

Arduino puede tomar información del entorno a través de sus pines de entrada de toda una gama de sensores y puede afectar aquello que le rodea controlando

luces, motores y otros actuadores. El microcontrolador de la placa Arduino se programa mediante el lenguaje de programación Arduino (basado en Wiring) y el entorno de desarrollo Arduino (basado en Processing). Los proyectos hechos con Arduino pueden ejecutarse sin necesidad de conectar a un ordenador, si bien tienen la posibilidad de hacerlo y comunicar con diferentes tipos de software.

Las placas pueden ser hechas a mano o comprarlas montadas de fábrica; el software puede ser descargado de forma gratuita. Los ficheros de diseño de referencia CAD están disponibles bajo una licencia abierta, así pues se es libre de adaptarlos a las necesidades.

“Arduino recibió una Mención Honorífica en la sección *Digital Communities* de la edición del 2006 del *Ars Electrónica Prix*”.

2.7.1 Hardware de Arduino

Hay diferentes versiones de placas Arduino. La actual placa básica, la Uno (figura 2.6), usa Atmel ATmega328, al igual que la Duemilanove. La anterior Diecimila, y las primeras unidades de Duemilanove usaban el Atmel ATmega168, mientras que las placas más antiguas usan el ATmega8. El Arduino Mega está basado en el ATmega1280.

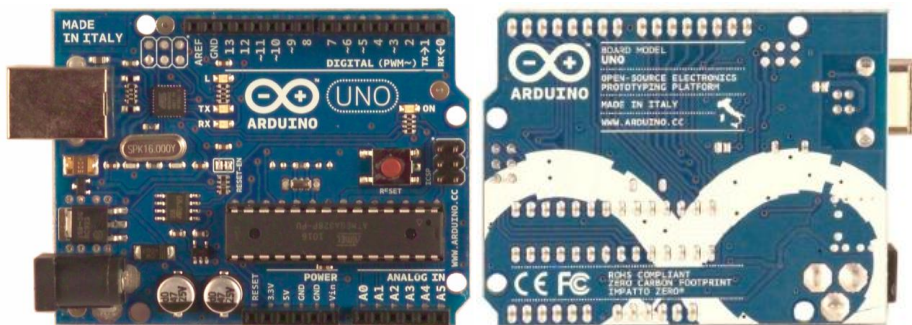


Fig. 2.6. Placa Arduino Uno

2.7.1.1 Placas de E/S

- Uno.- Esta es la última revisión de la placa Arduino USB básica. El Uno se diferencia de todas las placas anteriores, ya que no utiliza el chip FTDI USB - serie. Por el contrario, utiliza ATmega8U2 programado como un convertidor de USB a serie. Figura 2.6.
- Duemilanove.- "Duemilanove" significa 2009 en italiano que fue el año cuando salió al mercado. El Duemilanove es el más popular dentro de las series de placas con USB. Figura 2.7.

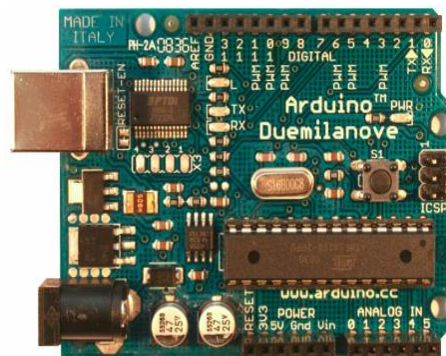


Figura 2.7. Placa Arduino Duemilanove

- Diecimila.- Fue la primera placa Arduino en integrar el chip FTDI para convertir de USB a serie y alimentarse con la energía proporcionada por la salida USB de la PC.

2.7.1.2 Shields Arduino

Un shield es una placa impresa que se pueden conectar en la parte superior de la placa Arduino para ampliar sus capacidades, pudiendo ser apilada una encima de la otra.

Las shields suelen ser diseños bastante simples y en general de código abierto y publicados libremente.

Entre las shields más conocidas se tiene las siguientes:

- ✦ Arduino Ethernet Shield
- ✦ Arduino microSD Shield
- ✦ Arduino Celular Shield SM5100B
- ✦ Arduino GPS Shield

2.7.1.2.1 Arduino GPS Shield

El Arduino GPS Shield (figura 2.8) permite a una placa Arduino conectarse a un sistema GPS. La placa dispone de un conector específico para un módulo receptor GPS EM-406.

Están disponibles en las placas todos los pines del receptor tales como RX, TX, PPS etc; así como una pequeña área para prototipo donde se puede soldar algunos componentes.

El interruptor DLINE/UART permite conmutar las salidas TX/RX del módulo GPS hacia el UART del Arduino o hacia los pines digitales 2 y 3 del Arduino. Otro interruptor ON/OFF permite alimentar o no el GPS. El pulsador de RESET de Arduino también está montado en la superficie.

La shield también dispone de un espacio para un zócalo para una batería de botón para la alimentación backup del GPS.



Fig. 2.8 Arduino GPS Shield

2.7.1.2.2 Arduino Shield MicroSD

Esta Shield permite conectar fácilmente una tarjeta microSD de alta capacidad para almacenar cantidades enormes de información como por ejemplo en proyectos de data-logging o similares.

La comunicación con la tarjeta de memoria se realiza mediante SPI. Los pines SCK, DI y DO están conectados al SPI del chip Atmega del Arduino (pines digitales 11-13), mientras que el pin CS está conectado al pin digital D8. Si se utiliza algunas de las librerías Open source disponibles tales como FAT16 o SDFat, es necesario asegurarse modificar la asignación de pines correctamente (la mayoría asume CS en el pin D10).

La microSD shield también dispone de una amplia área de prototipado donde se puede soldar componentes extra en caso de que sea necesario en un proyecto.

La placa Shield consta con un zócalo para tarjetas microSD soldado, así como un LED indicador de alimentación y un pulsador de reset.

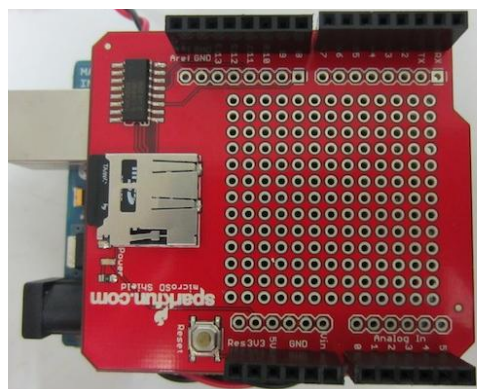


Fig. 2.9 Arduino Shield Micro SD

2.7.2 Arduino Uno

El Arduino Uno (figura 2.10) es una placa con microcontrolador basado en el ATmega328. Tiene 14 pines con entradas/salidas digitales (6 de las cuales

pueden ser usadas como salidas PWM), 6 entradas analógicas, un cristal oscilador a 16Mhz, conexión USB, entrada de alimentación DC, una cabecera ICSP, y un botón de reset. Contiene todo lo necesario para utilizar el microcontrolador; simplemente se conecta a un ordenador a través del cable USB para alimentarlo también se puede utilizar un adaptador o una batería para empezar a trabajar.

En la figura 2.10 se observa la placa Arduino Uno con sus componentes físicos y todos los pines disponibles.

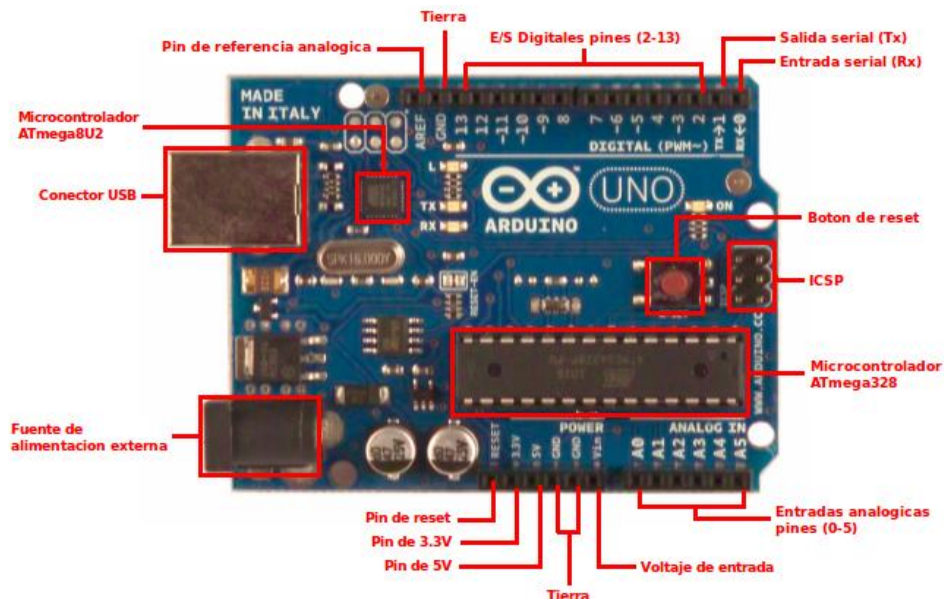


Figura 2.10. Componentes de la placa Arduino Uno

2.7.2.1 Características Generales

Microcontrolador	ATmega368
Voltaje de funcionamiento	5V
Voltaje de entrada (recomendado)	7-12V
Voltaje de entrada (limite)	6-20V

Pines E/S digitales	14 (6 proporcionan salida PWM)
Pines de entrada analógica	6
Intensidad por pin	40 mA
Intensidad en pin 3.3V	50 mA
Memoria Flash	32 KB de las cuales 2 KB las usa el gestor de arranque(bootloader)
SRAM	2 KB
EEPROM	1 KB
Velocidad de reloj	16 MHz

Tabla 2.2 Características generales de la placa Arduino

2.7.2.1.1 Alimentación

Puede ser alimentado vía conexión USB o con una fuente de alimentación externa DC. El origen de la alimentación se selecciona automáticamente.

Las fuentes de alimentación externas (no USB) pueden ser tanto un adaptador de pared o una batería. El adaptador se puede conectar usando un conector macho de 2.1 mm con centro positivo en el conector hembra de la placa. Los cables de la batería a los pines GND y Vin en los conectores de alimentación POWER.

La placa puede trabajar con una alimentación externa de entre 6 a 20 V. Si el voltaje suministrado es inferior a 7V el pin de 5V puede proporcionar menos de 5V y la placa puede volverse inestable, si se usan más de 12V los reguladores de voltaje se pueden sobre calentar y dañar la placa. El rango recomendado es de 7 a 12 V.

Los pines de alimentación son los siguientes:

- ⤴ **Vin:** Se puede proporcionar voltaje a través de este pin, o, si se está alimentando a través de la conexión de 2.1 mm, acceder a ella a través de este pin (7 a 12 V).
- ⤴ **5V:** Es el pin de salida de voltaje estabilizado, que es proporcionado por el Vin a través de un regulador integrado a la placa o directamente de la USB.
- ⤴ **3V3:** Es una fuente de 3.3V, generada por el regulador incluido en la placa. La corriente máxima soportada es de 50 mA.
- ⤴ **GND:** Pines de toma a tierra.

2.7.2.1.2 Memoria

El ATmega328 tiene 32 KB de memoria flash para almacenar códigos, 2 KB son usados para el arranque del sistema (bootloader). Tiene 2 KB de memoria SRAM. Posee 1 KB de EEPROM, a la cual se puede acceder para leer o escribir.

2.7.2.1.3 Entradas y Salidas

Cada uno de los 14 pines digitales pueden utilizarse como entradas o salidas usando las funciones `pinMode()`, `digitalWrite()` y `digitalRead()`. Las E/S operan a 5 V. Cada pin puede proporcionar o recibir una intensidad máxima de 40 mA y tienen una resistencia interna, pull up, (desconectada por defecto) de 20 K Ω . Además, algunos pines tienen funciones especializadas:

- **Serie: pin 0 (RX) y pin 1 (TX).** Usado para recibir (RX) y transmitir (TX) datos a través del puerto serie TTL. Estos pines están conectados en paralelo a los pines correspondientes del Atmega8U2 y a los pines RXD y TXD del Atmega.

- **Interrupciones Externas: pin 2 y pin 3.** Estos pines se pueden configurar para lanzar una interrupción en un valor LOW (0V), en flancos de subida o bajada (cambio de LOW a HIGH o viceversa), o en cambios de valor.
- **PWM: pines 3, 5, 6, 9, 10 y 11.** Proporciona una salida PWM (Pulse Wave Modulation, modulación por onda de pulso) de 8 bits de resolución con valores de 0 a 255. Se los identifica por el símbolo ~ en la placa Arduino.
- **SPI: pines 10 (SS), 11 (MOSI), 12 (MISO) y 13 (SCK).** Estos pines proporcionan comunicación SPI, que a pesar de que el hardware la proporcione actualmente no está incluido en el lenguaje Arduino.
- **LED: pin 13.** Hay un led integrado en la placa conectado al pin digital 13, cuando este pin tiene un valor HIGH (5V) el led se enciende y cuando éste tiene un valor LOW (0V) este se apaga.

La Uno tiene 6 entradas analógicas y cada una de ellas proporciona una resolución de 10 bits (1024 valores). Por defecto se mide de tierra a 5 V, aunque es posible cambiar la cota superior de este rango usando el pin AREF y la función `analogReference()`. Además algunos pines tienen funciones especializadas:

- **I²C: pin 4 (SDA) y pin 5 (SCL).** Soporte del protocolo de comunicaciones I²C (TWI) usando la librería `Wire`.

Hay otros pines en la placa:

- **AREF.** Voltaje de referencia para las entradas analógicas. Configura el voltaje de referencia usado por la entrada analógica. La función `analogRead()`

devolverá un valor de 1023 para aquella tensión de entrada que sea igual a la tensión de referencia. El valor del voltaje debe estar en el rango de 0 a 5 V.

- **Reset.** Suministra un valor LOW (0V) para reiniciar el microcontrolador. Típicamente usado para añadir un botón de reset a los Shields que no permiten acceso a la placa.

2.7.2.1.4 Comunicación

EL Arduino Uno facilita en varios aspectos la comunicación con el ordenador, otro Arduino u otros microcontroladores. El ATmega328 proporciona comunicación vía serie UART TTL (5V), disponible a través de los pines digitales 0(RX) y 1(TX). Un microcontrolador ATmega8U2 integrado en la placa que canaliza la comunicación serie a través del USB y proporcionan un puerto serie virtual en el ordenador. El software incluye un monitor de puerto serie que permite enviar y recibir información textual de la placa Arduino. Los LEDs RX y TX de la placa parpadearán cuando se detecte comunicación transmitida a través del ATmega8U2 y la conexión USB (no parpadearán si se usa la comunicación serie a través de los pines 0 y 1).

La librería SoftwareSerial permite comunicación serie por cualquier par de pines digitales del Uno.

El ATmega328 también soportan la comunicación I²C (TWI) y SPI . El software de Arduino incluye una librería Wire para simplificar el uso el bus I²C.

2.7.2.1.5 Protección contra sobre tensiones en el USB

El Arduino Uno tiene un multi fusible reiniciable que protege la conexión USB del ordenador de cortocircuitos y sobre tensiones. A parte que la mayoría de ordenadores proporcionan su propia protección interna, el fusible proporciona una capa extra de protección. Si más de 500 mA son detectados en el puerto USB, el fusible automáticamente corta la conexión hasta que el cortocircuito o la sobre tensión desaparece.

2.7.3 Software para Arduino

El entorno de código abierto Arduino hace fácil escribir el código y cargarlo a la placa de E/S. Funciona en Windows, Mac OS X, Linux y Androide. El entorno está escrito en Java y basado en Processing, avr-gcc y otros programas también de código abierto.

2.7.3.1 Entorno Arduino

El entorno Arduino también es conocido como IDE para Arduino, que se puede ver en la figura 2.9.

El entorno de desarrollo está constituido por un editor de textos para escribir el código, una área de mensajes, una consola de textos, una barra de herramientas con botones para las funciones comunes y una serie de menús. Permite la conexión con el hardware de Arduino para cargar los programas y comunicarse con ellos.

Arduino utiliza para escribir el software lo que se denomina sketch (programa). Estos programas son escritos en el editor de texto. Existe la posibilidad de cortar,

pegar y buscar/reemplazar texto. En el área de mensajes se muestra información mientras se carga el programa y también muestra errores. La consola muestra el texto de salida para el entorno de Arduino incluyendo los mensajes de error completos y otras informaciones. La barra de herramientas permite verificar el proceso de carga, creación, apertura y guardado de programas, y la monitorización serie.

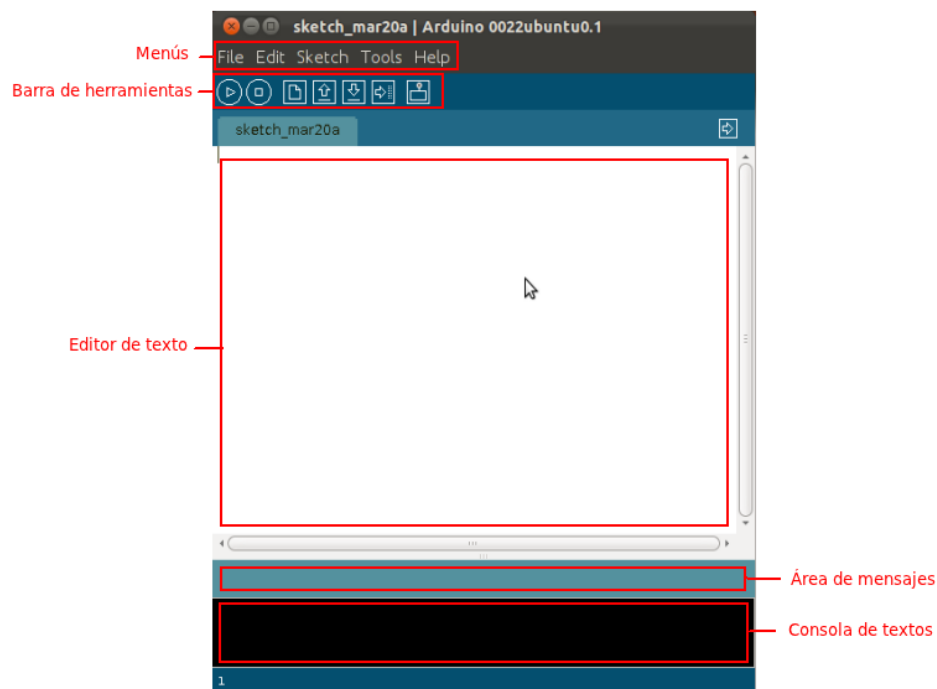





Figura 2.11. IDE para Arduino

2.7.3.1.1 Barra de herramientas

- 

Verify/Compile
Chequea el código en busca de errores.
- 

Stop
Finaliza la monitorización serie y oculta otros botones.
- 

New
Crea un nuevo sketch.



Open

Presenta un menú de todos los programas sketch “sketchbooks” (librería sketch).



Save

Guarda el programa sketch.



Upload to I/O Board

Compila el código y lo vuelca en la placa E/S Arduino.



Serial Monitor

Inicia la monitorización serial.

2.7.3.1.2 Menús

El entorno de Arduino contiene una serie de menús que son sensibles al contexto, esto quiere decir que sólo se habilitan de acuerdo a la acción que se esté realizando. Los menús son los siguientes:

File

New

Open

Sketchbook: Abre los sketch creados.

Examples: Abre los sketch que vienen de ejemplo al descargar el IDE para Arduino.

Close

Save

Save As

Upload to I/O Board: Compila el código y lo vuelca en la placa E/S Arduino, esta opción también se la encuentra en la barra de herramientas.

Page Setup

Print

Preferences: Permite modificar opciones del editor, sobre todo la ubicación para almacenar los sketch creados.

Quit

Edit

Undo

Redo

Cut

Copy

Copy for Forum: Copia al portapapeles el texto del sketch seleccionado, el texto posee el formato usado en los foros.

Copy as HTML: Copia al portapapeles el texto del sketch seleccionado, el texto tiene el formato HTML para páginas web.

Paste

Select All

Comment/Uncomment: Inicia y finaliza los comentarios en el sketch.

Increase Indent: Aumentar el guión – sangría.

Decrease Indent: Disminuir el guión - sangría

Find

Find Next

Sketch

Verify/Compile: Verifica los errores del programa (sketch).

Stop: Detiene la verificación de los errores.

Show Sketch Folder: Abre la carpeta de programas (sketch) en el escritorio.

Import Library: Añade una librería al programa, se incluye la sentencia `#include` en el código.

Add File: Añade un fichero fuente al programa que se esté realizando.

Tools

Auto Format: Da formato al código proporcionando estética, por ejemplo realiza tabulaciones entre la apertura y cierre de llaves, y las sentencias que tengan que ser tabuladas lo estarán.

Archive Sketch

Fix Encoding&Reload: En caso de que el sketch se lo realice con una versión antigua del Arduino/Processing y se quiera compilar en un IDE más actualizado esto da un conflicto y se presenta mensajes de error, esta opción permite las actualizaciones necesarias para que se pueda compilar el sketch en la placa E/S Arduino.

Serial Monitor: Inicia la monitorización serial, esta opción también se la encuentra en la barra de herramientas.

Board: Permite seleccionar la placa que se esté utilizando.

Serial Port: Contiene todos los dispositivos seriales (virtuales o reales).

Burn Bootlander: Permite grabar un gestor de arranque (bootloader) dentro del microcontrolador de la placa Arduino. Aunque no es un requisito para el normal funcionamiento de la placa Arduino, es útil si se compra un nuevo ATmega (el cual viene normalmente sin gestor de arranque). Asegurándose que se ha seleccionado la placa correcta en el menú Boards antes de grabar el bootloader.

Help: Permite revisar la información de Arduino, página web, preguntas frecuentes y otras opciones de ayuda; para lo cual se necesita conexión a internet.

La programación de la placa Arduino se la realiza a través del IDE para crear el sketch, el lenguaje usado es el Processing manteniendo la estructura de este.

2.7.4.1 Estructura

La estructura básica del lenguaje de programación de Arduino se compone de al menos dos partes. Estas dos partes necesarias, o funciones, encierran bloques que contienen instrucciones y son las siguientes:

⤴ **voidsetup()**

⤴ **voidloop()**

setup() *inicialización*

La función `setup()` se establece cuando se inicia un programa (sketch). Se emplea para iniciar variables, establecer el estado de los pines e inicializar las comunicaciones. Esta función se ejecutará una única vez después de que se conecte la placa Arduino a la fuente de alimentación, o cuando se pulse el botón de reinicio de la placa.

```
voidsetup()
{
  pinMode(pin, OUTPUT);    // ajusta a "pin" como salida
}
```

loop() *bucle*

Luego de crear la función `setup()`, la cual inicializa y prepara los valores iniciales, la función `loop()` hace justamente lo que su nombre sugiere, por lo tanto se ejecuta continuamente, permitiéndole al programa variar y responder, leyendo

entradas, activando salidas, etc. Esta función es el núcleo de todos los programas de Arduino y hace la mayor parte del trabajo.

```
void loop()
{
  digitalWrite(pin, HIGH); // activa "pin"
  delay(1000);             // espera 1000 milisegundos
  digitalWrite(pin, LOW); // desactiva "pin"
  delay(1000);             // espera 1000 milisegundos
}
```

2.7.4.2 Sintaxis

La sintaxis del lenguaje es muy parecida a la de C y C++, manteniendo las mismas estructuras, que son las siguientes:

- ⤴ Funciones
- ⤴ Llaves
- ⤴ Punto y coma
- ⤴ Bloques de comentarios
- ⤴ Comentarios de línea

2.7.4.2.1 Funciones

Una función es un bloque de código que tiene un nombre y un grupo de declaraciones que se ejecutan cuando se llama a la función. Se puede hacer uso de funciones integradas como **voidsetup()** y **voidloop()** o escribir nuevas.

Las funciones se escriben para ejecutar tareas repetitivas y reducir el desorden en un programa. En primer lugar se declara el tipo de la función, que será el valor retornado por la función (int, void...). A continuación del tipo, se declara el nombre de la función y, entre paréntesis, los parámetros que se pasan a la función.

```

tipo nombre_funcion(parametros)
{
  realizar esta acción;
}

```

La siguiente función `intretardo()`, asigna un valor de retardo en un programa por lectura del valor de un potenciómetro.

```

int retardo()
{
  int v; // crea una variable temporal "v"
  v = analogRead(pot); // lee el valor analógico de "pot" y lo asigna a "v"
  v /= 4; // convierte de 0-1023 a 0-255
  return v; // devuelve el valor final de "v"
}

```

2.7.4.2.2 Llaves {}

Las llaves definen el comienzo y el final de bloques de función y bloques de declaraciones como `voidloop()` y sentencias `for` e `if`. Las llaves deben estar balanceadas (a una llave de apertura `{` debe seguirle una llave de cierre `}`). Las llaves no balanceadas provocan errores de compilación.

```

voidloop()
{
  realizar esta acción;
}

```

El entorno Arduino incluye una práctica característica para chequear el balance de llaves. Sólo selecciona una llave y su compañera lógica aparecerá resaltada.

2.7.4.2.3 Punto y coma ;

Un punto y coma debe usarse al final de cada declaración y separa los elementos del programa. También se usa para separar los elementos en un bucle `for`.

```

int x = 13; //declara la variable "x" como el entero 13

```

Nota: Olvidar un punto y coma al final de una declaración producirá un error de compilación.

2.7.4.2.4 Bloques de comentarios /*...*/

Los bloques de comentarios, o comentarios multilínea, son áreas de texto ignoradas por el programa y se usan para grandes descripciones de código o comentarios que ayudan a otras personas a entender partes del programa. Empiezan con `/*` y terminan con `*/` y pueden abarcar múltiples líneas. Los comentarios son ignorados por el programa y no ocupan espacio en memoria.

```
/* Parpadeo de un led con intervalos de un segundo */
```

2.7.4.2.5 Comentarios de línea //

Comentarios de una línea empiezan con `//` y terminan con la siguiente línea de código. Como los bloques de comentarios son ignorados por el programa, no toman espacio en memoria. Comentarios de una línea se usan a menudo después de declaraciones válidas para proporcionar más información sobre qué lleva la declaración o proporcionar un recordatorio en el futuro.

```
int x = 13;           //declara la variable "x" como el entero 13
```

2.7.4.3 Variables

Una variable es una forma de llamar y almacenar un valor numérico para usarse después por el programa. Como su nombre indica, las variables son números que pueden cambiarse continuamente al contrario que las constantes, cuyo valor nunca cambia. Una variable necesita ser declarada y, opcionalmente, asignada al valor que necesita para ser almacenada.

```
int valor = 0;           // declara una variable y asigna el valor a 0
valor = analogRead(2); // ajusta la variable al valor del pin analógico 2
```

Una vez que una variable ha sido asignada, o reasignada, puedes testear su valor para ver si cumple ciertas condiciones, o puede usarse directamente.

```
if(valor < 100)        // comprueba si la variable es menor que 100
{
  valor = 100;         // si es cierto asigna el valor 100
}
delay(valor);         // usa la variable como retardo
```

2.7.4.3.1 Declaración de variable

Todas las variables tienen que ser declaradas antes de que puedan ser usadas. Declarar una variable significa definir su tipo de valor, como int, long, float, etc., definir un nombre específico, y, opcionalmente, asignar un valor inicial. Esto sólo necesita hacerse una vez en un programa pero el valor puede cambiarse en cualquier momento usando aritmética y varias asignaciones.

```
int valor = 0;           // declara una variable y asigna el valor a 0
```

Una variable puede ser declarada en un número de posiciones en todo el programa y donde esta definición tiene lugar determina que partes del programa pueden usar la variable.

2.7.4.3.2 Ámbito de la variable

Una variable puede ser declarada al comienzo del programa antes del `void setup()`, localmente dentro de funciones, y algunas veces en un bloque de declaración, por ejemplo bucles for. Donde la variable es declarada determina el ámbito de la variable, o la habilidad de ciertas partes de un programa de hacer uso de la variable.

Una variable global es aquella que puede ser vista y usada por cualquier función y declaración en un programa. Esta variable se declara al comienzo del programa, antes de la función `setup()`.

Una variable local es una que se define dentro de una función o como parte de un bucle `for`. Sólo es visible y sólo puede ser usada dentro de la función en la cual fue declarada. Además, es posible tener dos o más variables del mismo nombre en diferentes partes del programa que contienen diferentes valores.

```
int valor;                // 'valor' es visible por cualquier función
void setup() { }
void loop()
{
  for(int i=0; i<20;)      // 'i' es sólo visible dentro del bucle for
  {
    i++;
  }
  float f;                // 'f' es sólo visible dentro de loop
}
```

2.7.4.4 Tipos de datos

Byte

Byte almacena un valor numérico de 8 bits sin puntos decimales. Tienen un rango de 0 a 255.

```
byte x = 180;           // declara 'x' como un tipo byte y asigna el valor de 180
```

char

Es un tipo de dato que ocupa un byte de memoria y almacena un valor de carácter. Los caracteres literales se escriben con comillas simples: 'A' (para varios caracteres -strings- se utiliza dobles comillas "ABC").

De todas maneras los caracteres son almacenados como números. Se puede ver su codificado en la tabla ASCII. Con esto se puede entender que es posible realizar cálculos aritméticos con los caracteres, en este caso se utiliza el valor ASCII del carácter (por ejemplo 'A' + 1 tiene el valor de 66, ya que el valor ASCII de la letra mayúscula A es 65)

El tipo de datos char tiene signo, esto significa que codifica números desde -128 hasta 127. Para un dato de un byte (8 bits), utiliza el tipo de dato "byte".

```
char x = 'A';  
char y = 65;           // el valor de 'x' y 'y' equivalen a lo mismo
```

int

Enteros son los tipos de datos primarios para almacenamiento de números sin puntos decimales y almacenan un valor de 16 bits con un rango de -32,768 a 32,767.

```
int x = 1000; // declara 'x' como tipo int y asigna el valor de 1000
```

long

Tipo de datos de tamaño extendido para enteros largos, sin puntos decimales, almacenados en un valor de 32 bits con un rango de -2,146,483,648 a 2,147,483,647.

```
long x = 90000; // declara 'x' como tipo long y asigna el valor de 90000
```

float

Un tipo de datos para números en punto flotante, o números que tienen un punto decimal. Los números en punto flotante tienen mayor resolución que los enteros y

se almacenan como valor de 32 bits con un rango de $-3.4028235E+38$ a $3.4028235E+38$.

```
float x = 3.14; // declara 'x' como tipo float y asigna el valor de 3.14
```

arrays

Un array es una colección de valores que son accedidos con un índice numérico. Cualquier valor en el array debe llamarse escribiendo el nombre del array y el índice numérico del valor. Los arrays están registrados a cero, con el primer valor en el array comenzando con el índice número 0. Un array necesita ser declarado y opcionalmente asignarle valores antes de que puedan ser usados.

```
intmatriz[] = {value0, value1, value2...};
```

Asimismo es posible declarar un array declarando el tipo y el tamaño y luego asignarle valores a una posición del índice.

```
int matriz[5]; //declara un array de enteros con 6 posiciones  
matriz[3] = 10; //asigna a la cuarta posición del índice el valor 10
```

Para recibir un valor desde un array, asignamos una variable al array y la posición del índice:

```
x = myArray[3]; // x ahora es igual a 10
```

Constantes

El lenguaje Arduino tiene unos cuantos valores predefinidos que se llaman constantes. Se usan para hacer los programas más legibles. Las constantes se clasifican en grupos.

- ⤴ TRUE / FALSE
- ⤴ HIGH / LOW
- ⤴ INPUT / OUTPUT

True / False

Estas son constantes booleanas que definen niveles lógicos. FALSE se define como 0 (cero) mientras TRUE es 1 o un valor distinto de 0.

```
if(b == TRUE)          // si 'b' es verdadero
{
digitalWrite(9, HIGH) // envía un valor HIGH al pin 9;
}
```

HIGH / LOW

Estas constantes definen los niveles de pin como HIGH o LOW y se usan cuando se leen o se escriben los pines digitales. HIGH está definido como el nivel 1 lógico, ON ó 5 V, mientras que LOW es el nivel lógico 0, OFF ó 0 V.

```
digitalWrite(13, HIGH); // envía un valor HIGH al pin 13
```

INPUT / OUTPUT

Constantes usadas con la función pinMode() para definir el modo de un pin digital como INPUT u OUTPUT.

```
pinMode(13, OUTPUT); // define el pin 13 como salida
```

2.7.4.4.1 Conversión de datos

Permite cambiar un valor dado a otro de un tipo de dato específico.

char(x), byte(x), int(x), long(x), float(x)

La variable “ x ” es el valor a convertir.

```
int i;           // declara 'i' como variable int
float f;        // declara 'f' como variable float
```

```
f = 3.14;           // asigna el valor de 3.14 a 'f'
i = int(f);        // convierte a 'f' en entero y se asigna a 'i', i es igual a 3
```

2.7.4.5 Operadores Booleanos

Los operadores booleanos o lógicos son normalmente una forma de comparar dos expresiones y devuelven TRUE o FALSE dependiendo del operador. Hay tres operadores lógicos, AND, OR y NOT, que se usan a menudo en declaraciones if.

AND lógico (&&):

```
if (x>0 && x<5)      //verdadero sólo si las dos expresiones son ciertas
```

OR lógico (||):

```
if (x>0 || y>0)     //verdadero si al menos una expresión es cierta
```

NOT lógico (!):

```
if (!(x>0))         //verdadero sólo si la expresión es falsa
```

2.7.4.6 Control de Flujo

if

Las sentencias if comprueban si cierta condición ha sido alcanzada y ejecutan todas las sentencias dentro de las llaves si la declaración es cierta. Si es falsa el programa ignora la sentencia.

```
if (x < 2)          // si 'x' es menor a 2, realizar la siguiente acción
{
    x++;            // suma el valor de 1 a 'x'
}
```

if - else

if - else permite tomar decisiones entre una opción u otra.

```

if (pin == HIGH)      // si 'pin' es HIGH, realizar la siguiente acción
{
digitalWrite(9, HIGH) // envía un valor HIGH al pin 9;
}
else                  // caso contrario, realizar la siguiente acción
{
digitalWrite(9, LOW)  // envía un valor LOW al pin 9;
}

```

else puede preceder a otra comprobación if, por lo que múltiples y mutuas comprobaciones exclusivas pueden ejecutarse al mismo tiempo.

```

if (x < 2)            // si 'x' es menor a 2, realizar la siguiente acción
{
digitalWrite(9, HIGH) // envía un valor HIGH al pin 9;
}
elseif (x >= 5)      // en el caso que 'x' sea mayor o igual a 5, haga
{
digitalWrite(10, HIGH) // envía un valor HIGH al pin 10;
}
else                  // caso contrario, realizar la siguiente acción
{
digitalWrite(9, LOW)   // envía un valor LOW al pin 9;
digitalWrite(10, LOW) // envía un valor LOW al pin 10;
}

```

for

La sentencia for se usa para repetir un bloque de declaraciones encerradas en llaves un número específico de veces. Un contador de incremento se usa a menudo para incrementar y terminar el bucle. Hay tres partes separadas por punto y coma (;), en la cabecera del bucle.

```

for (inicialización; condición; expresión)
{
// realizar esta acción
}

```

La inicialización de una variable local, o contador de incremento, sucede primero y una sola una vez. Cada vez que pasa el bucle, la condición siguiente es comprobada. Si la condición devuelve TRUE, las declaraciones y expresiones que

siguen se ejecutan y la condición se comprueba de nuevo. Cuando la condición se vuelve FALSE, el bucle termina.

```
for (int i = 0; i < 20; i++) // declara i, comprueba si es menor
{ // que 20, incrementa i en 1
digitalWrite(13, HIGH); // activa el pin 13
delay(250); // pausa por 250 milisegundos
digitalWrite(13, LOW); // desactiva el pin 13
delay(250); // pausa por 250 milisegundos
```

while

El bucle while se repetirá continuamente, e infinitamente, hasta que la expresión dentro del paréntesis se vuelva falsa. Algo debe cambiar la variable comprobada o el bucle while nunca saldrá. Lo que modifique la variable puede estar en el código, como una variable que se incrementa, o ser una condición externa, como el valor que da un sensor.

```
while (x < 1000) // comprueba si es menor que 1000
{
digitalWrite(9, HIGH); // activa el pin 9
delay(x); // espera el tiempo que de la variable x
digitalWrite(9, LOW); // desactiva el pin 9
x++; // incrementa la variable en 1
}
```

do . . while

El bucle do . . while es un bucle que trabaja de la misma forma que el bucle while, con la excepción de que la condición es testada al final del bucle, por lo que el bucle do . . while siempre se ejecutará al menos una vez.

```
do
{
x = analogRead(A0); //asigna el valor de analogRead(A0) a 'x'
delay(50); // pausa de 50 milisegundos
}
while(x < 100); // repite si 'x' es menor que 100
```

break

Es usado para salir de los bucles **do**, **for**, o **while**, pasando por alto la condición normal del bucle.

```
for (int i = 0; i < 20; i++)    // declara 'i', comprueba si es menor
{                               // que 20, incrementa 'i' en 1
  digitalWrite(13, HIGH);      // activa el pin 13
  delay(250);                  // pausa por 250 milisegundos
  digitalWrite(13, LOW);       // desactiva el pin 13
  delay(250);                  // pausa por 250 milisegundos
  if (i == 10)                 // si 'i' = 10
  {
    i = 0;                     // asigna a 'i' = 0
    break;                     // sale del bucle for
  }
}
```


CAPITULO 3

DISEÑO E IMPLEMENTACIÓN

3.1 Introducción

Para el diseño e implementación del sistema de control de ruta vehicular mediante comparación con el uso de GPS empleando la plataforma Arduino UNO se tomó en cuenta los siguientes parámetros:

- Un medio para capturar la posición del vehículo.
- La utilización de un elemento electrónico que permita almacenar los datos de la ubicación del vehículo.
- Investigar y/o desarrollar un dispositivo que cumpla con la función de controlar la posición de un vehículo y posteriormente almacenarlo.
- Realizar experimentos reales a través de pruebas con monitoreo de ruta específico.
- Establecer el sistema para que sea portátil con el uso de baterías, se emplee con un cargador o que sea posible observar directamente los datos obtenidos en un computador.
- Visualizar posteriormente los datos obtenidos por el GPS LOCARD, almacenados en la MicroSD como documento de Excel.

3.2 Representación del comportamiento

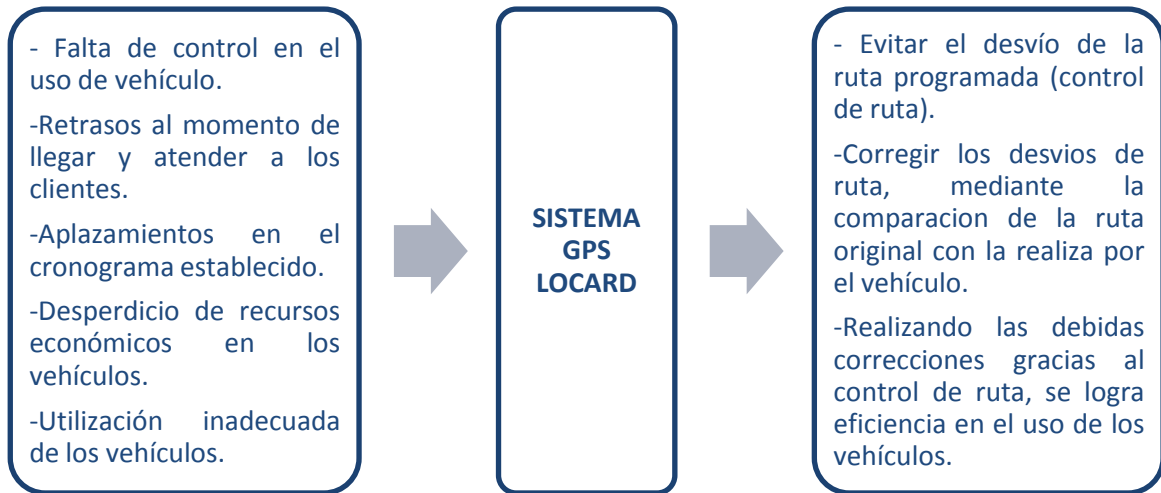


Fig. 3.1 Funcionamiento de Sistema GPS LOCARD

Basándose en los parámetros y necesidades mencionadas en la introducción, el sistema de control de ruta GPS LOCARD incluye las siguientes etapas:

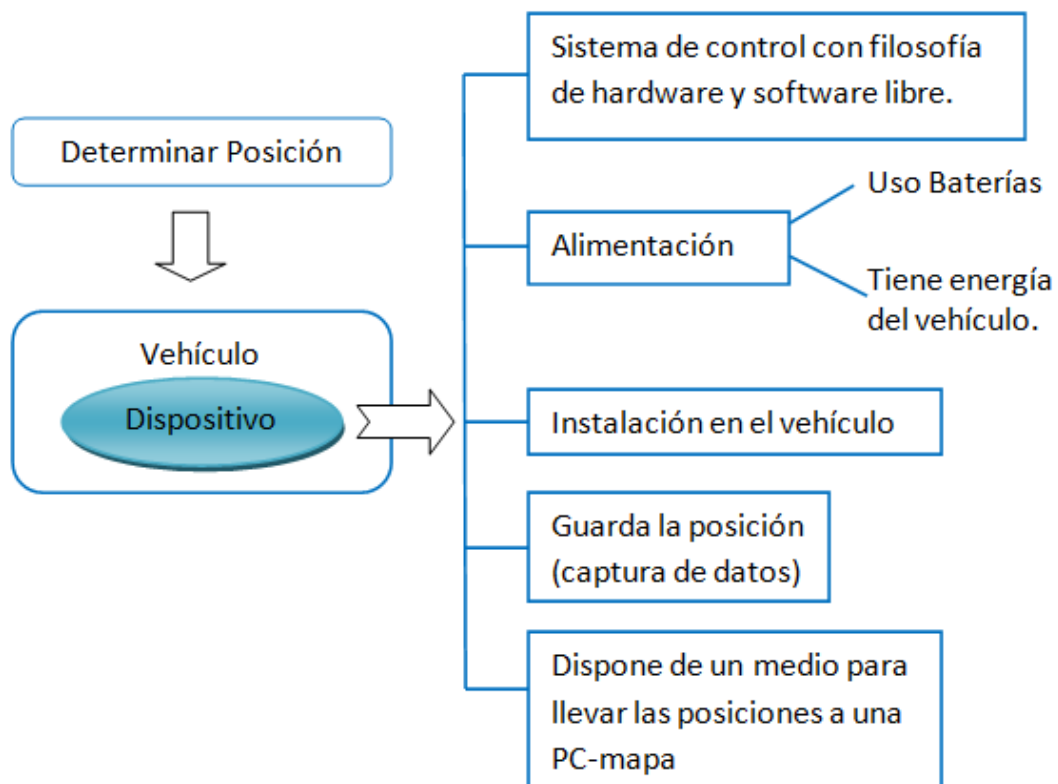


Fig. 3.2 Diagrama General Sistema GPS LOCARD

En la figura 3.2 se puede observar cómo se desarrolla el sistema GPS LOCARD, viendo las necesidades de los usuarios, debido a que es preciso tener un control adecuado de los vehículos, así se analiza las diversas posibilidades del dispositivo como por ejemplo modo de alimentación, análisis de software y hardware a ser empleado, buscando una manera adecuada de almacenar los datos, etc.

Todas estas opciones se toman en cuenta para a posteriormente ir descartando y llegar a una solución óptima para el sistema GPS LOCARD.

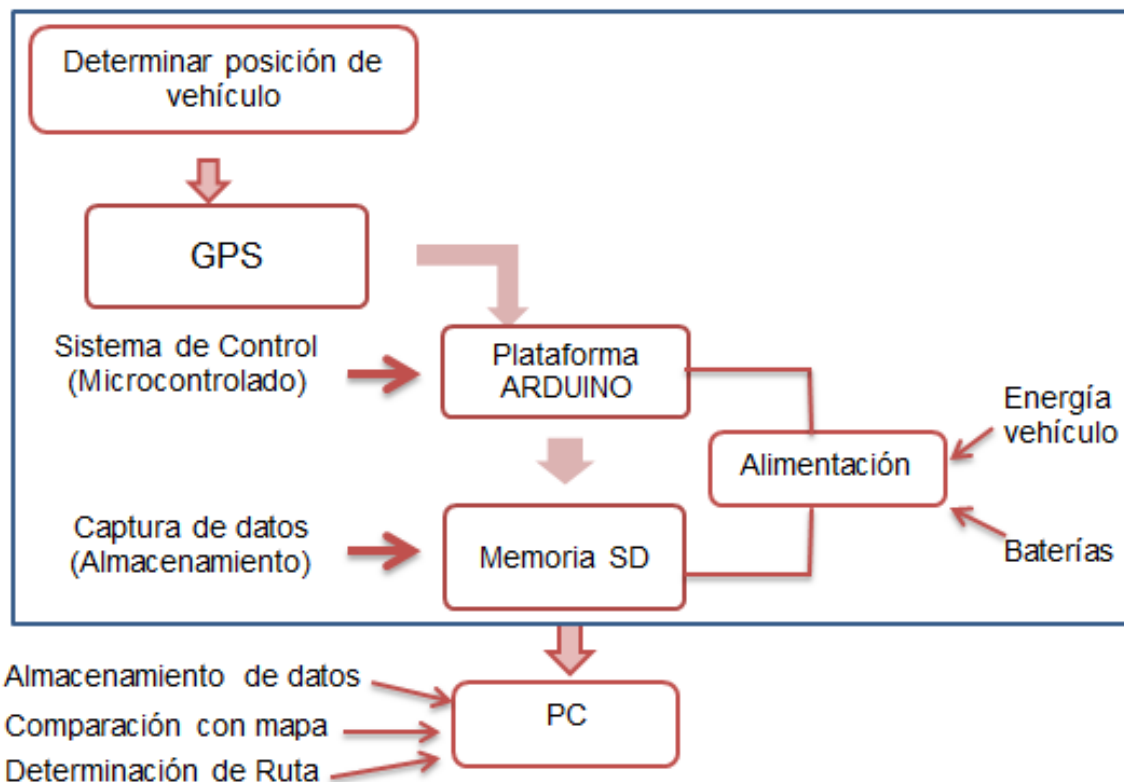


Fig. 3.3 Diagrama de Etapas y Funciones Sistema GPS LOCARD

Una vez analizadas todas las posibilidades de los elementos a ser empleados se escoge la mas adecuada para cada etapa, así se escoge para determinar la posición de vehiculos es necesario un GPS en este caso el modelo EM-406A, para el sistema de control se emplea software y hardware Arduino UNO.

Para almacenar los datos y posteriormente visualizarlos en el computador se opta por una microSD, en el computador se podrá tener un respaldo de todos los datos recogidos por la microSD, una comparación con la ruta establecida y posteriormente determinar una ruta optima.

Se toma en cuenta que el dispositivo debe ser portatil y su alimentacion debe ser constante es por ello que se establecen dos maneras de alimentarlo ya sea por baterias o por un cargador para vehiculos.



Fig. 3.4 Diagrama de Bloques Sistema GPS LOCARD

Como se observa en el diagrama de bloques el Sistema Control de Ruta mediante GPS consta de cuatro etapas que se detallan a continuación:

- **Etapa de Alimentación:** Está conformada por un conjunto de 4 pilas +AA recargables cada una de 1,2 voltios con 2500mAh de corriente para proporcionar el voltaje adecuado que en este caso sería aproximadamente 5 Voltios. Otra forma de proporcionar el voltaje y corriente requeridos es mediante un cargador para vehículo el cual toma el voltaje del plug de la cigarrera y lo transforma de 12 voltios a los 5 voltios requeridos por el sistema GPS LOCARD.

Adicionalmente si se desea se puede alimentar al dispositivo utilizando el puerto USB tipo B proporcionado en la placa Arduino Uno y que lo conecta al computador.

- **Sistema Microcontrolado:** Primero graba el programa correspondiente al monitoreo del GPS, luego para almacenar la información en la MicroSD es necesario cargar otro archivo en la misma con sus respectivas librerías en Arduino UNO. Una vez hecho esto el sistema analiza las señales captadas por el GPS mediante transmisión y recepción entre las 3 diferentes placas existentes pasando a guardar en la MicroSD los datos recopilados.
- **Dispositivo de Recepción:** Recibe la información enviada por los satélites de la red GPS y a continuación envía los datos al Sistema Microcontrolado.
- **Dispositivo de Almacenamiento:** Esta etapa al recibir los datos del Sistema Microcontrolado procede a guardar la información en un archivo de Excel de extensión .CV compatible para ser ejecutado con la aplicación Google Earth.
- **Etapa de Comparación:** Una vez cargado los datos en la MicroSD se procede a descargar los archivos en un computador para visualizar los datos obtenidos, creando la ruta en Google Earth.



Fig. 3.5 Diagrama de etapas con elementos utilizados en sistema GPS LOCARD

Para poder describir muy claramente el funcionamiento y el diseño del sistema GPS LOCARD se iniciara por el estudio del Sistema Microcontrolado.

3.2.1 SISTEMA MICROCONTROLADO

Viendo la necesidad de trabajar en software y hardware libre se procede a realizar una búsqueda entre todos los posibles sistemas microcontrolados existentes en el mercado, encontrar el adecuado para que cumpla con las necesidades mencionadas en la introducción, con este fin se toma en cuenta la opción de Arduino.

Dentro de Arduino existen diversos diseños siendo el de última tecnología el Arduino UNO, el mismo que ha sido mejorado para ser compatible con mayor cantidad de shields, y del cual se dispone más códigos y soluciones.

Arduino UNO cuenta con 14 pines entradas/salidas digitales, 6 entradas analógicas, un oscilador de cristal de 16 MHz, una conexión USB, un conector de alimentación, una cabecera ICSP, y un botón de reset. Contiene todo lo necesario para apoyar al microcontrolador ATmega 328P-PU el mismo que se encuentra programado como un convertidor de USB a serie.

Se describe gráficamente cada una de las partes mencionadas anteriormente:

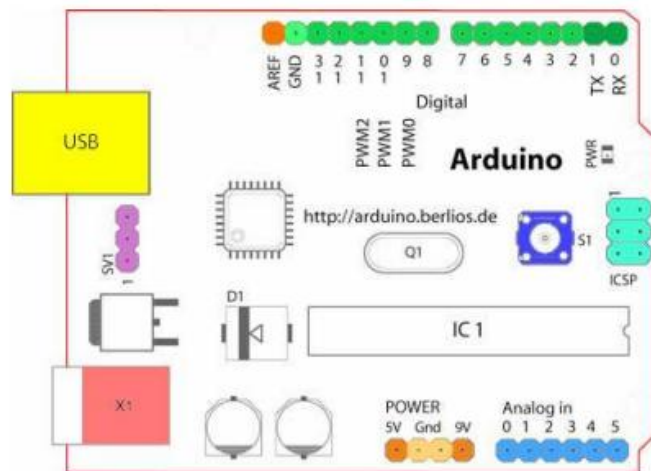


Fig. 3.6 Descripción grafica de distribución de elementos en placa Arduino UNO

Aref- Pin de referencia analógica (naranja)

GND- Señal de tierra digital (verde claro)

Pines digitales 2-13. Entrada y salida (verde)

Pines digitales 0-1 / entrada y salida del puerto serie: TX/RX (verde oscuro)

Botón de reset- Pulsador (azul oscuro)

Pines de entrada analógica 0-5 (azul claro)

Pines de alimentación y tierra (fuerza: naranja, tierra: naranja claro)

Entrada de la fuente de alimentación externa (9-12V DC) X1 (rosa)

Puerto USB (amarillo)

Para empezar a utilizar Arduino UNO es necesario instalar el programa en un computador tomando en cuenta el sistema operativo con el que cuenta, es posible descargar el programa de la página oficial de Arduino www.arduino.cc, donde existen códigos, librerías, hardware existentes, entre otras.

En este caso se escogió el software Arduino 0023 el cual proporciona una interfaz y compatibilidad con Arduino UNO, cabe aclarar que este software no es el más actual para Arduino UNO pero para los fines a ser empleado es el adecuado.

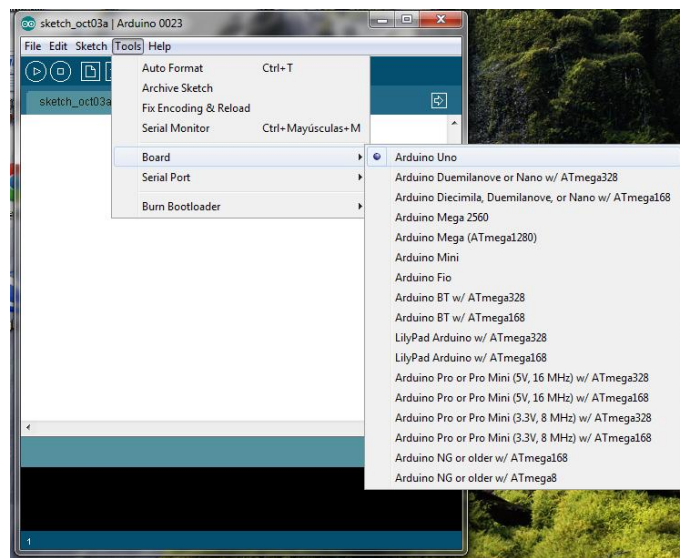


Fig. 3.7 Programa para Arduino UNO

Una vez instalado el programa se procede a conectar la placa Arduino Uno al computador para que sea asignado un puerto de comunicación (COM) se puede comprobar en el menú del programa como se muestra en la figura.

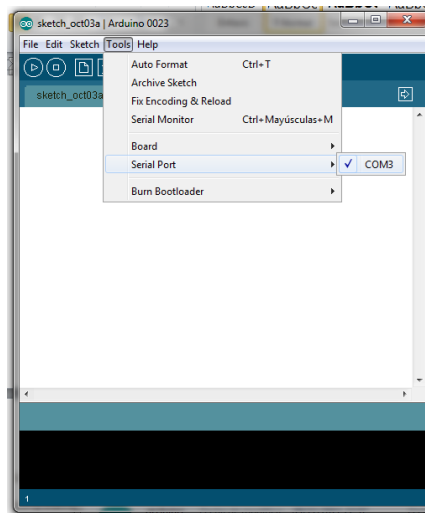


Fig. 3.8 Asignación de puerto Serial en Arduino UNO

Este proceso es indispensable puesto que si no reconoce el puerto serial se tendrá problemas el momento de grabar los programas en el Arduino.

Realizados estos pasos se procede a la programación en Arduino siendo el primer programa el siguiente con su detalle:

Este programa analiza las sentencias NMEA desde un GPS modelo EM-406A funcionando a 4800 bps en lectura valores de latitud, longitud, altitud, fecha, hora, y velocidad.

Para la Shield GPS modelo EM-406A se debe conectar en primera instancia a los puertos 2 y 3 para tomar en cuenta los datos captados por el GPS y que estos sean visualizados en el computador.

Una vez que obtenga su longitud y latitud se puede pegar las coordenadas de la ventana de terminal en Google Earth.

Para ejecutar correctamente este programa es necesario descargar y agregar al programa la librería NewSoftSerial para la comunicación serial con el GPS, para así conectar los pines TX y RX del GPS a cualquier pin digital en la placa Arduino

UNO, sólo es necesario asegurarse de definir los pines que se utiliza en el Arduino para la comunicación con el módulo GPS, adicionalmente es indispensable cargar la librería TinyGPS para que ejecute el programa como se muestra en la figura

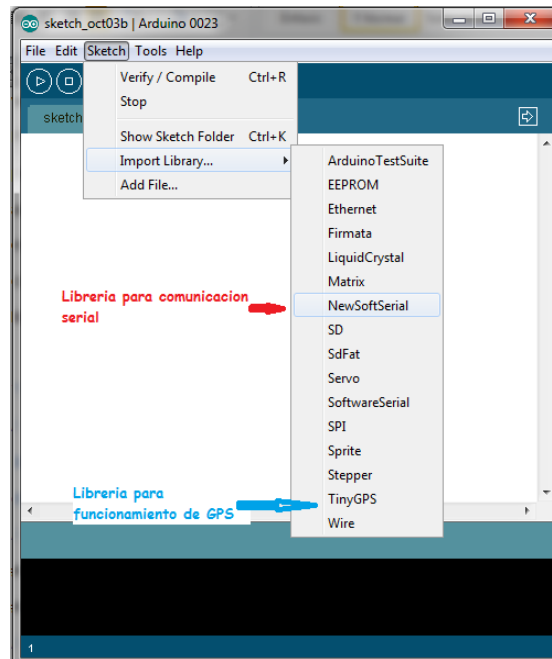


Fig. 3.9 Librerías necesarias para ejecutar el programa

```

*/
// Para que este esquema funcione, se debe descargar NewSoftSerial y la librería
TinyGPS de arduiniana.org y colocar en el hardware-> carpeta directorio de librerías de
Arduino UNO.
// Estas son las líneas de código que apuntan a esas bibliotecas.
#include <NewSoftSerial.h>
#include <TinyGPS.h>
// Define qué pines se van a utilizar en el Arduino para comunicarse con el GPS. En este
caso, el pin TX del módulo GPS se conectará a la RXPIN Arduino que es el pin 3.
#define RXPIN 2
#define TXPIN 3
#define GPSBAUD 4800 // Establezca este valor igual a la velocidad de transmisión del
GPS
TinyGPSSgps; // Crea una instancia del objeto TinyGPS
NewSoftSerialuart_gps (RXPIN, TXPIN); // Inicializa la biblioteca NewSoftSerial a los
pines que se ha definido anteriormente

getgpsvoid (TinyGPS y GPS); // Declaración de prototipos para las funciones que serán
usadas de la librería TinyGPS

```

```

// En la función de configuración, es necesario inicializar dos puertos serie, el puerto
serie de hardware estándar (Serial ()) para que tenga comunicación con el terminal del
programa a otro puerto serie (NewSoftSerial ()) para el GPS.
voidsetup ()
{
Serial.begin (115200); // Este es el tipo de serie terminal del programa. Este es rápido
porque se debe imprimir todo antes de un nuevo registro de datos. Si entra más
despacio, los datos no pueden ser válidos y probablemente se producirán errores de
suma de comprobación.
uart_gps.begin (GPSBAUD); // Establece la velocidad en baudios del GPS
Serial.println ("");
Serial.println ("CONTROL DE RUTA GPS LOCARD");
Serial.println ("... en espera de datos ...");
Serial.println ("");
}
// Este es el bucle principal del código. Todo lo que hace es comprobar si hay datos
sobre el pin RX del Arduino, se asegura de que los datos sean sentencias válidas de
NMEA, caso contrario salta a la función getgps ().
voidloop ()
{
mientras que (uart_gps.available ()) // Si hay datos en el pin RX ...
{
int c = uart_gps.read (); // cargar los datos en una variable ...
if (gps.encode (c)) // si hay una sentencia válida nueva ...
{
getgps (gps) // a continuación, capturar los datos.
}
}
}
getgpsvoid (TinyGPS y gps) // La función getgps se encarga de recibir e
imprimir los valores deseados.
{
// Para obtener todos los datos en variables que se puede utilizar en el código, es
necesario definir las variables y consultar el objeto de los datos.
latitud flotador, longitud; // Se define las variables que se utilizarán
gps.f_get_position (y latitud, y longitud); // A continuación, llamar a esta función
// Ahora se puede imprimir las variables de latitud y longitud
Serial.print ("Lat / Long:");
Serial.print (latitud, 5);
Serial.print (",");
Serial.println (longitud, 5);
int año; // Llama a la función fecha y la hora
byte mes, día, hora, minuto, segundo, centésimas;
gps.crack_datetime (y año, y mes, y día y hora, los minutos y, y en segundo lugar, y
centésimas);

Serial.print ("Fecha:"); Serial.print (mes, DEC); Serial.print ("/"); // Datos de impresión y
tiempo

```

```

Serial.print (día, DEC); Serial.print ("/"); Serial.print (año);
Serial.print ("Tiempo:"); Serial.print (hora, DEC); Serial.print (":");
Serial.print (minuto, DEC); Serial.print (":"); Serial.print (segundo, DEC);
Serial.print ("."); Serial.println (centésimas, DEC);
Serial.print ("Altitud (metros:"); Serial.println (gps.f_altitude ()); // Se imprimen los
valores de altitud y rumbo directamente
Serial.print ("Curso (grados:"); Serial.println (gps.f_course ()); // Se imprimen los
valores del curso.

Serial.print ("Speed(kmph:"); Serial.println (gps.f_speed_kmph ()); // Se imprimen los
valores de la velocidad
Serial.println ();
gps.stats caracteres (&, &, frases y failed_checksum);
}

```

Este programa despliega en primera instancia los datos captados por el GPS sin la opción de guardar los datos simplemente siendo visualizados en el computador directamente conectado el sistema GPS LOCARD mostrando lo siguiente:

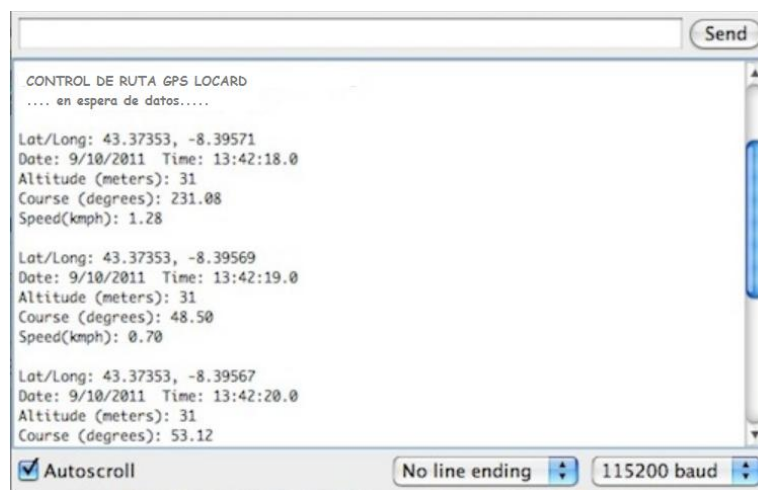


Fig. 3.10 Despliegue de datos con GPS LOCARD conectado al computador.

Esto se mostrará siempre y cuando esté conectado el modulo GPS al Arduino como se observa en la figura:



Fig. 3.11 Modo de conexión placa Arduino UNO con modulo GPS EM-406A

Posteriormente como es necesario almacenar los datos en una MicroSD es preciso cargar una nueva librería al programa que en este caso sería SdFat la que permite ejecutar el siguiente programa:

Esto es un registrador genérico que hace pruebas de comprobación para que los datos escritos sean buenos.

// Asume un registrador de chipset SiRF III adjunta al pin 0 y 1

sensorCount uint8_t = 1 // numero de Sensores analógicos a loguearse

// Librerias Para La SD

Include<SdFat.h>

Include<SdFatUtil.h>

Include<avr/pgmspace.h>

Define isdigit (x) (x> = '0' && x <= '9')

Tarjeta Sd2Card;

SdVolume volumen;

Raíz SdFile;

SdFile f;

Define BUFFSIZE 73 // Cada Vez se mete en el búfer Una Sentencia NMEA, 73bytes es más grande de la longitud máxima

carbón buffer [BUFFSIZE];

charlas buffer2 [12];

uint8_t bufferidx = 0;

tmp uint32_t;

// CONFIGURACION DEL MODULO GPS //

Define LOG_RMC 1 // Datos de localización Esenciales RMC

Define RMC_ON "\$PSRF103,4,0,1,1 * 21 \r\n" // Activa RMC (1 Hz)

```

# Define RMC_OFF "$ PSRF103, 4,0,0,1 * 20 \r\n" // Desactiva RMC
# Define LOG_GGA 0 // contiene fijos, HDOP y VDOP datos
# Define GGA_ON "$ PSRF103, 0,0,1,1 * 25 \r\n" // Activa GGA (1 Hz)
# Define GGA_OFF "$ PSRF103, 0,0,0,1 * 24 \r\n" // Desactiva GGA
# Define LOG_GSA 0 / data / satélite
# Define GSA_ON "$ PSRF103, 2,0,1,1 * 27 \r\n" // Activa GSA (1 Hz)
# Define GSA_OFF "$ PSRF103, 2,0,0,1 * 26 \r\n" // Desactiva GSA
# Define LOG_GSV 0 datos de satélites // detalladas
# Define GSV_ON "$ PSRF103, 3,0,1,1 * 26 \r\n" // Activa GSV (1 Hz)
# Define GSV_OFF "$ PSRF103, 3,0,0,1 * 27 \r\n" // Desactiva GSV
# Define LOG_GLL 0 // Compatibilidad
# Define USE_WAAS 1 // Útil para conseguir más precisión, pero más lento y consume
más batería.
# Define WAAS_ON "$ PSRF151, 1 * 3F \r\n" // Activa WAAS
# Define WAAS_OFF "$ PSRF151, 0 * 3E \r\n" // Desactiva WAAS
# Define LOG_RMC_FIXONLY 1 // Captura solo la posición fijada en el RMC
fix uint8_t = 0; // dato real de fijación de posición.
// Macros para utilizar PSTR
# Define putstring (str) SerialPrint_P (PSTR (str))
# Define putstring_nl (str) SerialPrintln_P (PSTR (str))
// Lee sin valor hexadecimal y devuelve Su valor decimal
uint8_t parseHex (char c) {
    if (c <'0 ') return 0;
    if (c <= '9 ') return c - '0';
    if (c <'A') return 0;
    if (c <= 'F') return (c - 'A') +10;
}
}
voidsetup ()
{
    Serial.begin (4800), // Frecuencia que Funciona el GPS
    pinMode (10, OUTPUT); // Esencial para que funcione la shield microSD!
    putstring_nl ("CONTROL DE RUTA GPS LOCARD");
    pinMode (led1Pin, OUTPUT); // establece el pin digital como salida
    pinMode (13, OUTPUT);
    if (! card.init ()) {
        putstring_nl ("La inicialización de la microSD ha fallado!");
    }
}

```

```

    error (1);
}
if (! volume.init (tarjeta)) {
    putstring_nl ("No existe partición en la tarjeta");
    error (2);
}
if (! root.openRoot (volumen)) {
    putstring_nl ("No Se Puede abrir el Directorio raíz");
    error (3);
}
strcpy (buffer, "GPSLOG00.CSV");           // empezara a almacenar un ARCHIVO desde
00, si ya existe este Archivo previamente incrementa la cuenta.
for (i = 0; i <100; i ++ ) {
    buffer [6] = '0 ' + I/10;
    buffer [7] = '0 ' + % i 10;
    if (f.open (raíz, tampón, O_CREAT | O_EXCL | O_WRITE)) break;
}
if (! f.isOpen ()) {
    putstring ("No ha podido crear el archivo"); Serial.println (buffer);
    error (3);
}
putstring ("ESCRIBIENDO en"); Serial.println (buffer);
putstring_nl ("Preparado");
// Escritura de la cabecera
si (sensorCount > 6) sensorCount = 6;
strncpy_P (buffer, PSTR ("tiempo, lat, long, velocidad, fecha, SENS0, sens1, sens2,
sens3, sens4, sens5"), 24 + 6 * sensorCount);
Serial.println (buffer);
// Claro error de impresión
f.writeError = 0;
f.println (buffer);
if (f.writeError || ! f.sync ()) {
    putstring_nl ("No Se Pudo ESCRIBIR la cabecera");
    error (5);
}
delay (1000);                               // Tiempo en el que se almacena en la microSD
// Se hace la CONFIGURACION DEL GPS

```

```

    putstring ("\r\n");
# Si USE_WAAS == 1
    putstring (WAAS_ON) // en WAAS
# Else
    putstring (WAAS_OFF) // en WAAS
# Endif
# Si LOG_RMC == 1
    putstring (RMC_ON) // en RMC
# Else
    putstring (RMC_OFF) // apagado RMC
# Endif
# Si LOG_GSV == 1
    putstring (GSV_ON) // en GSV
# Else
    putstring (GSV_OFF) // apagado GSV
# Endif
# Si LOG_GSA == 1
    putstring (GSA_ON) // en GSA
# Else
    putstring (GSA_OFF) // de GSA
# Endif
# Si LOG_GGA == 1
    putstring (GGA_ON) // en GGA
# Else
    putstring (GGA_OFF) // apagado GGA
# Endif
}
voidloop ()
{
    char c;
    suma uint8_t;
    // Lee Una linea NMEA del GPS
    if (Serial.available ()) {
        c = Serial.read ();
        // Serial.print (c, BYTE);
        if (bufferidx == 0) {
            while (c! = '$')

```



```

    c = Serial.read (); // Espera Que llegue sin nn $
}
tampón [bufferidx] = c;
// Serial.print (c, BYTE);
if (c == '\n') {
    // Putstring_nl ("EOL");
    // Serial.print (buffer);
    buffer [bufferidx +1] = 0; // terminación
    if (buffer [bufferidx-4] != '*') {
        Serial.print ('*', BYTE);
        bufferidx = 0;
        volver;
    }
    // Cálculo de suma de comprobación
    = suma parseHex (buffer [bufferidx-3]) * 16;
    suma + = parseHex (buffer [bufferidx-2]);
    // Comprueba la suma de comprobación
    for (i = 1; i <(bufferidx-4); i + +) {
        suma ^ = buffer [i];
    }
    if (suma != 0) {
        // Putstring_nl ("error de suma de comprobación");
        Serial.print ('~', BYTE);
        bufferidx = 0;
        volver;
    }
    // Datos obtenidos
    // Serial.println (buffer);
    if (strstr (buffer, "GPRMC")) {
        // Buscamos si tenemos la posición fijada
        char * p = buffer;
        p = strchr (p, ',') +1;
        p = strchr (p, ',') +1; // Vamos a El Tercer dato
        if (p [0] == 'V') {
            digitalWrite (led1Pin, LOW); // no tenemos fijada la posición
            fix = 0;
        }
    }
}

```

```

else {
    digitalWrite (led1Pin, HIGH); // tenemos fijada la posición
    fix = 1;
}
}

# Si LOG_RMC_FIXONLY
if (! fix) {
    Serial.print ('_', BYTE) // Esta en modo espera de datos y configurado para que
solo coja Datos con la posición fijada
    bufferidx = 0;
    volver;
}

# Endif
Serial.println ();
Serial.print ("Secuencia NMEA recibida:");
Serial.print (buffer);
// Buscando los Datos
// Encuentra El Tiempo
char * p = buffer;
p = strchr (p, ',') +1;
buffer [0] = p [0];
buffer [1] = p [1];
buffer [2] = ':';
buffer [3] = p [2];
buffer [4] = p [3];
buffer [5] = ':';
buffer [6] = p [4];
buffer [7] = p [5];
buffer [8] = ','; // Ignoramos milisegundos
p = strchr (buffer 8, ',') +1;
p = strchr (p, ',') +1;
p = strchr (p, ',') +1; // Encuentra Latitud
buffer [9] = '+';
tampón [10] = p [0];
buffer [11] = p [1];
buffer [12] = ";";
strncpy (tampón 13, p +2, 7);

```

```

buffer [20] = ',';

p = strchr (buffer 21, ',') +1;
if (p [0] == 'S')
    buffer [9] = '-';
p = strchr (p, ',') +1;    // Encuentra Longitud
buffer [21] = '+';
buffer [22] = p [0];
buffer [23] = p [1];
buffer [24] = p [2];
buffer [25] = ";
strncpy (tampón 26, p 3, 7);
buffer [33] = ',';
p = strchr (buffer 34, ',') +1;
if (p [0] == 'W')
    buffer [21] = '-';    // Encuentra Velocidad
p = strchr (p, ',') +1;
tmp = 0;
if (p [0] != ',') {
    // Convertimos la VELOCIDAD
    while (p [0] != '.' && p [0] != ',') {
        tmp * = 10;
        tmp + = p [0] - '0 ';
        p + +;
    }
    tmp * = 10;
    if (isdigit (p [1]))
        tmp + = p [1] - '0 ';
    tmp * = 10;
    if (isdigit (p [2]))
        tmp + = p [2] - '0 ';
    * tmp = 185, // la convertimos en kmh
}
tmp / = 100;
buffer [34] = (tmp / 10000) + '0 ';
tmp% = 10000;
buffer [35] = (tmp / 1000) + '0 ';

```

```

tmp% = 1000,
buffer [36] = (tmp / 100) + '0 ';
tmp% = 100;
buffer [37] = '.';
buffer [38] = (tmp / 10) + '0 ';
tmp% = 10;
buffer [39] = tmp + '0 ';
buffer [40] = ',';
p = strchr (p, ',') +1;
// Saltar rodamiento pasado
p = strchr (p, ',') +1;
// Modo para Evitar Problemas when Falta algún dato
fecha uint8_t [6];
por (id uint8_t = 0; id <6; id + +) fecha [id] = p [id];
// Formatea la FECHA asi 01.31.2001
buffer [41] = '2 ';
buffer [42] = '0 ';
buffer [43] = fecha [4];
buffer [44] = fecha [5];
buffer [45] = '-';
buffer [46] = date [2];
buffer [47] = fecha [3];
buffer [48] = '-';
buffer [49] = date [0];
buffer [50] = date [1];
tampón [51] = 0;
si (f.write ((uint8_t *) tampón, 51)! = 51) {
    putstring_nl ("no ha podido arreglar sí ESCRIBIR!");
volver;
}
Serial.print ("Datos Escritos en la SD:");
Serial.print (buffer);
f.writeError = 0;
/// AQUÍ SE AÑADE LA INFORMACION DEL GPS
(uint8_t ia = 0; ia<sensorCount; ia + +) {
    Serial.print (','); // escribimos Por serie
    f.print (','); // escribimos en el Archivo

```

```

uint16_t data = analogRead (ia);
Serial.print (datos) // escribimos Por serie
f.print (datos); // escribimos en el Archivo
}
Serial.println ();
f.println ();
if (f.writeError || f.sync ()) {
    putstring_nl ("no ha podido ESCRIBIR sí el dato");
    error (4);
}
bufferidx = 0;
volver;
}
bufferidx + +;
si (== bufferidx BUFFSIZE-1) {
    Serial.print (, BYTE '!');
    bufferidx = 0;
}
}
}
}

```

Cargado este programa al modulo Arduino UNO podrá ir almacenando los datos recibidos por el GPS en la MicroSD, pudiendo visualizar en el modulo mediante alertas con leds si se está escribiendo en la tarjeta, si está transmitiendo el modulo, o si tiene algún fallo.

3.2.2 DISPOSITIVO DE RECEPCIÓN GPS

Para determinar el dispositivo GPS se realiza una búsqueda exhaustiva entre los diversos equipos existentes en el mercado, viendo las diversas necesidades se determinó el GPS modelo EM-406A debido a que se acopla perfectamente a la plataforma Arduino UNO, este modelo es el único que consta de antena integrada fácil de manipular, pequeño, potente y el más actual existente en el mercado.

Este GPS captura los datos cada segundo, se comunica por puerto serie USB, se alimenta con 5 V y es compatible con el sistema WAAS que permite incluso mayor precisión.

Los datos obtenidos en este caso por el GPS se explican a continuación:

\$GPRMC,044235.000,A,4322.0289,N,00824.5210,W,0.39,65.46,020912,,,A*44

Empiezan por "\$" y acaban en un "A*" seguido de dos números, éste es el checksum para poder saber si el dato recibido es correcto. Los datos se separan por comas y el primero es el tipo de transmisión, en este caso el GPRMC (o RMC), uno de los más usados. Todo el protocolo lo enlazo en la sección de descargas. A continuación se revisan los datos para tener un mejor entendimiento:

044235.000 es la hora GMT (04:42:35)

A es la indicación de que el dato de posición está fijado y es correcto. V sería no válido

4322.0289 es la longitud (43° 22.0289´)

N Norte

00824.5210 es la latitud (8° 24.5210´)

W Oeste

0.39 velocidad en nudos

65.46 orientación en grados

020912 fecha (2 de septiembre del 2012)

```

COM3
Secuencia NMEA recibida: $GPRMC,031535.000,A,0007.1923,S,07828.8717,W,0.55,S,67.031012,,,A*67
Datos escritos en la SD: 03:15:35,-00 07.1923,-078 28.8717,001.01,2012-10-03,302
Secuencia NMEA recibida: $GPRMC,031536.000,A,0007.1921,S,07828.8718,W,0.68,S,67.031012,,,A*6E
Datos escritos en la SD: 03:15:36,-00 07.1921,-078 28.8718,001.25,2012-10-03,303
Secuencia NMEA recibida: $GPRMC,031537.000,A,0007.1925,S,07828.8716,W,1.94,S,67.031012,,,A*6E
Datos escritos en la SD: 03:15:37,-00 07.1925,-078 28.8716,003.58,2012-10-03,303
Secuencia NMEA recibida: $GPRMC,031538.000,A,0007.1926,S,07828.8715,W,1.97,S,67.031012,,,A*6E
Datos escritos en la SD: 03:15:38,-00 07.1926,-078 28.8715,002.53,2012-10-03,300
Secuencia NMEA recibida: $GPRMC,031539.000,A,0007.1930,S,07828.8713,W,2.00,S,67.031012,,,A*6F
Datos escritos en la SD: 03:15:39,-00 07.1930,-078 28.8713,003.70,2012-10-03,303
Secuencia NMEA recibida: $GPRMC,031540.000,A,0007.1933,S,07828.8710,W,1.87,S,67.031012,,,A*69
Datos escritos en la SD: 03:15:40,-00 07.1933,-078 28.8710,003.45,2012-10-03,293
Secuencia NMEA recibida: $GPRMC,031541.000,A,0007.1936,S,07828.8706,W,1.40,S,67.031012,,,A*61
Datos escritos en la SD: 03:15:41,-00 07.1936,-078 28.8706,002.59,2012-10-03,301
Secuencia NMEA recibida: $GPRMC,031542.000,A,0007.1935,S,07828.8702,W,1.07,S,67.031012,,,A*6A
Datos escritos en la SD: 03:15:42,-00 07.1935,-078 28.8702,001.97,2012-10-03,317
Secuencia NMEA recibida: $GPRMC,031543.000,A,0007.1930,S,07828.8700,W,1.67,S,67.031012,,,A*52
Datos escritos en la SD: 03:15:43,-00 07.1930,-078 28.8700,003.08,2012-10-03,300
Secuencia NMEA recibida: $GPRMC,031544.000,A,0007.1924,S,07828.8699,W,1.76,S,67.031012,,,A*55
Datos escritos en la SD: 03:15:44,-00 07.1924,-078 28.8699,003.25,2012-10-03,303
Secuencia NMEA recibida: $GPRMC,031545.000,A,0007.1915,S,07828.8700,W,1.93,S,67.031012,,,A*60
Datos escritos en la SD: 03:15:45,-00 07.1915,-078 28.8700,003.57,2012-10-03,303
Secuencia NMEA recibida: $GPRMC,031546.000,A,0007.1908,S,07828.8703,W,2.40,S,67.031012,,,A*62
Datos escritos en la SD: 03:15:46,-00 07.1908,-078 28.8703,004.81,2012-10-03,304
Secuencia NMEA recibida: $GPRMC,031547.000,A,0007.1904,S,07828.8708,W,1.08,S,67.031012,,,A*60
Datos escritos en la SD: 03:15:47,-00 07.1904,-078 28.8708,001.99,2012-10-03,303
Autoscroll
No line ending
4800 baud

```

Fig. 3.12 Datos captados por el GPS

Para poder utilizar adecuadamente el GPS modelo EM406A es necesario revisar la hoja de datos como se observa en la figura:

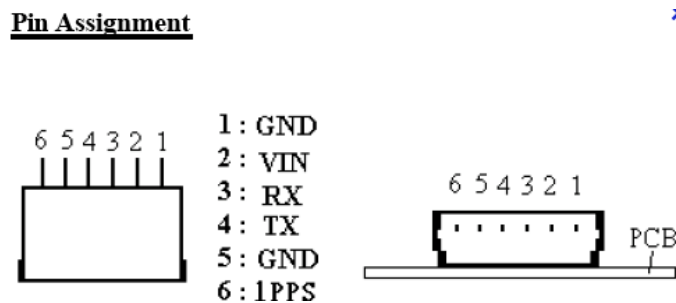


Fig. 3.13 Distribución de pines GPS modelo EM-406A

Cada uno de estos pines debe enlazarse con el modulo Arduino UNO para que tenga una comunicación adecuada, en primera instancia se conecta el pin de transmisión y recepción del GPS a los pines digitales 2 y 3 del modulo Arduino UNO los mismo que recibirán los datos captados por el GPS.

El GPS consta de un led indicador que define si está o no captando señal de satélite, cuando el led se encuentre parpadeando quiere decir que encontró la señal adecuada para transmitir, esta señal será enviada al modulo Arduino UNO

para que analice y determine si es válida o no, pudiendo almacenar los datos posteriormente en la microSD.



Fig. 3.14 Modulo GPS modelo EM-406A

3.2.3 Dispositivo de Almacenamiento MicroSD

En esta parte se selecciona la manera más adecuada para almacenar los datos para que estos posteriormente sean visualizados y monitoreados en un computador.

Se coloca la microSD en la shield adaptada para Arduino UNO como se muestra en la figura 3.15

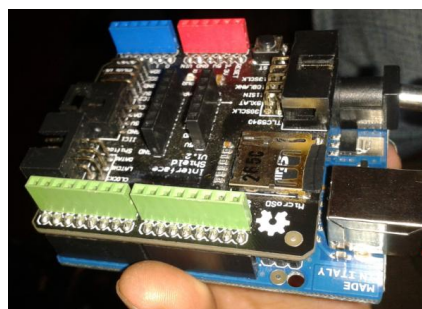


Fig. 3.15 Modulo Shield MicroSD para Arduino

Una vez colocada esta shield los datos recibidos por el GPS irán siendo almacenados en la MicroSD debido al programa previamente cargado en la placa Arduino UNO el mismo que selecciona si los datos guardados son o no validos.

Los datos almacenados en la microSD se van guardando en archivo de Excel con extensión .CV el mismo que es uno de las extensiones más compactas.

En el caso que se pierda comunicación o se presione el botón de reset tanto de la placa de Arduino como de la Shield MicroSD los datos capturados se almacenarán en otro archivo .CV variando los nombres de cada uno de los archivos.

Este botón podrá ser pulsado siempre y cuando se desee tener una nueva lectura desde una nueva posición creando así una nueva ruta.

3.2.4 Etapa de Comparación

En esta etapa ya es posible visualizar la ruta recorrida por el vehículo mediante la introducción de la microSD al computador por medio de un adaptador para microSD como se muestra en la figura.

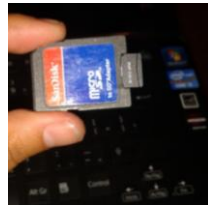


Fig. 3.16 Adaptador para MicroSD

El momento de abrir la MicroSD en el computador se podrá observar todos los archivos allí almacenados como por ejemplo los siguientes:

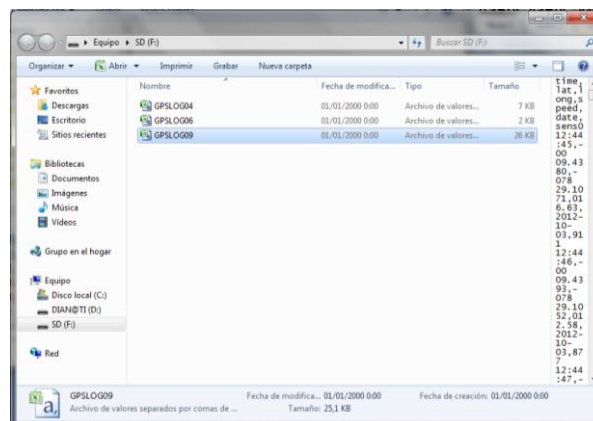


Fig. 3.17 Archivos .CV guardados en la MicroSD

Se podrá ver datos como tiempo, longitud, latitud, velocidad y altura desplegados de la siguiente manera:

	A	B	C	D	E	F
1	time,lat,long,speed,date,sens0					
2	12:44:45,-00 09.4380,-078 29.1071,016.63,2012-10-03,911					
3	12:44:46,-00 09.4393,-078 29.1052,012.58,2012-10-03,877					
4	12:44:47,-00 09.4402,-078 29.1037,009.43,2012-10-03,872					
5	12:44:48,-00 09.4411,-078 29.1022,008.51,2012-10-03,891					
6	12:44:49,-00 09.4420,-078 29.1009,009.95,2012-10-03,896					
7	12:44:50,-00 09.4428,-078 29.0997,010.95,2012-10-03,891					
8	12:44:51,-00 09.4438,-078 29.0990,010.28,2012-10-03,885					
9	12:44:52,-00 09.4452,-078 29.0987,011.21,2012-10-03,879					
10	12:44:53,-00 09.4463,-078 29.0982,009.37,2012-10-03,797					
11	12:44:54,-00 09.4476,-078 29.0987,007.95,2012-10-03,912					
12	12:44:55,-00 09.4485,-078 29.0992,006.19,2012-10-03,918					
13	12:44:56,-00 09.4492,-078 29.0993,005.95,2012-10-03,913					
14	12:44:57,-00 09.4500,-078 29.0994,004.95,2012-10-03,917					
15	12:44:58,-00 09.4505,-078 29.0994,003.57,2012-10-03,915					
16	12:44:59,-00 09.4507,-078 29.0989,002.70,2012-10-03,911					
17	12:45:00,-00 09.4512,-078 29.0991,003.10,2012-10-03,909					
18	12:45:01,-00 09.4515,-078 29.0992,002.47,2012-10-03,909					
19	12:45:02,-00 09.4527,-078 29.0983,006.17,2012-10-03,909					
20	12:45:03,-00 09.4531,-078 29.0979,004.69,2012-10-03,909					
21	12:45:04,-00 09.4536,-078 29.0975,003.57,2012-10-03,907					
22	12:45:05,-00 09.4538,-078 29.0970,003.60,2012-10-03,909					
23	12:45:06,-00 09.4540,-078 29.0963,004.68,2012-10-03,909					
24	12:45:07,-00 09.4535,-078 29.0961,002.90,2012-10-03,908					
25	12:45:08,-00 09.4524,-078 29.0964,001.07,2012-10-03,909					

Fig. 3.18 Datos en archivo .CV

Una vez revisados los datos se procede a cargar el archivo para ser visualizada la ruta en el programa Google Earth ingresando a la siguiente página

http://www.gpsvisualizer.com/map_input?form=googleearth debiendo ser configurada de la manera siguiente:

Fig. 3.19 Configuración para despliegue de datos en Google Earth

Es importante en la pestaña Desc. Template introducir la **Latitud:** {lat}⁰ **Longitud:** {long}⁰ **Velocidad:**{speed}Km/h **Tiempo:** {Time} **Elevación:**{altitude}m **Sens0:** {sens0}. Esto permite en cada punto tener todos los datos. El nombre entre {} es la variable de cada columna de el archivo. Para las pruebas realizadas no es necesario saber la altura ya que se está sobre el suelo, GPS visualizer saca el dato {altitude} de sus bases de datos para cada posición.

Posteriormente se carga el archivo de la microSD a la página en la parte "Create KML file", se descarga y se lo abre con el Google Earth obteniendo lo siguiente:

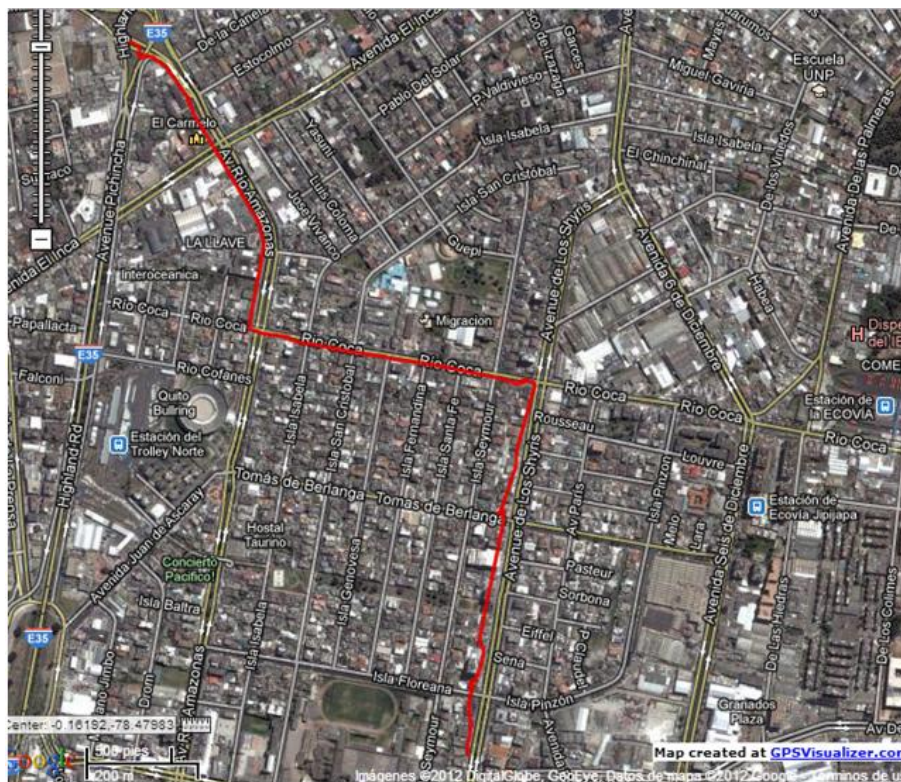


Fig. 3.20 Visualización en Google Earth realizada por el vehículo

Pudiendo así observar los datos captados por el vehículo, pudiendo comparar con la ruta establecida primordialmente.

3.2.5 Etapa de Alimentación

Esta etapa es de usuario, en ésta el usuario decide como desea suministrar energía al dispositivo GPS LOCARD debido a que se puede conectar mediante las siguientes opciones:

Comunicación serial USB 5V

Es el pin de salida de voltaje estabilizado de 5 V, que es proporcionado por el Vin directamente a través de la USB.

Cargador de 5V para vehículos.

El cual transforma la energía del compartimiento de la cigarrera a través de un cargador vehicular proporcionando 5 V fijos al dispositivo para que este funcione.

Baterías



Fig. 3.21 Baterías recargables

Estas baterías en este caso son recargables y tienen una duración de 10 horas aproximadamente pudiendo ser estas una manera fácil de transportar el dispositivo.

Una vez seleccionado alguna de estas opciones es posible trabajar con el dispositivo GPS LOCARD.

CAPITULO 4

ANÁLISIS FINANCIERO

4.1. Matriz FODA

La matriz FODA refleja los puntos fuertes y débiles del proyecto TTP, en la tabla

4.1 se observa detalladamente.

Fortalezas	Debilidades
<ul style="list-style-type: none"> • El sistema GPS LOCARD ayuda a controlar las rutas de vehículos. • Mejora en tiempos de ruta debido a posibles correcciones de desvió de ruta en los vehículos. • Fácil uso y accesibilidad a los clientes. • El GPS LOCARD es de sencillo manejo y bajo costo, sus placas son de open hardware. • Se puede configurar a través software libre. • Es una tecnología nueva que garantiza escalabilidad para futuro. • Es el primer Sistema de control de ruta por GPS basado en software y hardware libre empleado en el Ecuador. 	<ul style="list-style-type: none"> • Ya existen varios modelos de GPS en el mercado con algunas otras prestaciones, ingresar el producto en el mercado puede ser complicado. • Poca accesibilidad a elementos necesarios para su construcción.
Oportunidades	Amenazas
<ul style="list-style-type: none"> • Personas interesadas en tener un mejor control de sus vehículos adquiriendo un GPS LOCARD. • En el Ecuador no existe un proyecto de estas características es decir un control de ruta con GPS en software y hardware libre. • Ya que funciona con Arduino este se sigue innovando y se tiene actualizaciones constantes en software y hardware. 	<ul style="list-style-type: none"> • El sistema de control de ruta mediante GPS puede ser susceptible a ser copiados en su diseño, por no haber restricciones en este sentido. • Debido a que funcionan mediante la captación de señales de satélite éstos pueden sufrir algún desperfecto ocasional y dejar de emitir señales útiles para el GPS.

Tabla 4.1. Matriz FODA

4.2. Análisis de costo - beneficio

Debido a que se trata de hardware y software libre el análisis de costo beneficio se limita al costo generado en adquisición de materiales, puesto que la idea de software y hardware libre es que se puede adquirir gratuitamente por lo que la venta del sistema GPS LOCARD estaría en el armado, programado, realizar soporte al mismo, tiempo empleado y su depreciación.

Se detalla a continuación la lista de materiales empleados en la construcción del sistema GPS LOCARD:

Ítem	Descripción	Cant.	V. Unit(USD)	V. Total
Placas Arduino				
1	Modulo entrenador Arduino UNO	1	39	39,00
2	GPS modelo EM-406	1	59,94	59,94
3	Shield Micro SD para Arduino UNO	1	22,50	22,50
Materiales Empleados				
4	Resistencias 330 ohm 1/4w	2	0,02	0,04
5	Pulsador encendido/apagado	1	0,11	0,11
6	Porta Pilas 4 AA	1	0,36	0,36
7	Plug Fuente	1	0,22	0,22
8	Cable rojo/negro #16	1	0,31	0,31
9	Pilas Recargables	2	9,66	19,32
10	Cargador con pilas recargables	1	18,92	18,92
11	Cargador de 5 V para Arduino UNO	1	10,00	10,00
12	Micro SD de 1 Gb	1	20,00	20,00
13	Baquelita de 10X10	1	2,10	2,10
14	Lamina de papel de transferencia	1	2,80	2,80
15	Funda de cloruro férrico	1	0,50	0,50
16	Broca	1	0,75	0,75
17	Importación y costos de envío	1	20,22	20,22
18	Caja de Acrílico	1	8,50	8,50
			TOTAL	225,59

Tabla 4.2. Costo de materiales utilizados en el módulo de entrenamiento

Los materiales pueden variar dependiendo de la necesidad del usuario ya que en el caso de las pilas recargables podrían ser remplazadas por un cargador conectado al vehículo.

Actividad	Detalle	Costo (USD)
Adquisición de materiales	El costo total de los materiales utilizados para realizar un sistema de control de ruta por GPS se observa en la tabla 4.3, con su respectivo detalle	225,59
Investigación	Se investigó los elementos electrónicos a ser incluidos en el módulo, de acuerdo a los carentes en el laboratorio de la U. Israel y que puedan dar funcionalidad a la Arduino.	40
	TOTAL:	265,59

Tabla 4.3. Costo de la implementación del sistema de control de Ruta GPS LOCARD

Como se observa en las tablas anteriores, el valor total del módulo es de 265,73 dólares americanos, la ganancia en el sistema GPS LOCARD se encuentra en el armado, programado y dar soporte al mismo, el cual dependerá del cliente que va a adquirir el sistema.

En el mercado no existen módulos de Arduino, así que no se tiene una referencia del precio con el cual evaluar el porcentaje del costo y obtener algún parámetro financiero, debido a que el equipo fue traído de los Estados Unidos.

CAPITULO 5

CONCLUSIONES Y RECOMENDACIONES

5.1 CONCLUSIONES

- Luego de realizar varias pruebas se logró implementar el sistema GPS LOCARD el cual cumple a cabalidad un monitoreo de ruta vehicular mediante comparación con una ruta establecida.
- El momento de programar en Arduino UNO el código empleado es relativamente fácil de emplear además que es de acceso libre no fue necesario realizar mayores modificaciones a los mismos.
- El modulo GPS modelo EM-406 es el encargado de captar la posición vehicular este posiblemente tarde unos segundos hasta ubicar los satélites adecuados los mismo que le darán la posición exacta del vehículo, la configuración de éste cambia si se conecta la shield de almacenamiento MicroSD debido a que deberá estar sincronizado el receptor del GPS con el transmisor de la MicroSD y viceversa.
- Se tuvo algunos inconvenientes el momento de guardar los datos en la MicroSD debido a que no había correcta comunicación entre la placa de Arduino y de GPS fue necesario modificar la conexión de la placa de GPS para que guarde los datos en la MicroSD
- Debido a que en el país no se cuenta con todos los materiales fue necesario importar la mayoría de elementos para realizar el sistema GPS LOCARD a futuro se pretende tener mayor accesibilidad en el país.

5.2 RECOMENDACIONES

- Tener siempre cuidado al instalar o configurar cualquier dispositivo electrónico con la alimentación de las mismas
- El momento de programar en Arduino UNO es necesario tomar en cuenta los puertos de comunicación a ser empleados, debido a que si no reconoce el puerto serial será necesario buscar los instaladores necesarios para cargar el programa en la placa de Arduino UNO.
- Verificar siempre las polarizaciones y modos de conexión del modulo GPS debido a que si se conecta de manera equivocada no captara las señales de satélite adecuadas dando datos erróneos.
- Tomar en cuenta si se trabaja en el monitor Serial de Arduino UNO para visualizar los datos en tiempo real es necesario ver la configuración que se muestra en el manual de usuario.
- El momento de cargar los programas en la placa de Arduino es imprescindible seguir cada uno de los pasos sugeridos en el manual debido a que si no se lo realiza de esta manera surgen errores el momento de almacenar o captar los datos obtenidos por el GPS.
- Verificar que no existan sueldas frías en los circuitos impresos.

REFERENCIAS DE PÁGINAS WEB

1. <http://dspace.ups.edu.ec/handle/123456789/70>
2. http://www.robosafe.com/personal/mocana/papers/waf08_fernando.pdf
3. http://oa.upm.es/2957/1/INVE_MEM_2008_60672.pdf
4. <http://www.clubdelamar.org/sistemagps.htm>
5. <http://es.wikipedia.org/wiki/Microcontrolador>
6. http://es.wikipedia.org/wiki/Universal_Serial_Bus
7. [http://es.wikipedia.org/wiki/Universal_Serial_Bus#Velocidades_de_transmisi
i.C3.B3n](http://es.wikipedia.org/wiki/Universal_Serial_Bus#Velocidades_de_transmisi%C3%B3n)
8. <http://www.gnu.org/philosophy/free-sw.es.html> Autor: Hernán Giovagnoli
9. http://es.wikipedia.org/wiki/Hardware_libre Autor: Antonio Delgado
10. <http://www.wiring.org.co>
11. <http://www.processing.org>
12. <http://arduino.cc/en/Main/ArduinoBoardUno>
13. <http://arduino.cc/es/Main/ArduinoBoardDuemilanove>
14. <http://www.bricogeek.com/shop/237-arduino-gps-shield.html>
15. <http://www.gpsinformation.org/dale/nmea.htm>
16. http://es.wikipedia.org/wiki/Field_Programmable_Gate_Array

GLOSARIO DE TERMINOS

GPS: Sistema de Posicionamiento Global

Shield: Son placas impresas que se pueden conectar en la parte superior de la placa Arduino para ampliar sus capacidades, pudiendo ser apiladas una encima de otra, son fáciles de montar, y barato de producir.

GSM: Sistema Global para Comunicaciones Móviles

SMS: Servicio de mensajes cortos

WIFI: Es un mecanismo de conexión de dispositivos electrónicos de forma inalámbrica.

RAM: Memoria de acceso aleatorio (en inglés: random-access memory), se utiliza como memoria de trabajo para el sistema operativo, los programas y la mayoría del software. Es allí donde se cargan todas las instrucciones que ejecutan el procesador y otras unidades de cómputo.

Waypoint: Punto de paso o punto de referencia es la posición de un único lugar sobre la superficie de la tierra expresada por sus coordenadas.

CPU: Unidad Central de Procesamiento es el componente principal del ordenador y otros dispositivos programables, que interpreta las instrucciones contenidas en los programas y procesa los datos.

DSP: Procesador Digital de Señal es un sistema basado en un procesador o microprocesador que posee un conjunto de instrucciones, un hardware y un software optimizados para aplicaciones que requieran operaciones numéricas a muy alta velocidad.

ROM: Memoria de Solo Lectura es un medio de almacenamiento utilizado en ordenadores y dispositivos electrónicos, que permite sólo la lectura de la

información y no su escritura, independientemente de la presencia o no de una fuente de energía.

EEPROM: Es un tipo de memoria ROM que puede ser programada, borrada y reprogramada eléctricamente. Son memorias no volátiles.

USB: Bus Universal en serie es un estándar industrial desarrollado en los años 1990 que define los cables, conectores y protocolos usados en un bus para conectar, comunicar y proveer de alimentación eléctrica entre ordenadores, periféricos y dispositivos electrónicos.

PCI: Interconexión de Componentes Periféricos es un bus de ordenador estándar para conectar dispositivos periféricos directamente a su placa base.

Plug and Play: se refiere a la capacidad de un sistema informático de configurar automáticamente los dispositivos al conectarlos. Permite poder enchufar un dispositivo y utilizarlo inmediatamente, sin preocuparte de la configuración.

Half Duplex: semidúplex, significa que el método o protocolo de envío de información es bidireccional pero no simultáneo.

Full Duplex: es un término utilizado en las telecomunicaciones para definir a un sistema que es capaz de mantener una comunicación bidireccional, enviando y recibiendo mensajes de forma simultánea.

GND: Tierra o polo negativo

Host: El término es usado en informática para referirse a las computadoras conectadas a una Red, que proveen y utilizan servicios de ella.

Bits: Un bit es un dígito del sistema de numeración binario. El bit es la unidad mínima de información empleada en informática, en cualquier dispositivo digital, o en la teoría de la información.

Bytes: es una secuencia de bits contiguos, cuyo tamaño depende del código de información o código de caracteres en que sea definido.

NRZI: No retorno a cero invertido es una forma de codificar una señal binaria en una señal digital para transmitirla por un medio. Las señales NRZI de dos niveles tienen una transición si el bit que se está transfiriendo es un uno lógico y no lo tienen si lo que se transmite es un cero.

HDL: Hardware description language, es un lenguaje definido por el IEEE (Institute of Electrical and Electronics Engineers) usado por ingenieros para describir circuitos digitales.

SoC: También referido como system-on-chip, describe la tendencia cada vez más frecuente de usar tecnologías de fabricación que integran todos o gran parte de los módulos componentes de un ordenador o cualquier otro sistema informático o electrónico en un único circuito integrado o chip.

FPGA: Field Programmable Gate Array es un dispositivo semiconductor que contiene bloques de lógica cuya interconexión y funcionalidad puede ser configurada 'in situ' mediante un lenguaje de descripción especializado.

ASIC: Circuito integrado de Aplicación Especifica es un circuito integrado hecho a la medida para un uso en particular, en vez de ser concebido para propósitos de uso general. Se usan para una función específica.

CAD: Ficheros de diseño de referencia.

Atmel: es una compañía de semiconductores, fundada en 1984. Su línea de productos incluye microcontroladores, dispositivos de radio frecuencia, memorias EEPROM y Flash, ASICs, WiMAX, entre otras.

FTDI: Future Technology Devices International, empresa especializada en la tecnología Universal Serial Bus (USB). Desarrolla, fabrica y soporta dispositivos y sus controladores de software relacionadas para la conversión de RS-232 o transmisiones TTL serie a señales USB, con el fin de permitir la compatibilidad con dispositivos heredados con las computadoras modernas.

RX: En la Placa GPS EM-406A es el principal canal para recibir comandos de software.

TX: En GPS EM-406A es el canal de transmisión principal para la salida de datos de navegación y medición para usuario.

PPS: Terminal de GPS modelo EM-406A este proporciona un pulso por segundo de salida de la placa Arduino que está sincronizada con el tiempo de GPS.

UART: Transmisor-Receptor Asíncrono Universal controla los puertos y dispositivos serie. Se encuentra integrado en la placa base o en la tarjeta adaptadora del dispositivo.

RESET: Se conoce como reset a la puesta en condiciones iniciales de un sistema. Este puede ser mecánico, electrónico o de otro tipo. Normalmente se realiza al conectar el mismo, aunque, habitualmente, existe un mecanismo, normalmente un pulsador, que sirve para realizar la puesta en condiciones iniciales manualmente.

Backup: Es la copia total o parcial de información importante del disco duro, CDs, bases de datos u otro medio de almacenamiento.

MicroSD: Las tarjetas microSD o Transflash corresponden a un formato de tarjeta de memoria flash más pequeña que la MiniSD, desarrollada por SanDisk.

Data-Logging: Es un dispositivo electrónico que registra datos en el tiempo o en relación con la ubicación ya sea con un sistema incorporado en el instrumento o sensor o por medio de instrumentos y sensores externos.

SPI: Serial Peripheral Interface, es un bus estándar de comunicaciones.

PWM: Modulación de Ancho de Pulso de una señal o fuente de energía es una técnica en la que se modifica el ciclo de trabajo de una señal periódica (una senoidal o una cuadrada, por ejemplo), ya sea para transmitir información a través de un canal de comunicaciones o para controlar la cantidad de energía que se envía a una carga.

ICSP: Programación Serial en Circuito es un método de programación en RAV , Propeller de Parallax, y microcontroladores PIC.

Bootloader: Gestor de arranque es un programa que se encarga de dejar todo listo para que comience la ejecución del sistema operativo.

Pull-up: Desconectada por defecto.

TTL: Lógica transistor a transistor es una tecnología de construcción de circuitos electrónicos digitales. En los componentes fabricados con tecnología TTL los elementos de entrada y salida del dispositivo son transistores bipolares.

ASCII: Es el Código Estándar Estadounidense para el Intercambio de Información, es un código de caracteres basado en el alfabeto latino, tal como se usa en inglés moderno y en otras lenguas occidentales.

COM: Puerto de comunicación son adaptadores que se utilizan para enviar y recibir información de BIT en BIT fuera del computador a través de un único cable y de un determinado software de comunicación.

NMEA: es una especificación combinada eléctrica y de datos entre aparatos electrónicos marinos y, generalmente, receptores GPS.

SiFR: compañía que fue pionera en el uso comercial de GPS tecnología para aplicaciones de consumo SiRF, ha fabricado una gama de patentados GPS chipsets y software para los aparatos y sistemas de navegación.

RMC: datos mínimos recomendados para gps.

GGA: Información FIX la sentencia GGA muestra un ejemplo que proporciona datos esenciales fijos.

GSA: Datos Globales por Satélite.

GSV: Son los Datos Detallados por Satélite.

GLL: Datos de Latitud y Longitud.

WAAS: Es un Sistema de Aumento Basado en Satélites desarrollado por Estados Unidos. Está ideado como un complemento para la red GPS para proporcionar una mayor exactitud y seguridad en las señales, permitiendo una precisión en la posición menor de dos metros.

PSTR: Comando que permite almacenar los datos del GPS en la memoria Flash de la microSD.

ANEXO 1

DATASHEET EM-406A



環天衛星科技股份有限公司

PRODUCT USER MANUAL

GPS RECEIVER ENGINE BOARD

EM-406A

Features:

SiRF star III high performance GPS Chip Set

Very high sensitivity (Tracking Sensitivity: -159 dBm)

Extremely fast TTFF (Time To First Fix) at low signal level

Support NMEA 0183 data protocol

Built-in SuperCap to reserve system data for rapid satellite acquisition

Built-in patch antenna

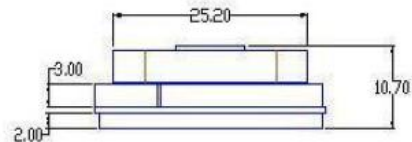
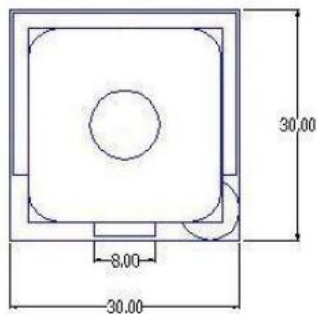
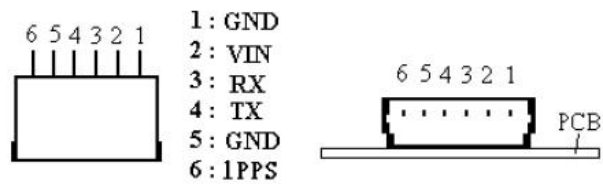
LED indicator for GPS fix or not fix

LED OFF:	Receiver switch off
LED ON:	No fixed, Signal searching
LED Flashing:	Position Fixed

Specification:**General**

Chipset	SiRF Star III
Frequency	L1, 1575.42 MHz
C/A code	1.023 MHz chip rate
Channels	20 channel all-in-view tracking
Sensitivity	-159 dBm

Pin Assignment



Dimension $\pm 0.2\text{mm}$

Pin description

* VIN (DC power input):

This is the main DC supply for a 4.5V ~6.5 DC input power.

* TX:

This is the main transmits channel for outputting navigation and measurement data to user's navigation software or user written software.

* RX:

This is the main receive channel for receiving software commands to the engine board from SiRFdemo software or from user written software.

* GND:

GND provides the ground for the engine board. Connect all grounds.

* 1PPS

This pin provides one pulse-per-second output from the engine board that is synchronized to GPS time.

ANEXO 2

FOTOGRAFÍAS DE SISTEMA GPS LOCARD

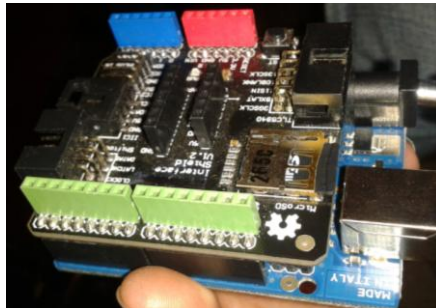
Vista superior GPS LOCARD



Vista Posterior GPS LOCARD



Shield MicroSD



Shield GPS



ANEXO 3

MANEJO DEL USB

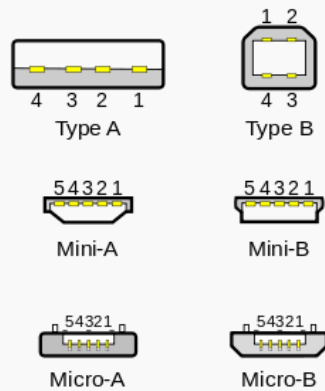


Símbolo USB

Tipo	Bus
Historia de producción	
Diseñador	Ajay Bhatt, Intel
Diseñado en	Enero 1996
Fabricante	IBM, Intel, Northern Telecom, Compaq, Microsoft, Digital Equipment Corporation y NEC
Sustituye a	Puerto serie, puerto paralelo, puerto de juegos, Apple Desktop Bus, PS/2
Sustituido por	Universal Serial Bus High Speed
Especificaciones	
Longitud	5 metros (máximo)
Ancho	11,5 mm (conector A), 8,45 mm (conector B)
Alto	4,5 mm (conector A), 7,78 mm (conector B, antes de v3.0)
Conectable en caliente	Sí
Externo	Sí
Electrico	5 voltios CC
	Voltaje máximo 5 voltios

	Corriente máxima	500 a 900 mA (depende de la versión)
Señal de Datos		Paquete de datos, definido por las especificaciones
	Ancho	1 bit
	Ancho de banda	1,5/12/480/5.000 Mbit/s (depende de la versión)
	Max nº dispositivos	127
	Protocolo	Serial
Cable		4 hilos en par trenzado; 8 en USB 3.0
Pines		4 (1 alimentación, 2 datos, 1 masa)
Conector		Único

Patillaje



Conectores tipo A (izquierda) y B (derecha)

Pin 1	V_{CC} (+5 V)
Pin 2	Data-
Pin 3	Data+
Pin 4	Tierra