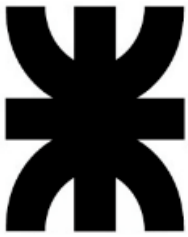

Universidad Tecnológica Nacional
Facultad Regional La Rioja

Carrera de Ingeniería Electrónica

Trabajo Final de Grado:

Módulo de Comunicaciones para Aplicaciones Generales y Geolocalización en Automotores

---

Autor:

Damián Leonel Corzi

La Rioja, Noviembre de 2017.

Catedra de Proyecto Final

Prof. Ing. Walter J. D. Cova

Trabajo Final de Grado:	Modulo de Comunicaciones para Aplicaciones Generales y Geolocalización en Automotores
Autor:	Damián Leonel Corzi
Docente Tutor:	Ing. Daniel Turra

Fecha de Examen: __ - __ - ____

(Leyenda “Aprobado”)……. EL EXAMEN DE DEFENSA DEL PRESENTE TRABAJO FINAL DE GRADO HA MERECIDO LA CALIFICACIÓN DE …… PUNTOS CONCEPTO: ……		
Tribunal Examinador:		
Presidente	Vocal 1°	Vocal 2°

Agradecimientos

A mis padres, hermana y abuela por su apoyo incondicional.

A mi amiga María Cristina Nieto Cano por su ayuda y sus grandes consejos.

A mi amigo Ángel Gutiérrez por su asesoramiento en electricidad del automóvil.

A mi compañero de trabajo Silvio Martínez y a mi jefe Ing. Aldo Morales por el aporte de buenas ideas.

A mis compañeros Gastón Razkevich y Cristian Vidal por el desarrollo conjunto del presente proyecto.

Al resto de mi familia y amigos por el acompañamiento a lo largo de varios meses de trabajo.

Contenido

Índice de tablas y figuras

Tablas

Figuras

1 Introducción

El presente proyecto final tiene por objeto construir un sistema capaz de brindar herramientas que ayuden a la prevención de accidentes de tránsito y reducir las pérdidas económicas relacionadas con los mismos, como así también las causadas por el robo de vehículos.

1.1 Diagnostico

Según la organización Luchemos por la vida[1], Argentina ostenta uno de los índices más altos de mortalidad producida por accidentes de tránsito en función de la cantidad de habitantes que posee, llegando en el año 2016 a la cifra de 7268 víctimas fatales[2].

Debido al hecho de que un accidente vial no solo involucra a las personas de forma directa, sino que también lo hace a través de las pérdidas económicas generadas por los daños a las propiedades privadas y públicas, es necesario determinar las principales causas que los provocan, siendo a nivel global según la Organización Mundial de la Salud[3] las siguientes:

- Velocidad excesiva.

- Conducción bajo los efectos del alcohol o sustancias psicoactivas.

- No utilización de cascos, cinturones de seguridad y medios de sujeción para los niños.

- Conducción distraída.

- Infraestructura vial insegura o vehículos inseguros.

Desde el punto de vista puramente económico, otra problemática que genera numerosas pérdidas es la que se encuentra relacionada con el robo de vehículos. Según la empresa internacional Ituran[4], de un balance realizado en 250000 vehículos monitoreados en el Área Metropolitana de Buenos Aires, se obtuvo que la frecuenta de robos anuales es de 2500 unidades.

1.2 Descripción del prototipo

El sistema construido permite realizar el monitoreo a distancia del vehículo y mantener al dueño o administrador al tanto de eventos de importancia, como en el caso de un exceso de velocidad, logrando una vigilancia continua del estado de un automóvil particular o de una flota perteneciente a una empresa de transporte.

Además, facilita el desarrollo y la interconexión de distintos dispositivos encargados de suministrar funciones adicionales, como el caso de alarmas, interruptores de arranque del motor, alcoholímetros, etc.

Como se trata de un sistema con características configurables, el mismo cuenta con interfaces fáciles de utilizar, lo que simplifica las tareas realizadas por el usuario o administrador.

De esta manera, con el sistema instalado, se obtiene información en tiempo real sobre el estado de vehículo, logrando que los conductores se vean obligados a disminuir la velocidad de circulación y permitir la localización del dispositivo en caso de robos o situaciones de emergencia.

1.3 Capítulos del informe

El informe se encuentra dividido en una serie de capítulos, incluyéndose además varios anexos, siendo el presente capitulo el número uno.

- Capítulo 2: “Sistemas de Comunicación”, presenta las bases teóricas de los distintos protocolos de comunicación utilizados en el sistema.

- Capítulo 3: “Microcontroladores”, presenta las bases teóricas de los microcontroladores y una descripción de los que serán utilizados en el proyecto.

- Capítulo 4: “Desarrollo”, presenta las distintas funcionalidades implementadas en el sistema.

- Capítulo 5: “Hardware del Sistema”, incluye diseños circuitales y fotografías de las placas construidas, como asimismo las mediciones eléctricas del consumo del dispositivo.

- Capítulo 6: “Ensayos”, presenta los ensayos realizados a las distintas partes del sistema durante su integración.

- Capítulo 7: “Costos”, detalla el valor monetario del sistema y los costos de su desarrollo.

- Capítulo 8: “Conclusión”, expone una conclusión del proyecto y detalla las posibles mejoras de futuras versiones.

- Anexo A: “Archivos auxiliares y de programación”, incluye las líneas de código utilizadas para la programación de los microcontroladores y la descripción de las distintas páginas web.

- Anexo B: “Internet móvil”, describe las bases de la transferencia de datos utilizando internet móvil, las cuales pueden implementarse en futuros desarrollos.

1.4 Bibliografía del capítulo

[1] Contenido web, fecha de acceso: 05/10/17. http://luchemos.org.ar/es/estadisticas/internacionales/muertos-en-argentina-comparacion-con-otros-paises-en-proporcion-a-vehiculos-y-o-poblacion

[2] Contenido web, fecha de acceso: 05/10/17. http://luchemos.org.ar/es/estadisticas/muertosanuales/muertos-en-argentina-durante-2016

[3] Contenido web, fecha de acceso: 05/10/17. http://www.who.int/mediacentre/factsheets/fs358/es/

[4] Contenido web, fecha de acceso: 05/10/17. http://www.ituran.com.ar/tag/robo-vehiculos/

2 Sistemas de comunicación

2.1 Introducción a los sistemas de comunicación electrónicos

Siguiendo a Tomasi [1], un sistema de comunicación electrónico es aquel que tiene como principal objetivo el transferir información de un lugar a otro. Por lo tanto, las comunicaciones electrónicas consisten en la transmisión, recepción y procesamiento de la información.

Un sistema de comunicaciones sencillo está formado por los siguientes elementos:

- Transmisor: Es aquel encargado de convertir la información original a una señal que pueda ser trasmitida por el medio.

- Medio de transmisión: Es aquel que transporta las señales del transmisor al lugar de recepción.

- Receptor: Es aquel que acepta del medio las señales transmitidas y las reconvierte obteniendo la información original.

Fig. 1: Elementos básicos de un sistema de comunicación.

Cada uno de los elementos debe poseer características particulares y seguir una serie de reglas definidas mediante protocolos para permitir el envío y la recepción de la información.

En todos los sistemas electrónicos, la información a transmitir es convertida en energía electromagnética a través de transductores (micrófonos, sensores de temperatura, etc.) la cual está comprendida en un intervalo de frecuencias conocido como banda base. Esta energía puede propagarse en forma de voltaje o corriente a través de un conductor, viajar en forma de ondas electromagnéticas a través del espacio, o en forma de ondas luminosas por medio de fibras ópticas.

En algunos casos no es practico realizar la transmisión de la banda base directamente debido a las características físicas del medio y es necesario utilizar técnicas de modulación, las cuales consisten en modificar por medio de la señal de información original a una o más propiedades de una señal conocida como portadora, la cual posee características que permiten su transmisión.

2.1.1 Modos de transmisión

Los modos de transmisión de un sistema definen el sentido en la cual fluirá la información entre dos dispositivos electrónicos llamados estaciones.

- Simplex: La transmisión tiene solo un sentido, se realiza de la estación trasmisora a la receptora.

- Semi dúplex: Las trasmisiones pueden realizarse en ambos sentidos pero no al mismo tiempo. Es decir, mientras una estación envía información la otra debe tomar el rol de receptor y esperar a que la transmisión finalice para poder realizar una respuesta.

- Full dúplex: Las trasmisiones pueden realizarse en ambos sentidos y al mismo tiempo entre dos estaciones.

- Full/full dúplex: Las trasmisiones pueden realizarse en ambos sentidos y al mismo tiempo entre dos o más estaciones.

2.1.2 Configuraciones de los circuitos

Las transmisiones realizadas a través de conductores eléctricos pueden tener distintos tipos de configuraciones según el número de hilos y la referencia eléctrica utilizada:

- Transmisiones de dos hilos no balanceados: Un conductor es utilizado para enviar la señal y el otro es utilizado como referencia. Se adapta de forma ideal a las transmisiones simplex y semi dúplex.

- Transmisiones de dos hilos balanceados: La señal es enviada a través de dos conductores de forma diferencial. Se adapta de forma ideal a las transmisiones simplex.

- Transmisiones de tres hilos no balanceados: Un conductor es utilizado como referencia y los restantes para enviar y recibir a las señales. Se adapta de forma ideal a las transmisiones full dúplex.

- Transmisiones de cuatro hilos balanceados: Un par de conductores se utiliza para enviar la señal de forma diferencial mientras que el par restante se utiliza para recibir. Se adapta de forma ideal a las trasmisiones full dúplex.

Las transmisiones del tipo diferencial son más robustas frente a las interferencias, debido a que estas afectan por igual a ambas líneas. Una desventaja de este tipo de configuraciones radica en el incremento de la cantidad de conductores utilizados.

En muchos casos, además de los hilos utilizados para la trasmisión, se agrega un blindaje para proteger la integridad de las señales frente al ruido electromagnético. Este blindaje se conecta a la masa de solo una de las estaciones, lo que permite evitar la formación de bucles de corrientes de masa.

2.1.3 Codificación digital

La forma más frecuente para realizar una transmisión de información digital consiste en asignar los estados lógicos a dos valores diferentes de tensión o de transiciones de tensión, generando de esta manera una trasmisión de banda base.

Los tipos de codificación más comunes son:

- Unipolar: Uno de los estados lógicos es asignado al valor de cero voltios mientras que el otro es representado por un valor de tensión diferente de cero. Las secuencias largas de un mismo estado generan problemas de sincronización en el receptor.

- Bipolar sin retorno a cero (NRZ): Se utilizan dos valores de tensión diferentes para representar a los estados lógicos. El valor de tensión se mantiene constante durante cada bit y la línea no regresa al valor cero. Las secuencias largas de un mismo estado generan problemas de sincronización en el receptor.

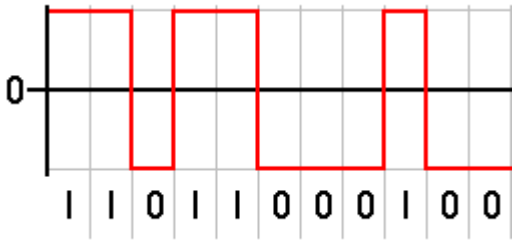

Fig. 2: Codificación NRZ[1].

- Bipolar con retorno a cero (RZ): Se utilizan dos valores de tensión diferentes para representar a los estados lógicos. El valor de la tensión no se mantiene constante durante cada bit sino que regresa al valor cero, por esta razón las secuencias largas de un mismo estado no plantean un problema de sincronización.

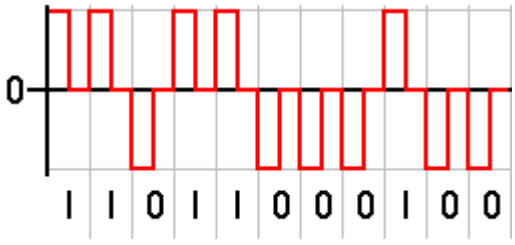

Fig. 3: Codificación RZ[2].

[1] NRZcode - Omegatron. Obtenido de: https://commons.wikimedia.org/wiki/File:NRZcode.png
[2] RZcode - Omegatron. Obtenido de: https://commons.wikimedia.org/wiki/File:RZcode.png

- Diferencial: Se utilizan dos valores de tensión diferentes para representar a los estados lógicos. El valor de la tensión no se mantiene constante durante cada bit sino que toma el valor del estado lógico opuesto. Al igual que la RZ no presenta problemas de sincronismo. Dos ejemplos de este tipo de codificación son la Manchester y la Manchester diferencial.

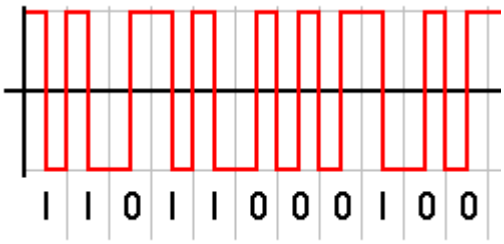

Fig. 4: Codificación Manchester[3].

2.1.4 Topologías

La topología o arquitectura de un sistema de comunicación define la forma en la que se interconectan las diferentes estaciones conformando lo que se conoce como red. Dentro de la red, cada una de las estaciones es conocida como nodo.

Las topologías más comunes son:

- Punto a punto: Solo dos estaciones se encuentran interconectadas entre sí.

- Estrella: Todos los nodos se encuentran conectados a una estación centrar conocida como anfitrión.

- Bus: Todos los nodos se encuentran interconectados entre sí a través de un único canal de comunicaciones.

- Anillo: Cada nodo se encuentra conectado a dos de sus vecinos, formando así, dos rutas alternativas para el envío de la información.

- Malla: Cada nodo se encuentra conectado con sus vecinos mediante canales independientes.

[3] Manchester code - Magnus. Obtenido de: https://commons.wikimedia.org/wiki/File:Manchester_code.png

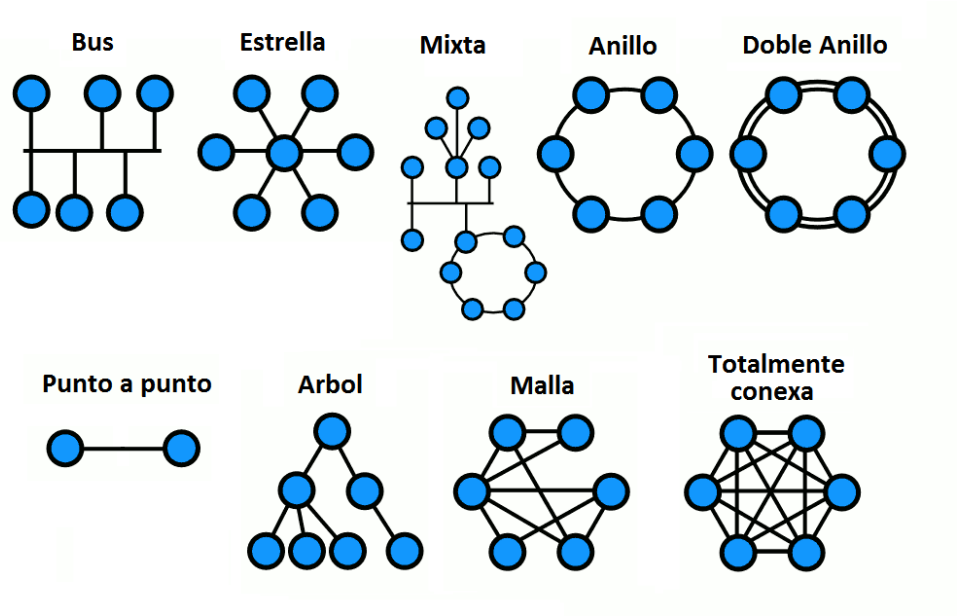

Fig. 5: Topologías de una red[4].

2.1.5 Modelo de referencia OSI

Según Tanenbaum [2], el modelo de interconexión de sistemas abiertos o modelo OSI, del inglés *Open System Interconnection*, es un modelo de referencia desarrollado por la Organización Internacional de Normalización (ISO/IEC 7498-1) para los protocolos de una red cuya arquitectura está formada por capas.

El modelo define una serie de capas en las cuales una red puede descomponerse. Cada una de estas debe brindar servicios a la que se encuentra por encima de ella y comunicarse con la que se encuentra debajo. Esto genera diferentes niveles de abstracción, lo que permite que una capa de un dispositivo conectado a la red pueda comunicarse directamente con la misma capa de un segundo dispositivo sin tener en cuenta a los procesos realizados por las capas inferiores.

Cada una de las capas del modelo se encarga de definir y realizar una serie de tareas:

- Capa física: Define el medio por el cual los bits serán trasmitidos, el tipo de conectores y las características de las señales eléctricas.

- Capa de enlace de datos: Define y realiza el direccionamiento físico, el control del acceso al medio, la detección de errores, la distribución de tramas y el control del flujo.

[4] Topología de red - Yearofthedragon. Obtenido de: https://commons.wikimedia.org/wiki/File:Topolog%C3%ADa_de_red.png

- Capa de red: El objetivo de esta capa es el de lograr que los datos lleguen desde el origen al destino, aun cuando ambos no estén conectados directamente. Esto se consigue determinando la ruta que los datos deben seguir.

- Capa de transporte: Encargada de efectuar el transporte de los datos, independizando a las capas superiores del tipo de red física utilizada.

- Capa de sesión: Encargada de mantener y controlar los enlaces establecidos entre dos dispositivos conectados a la red.

- Capa de presentación: Encargada de la representación de la información en los distintos dispositivos. Trabaja sobre el contenido de la información y no sobre la comunicación.

- Capa de aplicación: Encargada de ofrecer a las aplicaciones la posibilidad de acceder a los servicios de las demás capas y define los protocolos utilizados para el intercambio de datos.

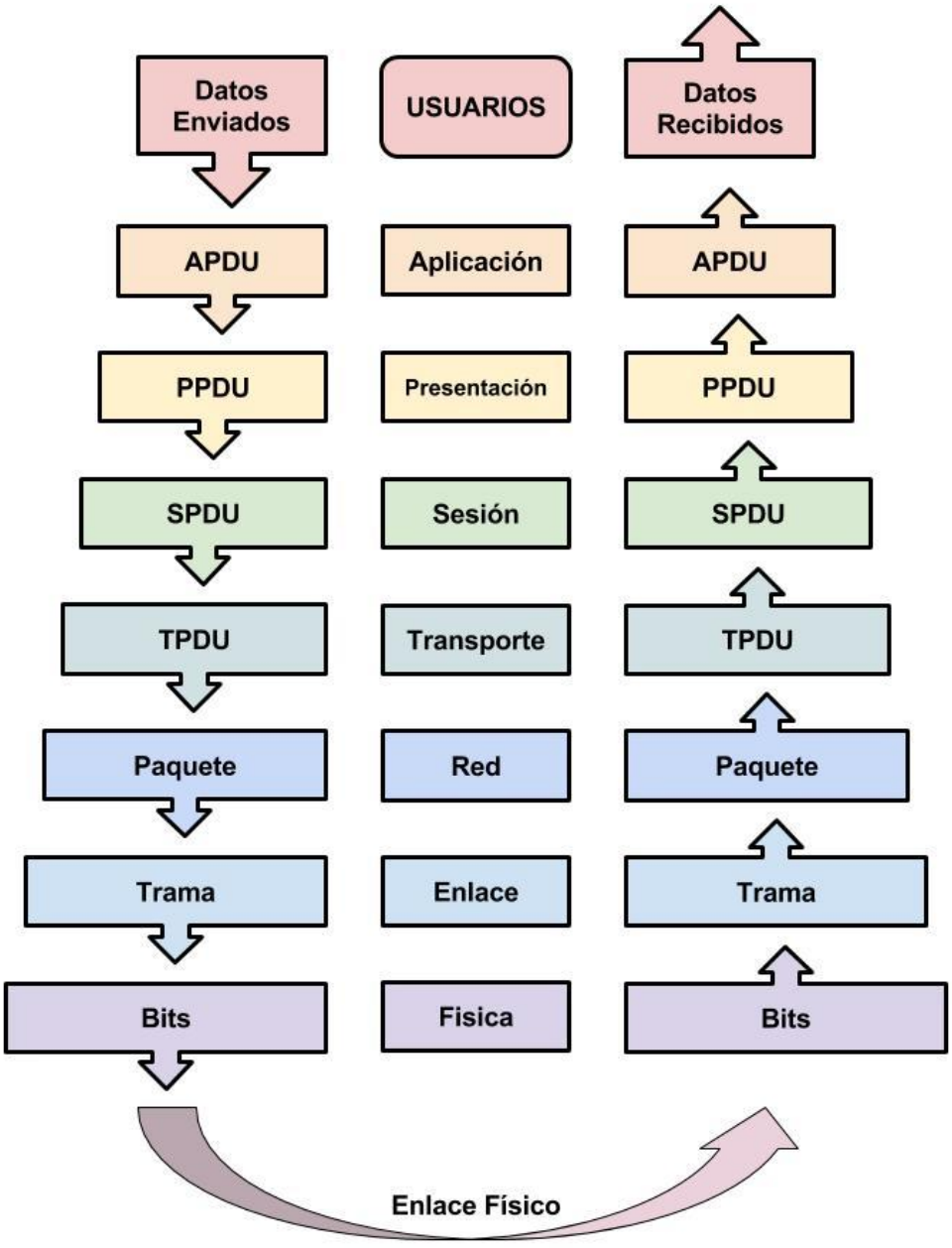

Fig. 6: Modelo de referencia OSI.

Durante la transmisión cada una de las capas de la red agrega información en forma de encabezados y colas a los datos recibidos desde la capa superior. A la información generada de esta manera se la conoce como unidad de datos y recibe distintos nombres según la capa que la produzca.

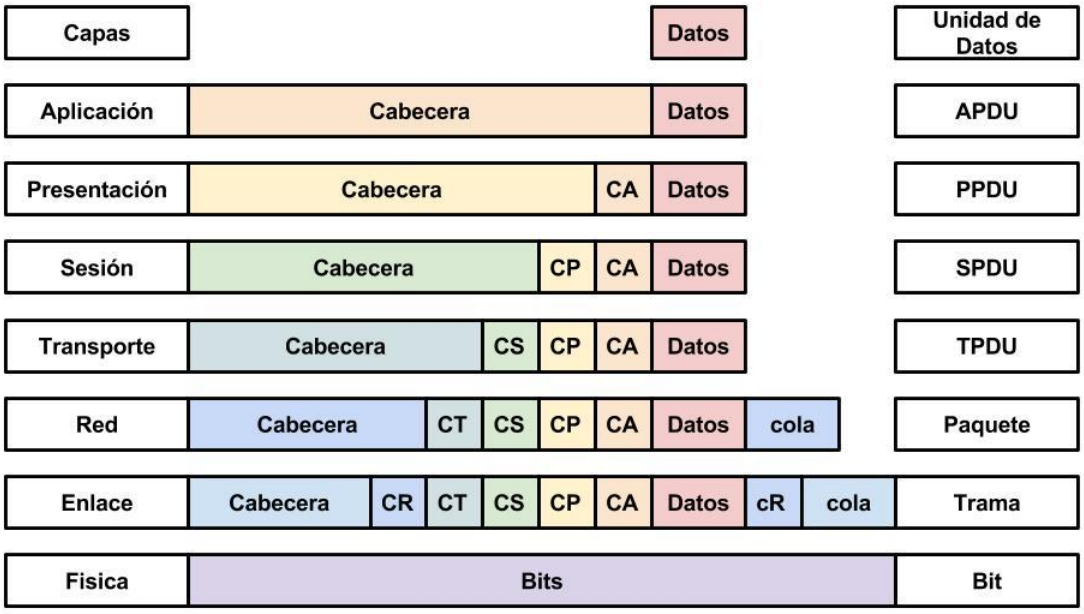

Fig. 7: Empaquetado de los datos durante una transmisión del modelo OSI.

2.2 CAN

CAN [3], del inglés *Controller Area Network* (ISO 11898), es un protocolo de comunicación desarrollado por la firma *Robert Bosch GmbH* para aplicaciones que demanden una transmisión robusta de información en el interior de un automóvil.

El protocolo ha sido adoptado por muchas áreas además de la industria automotriz. Numerosos estándares utilizan a CAN como base y esto es debido fundamentalmente a sus características principales:

- Define capa física y capa de enlace de datos (*Modelo OSI*).

- Topología del tipo bus.

- Orientado a mensajes, permite definir prioridad.

- Sistema Multi maestro, permite realizar multidifusion.

- Detección de errores y retransmisión automática de tramas.

- Desconexión autónoma de nodos defectuosos.

2.2.1 Capa física

La capa física de una red (referida al modelo *OSI*) define las características materiales, eléctricas y de transmisión del flujo de bits a través del bus.

2.2.1.1 Cables y conectores

Los distintos nodos del bus deben estar conectados mediante un par de cables trenzados con una impedancia característica de 120Ω. El estándar no especifica si el cable debe o no estar apantallado ni tampoco determina un tipo de conector a utilizar.

Debido a que las propiedades de la línea de transmisión determinan el ancho de banda del sistema, de manera aproximada, las longitudes máximas aceptadas en función de la tasa de transferencia son:

Tabla 1: Relación aproximada entre la longitud y la taza de transferencia en un bus CAN[5].

Longitud (m)	Tasa de transferencia (kbit/s)
40	1000
100	500
200	250
500	100
1000	50

2.2.1.2 Tipos de bus CAN

El estándar especifica dos tipos de buses según la velocidad de operación del sistema.

- Bus de alta velocidad: Utilizado para transmisiones de hasta 1 Mbit/s. Consiste en un único bus lineal (*backbone*) y dos resistencias de terminación para evitar reflexiones.

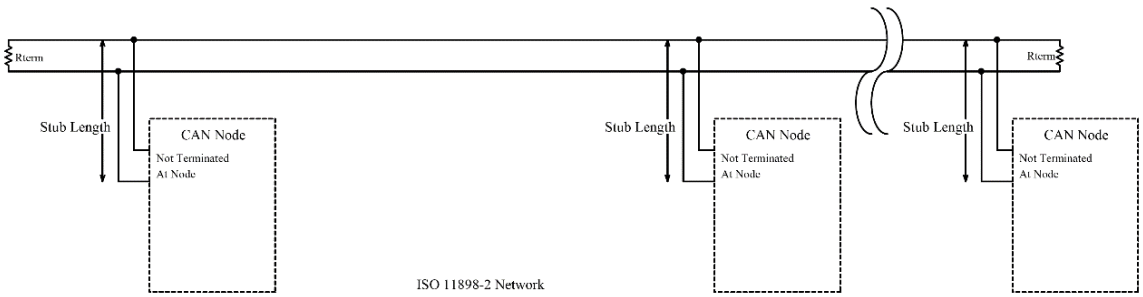

Fig. 8: Bus can de alta velocidad[6].

- Bus de baja velocidad: Utilizado para trasmisiones de hasta 125 kbit/s. Consiste en un bus del tipo estrella o múltiples buses del tipo estrella unidos entre sí por un único bus del tipo lineal. Las resistencias de terminación son colocadas en cada uno de los nodos del sistema.

[5] Bus CAN - Wikipedia. Obtenido de: https://es.wikipedia.org/wiki/Bus_CAN

[6] CAN ISO11898-2 Network - EE JRW. Obtenido de: https://commons.wikimedia.org/wiki/File:CAN_ISO11898-2_Network.png

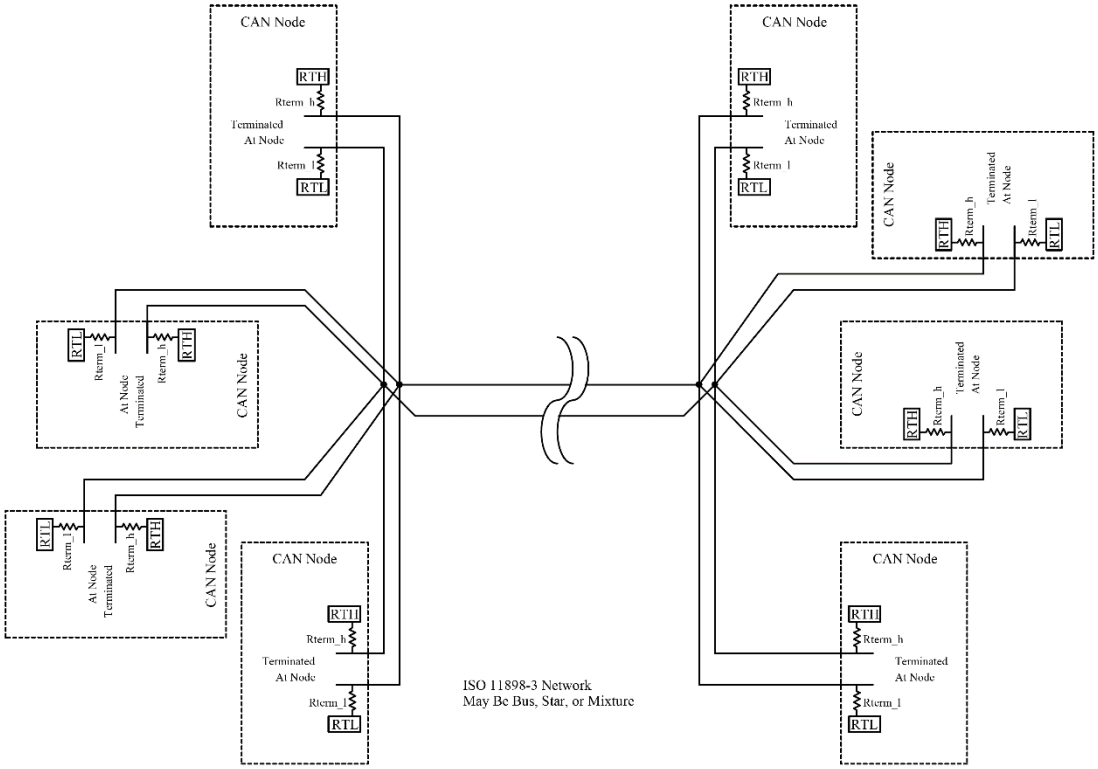

Fig. 9: Bus CAN de baja velocidad[7].

2.2.1.3 Niveles de tensión

Las señales de los cables del bus se denominan CAN *high* y CAN *low*. A partir de estas señales se definen dos estados del bus.

- Estado dominante: La diferencia de tensión entre ambos cables se encuentra comprendida entre 1.5 V y 3 V.

- Estado recesivo: Los dos cables se encuentran a un mismo nivel de tensión.

La especificación determina que la tensión en modo común deberá estar comprendida entre -2 V y 7 V, esto permite la conexión directa entre nodos que operen a diferentes valores de tensión.

[7] CAN ISO11898-3 Network - EE JRW. Obtenido de: https://commons.wikimedia.org/wiki/File:CAN_ISO11898-3_Network.png

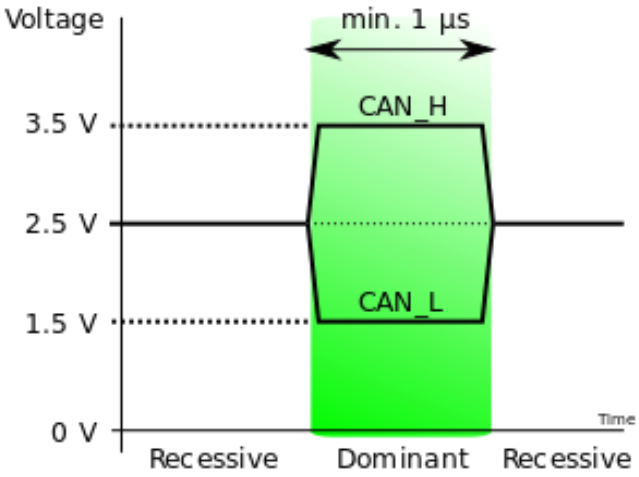

Fig. 10: Niveles de tensión de las señales en un bus CAN[8].

2.2.1.4 Sincronización del flujo de bits

La tasa de transferencia de todos los nodos de un bus CAN debe ser la misma. Debido a que el protocolo no define una señal de reloj independiente, factores como la deriva de reloj generan una diferencia entre las tasas de transferencia real de cada nodo. Para solucionar este inconveniente el protocolo define un método de sincronización para garantizar el correcto funcionamiento del sistema.

La sincronización es especialmente importante durante la fase de arbitraje, ya que en ella cada nodo debe ser capaz de observar los datos transmitidos por él y los datos transmitidos por los demás. Esto implica que todos los nodos deben muestrear el valor de cada bit en el instante adecuado.

El método utilizado para la sincronización establece que el controlador del nodo espere una transición del bus de recesivo a dominante. Si la misma no ocurre en el intervalo de tiempo esperado, el controlador reajusta la duración del siguiente bit. Dicho ajuste se lleva a cabo dividiendo al tiempo de bit en intervalos llamados *cuantos* y asignándolos a cuatro segmentos o intervalos de tiempo.

- Segmento de sincronización: Intervalo en el cual se supone que ocurre la transición del estado recesivo al dominante.

- Segmento de propagación: Intervalo que compensa los retardos de propagación a lo largo del bus.

- Segmentos de fase 1 y fase 2: Intervalos utilizados para la sincronización. El muestreo del bit se realiza al final de la fase 1, lo cual ocurre por lo general a un 75 % del tiempo de bit. Durante

[8] Canbus levels - Plupp. Obtenido de: https://commons.wikimedia.org/wiki/File:Canbus_levels.svg

la re sincronización el valor de ambos segmentos puede ser modificado para que el muestreo se realice en un instante de tiempo diferente.

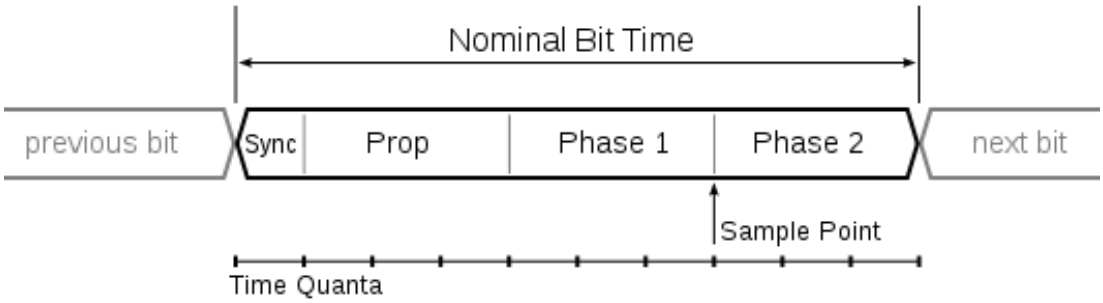

Fig. 11: Segmentos de un bit enviado a través del bus CAN[9].

2.2.2 Capa de enlace de datos

La capa de enlace de una red (referida al modelo *OSI*) define el método de arbitraje o acceso al medio y los tipos de tramas para el envío de mensajes.

2.2.2.1 Acceso al medio

La especificación del protocolo define al estado dominante el valor lógico 0 y al estado recesivo el valor lógico 1. Además define al estado recesivo como el estado en el cual el bus se encuentra inactivo.

Cuando dos nodos intentan transmitir bits diferentes al mismo tiempo ocurre lo que se denomina colisión. En este estado, el nodo que intento transmitir un bit recesivo detecta la colisión y detiene la transferencia pasando a un modo inactivo.

El arbitraje se realiza solo durante los primeros bits de una trama, evaluando lo que se conoce como identificador del mensaje. Al final del proceso de arbitraje solo debe quedar un nodo con el control del bus y los demás deben aplazar las transmisiones de sus datos. De esta manera, una trama con un valor de identificador bajo tendrá una prioridad mayor a la de una trama con un identificador alto.

2.2.2.2 Tipos de trama

Existen cuatro tipos de tramas definidas en el estándar. Algunas presentan diferencias según se trate de la versión base (2.0A) o de la versión extendida (2.0B) del protocolo.

2.2.2.3 Trama de datos

La trama de datos varía según la versión del protocolo. La principal diferencia se encuentra en la cantidad de bits definidos para el identificador del mensaje.

El estándar establece que un controlador CAN debe aceptar tramas en formato base, y puede o no aceptar tramas en formato extendido. Pero en cualquier caso debe tolerar las tramas de formato extendido y no generar una alerta de error, simplemente debe ignorar el mensaje.

[9] CAN Bit Timing2 - Rocketmagnet. Obtenido de: https://commons.wikimedia.org/wiki/File:CAN_Bit_Timing2.svg

2.2.2.4 Formato base

El formato base de una trama está formado por los siguientes campos:

Tabla 2: Formato base de la trama CAN[10].

Nombre del Campo	Longitud (bits)	Descripción
Inicio de trama	1	Comienzo de una transmisión.
Identificador de mensaje (ID)	11	Identificador único para cada mensaje, representa la prioridad de la trama.
Petición de transmisión remota (RTR)	1	Dominante para tramas de datos y recesivo para tramas de peticiones remotas.
Extensión de identificador (IDE)	1	Dominante para formato base.
Reservado (r0)	1	Debe ser dominante.
Longitud de datos (DLC)	4	Numero de bytes de datos del mensaje. Si el valor es superior a 8, su valor será interpretado como 8.
Datos	0 - 64	Datos de la trama, la longitud de este campo está determinada por el DLC.
CRC	15	Verificación por redundancia cíclica.
Delimitador de CRC	1	Debe ser recesivo.
Acuse de recibo (ACK)	1	El transmisor del mensaje lo emite como recesivo y cualquier receptor como dominante.
Delimitador de ACK	1	Debe ser recesivo.
Fin de trama (EOF)	7	Todos los bits deben ser recesivos.

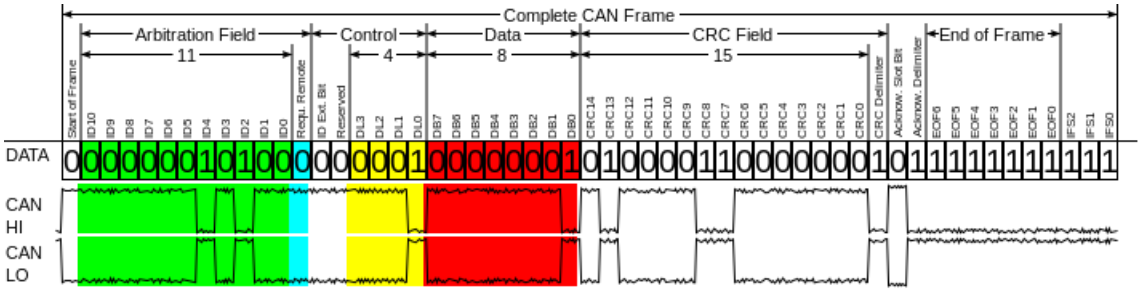

Fig. 12: Formato base de la trama CAN[11].

[10] Bus CAN - Wikipedia. Obtenido de: https://es.wikipedia.org/wiki/Bus_CAN

[11] CAN-Bus-frame in base format without stuffbits - OgreBot. Obtenido de: https://commons.wikimedia.org/wiki/File:CAN-Bus-frame_in_base_format_without_stuffbits.svg

2.2.2.5 Formato extendido

En el formato extendido el identificador del mensaje se divide en dos campos separados para permitir un mayor número de mensajes manteniendo la compatibilidad con el formato base.

Tabla 3: Formato extendido de la trama CAN[12].

Nombre del Campo	Longitud (bits)	Descripción
Inicio de trama	1	Comienzo de una transmisión.
Identificador de mensaje (ID_A)	11	Primera parte del Identificador único para cada mensaje, representa la prioridad de la trama.
Sustituto de transmisión remota (SRR)	1	Debe ser recesivo.
Extensión de identificador (IDE)	1	Recesivo para formato extendido.
Identificador de mensaje (ID_B)	18	Segunda parte del Identificador único para cada mensaje, representa la prioridad de la trama.
Petición de transmisión remota (RTR)	1	Dominante para tramas de datos y recesivo para tramas de peticiones remotas.
Reservados (r0, r1)	2	Deben ser dominantes.
Longitud de datos (DLC)	4	Numero de bytes de datos del mensaje. Si el valor es superior a 8, su valor será interpretado como 8.
Datos	0 - 64	Datos de la trama, la longitud de este campo está determinada por el DLC.
CRC	15	Verificación por redundancia cíclica.
Delimitador de CRC	1	Debe ser recesivo.
Acuse de recibo (ACK)	1	El transmisor del mensaje lo emite como recesivo y cualquier receptor como dominante.
Delimitador de ACK	1	Debe ser recesivo.
Fin de trama (EOF)	7	Todos los bits deben ser recesivos.

2.2.2.6 Trama remota

Las tramas remotas son utilizadas para realizar una petición de envío de un mensaje en particular. Se diferencian de las tramas de datos en que no contienen el campo de datos y en que el bit RTR se encuentra en estado recesivo.

2.2.2.7 Trama de error

La trama de error es una trama especial que se transmite cuando un nodo detecta un mensaje erróneo y provoca que los demás también transmitan una trama de error. Mediante un

[12] Bus CAN - Wikipedia. Obtenida de: https://es.wikipedia.org/wiki/Bus_CAN

algoritmo de conteo se consigue evitar que un nodo en particular bloquee el bus mediante continuas tramas de error.

2.2.2.8 Trama de sobrecarga

Es transmitida por un nodo que se encuentra ocupado y provoca que los demás generen retardos extra entre tramas.

2.2.2.9 Separación entre tramas

Las tramas de datos y remotas están separadas por al menos tres bits recesivos. Después de eso, si se detecta un bit dominante, es considerado como el inicio de una nueva trama.

Las tramas de error y de sobrecarga no respetan el espaciado entre las tramas.

2.2.2.10 Bits de relleno

Debido a que el protocolo utiliza una codificación sin retorno a cero (NRZ), durante la transmisión de una trama se insertan bits adicionales para generar transiciones de estado luego de cinco bits consecutivos de la misma polaridad, esto garantiza la sincronización en los nodos. Todos los bits de relleno son eliminados automáticamente por el nodo receptor.

No todos los campos de la trama son rellenados. Los bits delimitador CRC, ACK y EOF permanecen inalterados.

Cuando un nodo detecta seis bits consecutivos iguales en un campo susceptible de ser rellenado lo considera un error y genera una transmisión que consiste en seis bits consecutivos en estado dominante.

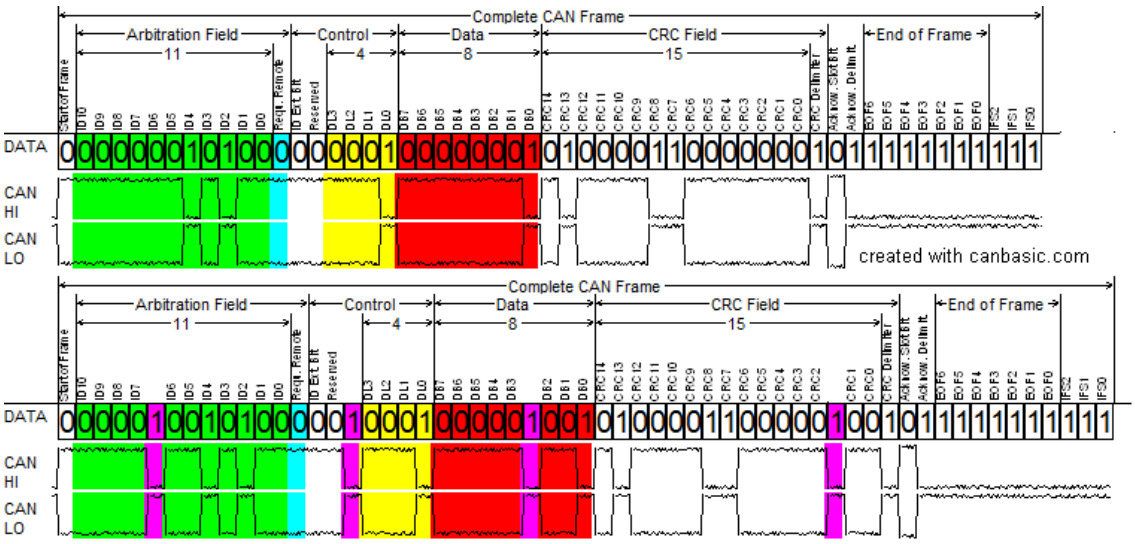

Fig. 13: Bits de relleno insertados en una trama CAN de formato base[13].

[13] CAN-Frame mit Pegeln mit Stuffbits - Ernst Otto Derwald. Obtenido de: https://commons.wikimedia.org/wiki/File:CAN-Frame_mit_Pegeln_mit_Stuffbits.png

2.2.3 Nodos CAN

Los elementos que componen a un nodo pueden o no estar integrados en un mismo dispositivo, pero sin importar la configuración física real, el sistema puede tratarse como la unión de varios componentes independientes.

- Transceptor: Realiza la conversión entre los niveles empleados por el Microcontrolador y los niveles definidos por el protocolo, por lo tanto, forma parte de la capa física.

- Controlador CAN: Realiza las tareas de la capa de enlace de datos y la sincronización de la tasa de transferencia. Además en algunos controladores se integran funcionalidades como el filtrado de mensajes o informes de estado.

- Microcontrolador: Realiza tareas relacionadas con las capas superiores y es quien gestiona la información transmitida.

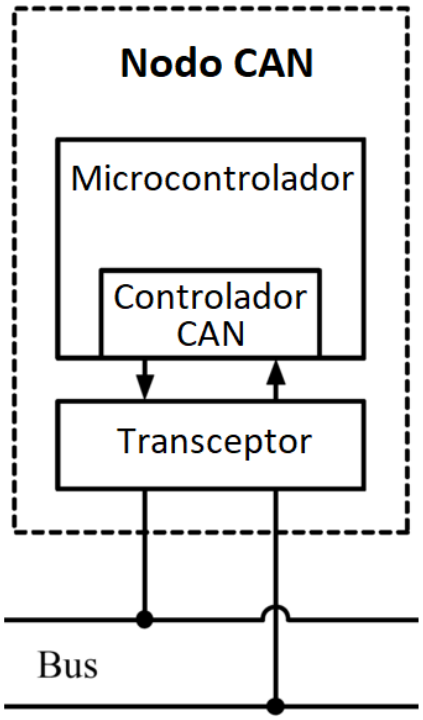

Fig. 14: Nodo de un bus CAN[14].

2.3 UART

Según National Instruments [4], UART del inglés *universal asynchronous receiver/transmitter*, consiste en la transmisión de datos de forma serial y asincrónica mediante dos líneas de datos independientes. Una línea es utilizada para transmitir (TxD) y otra es utilizada para recibir (RxD), por lo que la comunicación obtenida es del tipo Full Dúplex.

[14] CAN Node - EE JRW. Obtenido de: https://en.wikipedia.org/wiki/File:CAN_Node.png

2.3.1 Velocidad de transmisión

La velocidad de transmisión (*baud rate*) indica el número de bits por segundo que se transfieren, definiendo a esta relación con una unidad llamada baudio (*baud*). La velocidad de transmisión no se encuentra especificada, pero las más comunes incluyen al valor de 9600 baudios y a sus múltiplos.

Debido a que se trata de una transmisión asincrónica, tanto el emisor como el receptor deben trabajar a la misma velocidad para garantizar el correcto muestreo de los datos.

2.3.2 Trama UART

La trama de la transmisión no se encuentra especificada pero por lo general podemos encontrar en ella los siguientes elementos:

- Bits de inicio: Determinan el inicio de la trama y permite la sincronización con receptor.

- Bits de datos: La cantidad de bits de datos está directamente relacionada con la eficiencia de la línea, por lo que aumentando su valor aumenta la cantidad de información útil transmitida. Por otro lado, aumentar demasiado la cantidad de bits de datos genera problemas de sincronización. Debido a que la cantidad de bits utilizados en este campo no está especificada, por lo general se utilizan valores entre 5 bits y 8 bits.

- Bits de paridad: Determina una forma sencilla para encontrar errores en la transmisión y no necesariamente debe estar presente en la trama. El valor de este bit depende del cálculo de la paridad escogida.

- Bits de parada: Determina el final de la transmisión. Los valores típicos utilizados por lo general son 1 bit, 1.5 bit y 2 bit.

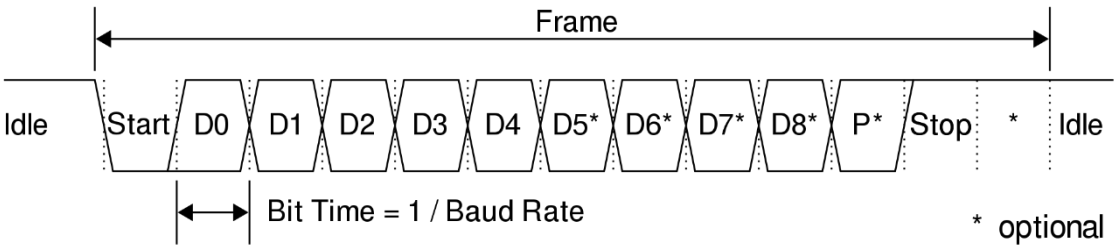

Fig. 15: Trama UART[15].

2.3.3 Dispositivos UART

Muchos dispositivos tienen incorporados controladores dedicados para la transmisión de datos mediante UART. En la mayoría de los casos el sistema está compuesto por un registro de desplazamiento con carga paralela y algunos registros adicionales donde la información es almacenada (buffer FIFO) [5].

[15] UART Frame - AmenophisIII. Obtenido de: https://commons.wikimedia.org/wiki/File:UART_Frame.svg

2.3.4 Condiciones especiales de funcionamiento

A las operaciones normales de envío y recepción de información podemos sumar algunos estados en los que el dispositivo advierte un error en su funcionamiento normal.

- Desbordamiento: Buffer de recepción lleno durante la llegada de una nueva trama.

- Enmarcado: Error al no encontrar los bits de inicio o de final de la trama.

- Paridad: Error que ocurre cuando el valor del bit de paridad no coincide con la paridad calculada por el receptor.

2.4 SPI

SPI [6], del inglés *Serial Peripheral Interface*, es un protocolo de comunicaciones desarrollado por Motorola para la transferencia de información entre circuitos electrónicos integrados.

El protocolo ha sido ampliamente adoptado por la industria debido a sus principales características:

- Comunicación Síncrona del tipo full dúplex.

- El sistema permite un solo maestro y múltiples esclavos.

- Campo de datos en la trama de longitud variable.

2.4.1 Interfaz

El protocolo especifica cuatro señales lógicas para establecer la comunicación entre los dispositivos de bus. Según la aplicación algunas de estas señales pueden o no ser requeridas y simplemente no estarán implementadas.

- SCLK: del inglés *Serial Clock*, es la señal encargada de transmitir los pulsos de reloj generados por el maestro a todos los dispositivos esclavos en el sistema.

- SS: del inglés *Slave Select*, es la señal encargada de definir cuál de todos los esclavos se encuentra activo.

- MOSI: del inglés *Master Output Slave Input*, es la señal utilizada por el maestro para enviar información a los dispositivos esclavos en el sistema.

- MISO: del inglés *Master Input Slave Output*, es la señal utilizada por un esclavo para enviar información al dispositivo maestro.

2.4.2 Polaridad y fase de la señal de reloj

Además del valor de la frecuencia otros parámetros que pueden configurarse en la señal del reloj son la polaridad y la fase con respecto a las señales de datos. Motorola define a estas configuraciones como CPOL y CPHA respectivamente.

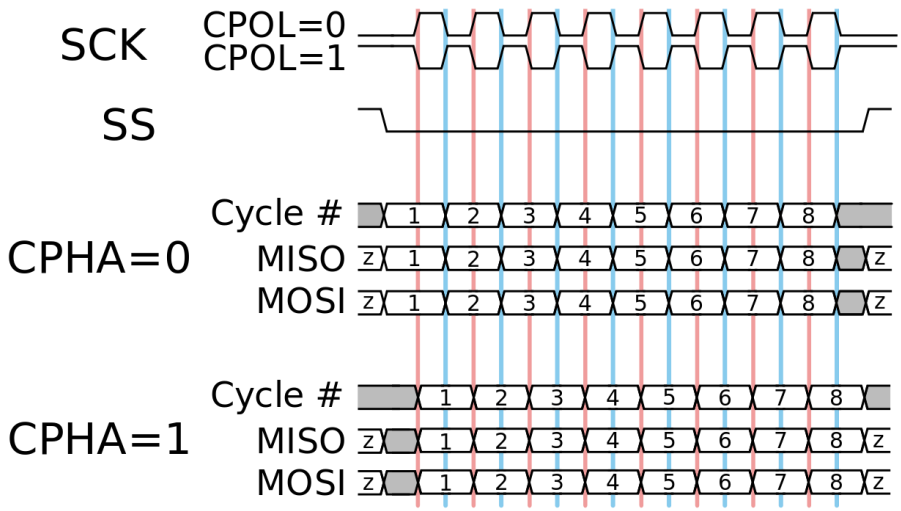

Fig. 16: Polaridades y fases de una transmisión SPI[16].

Muchos fabricantes han optado por definir modos de trabajo en los cuales se combinan ambas configuraciones.

Tabla 4: Modos de trabajo del protocolo SPI[17].

Modo	CPOL	CPHA
0	0	0
1	0	1
2	1	0
3	1	1

2.4.3 Operación de transmisión de datos

Luego de configurar de manera adecuada a los dispositivos conectados para que puedan operar con la misma frecuencia y en el mismo modo de trabajo, el proceso para realizar una transmisión de datos consiste en seleccionar mediante la línea SS al dispositivo esclavo al que se desea enviar la información.

Existen tres tipos principales de conexión entre el dispositivo maestro y los esclavos conectados en el bus.

- Un maestro y un esclavo: Es la conexión básica del bus SPI. En la mayoría de los casos la señal SS puede en el esclavo estar directamente conectada al valor lógico 0 del sistema.

[16] SPI timing diagram2 - Cburnett. Obtenido de: https://en.wikipedia.org/wiki/File:SPI_timing_diagram2.svg
[17] Serial Peripheral Interface Bus - Wikipedia. Obtenido de: https://en.wikipedia.org/wiki/Serial_Peripheral_Interface_Bus

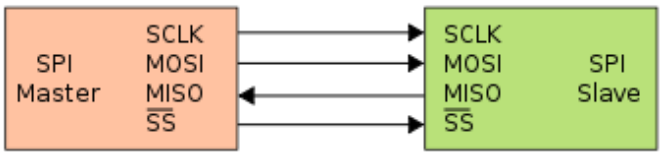

Fig. 17: Conexión básica del protocolo SPI. Un maestro y solo un esclavo[18].

- Un maestro y múltiples esclavos: Cada dispositivo esclavo posee su propia línea SS.

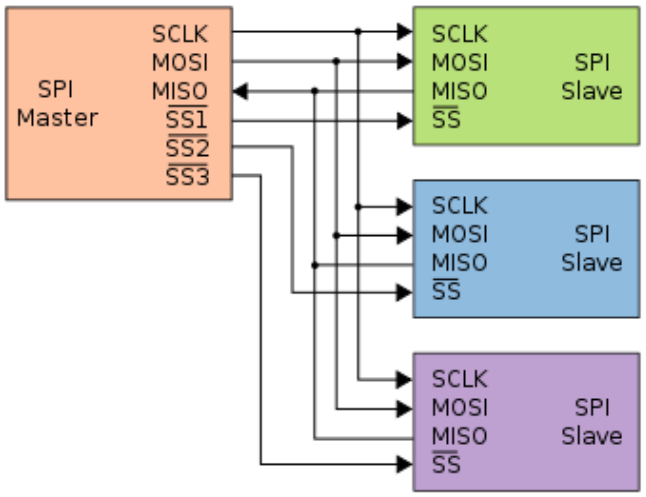

Fig. 18: Conexión estándar del protocolo SPI. Un maestro y múltiples esclavos[19].

- Un maestro y múltiples esclavos conectados en cadena de margaritas (*daisy chain*): En este tipo de conexión la salida de cada esclavo se conecta con la entrada del siguiente, de esta manera se consigue que todo el sistema se comporte como un único registro de desplazamiento.

[18] SPI single slave - Cburnett. Obtenido de: https://en.wikipedia.org/wiki/File:SPI_single_slave.svg
[19] SPI three slaves - Cburnett. Obtenido de: https://en.wikipedia.org/wiki/File:SPI_three_slaves.svg

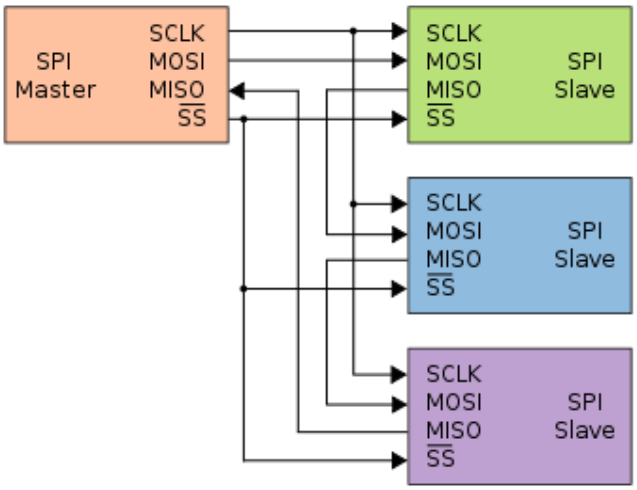

Fig. 19: Conexión mediante cadena de margaritas del protocolo SPI[20].

2.5 Wifi

Según Matthew Gast [7], wifi es el nombre que reciben las tecnologías utilizadas en la conexión inalámbrica de los distintos dispositivos basados en el estándar IEEE 802.11 [8] y comercializados por Wi-Fi Alliance.

Los dispositivos terminales que incorporan esta tecnología pueden conectarse a una red inalámbrica formada por un punto de acceso, el cual a su vez está por lo general conectado a un enrutador para permitir el acceso a internet.

Las características de la red varían según el estándar específico en el cual la versión de wifi esté basada, pero muchas de ellas son comunes entre las distintas versiones:

- Compatibilidad de conexión entre distintas versiones.

- Fácil conexión de los dispositivos.

- Redes de corto alcance.

2.5.1 IEEE 802.11

El estándar IEEE 802.11 define la capa física y la capa de enlace de datos para una red de área local inalámbrica, publicado y mantenido por el Instituto de Ingenieros Eléctricos y Electrónicos (*Institute of Electrical and Electronics*, IEEE).

Las distintas versiones del protocolo definen a las técnicas de modulación de las señales de radio frecuencia utilizada para transportar datos sobre canales del tipo semi dúplex, aunque la

[20] SPI three slaves Daisy chained - Cburnett. Obtenido de: https://en.wikipedia.org/wiki/File:SPI_three_slaves_daisy_chained.svg

versión original publicada en 1997 también específica a las transmisiones mediantes señales del tipo infrarrojas (actualmente en desuso).

Tabla 5: Características de las distintas versiones del protocolo 802.11[21].

Versión	Fecha publicación	Características
802.11	1997	Datos sin procesar a 1 Mbps o 2 Mbps.
		Originalmente definido la detección de portadora con acceso múltiple y prevención de colisiones (CSMA-CA).
		Tres canales no superpuestos a 2.4 GHz.
802.11a	1999	Velocidades de datos con distinta modulación a 6, 9, 12, 18, 24, 36, 48 y 54 Mbps.
		Multiplexion por división de frecuencia con 52 canales.
802.11b	1999	Velocidades de datos con distinta modulación a 1, 2, 5.5 y 11 Mbps.
		Tres canales no superpuestos a 2.4 GHz.
802.11g	2003	Velocidades de datos con distinta modulación a 6, 9, 12, 18, 24, 36, 48 y 54 Mbps.
		Multiplexion por división de frecuencia con 52 canales. (Mantiene compatibilidad con 802.11b).
		Tres canales no súper puestos a 2.4 GHz.
802.11n	2009	Velocidades de datos según el modo de operación hasta 450 Mbps.
		Sistema de antenas de múltiple entrada y múltiple salida (MIMO).
		Tres canales no súper puestos a 2.4 GHz.
802.11ac	2014	Velocidades de datos según el modo de operación hasta 1.3 Gbps.
		Sistema de múltiples antenas MIMO.

[21] Wi-Fi, diferentes protocolos y velocidades de datos - Intel. Obtenido de: https://www.intel.la/content/www/xl/es/support/network-and-i-o/wireless-networking/000005725.html

2.6 Protocolos de Internet

Según Sergio Lujan Mora [16], los protocolos de internet son un conjunto de protocolos de red en los que se basa internet y que permiten la transmisión de datos entre dispositivos interconectados.

En ocasiones a estos protocolos se los conoce como conjunto de protocolos TCP/IP en referencia a los dos protocolos más importantes que lo componen.

2.6.1 TCP/IP

TCP/IP, del inglés *Transmission Control Protocol / Internet Protocol*, es una familia de protocolos mantenidos por el Grupo de Trabajo de Ingeniería de Internet (*Internet Engineering Task Force*, IETF [10]) que define como los dispositivos electrónicos pueden ser conectados a internet y como la información será transmitida entre ellos.

Según Oracle [9], el modelo del protocolo TCP/IP está definido mediante una serie de capas muy similares a las del modelo OSI.

Tabla 6: Modelo TCP/IP[22].

Capa equivalente en el modelo OSI	Capa en TCP/IP	Ejemplos de protocolos de TCP/IP
Aplicación, sesión, presentación	Aplicación	DNS, FTP, HTTP, SMTP.
Transporte	Transporte	TCP, UDP.
Red	Internet	IPv4, IPv6.
Enlace de datos	Enlace de datos	PPP, IEEE 802.2.
Física	Red Física	Ethernet (IEEE 802.3), wifi (IEEE 802.11), Bluetooth (IEEE 802.15.1), RS-232, etc.

2.6.1.1 Capa de red física y capa de enlace de datos.

Al igual que en el modelo OSI, la capa física define las características del hardware que se utiliza en la red, mientras que la capa de enlace define al tipo de transferencia de los datos sobre la red, especifica al formato de la trama y al método de control de errores.

Debido a que TCP/IP no especifica a las capa inferiores de la red, estos niveles se pueden implementar mediante distintos protocolos. Los más utilizados en el mercado son el Ethernet y

[22] Tabla 1-2 Pila de protocolo TCP/IP - Guía de administración del sistema: servicios IP – Oracle. Obtenido de: https://docs.oracle.com/cd/E19957-01/820-2981/6nei0r0r6/index.html

el wifi, aunque numerosas aplicaciones también permiten realizar conexiones a internet a través de Bluetooth o conexiones RS-232.

2.6.1.2 Capa de internet

La capa de internet, también conocida como capa IP, acepta y transfiere paquetes por la red. Esta capa incluye a los siguientes protocolos:

- Protocolo de Internet (IP): Es el encargado de definir las direcciones IP de los distintos dispositivos conectados (*host*), de las comunicaciones entre *hosts* determinando el camino que debe seguir los datos dentro de la red, del agrupamiento de los datos formando paquetes y de la fragmentación de los mismos en caso de que su tamaño sea mayor al que puede ser transferido por la red.

- Protocolo de resolución de direcciones (ARP): Es el encargado de las comunicaciones entre la capa de internet y la cada de enlace de datos.

- Protocolo de mensajes de control de internet (ICMP): Es el encargado de detectar y registrar las condiciones de error de la red.

2.6.1.3 Capa de transporte

La capa de transporte es la encargada de garantizar que los paquetes lleguen en secuencia y sin errores intercambiando mensajes de confirmación de recepción y realizando el reenvío de los paquetes perdidos.

Esta capa incluye a los siguientes protocolos:

- Protocolo de control de transmisión (TCP): Es el encargado de establecer una comunicación fiable del tipo punto a punto entre las aplicaciones de dos dispositivos conectados a una red. Para conseguirlo, el protocolo agrega a los datos a transmitir una serie de encabezados que permiten establecer una conexión, determinar el orden de los paquetes, cerrar conexiones y establecer un mecanismo de confirmación de recepción.

- Protocolo de datagramas de usuario (UDP): Es el encargado de enviar mensajes sin verificar las conexiones entre el emisor y el receptor. Esto establece una comunicación poco fiable con un menor tráfico de datos.

- Protocolo de transmisión para el control de flujo (SCTP): Provee los mismos servicios que el protocolo TCP pero permite establecer conexiones entre múltiples hosts.

2.6.1.4 Capa de aplicación

La capa de aplicación define a las aplicaciones de red y a los servicios de internet estándar que puede utilizar el usuario.

2.6.2 Sistema de nombres de dominios

El Sistema de nombres de dominio o DNS, del inglés Domain Name System, es un protocolo de la capa de aplicación del modelo TCP/IP (IETF RFC-1034 [12], IETF RFC-1035) y administrado por *Internet Corporation for Assigned Names and Numbers* [11] (ICANN), define un sistema de nomenclaturas jerárquico y descentralizado para asociar información variada con un nombre de dominio.

Una de las aplicaciones de este sistema es la de traducir un nombre de dominio fácilmente memorizable a una dirección IP del tipo numérica, esto permite al usuario acceder a sitios web de forma sencilla.

Tabla 7: Equivalencia entre nombres de dominio y direcciones IP traducidas por un DNS.

Nombre	Nombre de dominio	Dirección IP (v4)
Google	www.google.com	172.217.28.238
UTN – FRLR	www.frlr.utn.edu.ar	190.114.205.4

Otra ventaja que presenta el sistema es la de poder cambiar la dirección IP de un sitio web sin tener que modificar al nombre de dominio, situación que generalmente ocurre al cambiar de proveedor de servicios de internet.

2.7 World Wide Web

Según Sergio Lujan Mora [16], la *World Wide Web* o simplemente web es un sistema de distribución de documentos de hipertexto o hipermedias accesibles mediante internet, desarrollado por Tim Berners-Lee en el CERN y actualmente mantenido por *World Wide Web Consortium* [13] (W3C).

La estructura de la web puede descomponerse en un sistema cliente/servidor[17], formado por los siguientes elementos:

- Un cliente, el cual es por lo general un navegador web que genera una petición.

- Uno o más intermediarios del mensaje, generalmente conocidos como proxis.

- Un servidor web que genera una respuesta y la envía de regreso al cliente.

Fig. 20: Estructura básica de la web.

El sistema de distribución está compuesto principalmente por el identificador de recursos uniformes (URI), el protocolo de transferencia de hipertexto (HTTP) y el lenguaje de marcado de hipertexto (HTML).

2.7.1 Identificador de recursos uniformes

El identificador de recursos uniformes o URI (IETF RFC-3986 [15]), del inglés *uniform resource identifier*, es una cadena de caracteres que identifica a los recursos (páginas web, documentos, imágenes, videos, etc.) de una red de forma univoca.

Un subconjunto de URI es el localizador de recursos uniformes (*Uniform Resource Locator*, URL) que además de proveer la identificación de un recurso permite su localización.

La sintaxis de un URI genérico consta de las siguientes partes:

- Esquema: Especifica los identificadores o el protocolo de acceso al recurso. Por ejemplo “http:”, “mailto:”, “ftp:”, etc.

- Doble Barra (//): Dependen del protocolo a utilizar, en algunos casos son opcionales.

- Autoridad: Está formada por una sección opcional conocida como autenticación en dónde se define un nombre de usuario y una contraseña, separados por el carácter “@” de una segunda sección donde se define al identificador de host por medio de su nombre de dominio o de su dirección IP.

- Ruta: Información jerárquica que determina la localización del recurso en el ámbito del dominio. Cada segmento de la cadena jerárquica debe ser separado con el carácter “/”.

- Consulta: Información iniciada por el carácter “?” con estructura formada por pares “atributo=valor” separados entre sí generalmente por el carácter “&”.

- Fragmento: Información iniciada por el carácter “#” que permite identificar a un recurso secundario, tal como a un título de cabecera dentro de una página web.

Tabla 8: Ejemplos de Identificadores de recursos uniformes.

Ejemplos de URI				
Esquema	Autoridad	Ruta	Consulta	Fragmento
http:	//ejemplo.com	/recursos		
https:	//172.152.0.14	/dia/hora	?min=2&seg=4	
ftp:	//jose:pass78@ejemplo.com			
http:	//ejemplo.com	/texto.html		#ejemplos
https://tools.ietf.org/html/rfc3986#section-1.1.3				
https:	// tools.ietf.org	/html/rfc3986		#section-1.1.3

2.7.2 Protocolo de transferencia de hipertexto

El protocolo de transferencia de hipertexto o HTTP, del inglés *Hypertext Transfer Protocol*, es un protocolo de la capa de aplicación del modelo TCP/IP (IETF RFC-7540 [14]) desarrollado por IETF y la W3C, define la transferencia de información entre los distintos elementos que componen a la arquitectura de la web.

El protocolo está orientado a transacciones y sigue un esquema de petición-respuesta entre un cliente y un servidor.

A la información enviada durante las transacciones se la conoce como mensajes HTTP y tiene la característica de ser un texto del tipo plano. Esto facilita su comprensión pero genera que el mensaje tenga una mayor longitud.

La estructura de un mensaje HTTP está compuesta por los siguientes campos:

- Línea inicial: En las peticiones indica el método de petición, el recurso solicitado y la versión HTTP que soporta el cliente. En las respuestas indica la versión del protocolo y un mensaje que indica que ha sucedido con la petición.

- Cabeceras: Metadatos utilizados por el cliente y el servidor.

- Cuerpo del mensaje: Es opcional, el contenido depende del tipo de petición.

Tabla 9: Ejemplo de un mensaje HTTP.

Ejemplo de un mensaje HTTP		
Campo	Petición	Respuesta
Primer línea	GET / HTTP/1.1	HTTP/1.1 200 OK
Cabeceras	Host: developer.mozilla.org Accept-language: en	Date: Wed, 17 Jul 2017 19:28:02 GTM Server: Apache Content-Length: 29769 Content-Type: text/html

Los métodos de petición definidos dependen de la versión del protocolo. Algunos de estos métodos son:

- GET: Pide al servidor por un recurso especifico. Por seguridad no debe ser usado por aplicaciones que causen efectos ya que trasmite la información de forma visible, es decir la URI contiene un campo de consulta.

- HEAD: Pide una respuesta idéntica a la de una petición GET pero sin el cuerpo de la respuesta.

- POST: Se utiliza para enviar información a un recurso específico, causando a menudo un cambio en el estado o un efecto en el servidor. Es utilizada en las tareas en las que el método GET no es adecuado, por ejemplo para realizar el envío de contraseñas.

Todas las características del protocolo definen al flujo de mensajes HTTP de la siguiente manera:

- El cliente inicia una conexión TCP.

- El cliente realiza una petición HTTP.

- El servidor realiza una respuesta.

- El cliente interpreta la respuesta.

- El cliente cierra la conexión TCP.

2.7.3 Lenguaje de marcado de hipertexto

El lenguaje de marcado de hipertexto[18], del inglés HyperText Markup Language, es un tipo de lenguaje de marcado desarrollado por Tim Berners-Lee y estandarizado por la W3C (ISO/IEC 15445) utilizado para la elaboración de páginas web.

Como todo lenguaje de marcado (el cual no debe ser confundido con un lenguaje de programación), además del texto que se desea mostrar en la página web, HTML incorpora marcas o etiquetas escritas entre un par de corchetes angulares (<etiqueta>) que permiten agregar información adicional al documento.

Una parte de la información adicional puede presentarse en forma de elementos externos (como imágenes, videos, archivos de audio) los cuales pueden ser agregados sin tener que ser insertados directamente en el código de la página. Para esto, el lenguaje incorpora la posibilidad de referenciar la ubicación del recurso en la red mediante texto (URL del recurso). Esta característica permite que el contenido de una página escrita en HTML sea solo texto, haciendo que la tarea de interpretar el código y unir todos los elementos referenciados para permitir la visualización de la página completa sea trabajo del navegador web.

El lenguaje HTML puede hasta cierto punto determinar la apariencia o el comportamiento de una página pero su tarea principal es la de definir el contenido y la estructura de la misma.

2.7.3.1 Elementos

Los elementos conforman la estructura básica del lenguaje. Un elemento genérico, por lo general, está formado por los siguientes campos:

- Etiqueta de inicio: Contiene al nombre del elemento y a sus atributos.

- Etiqueta de cierre: Según el elemento esta etiqueta estará o no presente. En su interior contiene al carácter “/” seguido del nombre del elemento.

- Contenido: Es la información que se encuentra entre las etiquetas de inicio y la de cierre.

Tabla 10: Elementos del lenguaje HTML.

Etiqueta de inicio	Contenido	Etiqueta de cierre
<nombre atributo=”valor”>	Texto del contenido	</nombre>
Ejemplo: Enlace a una dirección especifica.		
	UTN – La Rioja	

2.7.3.2 HTML básico

Una página básica posee los siguientes elementos en su estructura:

- Tipo de documento (DOCTYPE): Debe estar en la primera línea del documento y su función es la de informar al navegador web la versión del lenguaje que se utilizó para escribir el documento. No se considera como una etiqueta del lenguaje.

- Inicio del Documento (html): Indica al navegador web el inicio del documento.

- Cabecera (head): Define la cabecera del documento, la cual suele contener información relacionada al título de la página y a metadatos utilizados por el navegador.

- Cuerpo (body): Define el contenido principal del documento y todo aquello que el navegador web visualiza.

Tabla 11: Ejemplo de una página sencilla escrita en HTML5.

Página genérica	
Elementos	Descripción
<!DOCTYPE html>	Define un documento de HTML5.
<html>	Inicio del documento.
<head>	Inicio de la cabecera.
<title>Titulo de ventana </title>	Titulo mostrado en la ventana del navegador.
</head>	Cierre de la cabecera.
<body>	Inicio del cuerpo del documento.
Titulo Principal	Título principal visualizado en la pantalla del navegador.
Titulo Secundario	Titulo secundario visualizado en la pantalla del navegador.

Esto es un párrafo Que no tiene en cuenta a los saltos de línea, ni los espacios dobles.	La etiqueta “p” define párrafos del documento. El lenguaje HTML no tiene en cuenta los saltos de línea en el documento ni tampoco los espacios dobles.
Este párrafo contiene un salto de línea	La etiqueta produce un salto de línea, es un elemento especial que no posee una etiqueta de cierre.
</body>	Cierre del cuerpo.
</html>	Final del documento.

2.7.3.3 Referencia de Caracteres

Algunos caracteres presentan dificultades a la hora de su visualización, ya sea por que pertenecen a los caracteres especiales del lenguaje HTML o que debido a la codificación del documento escogida no formen parte de los caracteres disponibles. Para solucionar este inconveniente se debe hacer uso de referencias al carácter.

Existen dos tipos de referencias de caracteres:

- Referencias numéricas: Referenciando al carácter mediante su numeración en el sistema Unicode.

- Referencias a entidades HTML: Referenciando al carácter mediante un nombre que lo define. A este nombre se lo conoce como nombre de la identidad.

Tabla 12: Ejemplo de referencia de caracteres en HTML.

Carácter	Referencia numérica	Entidad
&	&#x0026	&amp
<	&#x003C	&lt
>	&#x003E	&gt

Para especificar el juego de caracteres con el cual el documento ha sido codificado podemos agregar información adicional a la cabecera de la página.

Tabla 13: Ejemplo de tipos de codificación utilizadas en HTML5.

Ejemplo en HTML5
<meta charset=”codificacion”>
Codificación UTF-8
<meta charset=”utf-8”>
Codificación Latin-1 (ISO-8859-1)
<meta charset=”ISO-8859-1”>

HTML5 tiene por defecto a la codificación UTF-8. Algunas aplicaciones pueden presentar inconvenientes con la versión estandarizada de esta codificación debido al *byte order mark* (BOM, Unicode xFEFF), por lo que en algunos casos es preferible retirar a este carácter codificando en UTF-8 sin BOM (no estandarizada).

2.7.4 JavaScript y CSS

Debido a que HTML tiene como objetivo principal definir la estructura y el contenido de una página web, se necesitan otros elementos para agregar funcionalidades y estilos a la presentación. Estas características adicionales son introducidas por JavaScript y por CSS respectivamente.

JavaScript [19] (JS) es un lenguaje de programación estandarizado (ECMAScript ISO/IEC 16262) cuya principal función es la de generar páginas web dinámicas y mejorar la interfaz de usuario. Posee una sintaxis similar a la del lenguaje C y convenciones adoptadas del lenguaje Java.

Cada uno de los programas escritos en JS recibe el nombre de Script y pueden ser agregados en la cabecera del documento HTML entre las etiquetas “script”.

CSS [20], del inglés *Cascading Style Sheets*, es un lenguaje de diseño gráfico utilizado para describir la presentación de los documentos estructurados escritos en un lenguaje de marcado (en este caso HTML). Está diseñado para poder definir una separación entre el contenido del documento y la forma en la será presentado.

A las distintas presentaciones de CCS se las conoce como estilos y pueden ser agregadas en la cabecera del documento HTML entre las etiquetas “style” o como atributos de las distintas etiquetas ubicadas en el documento.

Otro método para agregar estilos o Scripts al documento consiste en referenciarlos de forma externa, lo que permite su reutilización en múltiples documentos.

Tabla 14: Ejemplo de una página web sencilla que incluye estilos y scripts.

Ejemplo de página web genérica

<!DOCTYPE html>
<html>
<head>
<script>
function CambiarParrafo()
{
  document.getElementById("Parrafo1").innerHTML = "Párrafo modificado.";
}
</script>
 	
<style>
h2 { color: red;}
</style>
</head>
<body>
 Este título es de color rojo
 Párrafo inicial
 	
<button type="button" onclick="CambiarParrafo()">Modificar</button>
 	
</body>
</html>

2.7.5 Funcionamiento de la web

El funcionamiento básico de la web consiste en la ejecución de una serie de pasos por parte del sistema para obtener la información demandada por el usuario.

- El navegador realiza una consulta al servidor de DNS para traducir a la URL ingresada por el usuario en una dirección IP.

- Una vez conocida la dirección IP del servidor, el navegador le envía una petición HTTP solicitando al recurso especificado.

- El servidor procesa la solicitud y reenvía la información requerida.

- El navegador recibe los ficheros solicitados, los interpreta, renderiza y luego genera una presentación de la página web.

2.7.6 Programación del lado del servidor

Un servidor web, además de enviar un recurso específico, puede realizar un procesamiento a partir de las peticiones y generar lo que se conoce como una página web de contenido dinámico, cuya principal ventaja es la de poder interactuar con bases de datos.

La programación del lado del servidor consiste en la programación del procesamiento efectuado y se realiza mediante scripts escritos en un lenguaje específico para la tarea (PHP, Perl, etc.) ejecutados por un intérprete instalado en el servidor.

A la programación o escritura de la página web realizada por el usuario de forma directa se la conoce como programación del lado del cliente. Este tipo de programación genera páginas del tipo estáticas cuyo contenido permanecerá inalterado y debido a la ausencia de procesamiento, el consumo de los recursos informáticos del servidor será menor (lo que permite utilizar unidades de menores prestaciones).

2.8 GSM

Según Siegmund M.Redl [22], el sistema global de comunicaciones móviles, del inglés *Global System for Mobile communications* o simplemente GSM, es un estándar mantenido por la Asociación GSM (GSMA [21]) que define un sistema digital de comunicaciones móviles.

Las principales características del estándar son:

- Estándar abierto, no propietario y aun en desarrollo.

- Canales de voz y datos digitales.

- Mayor seguridad en la transmisión de información.

- Itinerancia y facilidad de cambio de operador sin tener que cambiar de dispositivo.

2.8.1 Arquitectura de la red GSM

La arquitectura de la red GSM se divide en varios subsistemas según su función:

- Estación móvil (*Mobile station*, MS): Formado por la el dispositivo móvil y la tarjeta SIM del cliente del servicio.

- Estación Base (*Base station subsystem*, BSS): Formado por las antenas y su controlador. Proporciona el acceso de los dispositivos móviles al espectro disponible y controla el envío y la recepción de los datos.

- Subsistema de red y conmutación (*Network and Switching Subsystem*, NSS): También conocido como núcleo de red (*core network*, CN) es la capa lógica del enrutamiento y de almacenamiento de datos. Permite establecer conexiones entre dispositivos de la misma red o de redes distintas.

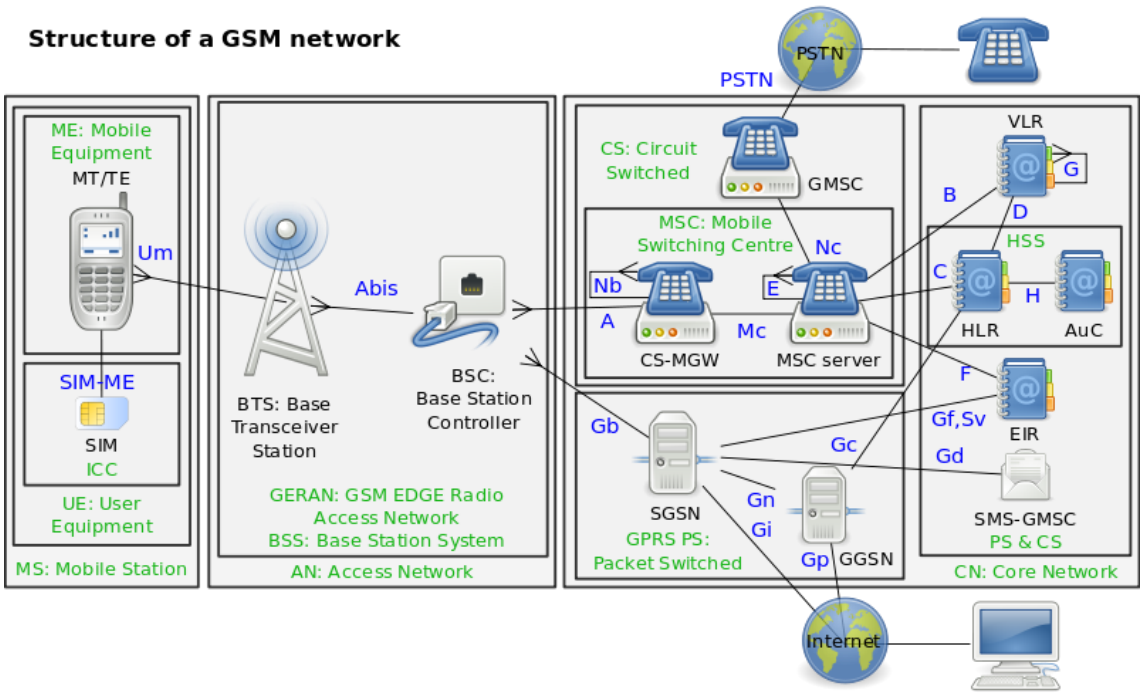

Fig. 21: Estructura de una red GSM[23].

2.8.2 Bandas de frecuencias

Como toda interfaz basada en comunicaciones inalámbricas, para poder operar junto a los demás servicios sin generar interferencias, las frecuencias de trabajo deben estar comprendidas en bandas definidas.

El estándar utiliza bandas diferentes para la subida y para la bajada de datos.

Tabla 15: Bandas de frecuencia utilizadas por GSM en Argentina.

Banda	Subida de datos (MHz)	Bajada de datos (MHz)	Notas
GSM 850	824 – 849	869 – 894	
GSM 1900	1850 – 1910	1930 – 1990	Incompatible con GSM1800 por solapamiento de bandas.

[23] Gsm structures - Tsaitgaist. Obtenido de: https://en.wikipedia.org/wiki/File:Gsm_structures.svg

2.8.3 Tarjetas SIM

La SIM o módulo de identidad del suscriptor es una tarjeta inteligente y desmontable que contiene almacenada la información del usuario del servicio, los parámetros de la red y el directorio telefónico.

La tarjeta brinda al estándar de una de sus características principales, la capacidad de cambiar de operador de servicio sin tener que cambiar de dispositivo móvil.

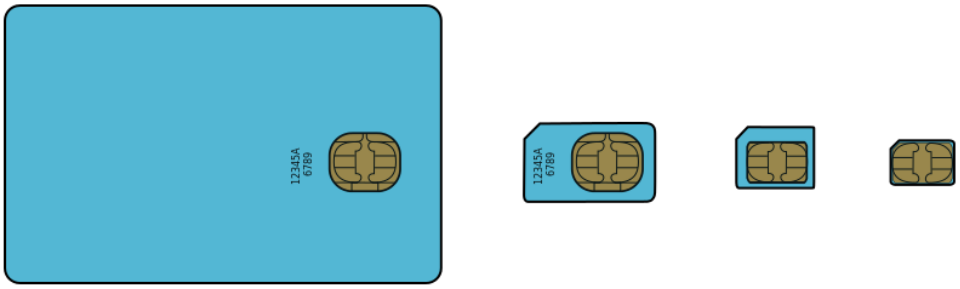

Fig. 22: Distintos formatos de las tarjetas SIM[24].

2.8.4 Servicio de mensajes cortos

El SMS, del inglés *Short Message Service*, es un servicio disponible en los dispositivos móviles de una red GSM. Permite al usuario enviar una cadena de hasta 140 caracteres de 8 bits. Los parámetros que constituyen a un mensaje de texto son los siguientes:

- Fecha de envío.

- Validez del mensaje.

- Número de teléfono del remitente y del destinatario.

- Numero de centro de mensajes cortos (SMSC).

2.9 GPS, NMEA 0183

Según la información oficial del gobierno de los Estados Unidos [23], el sistema de posicionamiento global, del inglés *Global Positioning System*, es un sistema desarrollado por el departamento de defensa de los Estados Unidos para determinar la ubicación de un dispositivo receptor en la superficie de la tierra. El sistema se compone por los satélites en órbita, las estaciones de control terrestres y por los dispositivos receptores.

El funcionamiento consiste en la transmisión de señales por parte de los satélites que los dispositivos receptores reconocen y utilizan para determinar la ubicación y la hora actual.

Las principales características del GPS son las siguientes:

[24] GSM SIM card evolution - Justin Ormont. Obtenido de: https://commons.wikimedia.org/wiki/File:GSM_SIM_card_evolution.svg

- Constelación de 24 satélites en órbita.

- Cobertura mundial.

- Capacidad ilimitada de dispositivos receptores.

Para la comunicación de los datos obtenidos por el dispositivo receptor, el mismo utiliza el protocolo NMEA 0183 [24], el cual es definido y controlado por la organización estadounidense *National Marine Electrinics Association* que especifica la comunicación entre instrumentos marinos y la mayoría de los receptores GPS.

El estándar especifica las siguientes características en la comunicación:

- Señales eléctricas requeridas.

- Velocidad de transferencia.

- Tiempos de actualización de los datos.

- Un emisor y uno o múltiples receptores en el bus.

- Información imprimible en formato ASCII.

2.9.1 Estructura de un mensaje estándar

La estructura de un mensaje enviado por un receptor GPS a un dispositivo de visualización consiste en una cadena de caracteres iniciados con "$" y seguidos con el campo de cabecera. Cada campo del mensaje se encuentra separado por comas y finaliza con un salto de línea y un retorno de carro.

NMEA soporta distintos tipos de mensajes, un ejemplo es el de DTM, utilizado para informar la Información del código de referencia.

Tabla 16: Ejemplo de un mensaje enviado por un GPS en formato NMEA[25].

Mensaje:					
Estándar		$GPDTM,LLL,LSD,LAT,N/S,LON,E/W,ALT,RRR*CS<CR><LF>			
Ejemplos		$GPDTM,W84,,0.0,N,0.0,E,0.0,W84*6F			
		$GPDTM,W72,,0.00,S,0.01,W,-2.8,W84*4F			
		$GPDTM,999,CH95,0.08,N,0.07,E,-47.7,W84*1C			
Campo	Ejemplo	Formato	Nombre	Unidad	Descripción
0	$GPDTM	Cadena de Caracteres	GPDTM	-	Cabecera del mensaje, identificador DTM.
1	W72	Cadena de Caracteres	LLL	-	Código de referencia local.
2	-	Cadena de Caracteres	LSD	-	Subdivisión del código de referencia local.
3	0.08	Numérico	LAT	Minutos	Latitud.

[25] Receiver Description Protocol Specification - U-blox. Obtenido de: https://www.u-blox.com/sites/default/files/products/documents/u-blox6_ReceiverDescrProtSpec_%28GPS.G6-SW-10018%29_Public.pdf

4	S	Carácter	NS	-	Indicador de Norte o Sur.
5	0.07	Numérico	LON	Minutos	Longitud.
6	E	Carácter	EW	-	Indicador de Este u Oeste.
7	-2.8	Numérico	ALT	Metros	Altitud
8	W84	Cadena de Caracteres	RRR	-	Código de referencia.
9	*67	Hexadecimal	CS	-	Suma de verificación.
10	-	Carácter	<CR><LF>	-	Retorno de carro y nueva línea.

Para aplicaciones en las cuales se requiera hacer uso de los datos obtenidos en lugar de solo visualizarlos, se necesita aplicar un filtro al mensaje recibido para conseguir la información necesaria.

2.10 Comandos Hayes

El conjunto de comandos Hayes o comandos AT es un protocolo abierto que especifica un método de comunicación con periféricos mediante órdenes simples. Originalmente fue desarrollado por la compañía *Hayes Communications* y luego estandarizado por International Telecommunication Union (ITU-T REC V.250 [25]) para permitir la configuración de módems. Actualmente ha sido adoptado por la mayoría de los proveedores de módulos GSM [26].

Según el parámetro a configurar, un mensaje genérico puede o no contener los siguientes campos:

Tabla 17: Estructura básica de un comando AT.

Inicio	+	Parámetro	=	Valor	Retorno de Carro
AT	+	COPS	=	?	<CR>

Los comandos AT especifican los mensajes u órdenes para realizar configuraciones, mientras que para establecer una comunicación el protocolo que era utilizado es conocido como RS-232 (TIA/EIA-232). Este protocolo define a la capa física de una red punto a punto entre el modem y la computadora, mientras que utiliza una comunicación UART para enviar y recibir información.

Los módulos GSM de la actualidad, a diferencia de los módems antiguos, no implementan al protocolo RS-232 pero si utilizan a la comunicación UART para realizar la transferencia de datos.

2.11 Bibliografía del capítulo

[1] Sistemas de comunicación electrónicas - Wayne Tomasi - Pearson Educación. (ISBN: 970-26-0316-1).

[2] Redes de computadoras - Andrew Tanenbaum, David Wetherall - Pearson. (ISBN: 978-607-32-0817-8).

[3] CAN Literature - Bosch Semiconductors & Sensors. Disponible en internet, fecha de acceso: 05/07/2017. http://www.bosch-semiconductors.de/en/automotive_electronics/ip_modules/can_literature_2.html

[4] Comunicación Serial: Conceptos Generales - National Instruments. Disponible en internet, fecha de acceso: 06/07/2017. http://digital.ni.com/public.nsf/allkb/039001258CEF8FB686256E0F005888D1

[5] Using the Universal Asynchronous Receiver Transmitter (UART) eTPU Function, Freescale Semiconductor. Disponible en internet, fecha de acceso: 06/07/2017. http://www.nxp.com/docs/en/application-note/AN2853.pdf

[6] SPI Block Guide V03.06 (Revisión: 04/02/2003) - Motorola, Inc. Disponible en internet, fecha de acceso: 07/07/2017. https://web.archive.org/web/20150413003534/http://www.ee.nmt.edu/~teare/ee308l/datasheets/S12SPIV3.pdf

[7] 802.11 Wireless Networks, The Definitive Guide - Matthew Gast - O'Reilly. (ISBN: 0-596-10052-3). Páginas: 31-33.

[8] IEEE 802.11-2016 - Institute of Electrical and Electronics. Disponible en internet, fecha de acceso: 10/07/2017. http://standards.ieee.org/getieee802/download/802.11-2016.pdf

[9] Guía de administración del sistema: servicios IP – Oracle. Disponible en internet, fecha de acceso: 11/07/2017. https://docs.oracle.com/cd/E19957-01/820-2981/6nei0r0r6/index.html

[10] Getting Started in the IETF – IETF. Disponible en internet, fecha de acceso: 11/07/2017. https://www.ietf.org/newcomers.html

[11] ¿Que hace ICANN? - Internet Corporation for Assigned Names and Numbers. Disponible en internet, fecha de acceso: 11/07/2017. https://www.icann.org/resources/pages/what-2012-02-25-es

[12] RFC-1034 - Internet Engineering Task Force. Disponible en internet, fecha de acceso: 11/07/2017. https://www.ietf.org/rfc/rfc1034.txt

[13] Help and FAQ - W3C. Disponible en internet, fecha de acceso: 11/07/2017. https://www.w3.org/Help/#activity

[14] RFC-7540 - Internet Engineering Task Force. Disponible en internet, fecha de acceso: 11/07/2017. https://tools.ietf.org/html/rfc7540

[15] RFC-3986 - Internet Engineering Task Force. Disponible en internet, fecha de acceso: 11/07/2017. https://tools.ietf.org/html/rfc3986

[16] Programación de aplicaciones web: historia, principios básicos y clientes web - Sergio Lujan Mora - Editorial Club Universitario. (ISNB: 978-84-8454-206-3). Capítulo 2, Historia de internet. Disponible en internet, fecha de acceso: 12/07/2017. https://rua.ua.es/dspace/bitstream/10045/16995/1/sergio_lujan-programacion_de_aplicaciones_web.pdf

[17] Programación en Internet: clientes web - Sergio Lujan Mora - Editorial Club Universitario. (ISNB: 978-84-8454-118-9). Capítulo 1, Arquitecturas cliente/servidor. Disponible en internet, fecha de acceso: 12/07/2017. https://rua.ua.es/dspace/bitstream/10045/16994/1/sergio_lujan-programacion_en_internet_clientes_web.pdf

[18] HTML 5.1 W3C Recommendation, 1 November 2016 - W3C. Disponible en internet, fecha de acceso: 12/07/2017. https://www.w3.org/TR/html51/

[19] JavaScript Guide - Mozilla Foundation. Disponible en internet, fecha de acceso: 12/07/2017. https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Introduction#What_is_JavaScript

[20] Guía Breve de CSS - W3C. Disponible en internet, fecha de acceso: 12/07/2017. http://www.w3c.es/Divulgacion/GuiasBreves/HojasEstilo

[21] The Creation of Standards for Global Mobile Communication - GSM, UMTS and LTE from 1982 to 2012, Global System for Mobile communications. Disponible en internet, fecha de acceso: 09/07/2017. https://www.gsma.com/aboutus/gsm-technology/gsm

[22] GSM and Personal Communications Handbook - Siegmund M.Redl, Matthias K, Weber, Malcolm W. Oliphant – Artech House. (ISBN: 0-89006-957-3). Disponible en internet, fecha de acceso: 09/07/2017. https://gctjaipur.files.wordpress.com/2015/08/gsm-and-personal-communications.pdf

[23] Sistema de Posicionamiento Global - GPS.GOV. Disponible en internet, fecha de acceso: 09/07/2017. http://www.gps.gov/spanish.php

[24] NMEA 0183 Standard - National Marine Electronics Association. Disponible en internet, fecha de acceso: 09/07/2017. https://www.nmea.org/content/nmea_standards/nmea_0183_v_410.asp

[25] Recommendation V.250 - International Telecommunication Union. Disponible en internet, fecha de acceso: 28/08/2017. http://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-V.250-200307-I!!PDF-

E&type=items

[26] Human Factors (HF); AT Commands for Assistive Mobile Device Interfaces. Disponible en internet, fecha de acceso: 28/08/2017. http://www.etsi.org/deliver/etsi_ts/102500_102599/102511/01.01.01_60/ts_102511v010101p.pdf

3 Microcontroladores

3.1 Introducción a los microcontroladores

Siguiendo a Jan Axelson[1], un microcontrolador es una computadora de un solo chip, esto significa que en un espacio reducido contiene todo lo necesario para la ejecución de instrucciones almacenadas en su memoria, lo que le permite realizar el control de objetos, procesos o eventos.

Al igual que una computadora personal, un microcontrolador contiene una serie de unidades básicas en su interior, pero debido a que todas ellas se encuentran en un único chip poseen características reducidas. Por ejemplo, un microcontrolador tendrá una menor cantidad de memoria que una computadora personal o su unidad central de procesamiento operara a una menor frecuencia.

El campo de aplicación de los microcontroladores es muy variado, especialmente en aquellas tareas donde se requieran las siguientes características:

- Bajo consumo energético.

- Menor costo.

- Espacio reducido.

- Tareas repetitivas y de poco procesamiento.

3.2 Unidades básicas

Un microcontrolador generalmente está formado por una serie de unidades básicas, las cuales se encuentran interconectadas entre sí a través de un bus o canal de comunicaciones:

- Unidad central de procesamiento (CPU): Está compuesta por la unidad aritmética-lógica (ALU), la unidad de control y los registros de trabajo. Es la encargada de ejecutar las instrucciones almacenadas en la memoria.

- Memoria: Podemos distinguir dos tipos de memoria, la memoria de programa, utilizada para almacenar a las instrucciones que deben ejecutarse, y la memoria de datos, destinada a guardar información utilizada durante la ejecución. Según el tipo de microcontrolador, la memoria de datos podrá ser volátil (RAM) o estática (ROM).

- Entradas y salidas digitales: Está formada por pines que permiten la comunicación del microcontrolador con el exterior mediante señales del tipo digital. En conjunto conforman lo que se conoce como puertos de comunicación paralelos.

- Interfaces: Permiten la comunicación del microcontrolador con el exterior mediante transmisiones del tipo serie. En la mayoría de los casos, además de permitir la transferencia de información son utilizadas para la programación del microcontrolador.

- Unidades adicionales: Confieren al microcontrolador características que le permiten realizar tareas de procesamiento de manera conjunta con la CPU, o agregan nuevas funcionalidades. Algunos ejemplos de este tipo de unidades son los módulos contadores o temporizadores, los

conversores analógicos, las unidades de depuración y los coprocesadores para el cálculo de punto flotante.

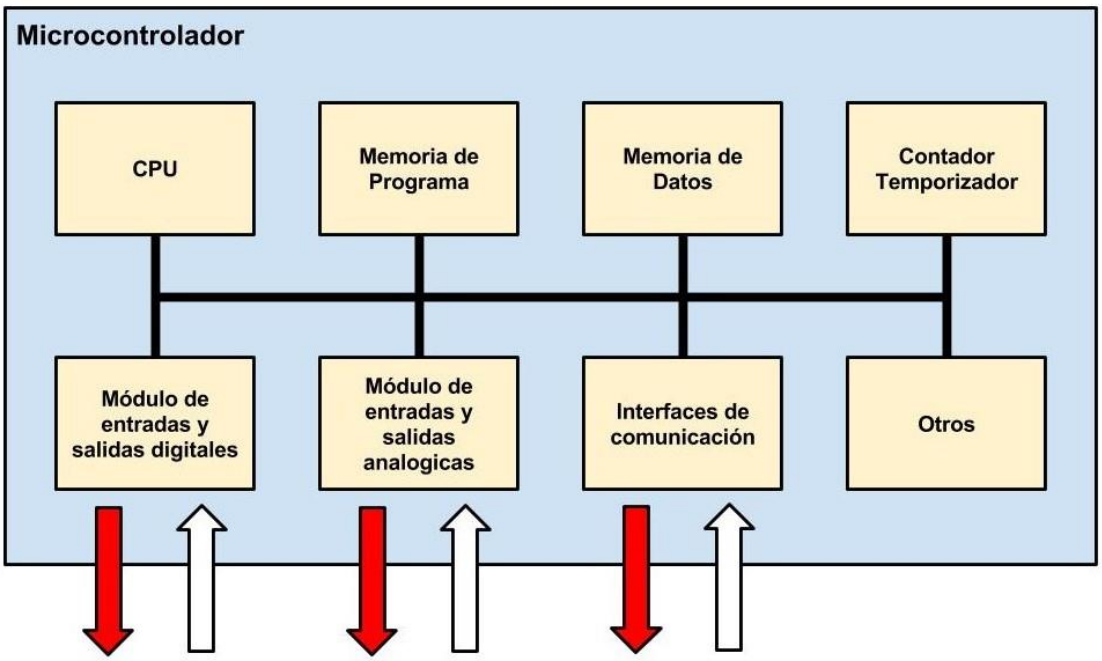

Fig. 23: Unidades básicas de un microcontrolador.

3.3 Etapas de desarrollo de un proyecto

Todo proyecto que demande de la utilización de un microcontrolador estará formado por una serie de etapas o fases en su desarrollo. En algunos casos, muchas de ellas serán omitidas o realizadas más de una vez.

Como en todo proyecto, la primera de ellas consiste en definir a la idea que dará solución al problema planteado. Determinar correctamente los requerimientos del problema conduce a escoger un microcontrolador con prestaciones de hardware adecuadas, lo cual se traduce en un menor costo total.

3.3.1 Diseño y construcción

Durante la fase de diseño se escoge al microcontrolador y se determina a los componentes que lo acompañaran en el circuito. Debido a que existe una gran variedad de microcontroladores en el mercado, un criterio para facilitar la elección consiste en evaluar sus principales características en función de la tarea a realizar:

- Poder de cálculo: Según la tarea demande un mayor o menor poder de cálculo podemos escoger a los procesadores según la cantidad de instrucciones que pueden ejecutar por segundo (medida en Mips), por el tamaño de los datos con los que trabaja (bits, nypples, bytes, words, etc.) o por el conjunto de operaciones que tienen disponible.

- Capacidad de memoria: Determina la cantidad de espacio disponible para el almacenamiento de instrucciones y por lo tanto el tamaño máximo del programa.

- Tipo de memoria: Determina la cantidad de ciclos de escritura y borrado que el dispositivo tiene disponibles además del método utilizado para realizarlos. Por ejemplo, las memorias del tipo EPROM tienen típicamente 100 ciclos disponibles y el método para realizar el borrado consiste en la exposición del microcontrolador a luz ultravioleta, mientras que las memorias del tipo FLASH tienen entre 10 000 y 100 000 ciclos y el borrado se realiza eléctricamente.

- Cantidad de pines de entrada y salida.

- Encapsulado: Determina la geometría del microcontrolador y la distribución de sus pines. Por ejemplo, el DIP (*Dual In-line Package*), el TQFP (*thin Quad Flat Package*), etc.

- Interfaces: Determina los protocolos de comunicación que es capaz de manejar el microcontrolador. Por ejemplo, la programación de algunos microcontroladores se realiza mediante protocolos especiales (ICSP, ISP, etc.) o UART, mientras que incorporan la posibilidad de comunicarse mediante CAN, UART, SPI, etc.

- Unidades adicionales: Determinan las funcionalidades extras que poseerá el microcontrolador, como la capacidad de interactuar con señales analógicas o la cantidad y el tamaño de los contadores internos.

- Consumo de energía.

- Rango de temperatura de trabajo.

- Rango de tensiones de trabajo.

3.3.2 Programación

Existen varias opciones para realizar un programa, cada una presenta distintos niveles de dificultad a la hora de escribir el código:

- Lenguaje de máquina: Esta formado directamente por las instrucciones escritas en binario. Es el único lenguaje que el microcontrolador puede ejecutar.

- Lenguaje de ensamblador: Este lenguaje permite reemplazar a las instrucciones escritas en binario por mnemónicos, lo cual simplifica la escritura y la comprensión de programas ya escritos. Debido a que el microcontrolador solo ejecuta lenguaje de máquina, existen programas llamados Ensambladores encargados de realizar la traducción entre ambos lenguajes.

- Lenguajes de alto nivel: Una desventaja del lenguaje ensamblador radica en que cada familia de microcontroladores posee su propio juego de nemónicos, lo cual significa que el usuario deberá aprender un nuevo vocabulario cada vez que desee programar una familia diferente. Una forma de resolver este inconveniente es recurrir a un lenguaje de alto nivel como C, C++ o Python (entre otros) y a un programa encargado de traducirlo a lenguaje de maquina conocido como Compilador.

Generalmente si el programa fue escrito correctamente teniendo en cuenta a todos los recursos del microcontrolador escogido, el lenguaje de maquina es aquel con el cual se conseguirá una mayor velocidad para la realización de las tareas y el que producirá un programa de menor tamaño. Por el contrario, un lenguaje de alto nivel facilita la escritura del programa pero genera un código de mayor longitud y de menor velocidad.

3.3.3 Testeo y depuración

Luego de escribir el programa o parte del mismo, se realizan pruebas para determinar su correcto funcionamiento. Al proceso de buscar errores en el programa y corregirlos se lo conoce como depuración y puede realizarse de diferentes maneras:

- Prueba en circuito: Consiste en grabar el programa en el microcontrolador y observar su funcionamiento directamente junto al resto del sistema.

- Depuración en circuito: Consiste en agregar líneas de código al programa o incluir un dispositivo externo que permita la monitorización del microcontrolador durante su funcionamiento en el circuito.

- Simulador: Consiste en un software para computadoras que permite simular al microcontrolador junto al circuito y observar el funcionamiento del programa.

3.4 Microcontroladores del sistema

Las tareas que el sistema debe realizar para implementar la totalidad de sus funciones definen las características que debe poseer el microcontrolador a utilizar, siendo mayoría aquellas que involucran a los datos enviados y recibidos por las interfaces de comunicación.

Debido a que las principales operaciones realizadas sobre los datos consisten en la lectura, el análisis, la edición, la escritura y la realización de operaciones matemáticas sobre los valores numéricos codificados en caracteres, y a que el volumen de los datos ingresados supera considerablemente al tamaño de los buffers de la mayoría de los microcontroladores, el procesamiento que debería ser realizado por una sola unidad resulta excesivo, pudiendo ocasionar una pérdida de información.

Una situación que ejemplifica la pérdida de datos seria durante la llegada simultanea de los mensajes provenientes de un GPS que utilice al protocolo NMEA y de un módulo wi-fi ante una petición del tipo POST, debido a que en ambos mensajes la información contenida supera los 400 bytes y por lo tanto, se requieren de al menos 400 instrucciones solo para realizar operaciones de lectura.

De esta manera, durante el desarrollo del sistema, las distintas funcionalidades a implementar serán divididas en dos microcontroladores, lo que implica que sus características técnicas pueden ser diferentes.

En el caso del primer microcontrolador, el mismo debe ser de alto nivel y poseer a los controladores de las distintas interfaces a utilizar, ya que será el encargado de la gestión de las tareas de comunicación. Mientras que el segundo, debe ser un microcontrolador capaz de realizar operaciones matemáticas de manera veloz.

Además de las características técnicas requeridas para cada microcontrolador, se deben tener en cuenta a una serie de agregados que faciliten las distintas tareas realizadas durante cada etapa de desarrollo:

- Herramientas que permiten su programación en lenguajes de alto nivel.

- Conectores que permiten el desarrollo del sistema en placas de pruebas (*protoboards*).

- Disponibilidad en el mercado local.

A partir de lo planteado, los microcontroladores seleccionados para conformar el sistema son:

- ATMEL ATSAM3X8E de la placa Arduino Due: Utilizado como microcontrolador principal, es el encargado de realizar todas las tareas de comunicación y de gestión de datos.

- DsPIC30F4013: Su principal tarea es la de convertir de manera constante, los parámetros entregados por un módulo GPS, al sistema de coordenadas requerido por las aplicaciones implementadas.

Parte de la información entregada por el GPS al segundo microcontrolador contiene al valor de la velocidad a la cual es sistema se desplaza, por lo que la misma puede ser suministrada al usuario mediante un visualizador.

El dispositivo visualizador puede variar según las necesidades del usuario, por lo que su única restricción de diseño será la de poder leer la información suministrada por el segundo microcontrolador. Durante el desarrollo del proyecto, este dispositivo será construido a partir de un tercer microcontrolador de características básicas, un PIC18F2550, encargado de mostrar los datos en una pantalla.

3.5 Microcontroladores PIC

A partir de lo publicado por Andrés Saravia[2], los PIC son microcontroladores pertenecientes a un conjunto de familias fabricadas por Microchip Technology, los cuales presentan una serie de características en común:

- Arquitectura Harvard: El programa y los datos se almacenan en distintos bloques de memoria a los cuales se puede acceder por canales o buses separados.

- RISC: Cuentan con un reducido conjunto de instrucciones de un tamaño fijo (la cantidad de bits que ocupa cada instrucción es constante).

- Cantidad de pulsos de reloj por instrucción constante.

- Registros de puertos y unidades adicionales mapeados en memoria.

Cada familia posee características adicionales que la distinguen y le confieren un campo de aplicación distinto.

3.5.1 Familias de PIC

Las familias de microcontroladores PIC se clasifican según el tamaño en bits de los datos con los que operan sus instrucciones[3].

3.5.1.1 Microcontroladores de 8 bits

Los microcontroladores que operan con datos de 8 bits de longitud pueden sub clasificarse en varios grupos según la cantidad de bits que ocupa cada instrucción:

- 12 bits: Comprende a la familia PIC10 y a algunos microcontroladores de la PIC12 y PIC16. Son considerados la gama baja. Tienen como principal característica a su reducido tamaño y a una pequeña cantidad de pines (8 para PIC10 y PIC12).

- 14 bits: Comprende a la familia PIC12 y PIC16. Implementan la capacidad de manejar interrupciones.

- 16 bits: Comprende a la familia PIC18. El juego de instrucciones esta optimizado para la programación utilizando lenguajes de alto nivel.

3.5.1.2 Microcontroladores de 16 bits

Existen dos familias que componen a este grupo de microcontroladores:

- PIC24: Esta familia comprende a los microcontroladores de propósito general, combinan un bajo consumo de potencia con un procesamiento de hasta 70 MIPS.

- DSPIC: Se presenta como una extensión de la familia PIC24. Implementa características que le permiten realizar operaciones de procesamiento de señales (DSP).

3.5.1.3 Microcontroladores de 32 bits

Solo una familia compone a este grupo, la PIC32. La misma representa a la gama alta de los microcontroladores PIC. Se caracterizan por un alto poder de procesamiento y una gran capacidad de memoria.

3.5.2 Características del microcontrolador dsPIC30F4013

El dsPIC30F4013[4] es un microcontrolador de la familia PIC que incorpora operaciones de procesamiento de señal y se encuentra disponible en varios tipos de encapsulados.

Tabla 18: Características principales del microcontrolador dsPIC30F4013.

Unidad	Características
CPU	Máxima cantidad de instrucciones por segundo de 30MIPs.
	84 instrucciones optimizadas para el uso de compiladores de lenguaje C.
	Tamaño de datos de 16 bits.
	Instrucciones especiales que permiten realizar operaciones de multiplicación y división directamente (DSP).
	Memoria de programa tipo Flash de 48KB con 10 000 ciclos de escritura y borrado.

Memoria	Memoria de datos EEPROM de 1KB con 100 000 ciclos de escritura y borrado.
	Memoria de datos RAM de 2KB.
Pines de entrada y salida	30.
Interfaces y cantidad de conexiones	ICSP (*In-Circuit Serial Programming*, utilizado para la programación) x1.
	SPI (soporta todos los modos de trabajo) x3.
	CAN extendido (2.0b) x1.
	UART x2.
	I2C x1.
Unidades Adicionales	Convertidor de señales analógicas a digitales de 12 bits y 200 ksps.
	Cuatro contadores de 16 bits que pueden combinarse para formar dos de 32 bits.
	Cuatro módulos CCP con PWM de 16 bits de resolución.
Rango de temperatura	-40ºC a 125ºC.
Rango de tensión	2.5V a 5.5V.
Encapsulados	DIP, TQFP y QFN.

3.5.3 Características del microcontrolador PIC18F2550

El PIC18F2550 es un microcontrolador de la familia PIC de gama media, incorpora características que reducen su consumo energético y lo hacen ideal para aplicaciones de conectividad. Se encuentra disponible en varios tipos de encapsulados.

Tabla 19: Características principales del microcontrolador PIC18F2550.

Unidad	Características
CPU	Máxima cantidad de instrucciones por segundo de 12MIPs.
	Instrucciones optimizadas para compiladores de lenguaje C.
	Tamaño de datos de 8 bits.
Memoria	Memoria de programa tipo Flash de 32KB con 100 000 ciclos de escritura y borrado.
	Memoria de datos EEPROM de 256 bytes con 1 000 000 ciclos de escritura y borrado.

	Memoria de datos RAM de 2KB.
Pines de entrada y salida	24.
Interfaces y cantidad de conexiones	ICSP (*In-Circuit Serial Programming*, utilizado para la programación) x1.
	SPI (soporta todos los modos de trabajo) x1.
	USB V2.0 x1.
	UART x1.
	I2C x1.
Unidades Adicionales	Convertidor de señales analógicas a digitales de 10 bits.
	Tres contadores de 16 bits y uno de 8 bits.
	Dos módulos CCP con PWM de 10 bits de resolución.
Rango de temperatura	-40ºC a 85ºC.
Rango de tensión	2V a 5.5V.
Encapsulados	DIP, TQFP y QFN.

3.6 Arduino

Siguiendo a Massimo Banzi[5], Arduino es una plataforma de código abierto de fácil utilización formada por dos partes. La primera corresponde a las placas físicas que integran en un circuito a todo lo necesario para el funcionamiento de un microcontrolador, mientras que la segunda corresponde al IDE (*Integrated Development Environment*) o entorno de desarrollo integrado, el cual es un software que permite la programación del microcontrolador desde una computadora personal.

3.6.1 Placas

Cada una de las distintas placas de la plataforma está formada por una serie de componentes que permiten el funcionamiento de un microcontrolador, su programación y su interacción con el mundo exterior:

- Microcontrolador: Generalmente de la familia AVR del fabricante ATMEL. El modelo y su encapsulado pueden variar según la placa y su revisión.

- Regulador de voltaje: En la mayoría de los casos el microcontrolador puede ser alimentado por una fuente externa o directamente desde el mismo USB de la computadora, por lo que la tensión debe ser regulada a un valor adecuado para el microcontrolador utilizado.

- Conectores de entrada y salida: Permiten realizar la conexión del microcontrolador con los distintos periféricos o elementos con los que se desee interactuar.

- Programador: Arduino reemplaza al protocolo nativo de programación de ATMEL (ISP) por un sistema formado por un convertidor USB a UART y un bootloader precargado en el microcontrolador. El bootloader es un pequeño programa que permite almacenar en una sección vacía de la memoria a las instrucciones que son enviadas por la computadora a través del USB. Una vez que el proceso de programación ha terminado, este sistema permite al microcontrolador comunicarse con la computadora durante su funcionamiento.

- Componentes adicionales: Además del cristal utilizado para generar a la señal de reloj del microcontrolador, como todo circuito electrónico, la placa cuenta con componentes discretos como resistencias, capacitores o diodos.

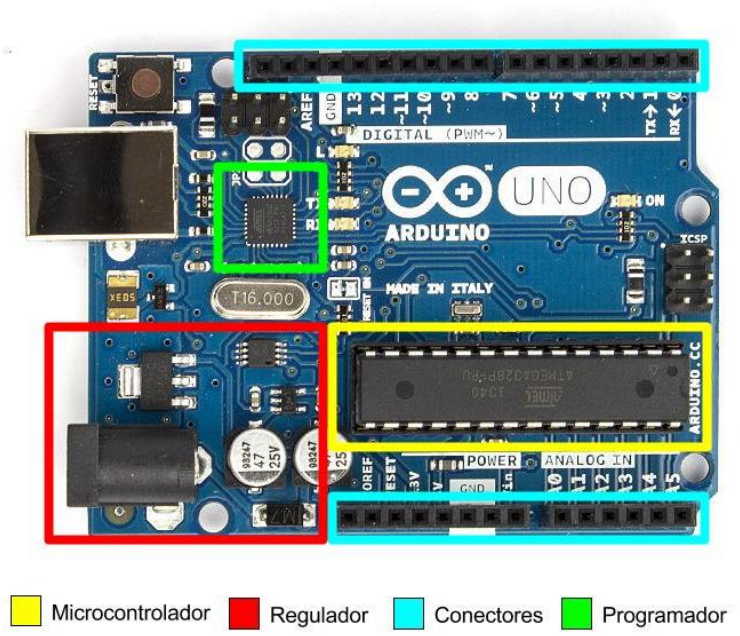

Fig. 24: Componentes principales de la placa Arduino Uno[26].

3.6.2 Entorno de desarrollo integrado

El IDE de Arduino es el software que permite al usuario escribir o editar programas, compilar y grabar en la memoria del microcontrolador. Además agrega opciones para configurar el entorno o gestionar librerías e incluye herramientas que facilitan la comunicación de la computadora con la placa.

[26] ArduinoUno R3 Front 450px - R.hampl (2015). Obtenido de: https://commons.wikimedia.org/wiki/File:ArduinoUno_R3_Front_450px.jpg

3.6.2.1 Edición del programa

Cada programa es conocido como sketch y está escrito en un lenguaje basado en Processing (similar a C++), el cual soporta a las funciones de C y C++.

Los sketches pueden estar formado por uno o varios archivos ubicados en un mismo directorio y, al igual que un programa escrito en C, puede incluir librerías y agregar opciones de pre procesamiento.

Un sketch básico posee una serie de secciones o partes entre las cuales podemos destacar a dos:

- Sección de configuración: Esta parte del programa es ejecutada solo durante el inicio del sistema y generalmente es utilizada para escribir las rutinas de configuración de la placa.

- Sección de bucle principal: Esta parte del programa es ejecutada una y otra vez de forma indefinida cuando las rutinas de la sección de configuración finalizan.

3.6.2.2 Compilado y programación

Una vez que el sketch ha sido escrito correctamente, el IDE recurre al compilador para traducirlo a lenguaje de máquina. Debido a que el lenguaje de maquina varía según cada microcontrolador, las opciones de configuración del entorno permiten elegir al adecuando.

El código de maquina es transferido por USB al convertidor ubicado en la placa, el cual gracias a los controladores (drivers) instalados en la computadora es reconocido como un puerto serie o COM.

3.6.2.3 Monitor serie

El monitor serie es una de las herramientas adicionales del IDE, permite establecer una comunicación con un puerto COM y por lo tanto también con el convertidor ubicado en la placa. Entre las opciones que tiene disponible permite definir la velocidad en baudios de la trama UART.

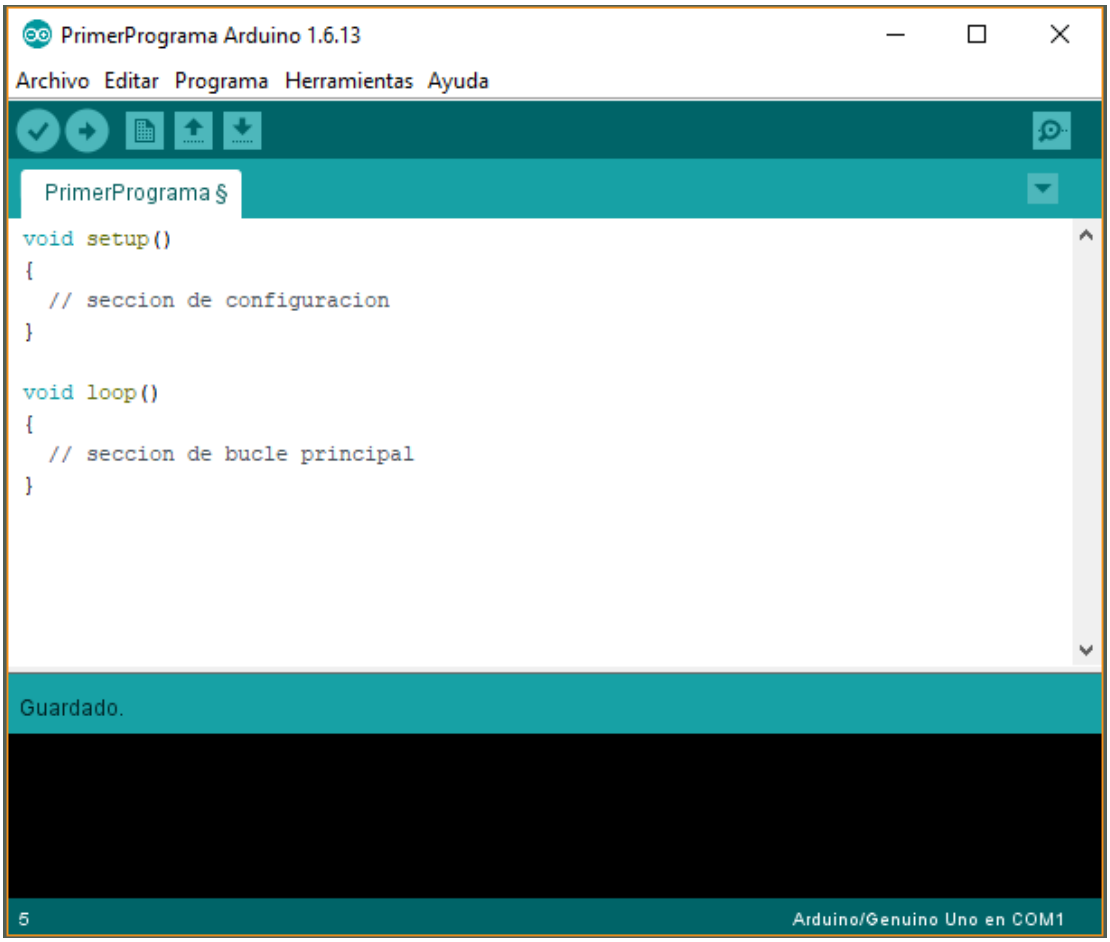

Fig. 25: Programa IDE de Arduino en Windows.

3.6.4 Arduino Due

El Arduino Due[6] es una placa basada en un microcontrolador ATMEL ATSAM3X8E[7], que incorpora una CPU ARM-M3 en lugar de utilizar un microcontrolador ATMEL AVR como la mayoría de las placas Arduino.

Tabla 20: Características principales de la placa Arduino Due.

Microcontrolador ATSAM3X8E	
Unidad	Características
CPU	ARM Cortex-M3 2.0 a 84 MHz.
	Juego de instrucciones Thumb-2.
	Tamaño de datos de 32 bits.
Memoria	Memoria de programa tipo Flash de 512KB.
	Memoria adicional para almacenamiento de bootloader de 16 KB.
	Memoria RAM de 96KB.

Pines de entrada y salida	103.
Interfaces y cantidad de conexiones	ICE (utilizado para la programación) x1.
	SPI (soporta todos los modos de trabajo) x4.
	CAN extendido (2.0b) x2.
	UART x5.
	I2C x2.
	USB 2.0 x1.
	Ethernet 10/100 x1.
Unidades Adicionales	Convertidor de señales analógicas a digitales de 12 bits y 1 Msps.
	Convertidor digital a analógico de 12 bits y 1 Msps.
	Contador de 32 bits de 9 canales.
	Módulo PWM de ocho canales de 16 bits de resolución.
Rango de temperatura	-40ºC a 85ºC.
Rango de tensión	1.62V a 3.6V.
Regulador de tensión: Sistema formado por un LM2734 y un NCP1117	
Rango de tensiones de entrada recomendadas	7V a 12V.
Tensión de salida	3.3V
Conectores de entrada y salida	
Conectores de pines digitales	54
Conectores de pines de entrada analógica	12

Conectores de pines de salida analógica	2
Programador: Microcontrolador ATMEGA16U2.	

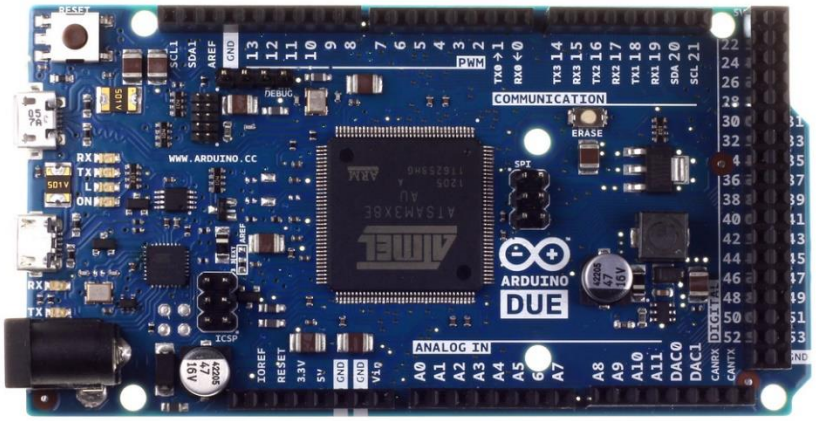

Fig. 26: Placa Arduino Due[27].

[27] ArduinoDue Front - Arduino SA (2015). Obtenido de: https://commons.wikimedia.org/wiki/File:ArduinoDue_Front.jpg

3.7 Bibliografía del capítulo

[1] The Microcontroller Idea Book - Jan Axelson - Lakeview Research. (ISBN: 0-9650819-0-7).

[2] Arquitectura y programación de microcontroladores PIC - Andres R. Bruno Saravia, Ariel Coria - MC electronics. (ISBN: 978-987-05-8658-6).

[3] Catalogo: Focus Product Selector Guide - Microchip. Disponible en internet, fecha de acceso: 25/07/2017. http://ww1.microchip.com/downloads/en/DeviceDoc/00001308Q.pdf

[4] Hoja de datos: dsPIC30F3014, dsPIC30F3013 - Microchip. Disponible en internet, fecha de acceso: 25/07/2017. http://ww1.microchip.com/downloads/en/devicedoc/70138c.pdf

[5] Getting Started with Arduino - Massimo Banzi, Michael Shiloh - Makermedia. (ISBN: 978-1-449-36333-8).

[6] Getting Started with Arduino Due - Arduino. Disponible en internet, fecha de acceso: 26/07/2017. https://www.arduino.cc/en/Guide/ArduinoDue

[7] SAM3X / SAM3A Series - Atmel. Disponible en internet, fecha de acceso: 26/07/2017. http://www.atmel.com/Images/Atmel-11057-32-bit-Cortex-M3-Microcontroller-SAM3X-SAM3A_Datasheet.pdf

4 Desarrollo

4.1 Introducción

El sistema desarrollado está formado por tres placas separadas, donde cada una cumple con una o múltiples funciones. En ellas encontramos un conjunto de microcontroladores, una serie de módulos y varios componentes electrónicos distribuidos que trabajan en conjunto para realizar las siguientes tareas:

- Determinar ubicación y velocidad del automóvil.

- Almacenar información relevante en forma de archivos.

- Ofrecer al usuario una serie de páginas web que permitan la configuración del sistema.

- Gestionar el envío y la recepción de mensajes de texto por GSM.

- Establecer una comunicación a través de un bus CAN con distintas placas interconectadas.

- Atender a la interfaz de usuario.

- Gestionar informes de errores.

4.1.1 Placas del sistema

Las distintas placas en las cuales se divide el sistema son:

- Placa de Alimentación: Encargada de establecer las tensiones de trabajo requeridas por el resto del sistema.

- Placa de Usuario: Encargada de facilitar el acceso del usuario a una parte de la interfaz de entrada y salida. Contiene al dispositivo visualizador, formado por un microcontrolador PIC18F2550 y una pantalla LCD16x2.

- Placa Principal: Encargada de la realización de las demás tareas del sistema.

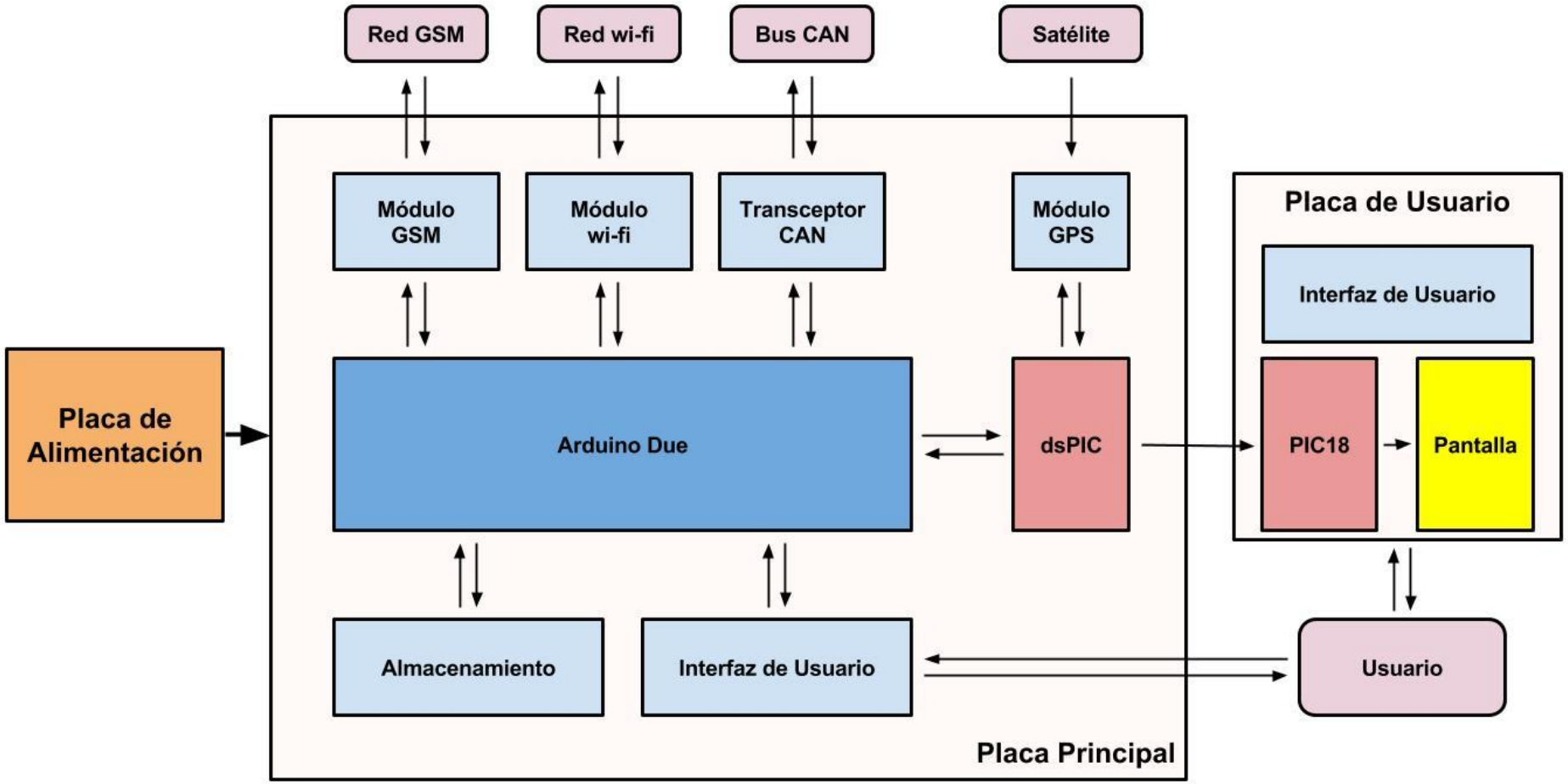

Fig. 27: Esquemático simplificado del sistema.

4.1.2 Modos de funcionamiento

El sistema posee tres modos de funcionamiento, los cuales son utilizados por las distintas placas conectadas al bus CAN para modificar las acciones que las mismas realizan.

Los modos de funcionamiento definidos son:

- Modo1: Sistema Apagado.

- Modo2: Sistema Automático.

- Modo3: Sistema Encendido.

El modo de funcionamiento habilitado o en uso, queda determinado a través de algunas de las interfaces implementadas en el sistema.

4.2 Ubicación y velocidad

Determinar la ubicación y la velocidad son algunas de las tareas realizadas por el microcontrolador dsPIC, lo cual permite disminuir la cantidad de procesamiento que debe efectuar el microcontrolador del Arduino Due.

4.2.1 Modulo GPS

El receptor GPS utilizado es el módulo NEO-6 del fabricante U-blox [1], el cual se encuentra en una placa junto a todos los componentes electrónicos que permiten su funcionamiento. Las principales características de la placa son:

- Rango de tensiones de alimentación entre 3.3V y 5V.

- Comunicación UART por defecto: 9600 baudios, campo de datos de 8 bits, 1 bit de inicio, 1 bit de paro y sin bit de paridad.

- Protocolo de mensajes NMEA[2].

- Actualización de la información cada 1 segundo.

- Rango de temperatura entre -40ºC y 85ºC.

Como lo especifica el protocolo NMEA, la información es enviada en forma de una cadena de caracteres a través de una comunicación UART. Todos los mensajes son enviados cada un segundo y su contenido varía según si la ubicación pudo o no ser calculada por el modulo.

Tabla 21: Mensajes NMEA del receptor GPS con ubicación indeterminada.

Cadena de caracteres con ubicación indeterminada (*unfixed*)
$GPGGA,,,,,,0,00,99.99,,,,,,*48
$GPGSA,A,1,,,,,,,,,,,,,99.99,99.99,99.99*30
$GPGSV,1,1,00*79
$GPGLL,,,,,,V,N*64
$GPRMC,,V,,,,,,,,,N*53
$GPVTG,,,,,,,,,N*30

Tabla 22: Mensajes NMEA del receptor GPS con ubicación determinada.

Cadena de caracteres con ubicación determinada (*fixed*)
$GPGGA,172519.00,2227.57131,N,07004.19415,E,2,07,1.42,34.0,M,-55.6,M,,0000*45
$GPGSA,A,3,25,29,12,21,18,05,15,,,,,,3.47,1.42,3.16*05
$GPGSV,3,1,10,02,15,062,07,05,38,038,17,12,36,184,33,15,32,144,21*70
$GPGSV,3,2,10,18,28,222,45,21,26,288,42,25,53,232,42,26,27,104,20*78
$GPGSV,3,3,10,29,52,344,32,39,34,249,40*70
$GPGLL,2227.57129,N,07004.19412,E,172518.00,A,D*65
$GPRMC,172519.00,A,2227.57131,N,07004.19415,E,0.114,,040614,,,D*70
$GPVTG,,T,,M,0.114,N,0.211,K,D*20

Fig. 28: Modulo receptor GPS U-blox NEO-6[28].

4.2.2 Mensajes al visualizador

El microcontrolador utiliza a sus interfaces UART para comunicarse con distintos dispositivos periféricos. Uno de estos es el visualizador, el cual se encarga de mostrar al usuario la información entregada por el dsPIC.

Independientemente de su funcionamiento, el periférico debe ser capaz de recibir y visualizar una serie de mensajes enviados por el microcontrolador seguidos del carácter <CR>.

[28] Modulo GPS Ublox GPS6MV2. Obtenido de: http://www.globeteck.com.br/produto/2456/161/modulo-gps-ublox-gps6mv2

Los tipos de mensajes son:

- Estado del servidor: Inician con el carácter '$', seguidos de 'S' o 'N' para indicar que el servidor implementado en la placa de comunicaciones se encuentra activo.

- Velocidad: Inicia con el carácter '>', seguido del valor de la velocidad entregado por el GPS o por la cadena "SAT" en caso de que el modulo no haya podido calcularla.

- Mensaje Auxiliar: Inicia con el carácter '#', seguido de una cadena de hasta 16 caracteres.

4.2.3 Ubicación y tiempo

Para determinar la ubicación y el tiempo, el microcontrolador analiza a la cadena de caracteres recibida, separa el contenido útil del mensaje GPGGA y convierte su información a un formato diferente. Estos procedimientos facilitan al microcontrolador del Arduino Due la utilización de los datos obtenidos.

Tabla 23: Campos útiles del mensaje GPGGA.

$GPGGA,172519.00,2227.57131,N,07004.19415,E,2,07,1.42,34.0,M,-55.6,M,,0000*45	
Información utilizada	Descripción
$GPGGA	Cabecera del mensaje.
172519.00	Tiempo.
2227.57131	Latitud.
N	Latitud.
07004.19415	Longitud.
E	Longitud.
2	Calidad, si es distinto de cero la posición pudo ser calculada.

4.2.3.1 Tiempo

La información del tiempo se encuentra en el formato GTM+0, por lo que debe ser convertida para su utilización en argentina a GTM-3. El procedimiento consiste simplemente en restar tres horas a la indicada por el GPS.

Tabla 24: Información del tiempo del mensaje GPGGA.

Tiempo	
Campo	Formato
172519.00	hhmmss.ss (horas, minutos y segundos).
Información de salida: 12:25:19.	

4.2.3.2 Ubicación

La información de la ubicación se encuentra expresada en coordenadas sexagesimales y debe ser convertida a coordenadas decimales para poder utilizarse en los servicios web.

Tabla 25: Información de la ubicación del mensaje GPGGA.

Latitud	
Campo	Formato
2227.57131	ggmm.mmmm (grados y minutos).
N	N/S (norte o sur, determina el signo en coordenadas decimales).
Información de salida: 22.4595218	
Longitud	
Campo	Formato
07004.19415	gggmm.mmmm (grados y minutos).
E	E/W (este u oeste, determina el signo en coordenadas decimales).
Información de salida: 70.0699025	

4.2.4 Velocidad

Para determinar la velocidad, el microcontrolador analiza a la cadena de caracteres recibida y separa el contenido útil del mensaje GPVTG. De esta manera, el dsPIC monitorea de manera permanente la velocidad del vehículo, informando al Arduino Due y al usuario cuando la misma supera al valor de la máxima establecida.

Tabla 26: Mensaje GPVTG.

$GPVTG,170.79,T,,M,4.213,N,7.802,K,A*3C	
Información utilizada	Descripción
7.802	Velocidad expresada en Kilómetros por Hora

4.2.5 Comunicación

El microcontrolador implementa un sistema de mensajes para comunicarse con el Arduino Due a través de UART y agrega una serie de pines utilizados para la transferencia de información en forma de notificaciones.

4.2.5.1 Mensajes

El procedimiento consiste en determinar la información solicitada por el Arduino Due a través de un reducido número de mensajes, los cuales están formados por una cadena de caracteres recibidos por UART.

- ATE: Es utilizado por el Arduino Due para determinar si es que el microcontrolador dsPIC se encuentra en funcionamiento. Devuelve el texto “OK” seguido del carácter <CR>.

- Hor: Devuelve la hora del sistema seguida del carácter <CR>.

- Lat: Devuelve la latitud del GPS seguida del carácter <CR>.

- Lon: Devuelve la longitud del GPS seguida del carácter <CR>.

- MV: Es utilizado por Arduino Due para configurar el valor de la velocidad máxima. A los caracteres “MV” le sigue la información de la velocidad codificada en un tercer carácter y por lo tanto, el rango de velocidades configurables abarca desde 0 a 255 Km/h. A este mensaje se responde con el texto “OK” seguido del carácter <CR>.

- ST: Es utilizado por Arduino Due para informar que debe recibir y luego re trasmitir una cadena de hasta 16 caracteres. A este mensaje el microcontrolador responde con el carácter ‘>’ para informar al Arduino que se encuentra listo para recibir la cadena, la cual debe finalizar con un <CR>. Una vez que la transferencia finalizo, el dsPIC envía el mensaje al visualizador.

4.2.5.2 Pines de estado

Estos pines permiten informar al Arduino Due sobre el estado del dsPIC, lo que evita que el mismo deba consultarlo constantemente a través de los mensajes. Los pines utilizados reciben distintos nombres según la función que realizan:

- PIN_CAL: Indica que el campo calidad del mensaje es distinto de cero y por lo tanto la ubicación es correcta.

- PIN_VEL: Indica que la velocidad medida ha superado el máximo establecido en la configuración.

- PIN_CLK: Genera una señal de reloj con un periodo de 10 segundos. Es utilizada por Arduino para tareas de temporizado.

4.2.5.3 Pines de entrada

Los pines de entrada permiten recibir información desde el Arduino Due sin tener que utilizar al sistema de mensajes. El único pin implementado para esta operación es el pin INWWW utilizado para indicar el estado del servidor implementado en la placa de comunicaciones.

4.2.5.4 Pines de salida

Los pines de salida permiten informar de forma directa sobre el estado del dsPIC al usuario. El único pin implementado para esta operación es el pin PIN_ZUM, utilizado para generar una señal del tipo sonora, que permite indicar al usuario que la velocidad medida ha superado el máximo establecido en la configuración.

4.3 Almacenamiento

El sistema utiliza una tarjeta de memoria SD para almacenar información, la cual es gestionada por el microcontrolador del Arduino Due. Gracias a esto, se consigue una gran capacidad de almacenamiento y la posibilidad de editar el contenido de la memoria con una computadora personal.

Una parte de la información almacenada es utilizada para definir los parámetros de los módulos conectados y realizar configuraciones en los distintos microcontroladores distribuidos en el sistema.

4.3.1 Tarjeta SD

Una tarjeta SD (*Secure Digital*) [6] es una memoria del tipo FLASH cuyo controlador incorpora al protocolo SPI para transferir información a baja velocidad y a una serie de comandos para realizar las operaciones de lectura o escritura de datos. De esta manera, la conexión entre el microcontrolador del Arduino Due y la tarjeta se realiza como la de cualquier otro periférico del bus SPI. Para el caso de los comandos utilizados, el IDE de Arduino incluye librerías dedicadas para efectuar operaciones directamente a nivel de archivo.

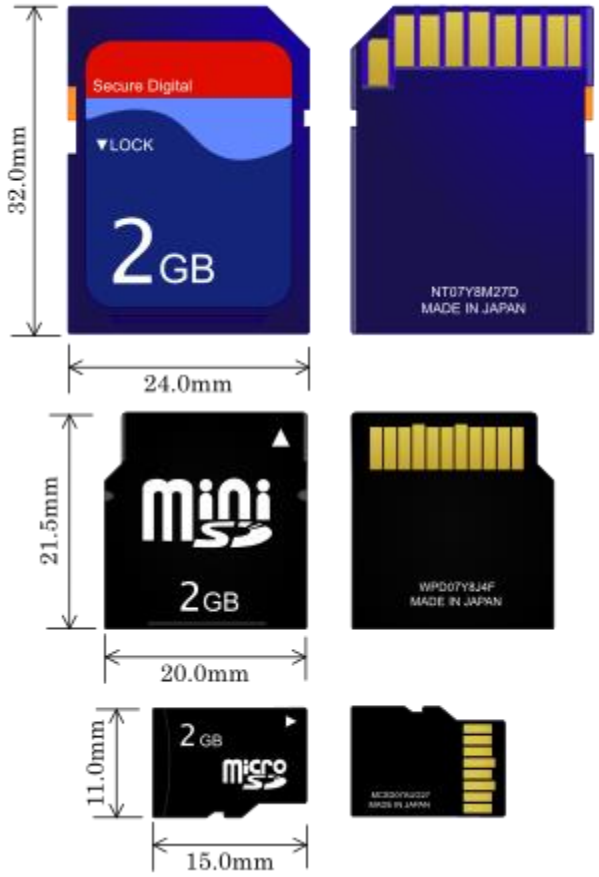

Fig. 29: Formatos físicos de las tarjetas SD[29].

4.3.2 Archivos

Los archivos almacenados en la tarjeta de memoria pueden clasificarse según la función que cumplan dentro del sistema en archivos de ubicación, archivos web y archivos de configuración.

4.3.2.1 Archivo de ubicaciones

Además del HTML, otro lenguaje de marcado ampliamente extendido es el XML (*extensible markup lenguaje*) [7] también desarrollado por el W3C. Este lenguaje es utilizado por el estándar abierto GPX (*GPS Exchange format*) [8] para almacenar y transferir datos GPS entre distintas aplicaciones.

[29] SD Cards - Tkgd2007 (2008). Obtenido de: https://commons.wikimedia.org/wiki/File:SD_Cards.svg

El GPX puede ser utilizado para describir lo que se conoce como puntos de control (*waypoints*), almacenando en el archivo las coordenadas de la ubicación del dispositivo al momento de ocurrir un evento importante para el sistema.

El formato del protocolo GPX, al igual que en las páginas escritas en HTML, define una estructura formada por una serie de elementos en los cuales se distribuye la información:

- Tipo de documento: Contiene atributos que definen las características del documento. Por ejemplo, el tipo de codificación, versión de XML, etc.

- Inicio de documento: Marca el principio del documento y agrega información adicional sobre las características del mismo. Por ejemplo, versión del archivo, autor, información de contacto, etc.

- Datos: Información GPS almacenada. Por ejemplo, puntos de control, rutas, etc.

Tabla 27: Contenido de un archivo GPX estándar.

Contenido	Descripción
<?xml version=”1.0” encoding=”UTF-8” standalone=”no” ?>	Tipo de documento.
<gpx creator=”Nombre” version=”Numero”>	Inicio del documento.
<wpt lat=Latitud" lon="Longitud">	Punto de control.
<ele>Altura</ele>	Altura del punto (msnm).
<time>Fecha-Hora</time>	Fecha y hora.
<name>Nombre</name>	Nombre del punto.
</wpt></gpx>	Etiquetas de cierre.

Según las aplicaciones utilizadas para crear y visualizar la información, algunos de los elementos o atributos de los elementos pueden ser omitidos.

Tabla 28: Contenido básico de un archivo GPX.

Contenido	Descripción
<?xml version=”1.0” encoding=”UTF-8” standalone=”no” ¿>	Tipo de documento.
<gpx>	Inicio del documento.
<wpt lat=-29.42081” lon=”-66.84476”>	Punto de control.
<time>2017-07-16T16:15:35Z</time>	Fecha y hora.
<name>VEL: 120</name>	Nombre del punto.
</wpt></gpx>	Etiquetas de cierre.

Los distintos eventos para los cuales el sistema almacena un punto de control son:

- Mensajes CAN: Cuando las placas conectadas a través del bus CAN envían un mensaje que requiere ser almacenado. La etiqueta de estos puntos está definida en el cuerpo del mensaje.

- Mensajes de texto: Cuando un mensaje de texto es recibido desde el número del administrador. Las etiquetas de estos puntos son “SIM: UBI”, “SIM: Modo1”, “SIM: Modo2”, “SIM: Modo3” o “SIM: NR”. Esta última utilizada cuando no se reconoce ninguna orden en el contenido del mensaje.

- Botón de pánico: Cuando el usuario pulsa el botón PANIC. La etiqueta de estos puntos es “PANIC”.

- Velocidad máxima: Cuando la velocidad del sistema supera a la máxima establecida. La etiqueta de estos puntos es “VEL:”, seguida del valor de la velocidad.

4.3.2.2 Archivos web

Los archivos web son todos aquellos archivos que se emplean para generar el contenido visualizado en un navegador. El sistema cuenta con tres tipos de archivos diferentes:

- Páginas web: Archivos de texto en formato HTML utilizados para almacenar el contenido de las páginas web del sistema.

- Favicon: Archivo de imagen utilizado por el navegador para personalizar la presentación de las páginas web.

- Vector de ubicaciones: Archivo de texto que almacena en un vector de JavaScript parte de la información de los puntos de control del archivo de ubicaciones, permite visualizar a los mismos en la página web de presentación de mapas.

Tabla 29: Vector genérico de JavaScript.

Vector genérico
[‘nombre1’,Latitud1,Longitud1],…,[‘nombreX’,LatitudX,LongitudX]
Ejemplo
[‘VEL: 120’, -29.42081 , -66.84476]

4.3.2.2 Archivos de configuración

Los archivos de configuración son utilizados para almacenar parámetros o información utilizada por los módulos y microcontroladores.

Durante el inicio del sistema, el microcontrolador del Arduino Due realiza una lectura de los archivos y almacenada el contenido en su memoria RAM. Mientras que la información de los archivos es actualizada durante las tareas de configuración del sistema. Por ejemplo, durante la llegada de un mensaje de texto del administrador o una acción realizada a través de una de las páginas web.

Los distintos archivos de configuración almacenan la siguiente información:

- CX: Contraseña del administrador del sistema.

- NX: Número de teléfono del administrador.

- MD: Modo de trabajo.

- MV: Velocidad máxima permitida.

- IX: Dirección IP que el módulo wi-fi tendrá en la red a la cual se conecta.

- SX: Nombre de la red wi-fi a la cual el modulo se conecta.

- PX: Contraseña de la red wi-fi a la cual el modulo se conecta.

- SP: Nombre del punto de acceso generado por el módulo wi-fi.

- PP: Contraseña del punto de acceso generado por el módulo wi-fi.

4.3.3 Directorios

Un directorio es un archivo de sistema que contiene en su interior referencias a otros archivos o incluso a otros directorios. Permiten crear una estructura jerárquica en forma de árbol facilitando la organización de todos los archivos almacenados en la unidad de memoria.

Desde el punto de vista del sistema operativo de una computadora personal, un directorio es un contenedor virtual que almacena a los archivos a los que hace referencia y posee una representación gráfica en forma de carpetas. En el caso de Arduino, debido a que las operaciones se realizan a nivel de archivo, el acceso a los mismos se realiza directamente de la forma utilizada en sistemas operativos UNIX.

Tabla 30: Ejemplo de un directorio en Arduino.

Acceso a archivos en directorios en el IDE de Arduino
/directorioprincipal/subdirectorio/archivo
/config/PP.log

Los directorios que contiene la tarjeta de memoria son:

- Web: Contiene a los archivos del tipo web.

- Ubic: Contiene al archivo de ubicaciones y al vector de ubicaciones.

- Config: Contiene a los archivos de configuración del sistema.

Tabla 31: Árbol de directorio de la tarjeta SD.

Directorio	Archivo
/web	a.txt: Página principal.
	c.txt: Pagina de configuración de número del administrador.
	e.txt: Página de error.
	g.txt: Página de gestión de archivos.
	i.txt: Configuración de los modos de funcionamiento.
	k.txt: Configuración de fecha.
	m.txt: Página de mapas.
	p.txt: Página para cambiar contraseña.
	q.txt: Pagina para indicar que los cambios fueron almacenados.
	r.txt: Pagina para indicar que la contraseña ingresada no es la correcta.
	s.txt: Pagina de estado.
	t.txt: Configuración del punto de acceso.
	v.txt: Pagina para la configuración de la velocidad máxima.
	w.txt: Pagina de configuración de la red wi-fi.
	f.txt: favicon (extensión cambiada para facilitar su lectura).

/ubic	y.txt: Vector de ubicaciones.
	z.txt: Archivo de ubicaciones.
/config	CX.log: Contraseña de administrador.
	MD.log: Modo de trabajo.
	MV.log: Velocidad máxima.
	NX.log: Teléfono de administrador.
	IX.log: Dirección IP en red existente.
	PP.log: Contraseña de punto de acceso creado.
	PX.log: Contraseña de red existente.
	SP.log: Nombre de punto de acceso creado.
	SX.log: Nombre de red existente.

4.4 Web

La mayoría de las configuraciones del sistema pueden ser realizadas a través de un navegador web, esto se logra gracias a que el microcontrolador del Arduino Due y un módulo de comunicaciones wi-fi trabajan en conjunto para implementar un servidor web sencillo.

El contenido de las páginas web enviadas por el servidor se encuentra almacenado en la tarjeta de memoria.

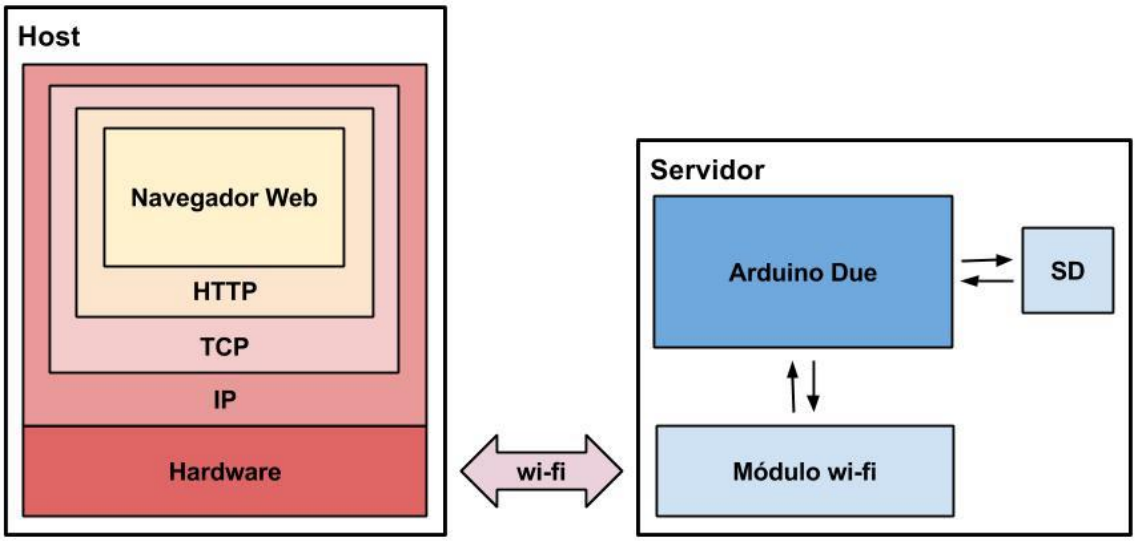

Fig. 30: Esquemático de la comunicación web.

4.4.1 Modulo wi-fi

El módulo wi-fi utilizado es un ESP8266-01 [3] del fabricante Espressif Systems, el cual se encuentra en una placa junto a todos los componentes electrónicos que permiten su funcionamiento. Las principales características de la placa son:

- Tensión de alimentación de 3.3V.

- Comunicación UART por defecto: 115200 baudios, campo de datos de 8 bits, 1 bit de inicio, 1 bit de paro y sin bit de paridad.

- Implementa comandos AT[4].

- Soporta IEEE 802.11/b/g/n.

- Soporta TCP/IP.

- Rango de temperatura entre -40ºC y 125ºC.

Las características del módulo permiten utilizarlo como un puente entre una comunicación del tipo TCP/IP sobre wi-fi con los dispositivos conectados a una red, y una comunicación de comandos AT sobre UART utilizada por el microcontrolador para manejarlo[5].

La configuración del módulo puede dividirse en dos partes, la primera utilizada para la configuración del sistema y la segunda para realizar la configuración de red. A su vez, en esta última podemos encontrar a aquellas que configuran al módulo como un punto de acceso y le permiten generar su propia red, y a las que lo configurar como un dispositivo host para conectarse a una red ya existente.

Ambos modos de funcionamiento (como punto de acceso o como host) pueden operar simultáneamente, lo que permite al usuario contar siempre con al menos una red para acceder a las páginas de configuración de todo el sistema.

Por defecto, el modulo responderá a cada comando con un eco seguido de la información asociada a la acción (“OK” o “ERROR” para las configuraciones comunes).

Tabla 32: Comandos básicos utilizados para la configuración del módulo wi-fi.

Configuraciones del sistema	
Comando	Descripción
AT	Utilizado para saber si el modulo está en funcionamiento, responde “OK”.
ATE0	Elimina el eco de cada comando.
AT+CWMODE=3	Configura al módulo en ambos modos de funcionamiento de manera simultánea.
Configuraciones de punto de acceso	
Comando	Descripción
AT+CWSAP=”*NombredenuevaRed*”,”*Contraseña*”,5,3	Parámetros de la nueva red.
Configuraciones de host	
Comando	Descripción
AT+CIPSTA=”*DireccionIP*”	Dirección IP escogida para el modulo (debe pertenecer a un rango de direcciones válidas para la red).
AT+CWJAP=”*NombredeRedexistente*”,”*Constraseña*”	Parámetros de la red a la cual el modulo se conectara.
AT+CIPMUX=1	Permite múltiples conexiones al modem.

AT+CIPSERVER=1,80	Activa la capacidad del módulo para funcionar como servidor en un puerto TCP especificado.
AT+CIPSERVER=0	Desactiva al servidor.
AT+CWQAP	Desconecta al módulo de la red.

Además de los comandos de configuración, el modulo también posee a aquellos utilizados para realizar una acción en respuesta a las peticiones de los demás hosts.

Tabla 33: Comandos básicos utilizados para realizar una respuesta del módulo wi-fi.

Comando	Descripción
AT+CIPSEND=*NumerodeHost*,*CantidadCaracteres*	Indica al módulo que debe enviar una cadena de caracteres de un tamaño especificado (máximo 2048) al host que realizo la petición del recurso. El modulo responderá con el carácter ">" y quedara en espera de los datos.
AT+CIPCLOSE=*NumerodeHost*	Cierra la conexión con el host.

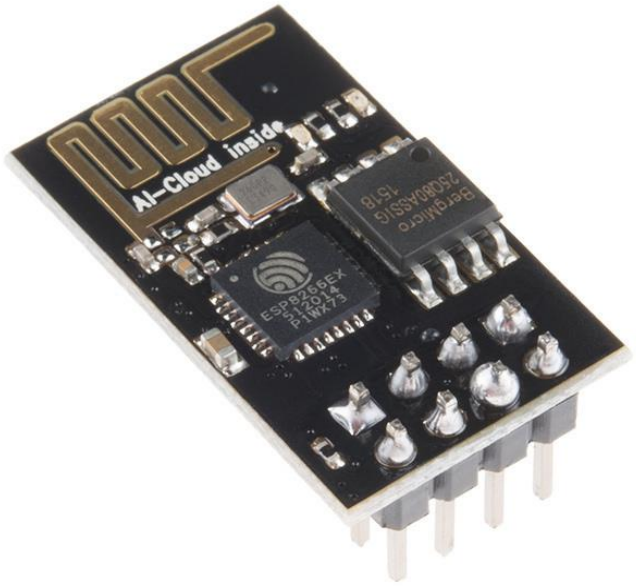

Fig. 31: Modulo wi-fi ESP8266-01[30].

[30] ESP-01 - Sparkfun Electronics (2017). Obtenido de: https://en.wikipedia.org/wiki/File:ESP-01.jpg

4.4.2 Servidor

4.4.2.1 Direccionamiento

Luego de realizar la configuración del módulo y obtener una conexión exitosa, el sistema tendrá dos direcciones IP diferentes para realizar las peticiones de las páginas:

- IP de punto de acceso: Es la dirección IP del módulo en la red que el mismo genera. Por defecto es 192.168.4.1.

- IP de host: Es la dirección del módulo en la red generada por un punto de acceso externo, generalmente un enrutador (*router*) conectado a internet. Por ejemplo, 192.168.0.17.

4.4.2.2 Peticiones

Una vez que un navegador web de un ordenador conectado a la red direcciona la IP del módulo, este envía la solicitud HTTP recibida al microcontrolador del Arduino Due en forma de una cadena de caracteres.

El contenido de la cadena es analizado para determinar:

- Host que realizo la petición.

- Recurso demandando.

- Método de la petición (GET o POST).

En el caso de las páginas almacenadas, los formularios completados por el usuario para realizar las configuraciones utilizan al método POST. Esto genera que el contenido de la cadena presente datos adicionales, similares a los de una consulta en un identificador de recursos uniformes. Esto significa que la cadena contendrá datos del tipo “atributo=valor” separados entre sí por el carácter “&” a los cuales el microcontrolador debe identificar.

Fig. 32: Formulario de la página de configuración wi-fi.

Tabla 34: Peticiones recibidas por el módulo wi-fi y enviadas al Arduino Due.

Petición GET de la página principal:
1,CONNECT +IPD,0,385:GET / HTTP/1.1 Host: 192.168.0.55 Connection: keep-alive Cache-Control: max-age=0 Upgrade-Insecure-Requests: 1 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/60.0.3112.113 Safari/537.36 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8 Accept-Encoding: gzip, deflate Accept-Language: es-ES,es;q=0.8,en;q=0.6,en-US;q=0.4
Petición POST generada por el formulario de la página de configuración wi-fi:
0,CONNECT +IPD,0,391:POST /w.html HTTP/1.1 Host: 192.168.0.55

```
Connection: keep-alive
Content-Length: 64
Cache-Control: max-age=0
Origin: http://192.168.0.55
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/60.0.3112.113 Safari/537.36
Content-Type: application/x-www-form-urlencoded
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
Referer: http://192.168.0.55/
Accept-Encoding: gzip, deflate
Accept-Language: es-ES,es;q=0.8,en;q=0.6,en-US;q=0.4

CX=232323&H=w&SX=NombredeRed&PX=123456&IX=192.168.0.1&E=Guardar+Cambios
```

4.4.2.3 Respuesta

Luego de que el microcontrolador determina el nombre del recurso solicitado, realiza una búsqueda del archivo en la memoria SD y envía su contenido al cliente que efectuó la petición.

Debido a que el comando utilizado para enviar la información al cliente tiene un máximo de caracteres permitidos, los recursos solicitados que superen al valor establecido serán enviados por partes. Es decir, la información es enviada en bloques que no superan los 2048 caracteres.

Una vez que la trasferencia del contenido del archivo finalizo, el microcontrolador le indica al módulo que debe finalizar la conexión.

4.4.2.4 Estado del servidor

Las tareas realizadas por el sistema funcionando como un servidor demandan mucho tiempo de procesamiento, por este motivo durante algunos eventos considerados de mayor importancia, la funcionalidad del servidor puede desactivarse.

Los eventos durante los cuales el servidor se encuentra desactivado son:

- Envío y recepción de mensajes de texto: Permite al microcontrolador atender de forma exclusiva al módulo GSM.

- Configuración de los parámetros de las redes wi-fi.

- Mensaje CAN: Permite al microcontrolador centrarse en las tareas relacionadas a la comunicación a través del bus CAN.

Además de la desactivación temporaria del servidor durante los eventos, el sistema puede activarse o desactivarse de manera permanente a través de mensajes especiales recibidos por el bus CAN. Esta función permite conectar una placa adicional al sistema para desactivar el servidor cuando el motor del vehículo se encienda, lo que permite al Arduino Due centrarse en tareas de comunicación a través de CAN y GSM.

4.4.3 Páginas web

El contenido de casi todas las páginas web del sistema ya está almacenado en la tarjeta SD y por lo tanto se tratan de páginas del tipo estáticas.

Las páginas que se encuentran disponibles son:

- Página principal: Es la página que se encuentra disponible en la raíz de la ruta de la URL, y contiene en ella enlaces a todas las demás páginas disponibles.

- Configuración de número del administrador: En ella se puede realizar el cambio del número telefónico del administrador del sistema.

- Página de error: Indica que el recurso solicitado no está disponible. Es decir, informa al usuario que la ruta de la URL ingresada no es válida.

- Página de gestión de archivos: Permite descargar o eliminar el archivo de ubicaciones.

- Configuración de los modos de funcionamiento: Permite seleccionar entre los distintos modos de funcionamiento del sistema.

- Configuración de fecha: Permite modificar la fecha del reloj del sistema.

- Página para cambiar contraseña: Permite modificar la contraseña del administrador del sistema.

- Configuración de la velocidad máxima.

- Pagina para indicar que los cambios fueron almacenados.

- Pagina para indicar que la contraseña ingresada no es la correcta.

- Configuración del punto de acceso: Permite configurar los parámetros del módulo cuando opera como punto de acceso.

- Configuración de la red wi-fi: Permite configurar los parámetros del módulo cuando opera como un host en una red ya disponible.

Fig. 33: Pagina de configuración de la velocidad máxima.

El sistema cuenta además con dos páginas cuyo contenido puede variar de forma dinámica:

- Página de estado: En esta página web el microcontrolador agrega contenido adicional al código HTML para mostrar al usuario información relevante acerca del estado del sistema.

- Página de mapas: Es la única que necesita de un cliente con acceso a internet para ser visualizada. Esto se debe a que el código posee un script que permite utilizar a los mapas de la empresa Google para mostrar los puntos almacenados en el vector de ubicaciones.

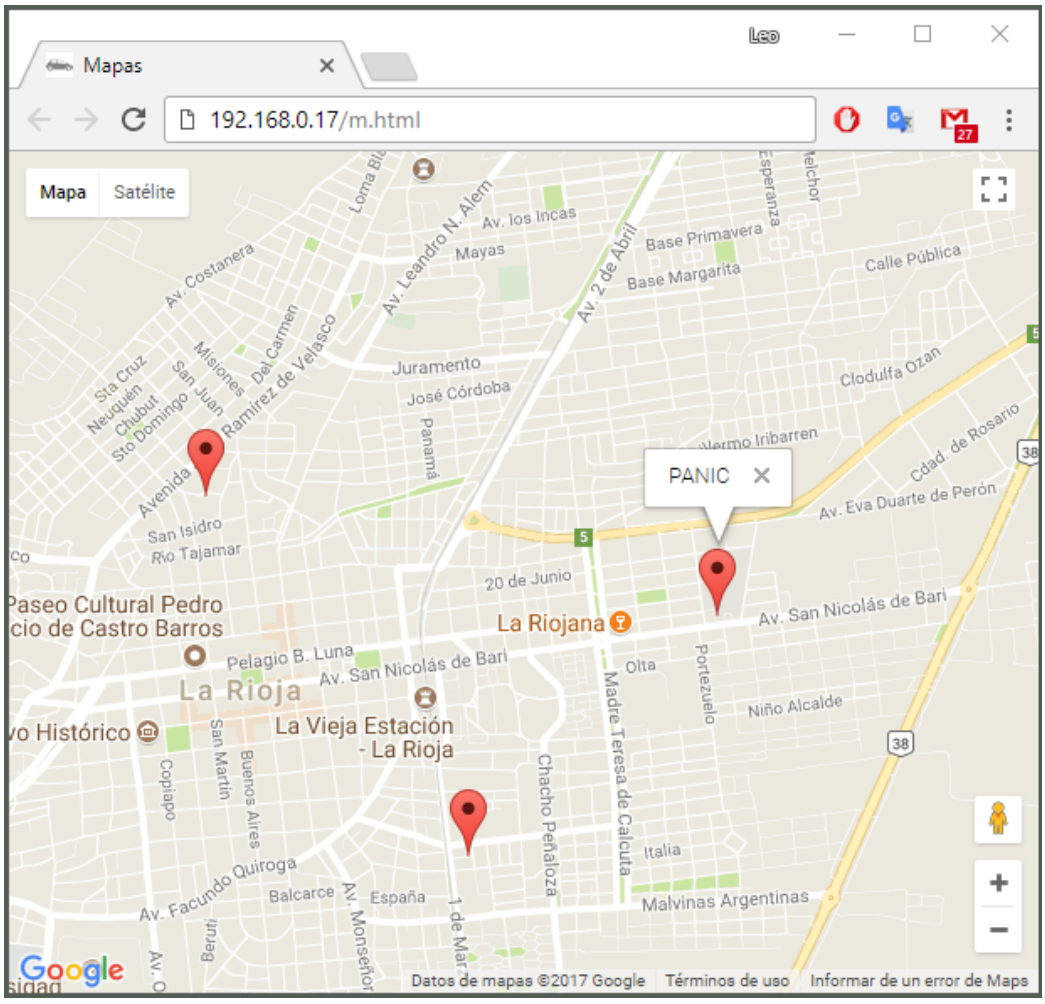

Fig. 34: Pagina de mapas.

4.4.4 Recursos adicionales

Además de las páginas web, el sistema posee dos archivos adicionales que pueden ser enviados como recursos, estos son el archivo favicon y el archivo de ubicaciones.

4.4.4.1 Favicon

El archivo favicon, del inglés *favorite icon*, es una pequeña imagen utilizada por el navegador para facilitar la identificación de la página en las listas de favoritos o historiales, en las cabeceras de pestañas o en las barras de direcciones.

La transferencia de la información del archivo favicon desde el interior de la memoria SD al navegador se realiza exactamente igual a la de los archivos HTML. Esto es debido a que sin importar el tipo de archivo, todos están formados por una sucesión de bits, los cuales pueden ser agrupados como si se trataran de simples caracteres y es el navegador quien se encarga de representarlos en su formato original.

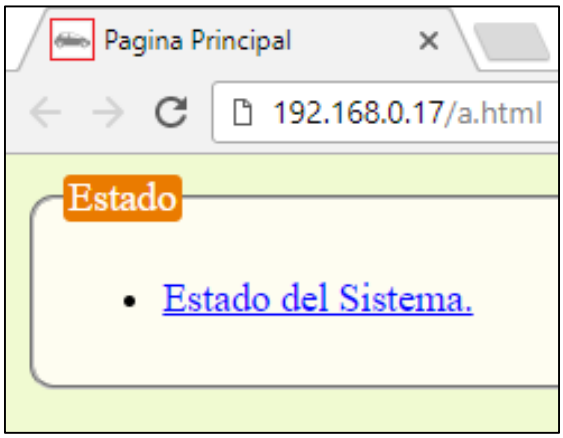

Fig. 35: Presentación del favicon en el navegador web Chrome.

4.4.4.2 Archivo de ubicaciones

El archivo de ubicaciones puede ser enviado como un recurso al navegador, pero debido a que se encuentra escrito en el lenguaje XML, el navegador al recibirlo lo presentara como si se tratara de una página web de texto. La forma de solucionar este inconveniente es agregar el atributo *download* en el enlace utilizado, lo que le indica al navegador que debe descargar el archivo en lugar de presentarlo.

4.5 Mensajes de texto

El sistema implementa una comunicación a través de mensajes de texto para alertar al administrador en el momento que se produzca un evento considerado como importante, permitir realizar algunas configuraciones o consultar la ubicación del vehículo.

4.5.1 Modulo GSM

El módulo utilizado es un SIM800L [9] del fabricante SIMCom, el cual se encuentra en una placa junto a todos los componentes electrónicos que permiten su funcionamiento y a un zócalo para tarjetas SIM. Las principales características de la placa son:

- Tensión de alimentación de 4.2V.

- Comunicación UART por defecto: 115200 baudios, campo de datos de 8 bits, 1 bit de inicio, 1 bit de paro y sin bit de paridad.

- Implementa comandos AT.

- Soporta GSM en bandas 850/900/1800/1900.

- Soporta TCP/IP.

- Rango de temperatura entre -40ºC y 85ºC.

- Contiene un reloj de tiempo real.

El modulo es utilizado principalmente para enviar y recibir mensajes de texto. Su comunicación con el microcontrolador se realiza a través de comandos AT sobre UART.

Por defecto, el modulo responderá a cada comando con un eco seguido de la información asociada a la acción (“OK” o “ERROR” para las configuraciones comunes).

Tabla 35: Comandos básicos del módulo SIM800L.

Configuraciones del módulo:	
Comandos	Descripción
AT	Utilizado para saber si el modulo está en funcionamiento, responde “OK”.
AT+CMEE=0	Los errores que ocurran en el sistema solo devolverán el mensaje “ERROR”.
ATE0	Elimina el eco de cada comando.
AT+GSMBUSY=1	Rechaza las llamadas entrantes.
AT+CMGF	Los mensajes de texto serán codificados en caracteres imprimibles.
AT+CNMO=,1	Activar notificaciones de mensajes de texto entrantes.
AT+IPR=19200	Configura la velocidad de trasmisión en 19200 baudios.
Comandos para mensajes de texto:	
Comandos	Descripción
AT+CMGDA=”DEL ALL”	Elimina todos los mensajes de texto almacenados en la memoria del módulo.
AT+CMGS=”*Numero*”	Informa al módulo que debe enviar un mensaje de texto al número indicado. El modulo responderá con el carácter “>” y quedara en espera del cuerpo del mensaje. Una vez finalizado el mensaje, el microcontrolador debe enviar el carácter 26 del código ASCII.
AT+CMGL=”REC UNREAD”	Solicita el contenido de todos los mensajes no leídos.
Comando utilizados por el reloj de tiempo real:	
Comandos	Descripción
AT+CCLK=”*Fecha*”,”*Hora*”	Define la fecha y la hora en el reloj de tiempo real.
AT+CCLK?	Consulta la hora del reloj de tiempo real.

Fig. 36: Modulo SIM800L[31].

4.5.2 Lectura de mensajes

Cuando el sistema recibe un mensaje de texto, el modulo informa al microcontrolador mediante una notificación formada por una cadena de caracteres.

Tabla 36: Notificación enviada por el módulo GSM.

Cadena de caracteres	Descripción
+CMTI: “SM”,*n*	Notificación enviada por el modulo para indicar la recepción de un mensaje. El carácter “n” determina la ubicación del mensaje en la memoria de la tarjeta SIM (“SM”, *SIM memory*).

Una vez que el microcontrolador recibe la notificación y se encuentra en condiciones de realizar una lectura del mensaje, solicita al módulo el contenido del mismo mediante el comando correspondiente.

Tabla 37: Respuesta del módulo GSM al comando de lectura de los mensajes de texto.

Cadena de caracteres	Descripción
+CMGL: 1,"REC UNREAD","*Numero*","*Fecha*” *Cuerpo del mensaje*	Respuesta del módulo a una solicitud de contenido de los mensajes no leídos.

El microcontrolador analiza la respuesta y en caso de que el número telefónico pertenezca al administrador del sistema, realiza una búsqueda excluyente de una serie de raíces gramaticales en el cuerpo del mensaje. Esto permite que el sistema interprete palabras que pertenecen a una misma familia gramatical y realice una acción correspondiente para cada conjunto.

[31] mod_sim800l_small_2 - Avelectronics (2016). Obtenido de: http://avelectronics.co/wp-content/uploads/2016/09/mod_sim800l_small_2-800x800.jpg

Las raíces buscadas en el cuerpo del mensaje son:

- UBI: El sistema envía al número de teléfono del administrador un mensaje de texto con una URL que direcciona a los servidores de Google, la dirección contiene información de la ubicación del vehículo en caso de estar disponible. Algunas palabras que contienen esta raíz son ubicación, ubicar, etc.

- APA: El modo de funcionamiento del sistema cambia a Modo1 y se envía al número de teléfono del administrador un mensaje informando que el cambio se realizó de forma satisfactoria. Algunas palabras que contienen esta raíz son apagar, apagado, etc.

- AUT: El modo de funcionamiento del sistema cambia a Modo2 y se envía al número de teléfono del administrador un mensaje informando que el cambio se realizó de forma satisfactoria. La principal palabras que contienen esta raíz es automático.

- ENC: El modo de funcionamiento del sistema cambia a Modo3 y se envía al número de teléfono del administrador un mensaje informando que el cambio se realizó de forma satisfactoria. Algunas palabras que contienen esta raíz son encender, encendido etc.

Debido a que la búsqueda es del tipo excluyente, se garantiza que ante la llegada de un mensaje que contenga a más de una raíz, el sistema solo reaccionara a la que posea una mayor importancia. De esta manera, para cada mensaje el sistema realizara solo una acción.

El método de búsqueda realizado analiza a las raíces como simples cadenas de caracteres, por lo que es posible encontrarlas cuando el cuerpo del mensaje está formado por más de una palabra. Por ejemplo, la frase “enviar ubicación del sistema” tendrá el mismo efecto que la cadena “ubi”.

En caso de que ninguna raíz sea encontrada en el cuerpo del mensaje, el sistema enviara un mensaje al número del administrador informando lo sucedido.

Tabla 38: Mensajes de texto enviados como respuesta al número del administrador.

Raíz en el mensaje	Respuestas del sistema enviadas al número del administrador
UBI	Ubicación: https://www.google.com.ar/maps/search/LAT,+LON
	Ubicación momentáneamente no disponible
APA	Sistema Apagado, Modo1
AUT	Sistema Automatico, Modo2
ENC	Sistema Encendido, Modo3
No encontrada	Comando no reconocido

4.5.3 Envío de mensajes

El sistema envía un mensaje de texto al número de teléfono del administrador en el momento en el que se producen algunos de los siguientes eventos:

- Mensajes CAN: Cuando las placas conectadas a través del bus CAN envían un mensaje que requiere de los servicios GSM. El contenido de los mensajes de texto se encuentra definido en el interior de los mensajes CAN.

- Botón de pánico: Cuando el usuario pulsa el botón PANIC. La etiqueta de estos puntos es “PANIC”.

- Velocidad máxima: Cuando la velocidad del sistema supera a la máxima establecida.

El contenido del mensaje informara lo sucedido e incluirá una URL que direcciona a los servidores de Google la información de la ubicación en caso de que la misma se encuentre disponible.

Tabla 39: Eventos que generan el envío de un mensaje de texto.

Evento	Mensaje al número del administrador
Mensajes CAN	*Definido por el contenido del mensaje CAN.*
Botón PANIC	Botón anti pánico presionando.
Velocidad Máxima	Límite de velocidad superado. Velocidad: *Valor* Km/h.

4.6 Reloj de tiempo real

Un reloj de tiempo real es un dispositivo que permite almacenar y mantener actualizada la información de la fecha y la hora del sistema.

La información de la hora es obtenida del GPS a través del microcontrolador dsPIC, mientras que la fecha es ingresada por el usuario a través de la página web correspondiente.

Los datos entregados por el reloj son utilizados por el sistema en el archivo de ubicaciones, lo que permite a las aplicaciones que trabajan con archivos del tipo GPX filtrar los puntos de control según el instante en cual se almacenaron.

4.7 Comunicación CAN

La comunicación mediante CAN se realiza sobre un protocolo orientado a mensajes, en donde cada uno posee un identificador único. A este protocolo base, debemos sumar un conjunto de reglas establecidas por el sistema para garantizar la correcta transferencia de información entre los dispositivos conectados al bus.

Debido a que el microcontrolador del Arduino Due posee un controlador CAN pero no un transceptor, el sistema utiliza un Sn65hvd230[10] para establecer la comunicación con el resto de las placas conectadas al bus.

4.7.1 IDmodo

El mensaje IDmodo es enviado por el sistema al bus CAN para informar a las placas conectadas el modo de funcionamiento utilizado. El identificador de este mensaje es el número 0 y al recibir el mensaje, las placas no generan respuesta.

Debido a que el modo del sistema se encuentra codificado en un carácter, los valores transmitidos para cada modo son:

- Modo1: Valor transmitido, 49.

- Modo2: Valor transmitido, 50.

- Modo3: Valor transmitido, 51.

El mensaje IDmodo será transmitido cada vez que el modo de funcionamiento varié o cuando el sistema reciba una petición.

4.7.2 IDinforma

El mensaje IDinforma es enviado por todos los dispositivos conectados al bus al momento de encender. El identificador de este mensaje es el número 10 y el cuerpo del mensaje contiene un valor único para cada placa, el cual es utilizado como el identificador de la misma.

En el caso del sistema de comunicación, el identificador de la placa es el número 0. Esto significa que al momento de finalizar las operaciones de configuración, el sistema envía por el bus CAN el mensaje IDinforma para indicar al resto de las placas que las funciones de comunicación se encuentran disponibles.

4.7.1 IDfinal

El mensaje IDfinal es utilizado por el sistema para informar al resto de las placas conectadas que ha finalizado una operación anteriormente solicitada. El identificador de este mensaje es el número 8 y el cuerpo del mensaje varía según la operación solicitada.

4.7.4 IDmodoRQST

El mensaje IDmodoRQST es utilizado por las placas conectadas al bus para solicitar al sistema el envío del mensaje IDmodo. El identificador de este mensaje es el número 1 y el cuerpo del mensaje contiene el valor que identifica a cada placa.

4.7.5 IDservidor

El mensaje IDservidor es utilizado por las placas conectadas al bus para encender o apagar el servidor web. El identificador de este mensaje es el número 2 y el cuerpo del mensaje debe contener a los caracteres ‘0’ para apagar o ‘1’ para encender el servidor.

La respuesta del sistema a este mensaje se realiza mediante un IDfinal, en donde el cuerpo del mensaje corresponde al identificador del IDservidor.

Un ejemplo donde se utiliza este mensaje es en el apagado del servidor durante el funcionamiento del vehículo, lo que permite al sistema dedicarse exclusivamente a las tareas de comunicación por CAN y GSM.

4.7.6 IDvelocidad

El mensaje IDvelocidad es utilizado por las placas conectadas al bus para modificar el valor de la velocidad máxima del sistema a 10Km/h. El identificador de este mensaje es el número 4 y el cuerpo del mensaje debe contener a los caracteres ‘1’ para configurar el nuevo valor o ‘0’ para volver a configurar el valor previamente establecido.

La respuesta del sistema a este mensaje se realiza mediante un IDfinal, en donde el cuerpo del mensaje corresponde al identificador del IDvelocidad.

Un ejemplo donde se utiliza este mensaje es cuando el vehículo se encuentra estacionado, en caso de que la velocidad supere los 10Km/h significa que el mismo está siendo remolcado y el sistema generara una alerta.

4.7.7 IDpantalla, IDalmacena e IDenvia

Los mensajes IDpantalla, IDalmacena e IDenvia son utilizados por las placas conectadas al bus para transferir una cadena de hasta 16 caracteres al sistema. El identificador del mensaje varia para cada uno y define lo que el sistema realiza con los caracteres recibidos:

- IDpantalla: El identificador de este mensaje es el número 5. Es utilizado para indicar al sistema que debe mostrar en el visualizador los caracteres recibidos.

- IDalmacena: El identificador de este mensaje es el número 6. Es utilizado para indicar al sistema que debe realizar las acciones de IDpantalla y almacenar un punto de control cuya etiqueta se encuentra definida por los caracteres recibidos.

- IDenvia: El identificador de este mensaje es el número 7. Es utilizado para indicar al sistema que debe realizar las acciones de IDalmacena y enviar un mensaje de texto con los caracteres recibidos.

La forma en la que se realiza la operación de transferencia consiste en:

- Una placa conectada al bus envía alguno de los tres mensajes definidos y agrega como cuerpo al valor del identificador de placa.

- El sistema recibe el mensaje y responde con una copia del mismo, esto indica a la placa que el sistema se encuentra listo para recibir la cadena de caracteres.

- La placa envía los 16 caracteres en dos mensajes de 8 bytes, cada uno con el mismo identificador utilizado al principio.

- El sistema recibe los mensajes y realiza las tareas necesarias, una vez que las mismas han finalizado, el sistema envía un IDfinal cuyo cuerpo corresponde al de la operación realizada.

4.8 Interfaz de usuario

La interfaz de usuario agrega una alternativa a la comunicación realizada a través de las páginas web. En el sistema, una parte de la interfaz se encuentra en la placa principal, mientras que el resto se ubica en la placa de usuario.

4.8.1 Interfaz de entrada

La interfaz de entrada está compuesta por una serie de botones que agregan funciones adicionales al sistema.

4.8.1.1 Botón RQST

El botón RQST, ubicado en la placa principal, permite enviar toda la información relacionada con el estado del sistema por una salida UART adicional.

La información sobre el estado del sistema incluye:

- Contenido de los archivos de configuración.

- Estado de los pines conectados con el microcontrolador dsPIC.

- Información de ubicación, velocidad, hora y valor de la constante utilizada.

- Fecha y hora del reloj en tiempo real.

- Contenido del archivo de ubicaciones.

- Información de las variables de estado utilizadas por el microcontrolador del Arduino Due.

4.8.1.2 Botón PANIC

Permite al usuario enviar un mensaje de texto al número del administrador del sistema con la información de la ubicación en la cual se encuentra el vehículo. Puede ser utilizado en caso de una situación de emergencia por parte del conductor o los acompañantes.

4.8.2 Interfaz de salida

La interfaz de salida está compuesta por una serie de indicadores, un dispositivo de visualización y una UART adicional que permiten informar al usuario el estado del sistema.

4.8.2.1 UART

La UART, ubicada en la placa principal, es utilizada constantemente para transmitir informes sobre los mensajes entrantes por el Bus CAN, las páginas web solicitadas o los errores ocurridos durante el funcionamiento del sistema.

Las características de la comunicación de la UART son:

- Velocidad de 115200 baudios.

- Campo de datos de 8 bits.

- Un bit de inicio y un bit de paro.

- Sin bits de paridad.

4.8.2.2 Indicadores luminosos

El indicador luminoso LEN, permite informar al usuario que el sistema se encuentra encendido. Además, en caso de que la tarjeta de memoria no pueda ser leída durante el inicio del programa, el mismo titilara de forma ininterrumpida para indicar lo sucedido.

4.8.2.3 Señal sonora

La señal sonora es utilizada para indicar al usuario que la velocidad del vehículo ha superado el valor de la máxima establecida.

4.8.2.4 Visualizador

El visualizador consiste en un dispositivo capas de recibir los mensajes enviados por el dsPIC y mostrarlos en una pantalla, en donde la primera línea de caracteres corresponde a la información de la velocidad del vehículo y a la del estado del servidor. En el caso de este último, para indicar que el mismo se encuentra activo en la pantalla aparecerán los caracteres

"WWW", mientras que si los mismos no se encuentran presentes, significa que el servidor está apagado.

La segunda línea de caracteres es utilizada para mostrar cualquier tipo de mensajes procedentes del sistema, en donde el contenido de los mismos es generado por las siguientes acciones o cambios en la configuración:

- Fin de las configuraciones iniciales: Los caracteres mostrados en pantalla son "Sistema Listo".

- Modo de funcionamiento: Los caracteres mostrados en pantalla son "Cambio: Modo *valor*".

- Mensajes CAN: Contenido personalizado enviado por alguna placa conectada al bus.

- Velocidad máxima: Los caracteres mostrados en pantalla son "Vel max: *valor* Km/h".

- Botón RQST: Los caracteres mostrados en pantalla son "RQST".

Tabla 40: Disposición de los caracteres en la pantalla del visualizador.

Caracteres en pantalla															
V	E	L	=		4	5							W	W	W
C	a	m	b	i	o	:		M	o	d	o	3			

4.9 Informes de errores y estado

Al momento de ocurrir errores durante el funcionamiento normal del sistema, el microcontrolador informa al usuario lo sucedido a través de la interfaz de salida.

Los errores ocurridos durante la comunicación con los distintos módulos o dispositivos, son informados a través de la UART de la interfaz de salida en forma de mensajes. Todos los mensajes inician con los caracteres "*-ERR*:" y son seguidos de información que indica la fuente del error, la cual por lo general señala a la zona del programa en la cual el mismo se origina.

Tabla 41: Ejemplos de mensajes de error enviados por la UART de la interfaz de salida.

Error durante la lectura de un archivo de configuración.	-ERR: Leer_Archivo *NombredeArchivo*
Error al enviar la página de estado del sistema.	-ERR: Enviar_Pagina_Estado
Error al enviar un mensaje de texto.	-ERR: Enviar_Mensaje

4.10 Bibliografía del capítulo

[1] Data Sheet: NEO-6, u-blox 6 GPS Modules - U-blox. Disponible en internet, fecha de acceso: 30/07/2017. https://www.u-blox.com/sites/default/files/products/documents/NEO-6_DataSheet_(GPS.G6-HW-09005).pdf

[2] u-blox 6, Receiver Description - U-blox. Disponible en internet, fecha de acceso: 30/07/2017. https://www.u-blox.com/sites/default/files/products/documents/u-blox6_ReceiverDescrProtSpec_%28GPS.G6-SW-10018%29_Public.pdf?utm_source=en%2Fimages%2Fdownloads%2FProduct_Docs%2Fu-blox6_ReceiverDescriptionProtocolSpec_%28GPS.G6-SW-10018%29.pdf

[3] Datasheet, ESP8266EX - Espressif Systems. Disponible en internet, fecha de acceso: 30/07/2017. https://espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf

[4] ESP8266 AT Instruction Set, versión 2.1.0 - Espressif Systems. Disponible en internet, fecha de acceso: 30/07/2017. https://espressif.com/sites/default/files/documentation/4a-esp8266_at_instruction_set_en.pdf

[5] ESP8266 AT Command Examples, versión 1.3 - Espressif Systems. Disponible en internet, fecha de acceso: 30/07/2017. https://espressif.com/sites/default/files/documentation/4b-esp8266_at_command_examples_en.pdf

[6] SanDisk SD Card, Product Manual, versión 2.2 - SanDisk Corporation. Disponible en internet, fecha de acceso: 30/07/2017. https://www.brokentoaster.com/arduinomp3/files/Sandiskmanual-SecureDigital2.2.pdf

[7] Extensible Markup Language (XML) 1.1, (Second Edition) - W3C. Disponible en internet, fecha de acceso: 30/07/2017. https://www.w3.org/TR/xml11/

[8] GPX: the GPS Exchange Format - TopoGrafix. Disponible en internet, fecha de acceso: 05/08/2017. http://www.topografix.com/gpx.asp

[9] SIM800 - SIMCom. Disponible en internet, fecha de acceso: 05/08/2017. http://simcomm2m.com/En/module/detail.aspx?id=138

[10] Sn65hvd23x - Texas Instruments. Disponible en internet, fecha de acceso: 30/08/2017. http://www.ti.com/lit/ds/slos346n/slos346n.pdf

5 Hardware del Sistema

5.1 Introducción

El hardware corresponde a la parte física del sistema, abarcando a todos componentes electrónicos, eléctricos y mecánicos (como soportes y conectores), por lo que determina características como el tamaño, la geometría y el consumo.

5.2 Placas del sistema

El sistema se divide en una serie de placas, cada una definida en el capítulo anterior.

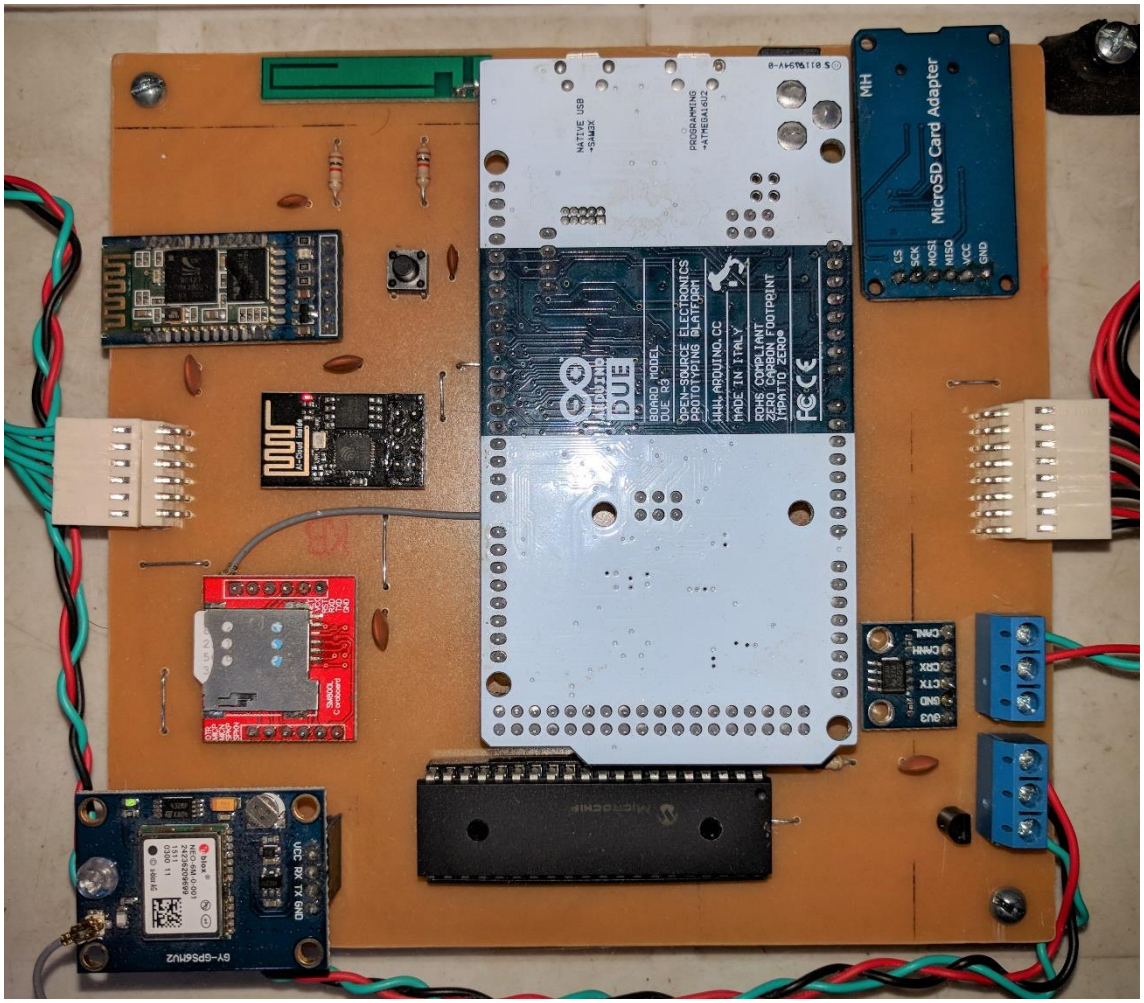

Fig. 37: Fotografía de la Placa Principal del sistema con un módulo Bluetooth conectado.

La Placa de Alimentación utiliza fuentes del tipo modular para generar cada uno de los voltajes requeridos, las mismas se encuentran basadas en el regulador LM2592 y en sus circuitos de aplicación[1].

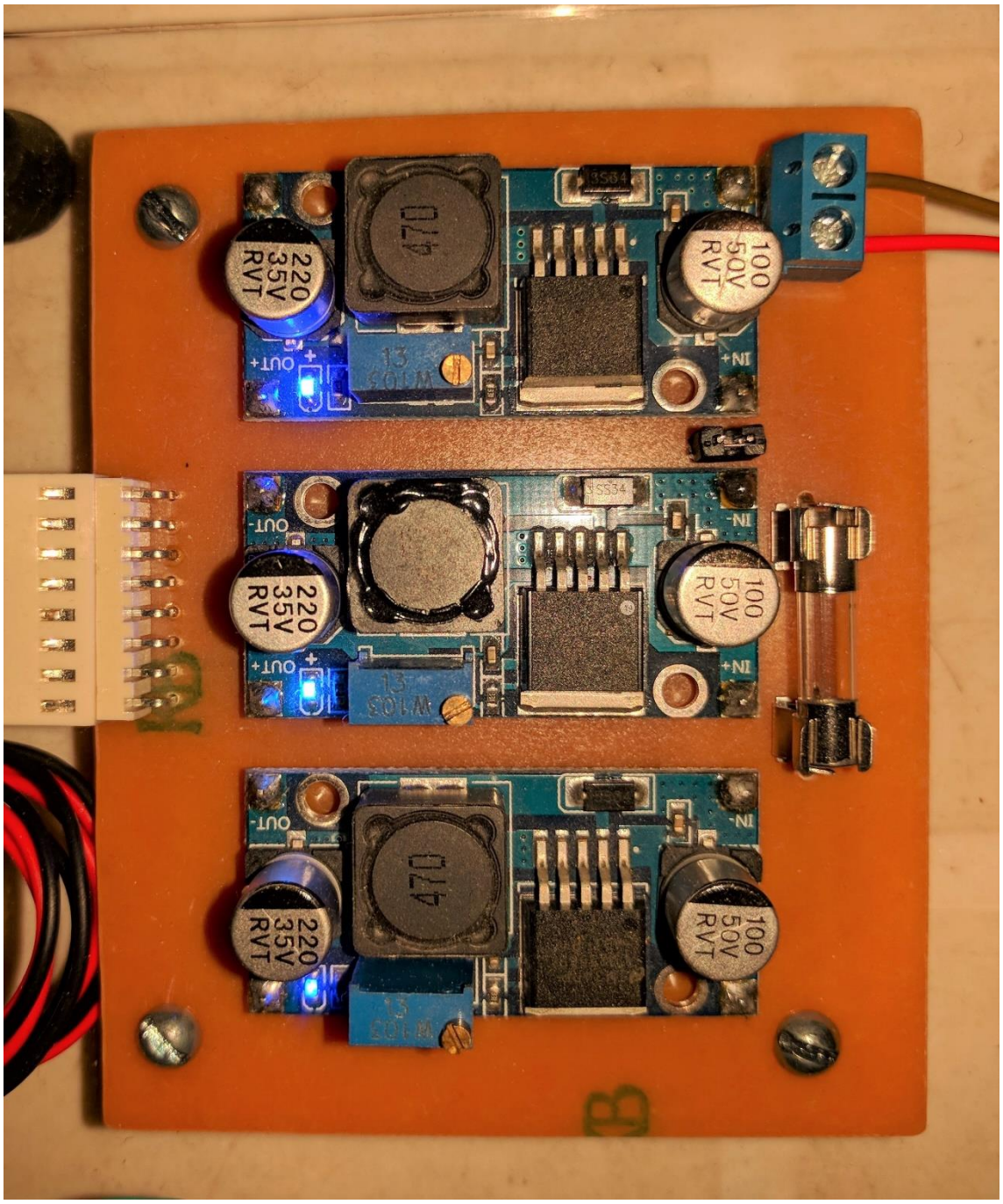

Fig. 38: Fotografía de la Placa de Alimentación.

Además de los componentes de la interfaz de usuario del sistema, la Placa de Usuario posee una entrada y una salida adicionales, las cuales pueden ser utilizadas por cualquier placa adicional que requiera de una comunicación con el usuario.

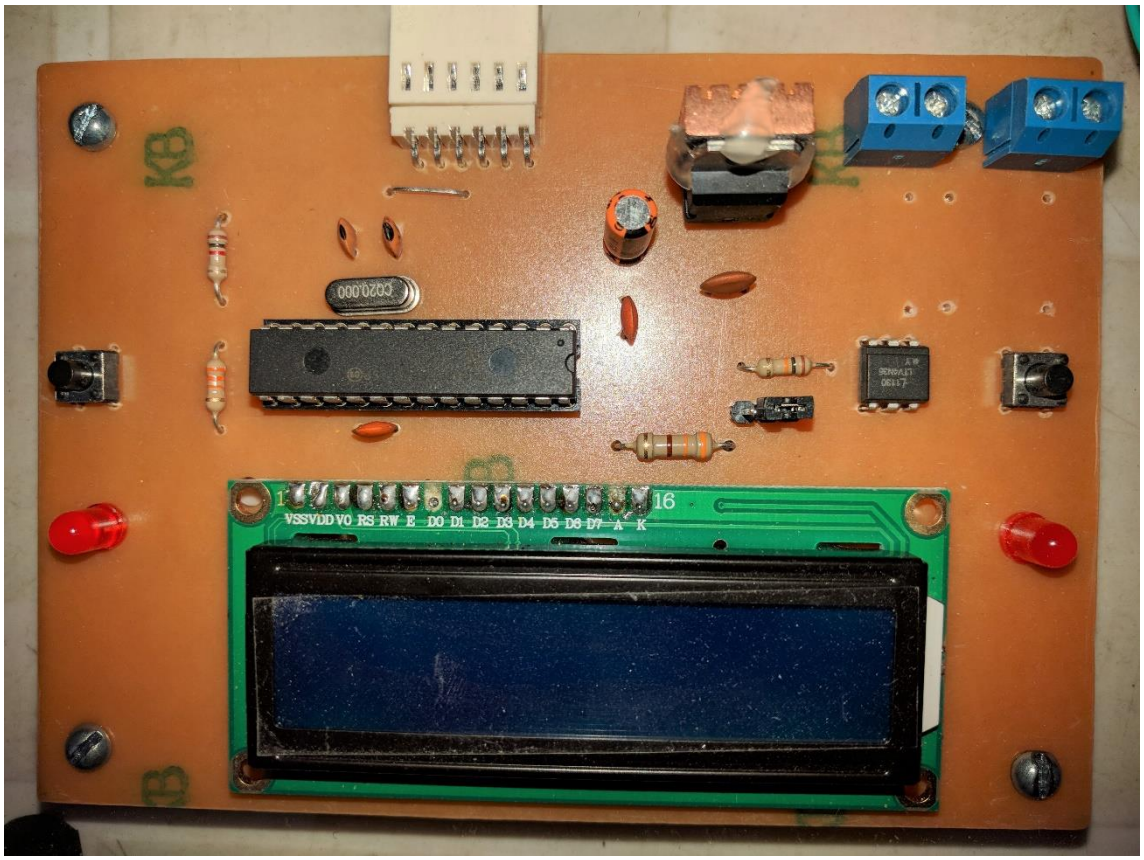

Fig. 39: Fotografía de la Placa de Usuario.

La placa generadora de la señal sonora está formada por un zumbador, junto al resto de la electrónica que permite su control a partir de la señal suministrada por el microcontrolador dsPIC de la Placa Principal.

Fig. 40: Fotografía de la Placa generadora de señal sonora.

La interconexión de las distintas placas mencionadas conforma el sistema completo.

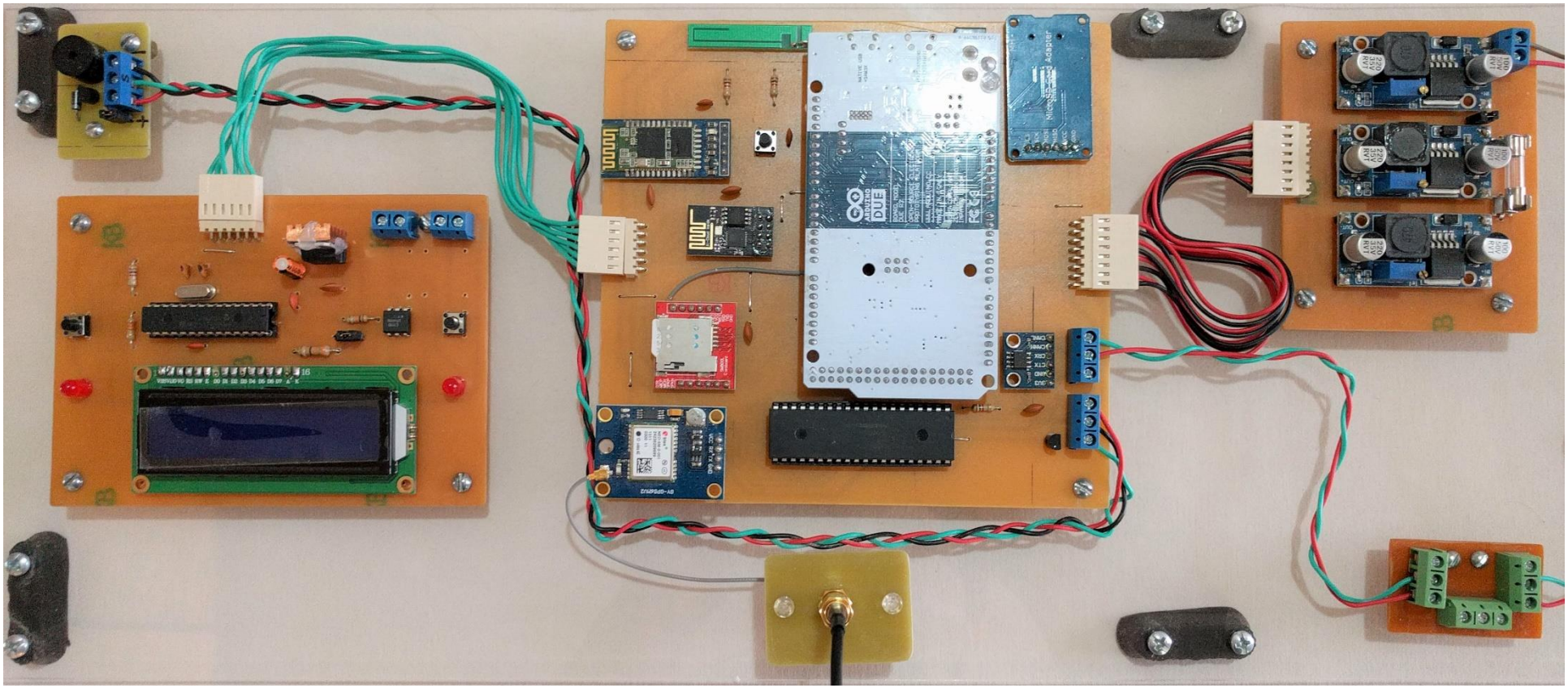

Fig. 41: Fotografía del sistema completo.

5.3 Esquemáticos y circuitos impresos

Los distintos esquemáticos y distribución de los componentes en los circuitos impresos se realizaron utilizado el programa Proteus.

Las imágenes de los circuitos impresos presentados en esta sección no se encuentran en escala real.

En el caso de la Placa Principal, se puede observar que la UART de la interfaz de usuario posee dos conectores diferentes, lo cual permite la conexión directa de un módulo Bluetooth HC-05[2].

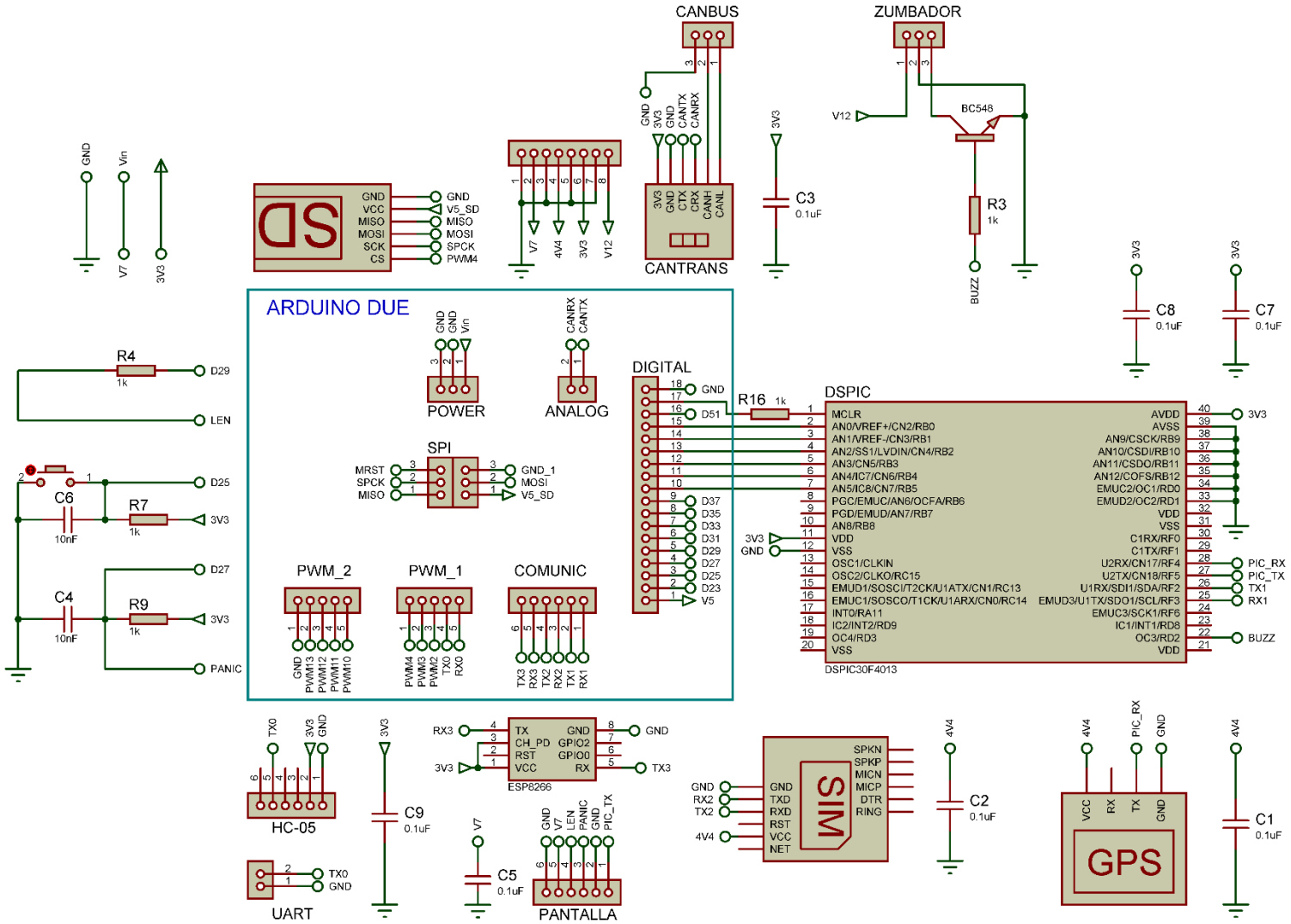

Fig. 42: Esquemático de la Placa Principal.

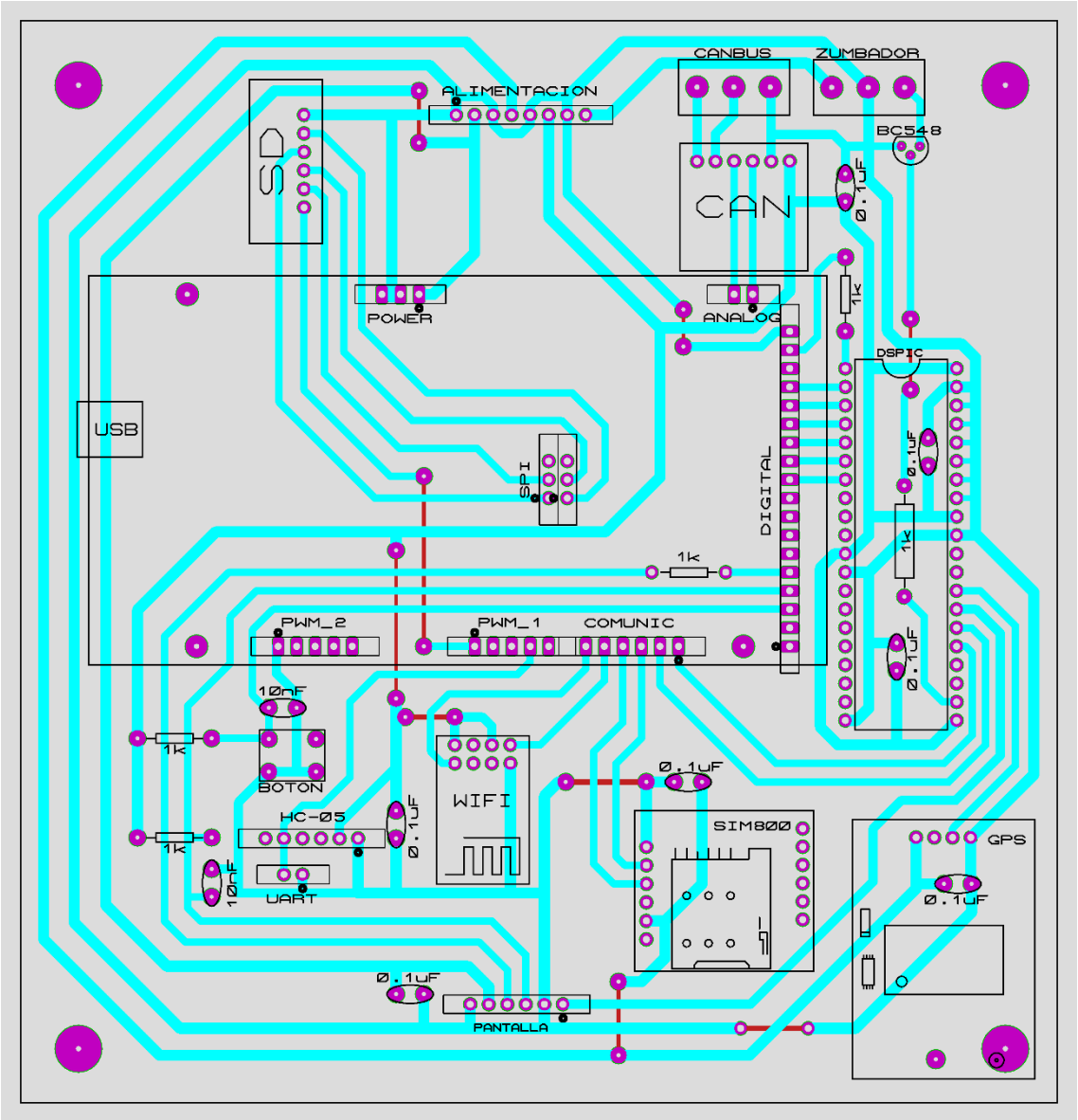

Fig. 43: PCB de la Placa Principal.

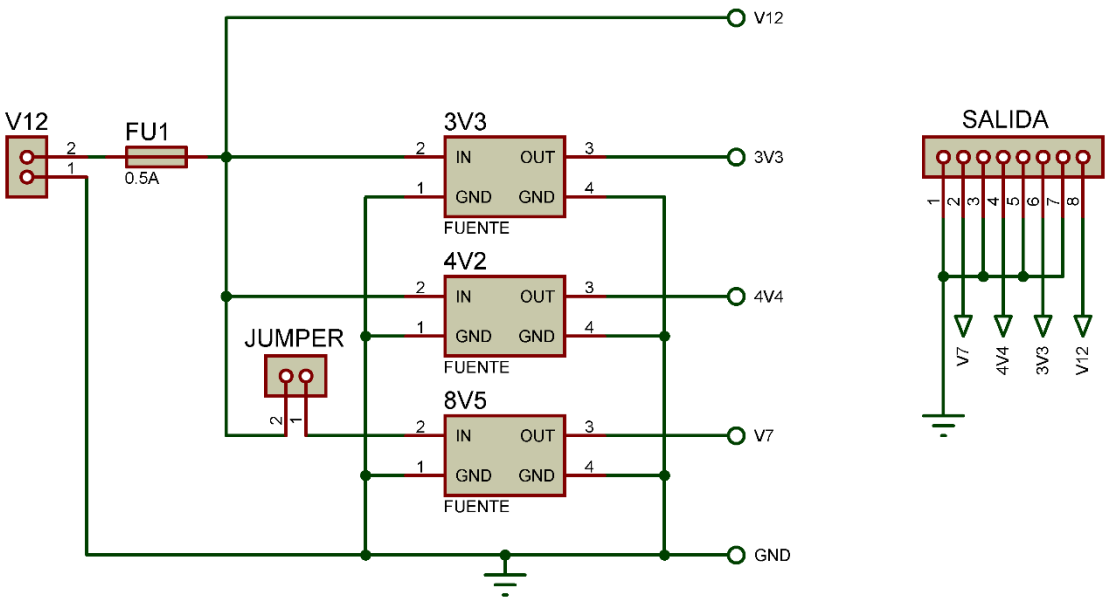

Fig. 44: Esquemático de la Placa de Alimentación.

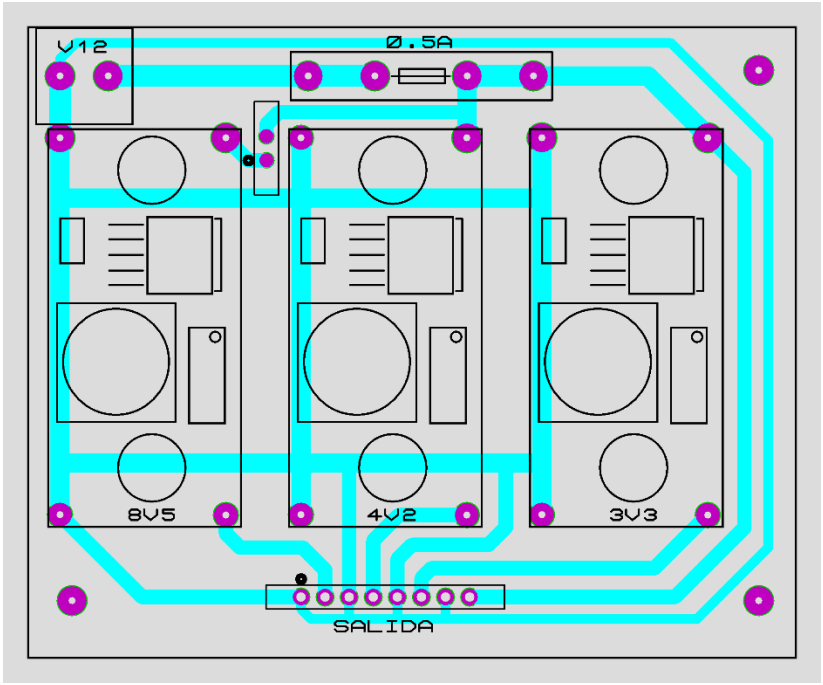

Fig. 45: PCB de la Placa de Alimentación.

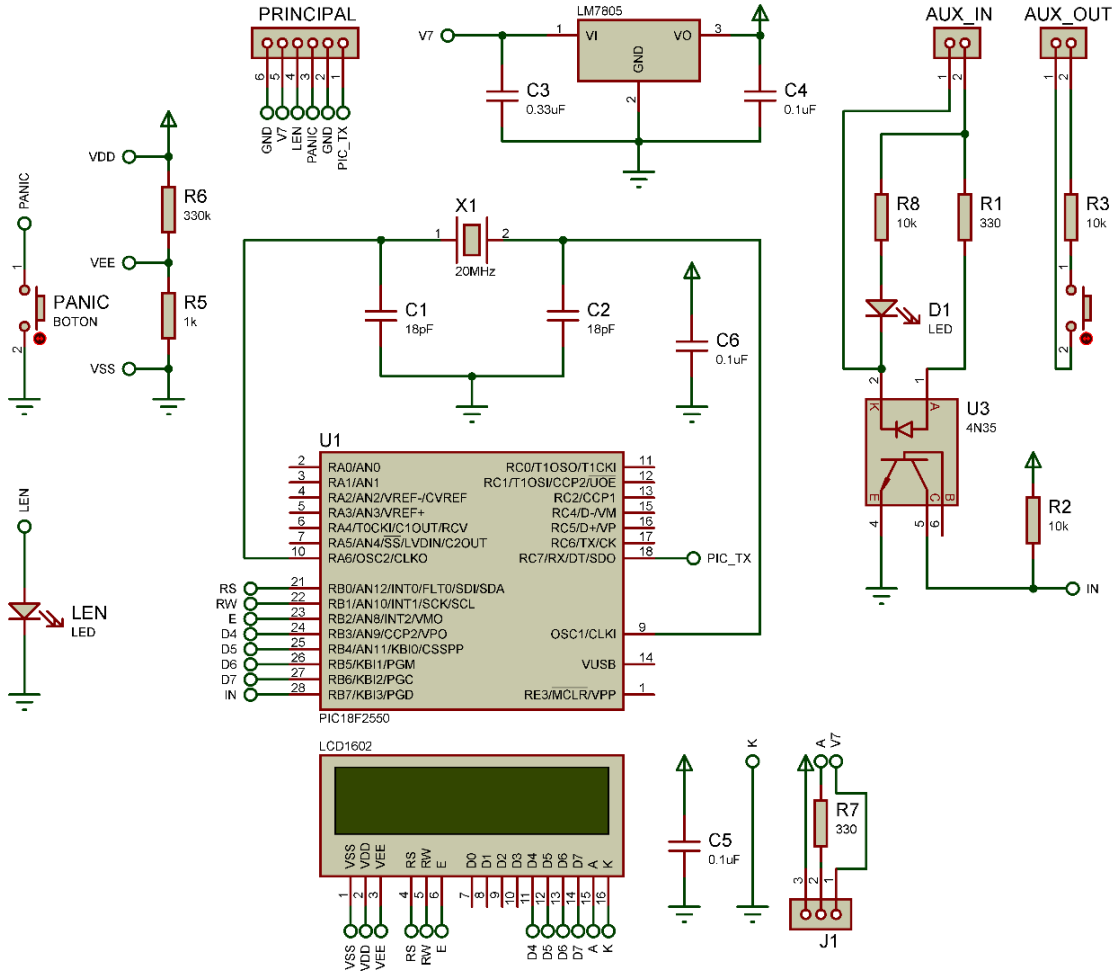

Fig. 46: Esquemático de la Placa de Usuario.

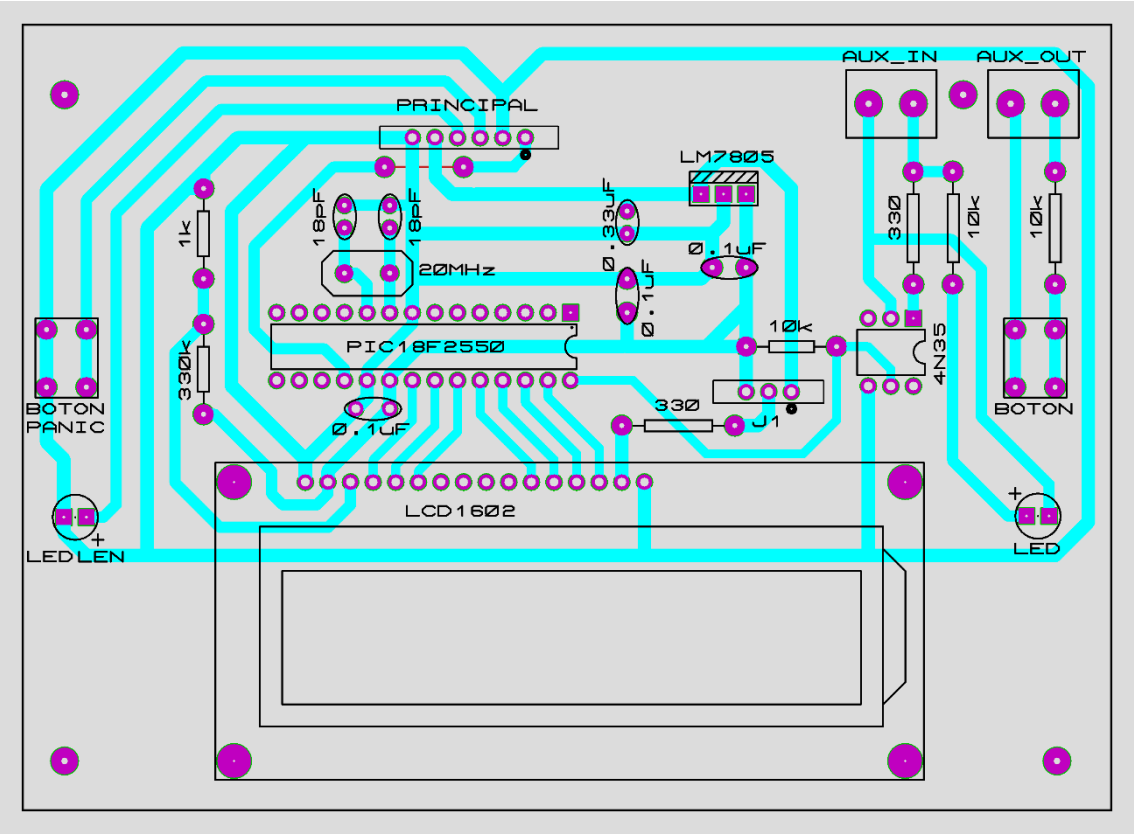

Fig. 47: PCB de la Placa de Usuario.

En el esquemático de La placa generadora de la señal sonora, se puede observar que la misma posee un pequeño jumper que permite anular completamente el funcionamiento del zumbador.

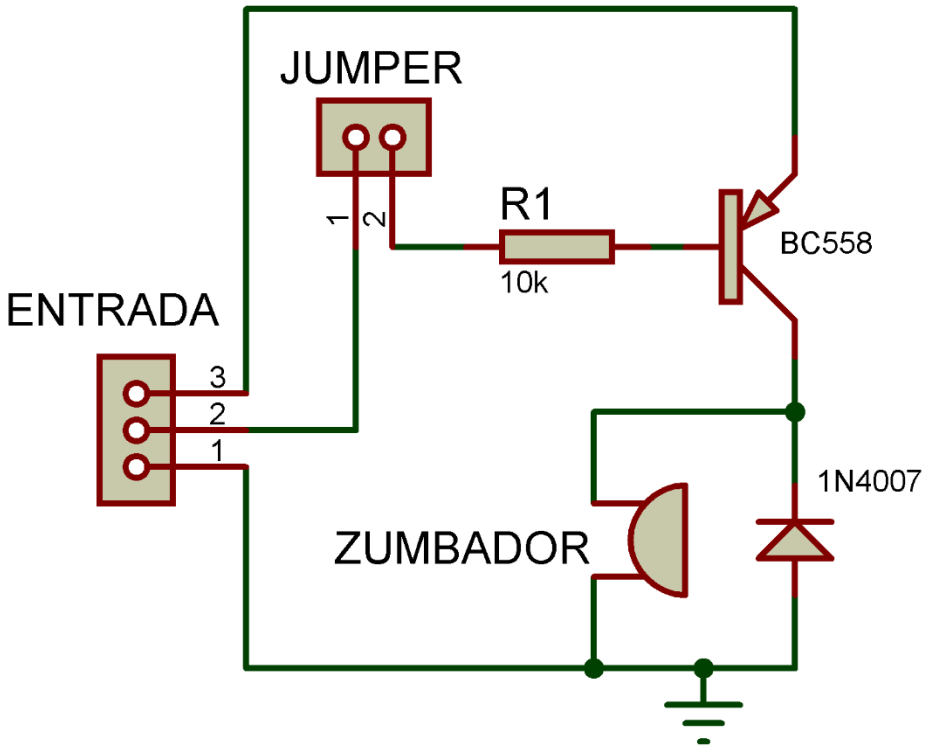

Fig. 48: Esquemático de la Placa generadora de señal sonora.

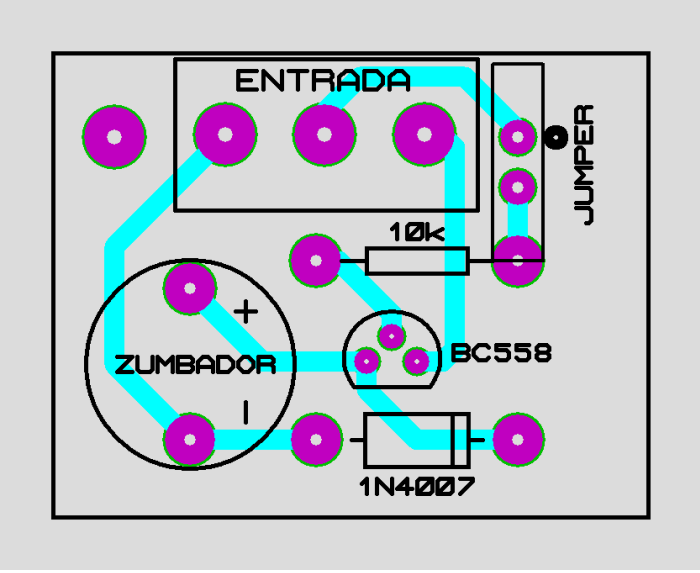

Fig. 49: PCB de la Placa generadora de señal sonora.

5.4 Alimentación del sistema

Debido a que la Placa de Alimentación suministra distintos niveles de tensión, utilizados para cumplir con los requerimientos de todos los componentes del sistema, se realizaron mediciones individuales a las corrientes suministradas por cada fuente.

El valor de la tensión recomendada para la entrada de la Placa de Alimentación es de 12 V, pero la misma puede estar en un rango entre los 9 V y los 13.5 V sin realizar ninguna adaptación en el hardware. Este rango de trabajo se encuentra limitado por las características del zumbador utilizado en la Placa generadora de señal sonora, en caso de que el mismo este anulado o desactivado, el rango se extiende entre los 7 V y los 35 V.

Tabla 42: Utilización de los voltajes de la Placa de Alimentación.

Tensión suministrada por la Placa de Alimentación	Componentes que la utilizan
12V	Placa generadora de la señal sonora.
8.5V	Arduino Due y Placa de Usuario.
4.2V	Modulo GPS y Modulo GSM.
3.3V	Alimentación del resto de los componentes de la placa Principal.

Las distintas mediciones se realizaron con un multímetro UT55 de la marca UNI-T[3].

Todos los valores de las tensiones fueron tomadas utilizando un solo Rango de trabajo, mientras que en las mediciones de corriente, el mismo fue seleccionado según la magnitud de cada valor.

Tabla 43: Características del rango de trabajo utilizado para las mediciones de tensión.

Rango	Resolución	Precisión
20V	10mV	±(0.5%+1)

Tabla 44: Mediciones de las corrientes de consumo.

Tensión de Alimentación [V]	Corriente [mA]	Instrumento		
		Rango	Resolución	Precisión
Suministrado por las fuentes de la Placa de Alimentación				
3.3	220	20A	10mA	±(2%+5)
4.2	180	200mA	100µA	±(1.5%+1)
8.5	136	200mA	100µA	±(1.5%+1)
12	58	200mA	100µA	±(1.5%+1)
Consumo total a la entrada de la Placa de Alimentación				
12	350	20A	10mA	±(2%+5)

Fig. 50: Valor de la medición de corriente en la pantalla del instrumento.

5.5 Bibliografía del capítulo

[1] Hoja de datos: LM2592HV - Texas Instruments. Disponible en internet, fecha de acceso: 24/10/2017. http://www.ti.com/lit/ds/symlink/lm2592hv.pdf

[2] HC Serial Bluetooth Products. Disponible en internet, fecha de acceso: 24/10/2017.

https://cdn.makezine.com/uploads/2014/03/hc_hc-05-user-instructions-bluetooth.pdf

[3] Manual de usuario: Model UT51-55 - UNI-T. Disponible en internet, fecha de acceso: 24/10/2017.

http://www.ageta.hu/pdf/UT51-55.pdf

6 Ensayos

6.1 Introducción

La realización de ensayos a las distintas partes del sistema permite evaluar el correcto funcionamiento del mismo, ayudando al desarrollador a encontrar errores en las funciones planteadas o comportamientos imprevistos en los módulos conectados.

Luego de verificar el funcionamiento eléctrico de cada uno de los módulos y microcontroladores del sistema (alimentación, conexiones, etc.) se realizaron los ensayos a las distintas funciones de comunicación programadas, para lo cual se utilizó un conjunto de herramientas de hardware y de software, junto a algunas líneas adicionales incluidas en código del Arduino Due.

Las principales herramientas de hardware utilizadas fueron dos conversores USB a UART con los drivers necesarios para seleccionarlos como puertos del tipo COM y un multímetro para realizar las mediciones eléctricas necesarias.

Debido a la cantidad de herramientas de software utilizadas, la mayoría serán nombradas directamente en la sección correspondiente del capítulo. En el caso de las terminales de comunicación con los puertos COM, los programas utilizados fueron Hercules[32], Putty[33] y el monitor serie incluido en el IDE de Arduino.

Las líneas adicionales incluidas en el código del programa cargado en el Arduino Due permiten utilizar a la UART de programación como una interfaz de depuración de alto nivel.

6.2 Ensayos de ubicación y velocidad

El ensayo realizado consiste en la simulación de los parámetros entregados por el módulo GPS durante su funcionamiento a lo largo de situaciones límites y representativas, para evaluar el correcto funcionamiento del sistema.

Las pruebas realizadas se dividen en dos etapas. En la primera, el microcontrolador dsPIC fue evaluado de manera aislada, realizando una serie de conexiones que le permitieron comunicarse con dos conversores USB a UART. Mientras que en la segunda, el mismo fue conectado directamente al Arduino Due.

[32] HW group. Disponible en internet: https://new.hwg.cz/software/hercules-setup-utility
[33] Simon Tatham. Disponible en internet: http://www.putty.org/

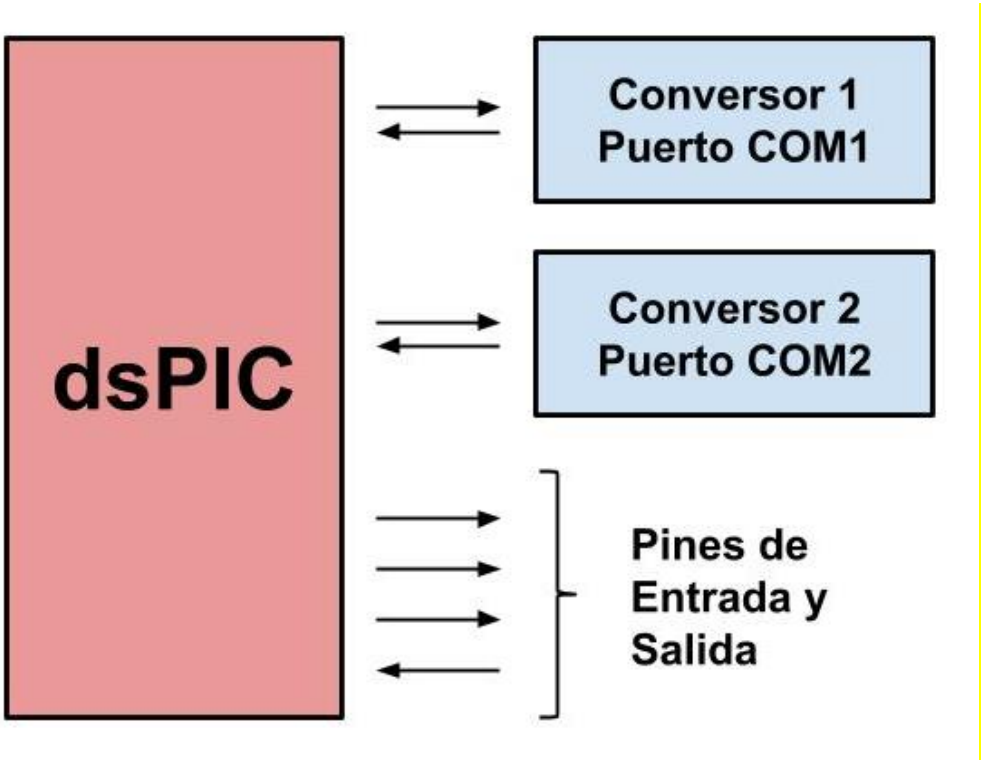

Fig. 51: Conexión entre el dsPIC y los conversores USB a UART.

Además de las herramientas principales, para realizar el envío de los parámetros al microcontrolador, se desarrolló un programa escrito en Python que permite una transferencia continua de la información almacenada en una computadora personal a los dispositivos conectados en los puertos COM. Mientras que para visualizar de forma continua la ubicación contenida en los parámetros, el programa que se utilizo es el U-Center[34], del mismo fabricante del módulo GPS.

El procedimiento utilizado durante la primera etapa consiste en la ejecución de una serie acciones:

- Escribir en un archivo de texto plano los parámetros a enviar, teniendo en cuenta que los mismos deben representar el funcionamiento real del módulo durante la situación evaluada.

- Establecer mediante un programa terminal una conexión al puerto COM1, para realizar el envío de los mensajes establecidos en el protocolo diseñado y la recepción de la información solicitada.

- Ejecutar en un intérprete de Python el programa desarrollado, seleccionando al puerto COM2 como destino de la información enviada.

- Evaluar que la información solicitada para cada mensaje sea la correcta, verificando que el funcionamiento de los pines de estado y de salida sea el previsto.

[34] U-blox. Disponible en internet: https://www.u-blox.com/en/product/u-center-windows

- Modificar el estado de los pines de entrada y analizar las acciones realizadas por el microcontrolador.

Tabla 45: Ejemplo del contenido del archivo utilizado durante la simulación.

Parámetros a simular
$GPGGA,234449.00,2925.41521,S,06650.71932,W,1,07,1.19,495.4,M,30.9,M,,*53
$GPGSA,A,3,31,29,12,21,05,25,20,,,,,2.66,1.19,2.38*09
$GPGSV,3,1,11,02,06,139,,05,21,103,25,12,31,069,30,14,03,288,*7E
$GPGSV,3,2,11,20,31,054,20,21,43,325,22,24,02,017,,25,66,098,19*77
$GPGSV,3,3,11,26,08,227,,29,56,174,24,31,38,241,16*47
$GPGLL,2925.41521,S,06650.71932,W,234449.00,A,A*6E
$GPRMC,234450.00,A,2925.41388,S,06650.71967,W,4.731,354.18,230917,,,A*60
$GPVTG,354.18,T,,M,4.731,N,8.762,K,A*3C

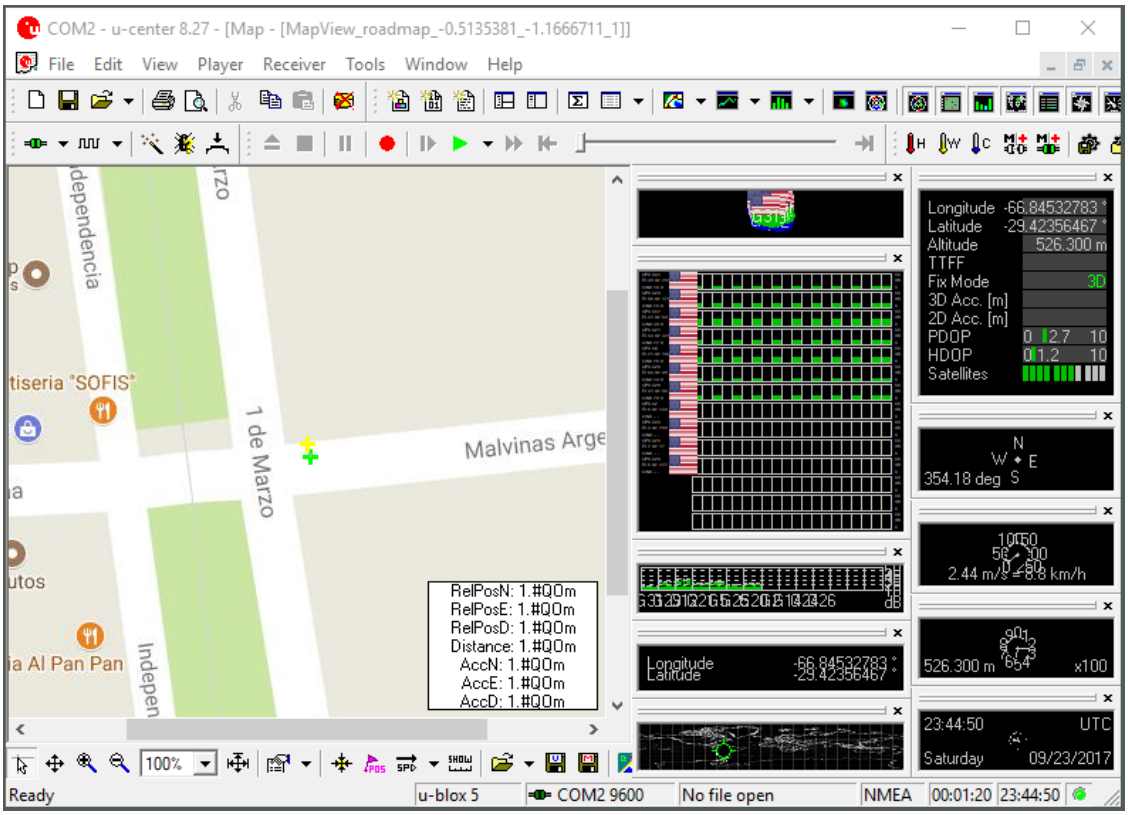

Fig. 52: Mapa generado por el programa U-Center a partir de los parámetros de una ubicación simulada.

La segunda etapa de la evaluación se realizó de una manera similar a la primera, con la diferencia de que el primer conversor fue reemplazado directamente por el Arduino Due, verificando desde una computadora a través de la UART de depuración que las funciones implementadas para la comunicación entre ambos microcontroladores fueran las correctas.

6.3 Ensayos de Almacenamiento

El procedimiento de ensayo al sistema de archivos y a sus funciones de lectura y escritura se realizó de manera separada para cada tipo.

6.3.1 Archivos de configuración

En el caso de los archivos de configuración, debido a que los mismos son los más simples del sistema, su evaluación se realizó directamente desde la UART de depuración.

6.3.2 Archivos web

Los archivos encargados de generar las distintas páginas web, fueron evaluados durante su desarrollo con el navegador Chrome desde una computadora personal, verificando su correcta visualización en distintos formatos de pantalla. Esto se logró gracias a las funciones incluidas en el programa conocidas como “Herramientas para desarrolladores”.

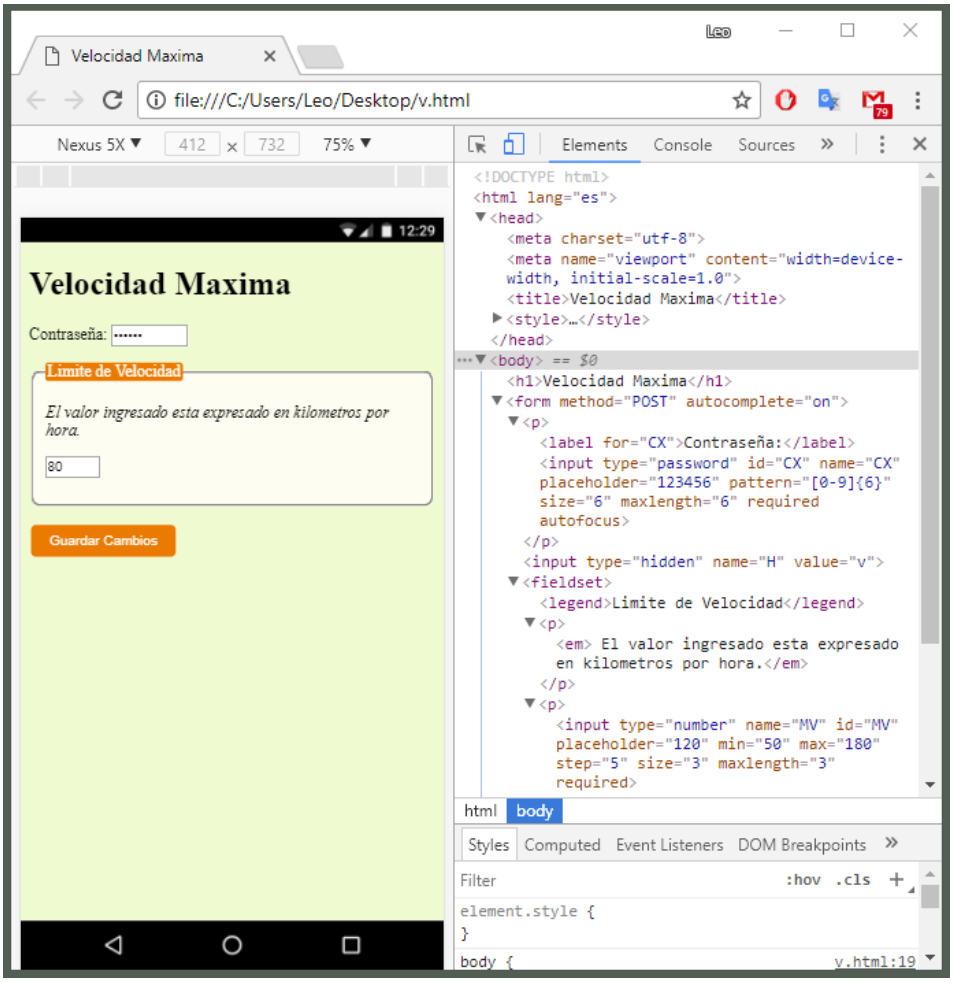

Fig. 53: Pagina abierta junto a Herramientas para Desarrolladores.

La evaluación del funcionamiento de los formularios de las páginas web se realizó creando un servidor TCP/IP dentro de la computadora con ayuda del programa Hercules. Agregando el atributo *action* a los formularios con la dirección IP del nuevo servidor, se pudo verificar que

los mensajes del protocolo HTTP eran correctos. Para realizar esta tarea, la dirección IP utilizada fue la del *localhost* (127.0.0.1) con el puerto 80.

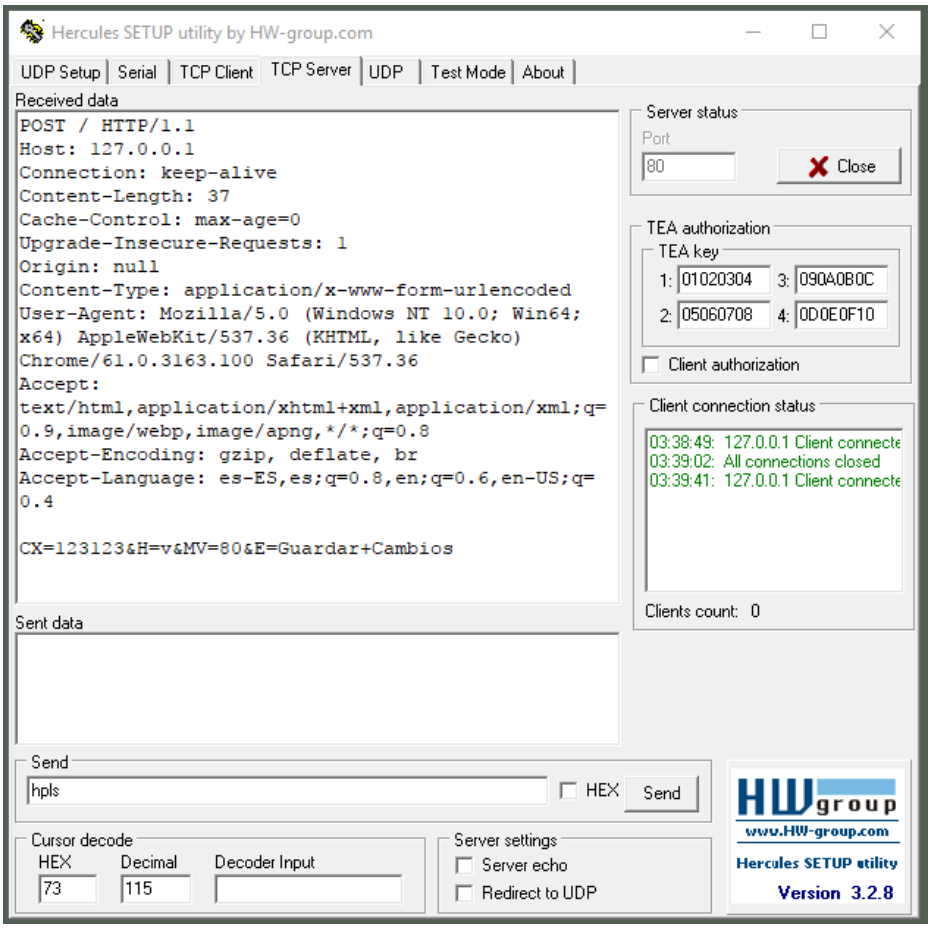

Fig. 54: Mensajes generados por el método POST del protocolo HTTP.

El funcionamiento de la página de visualización de mapas también se probó directamente en el navegador web, para ello se escribió manualmente un vector con ubicaciones previamente seleccionadas.

Tabla 46: Contenido del vector de ubicaciones.

Etiqueta	Latitud	Longitud
Plaza25	-29.41261	-66.85580
https://www.google.com.ar/maps/search/-29.41261,+-66.85580		
Monumento	-29.43808	-66.83625
https://www.google.com.ar/maps/search/-29.43808,+-66.83625		

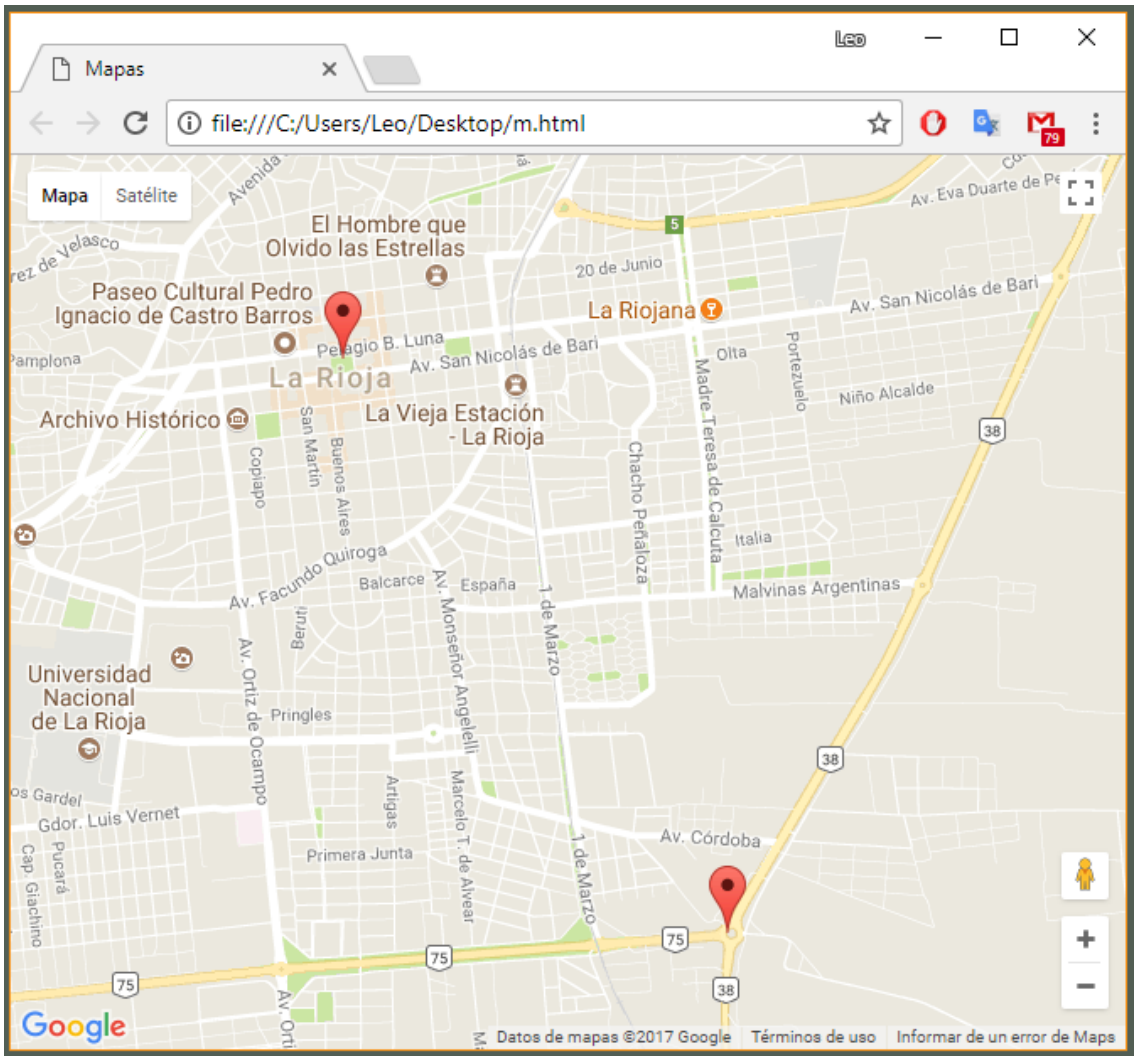

Fig. 55: Pagina de mapas con un vector de ubicaciones escrito manualmente.

6.3.3 Archivos de ubicaciones

La verificación de las funciones utilizadas para almacenar información en los archivos de configuraciones se realizó a través de un procedimiento similar al descripto en la segunda etapa de los ensayos de ubicación y velocidad, es decir, generando puntos de control en ubicaciones simuladas.

El vector de ubicaciones fue escrito directamente sobre la página de mapas, mientras que el archivo GPX fue visualizado en el programa Google Earth[35], comparando los puntos de control observados con las ubicaciones simuladas.

[35] Google Inc. Disponible en internet: https://www.google.com/intl/es/earth/desktop/

Tabla 47: Archivo de Ubicaciones GPX.

Archivo GPX:
<?xml version="1.0" encoding="UTF-8" standalone="no" ?> <gpx> <wpt lat="-29.41261" lon="-66.85580"><time>2000-01-01T20:58:48Z</time><name>Plaza25</name></wpt> <wpt lat="-29.43808" lon="-66.83625"><time>2000-01-01T21:02:21Z</time><name>Monumento</name></wpt> </gpx>

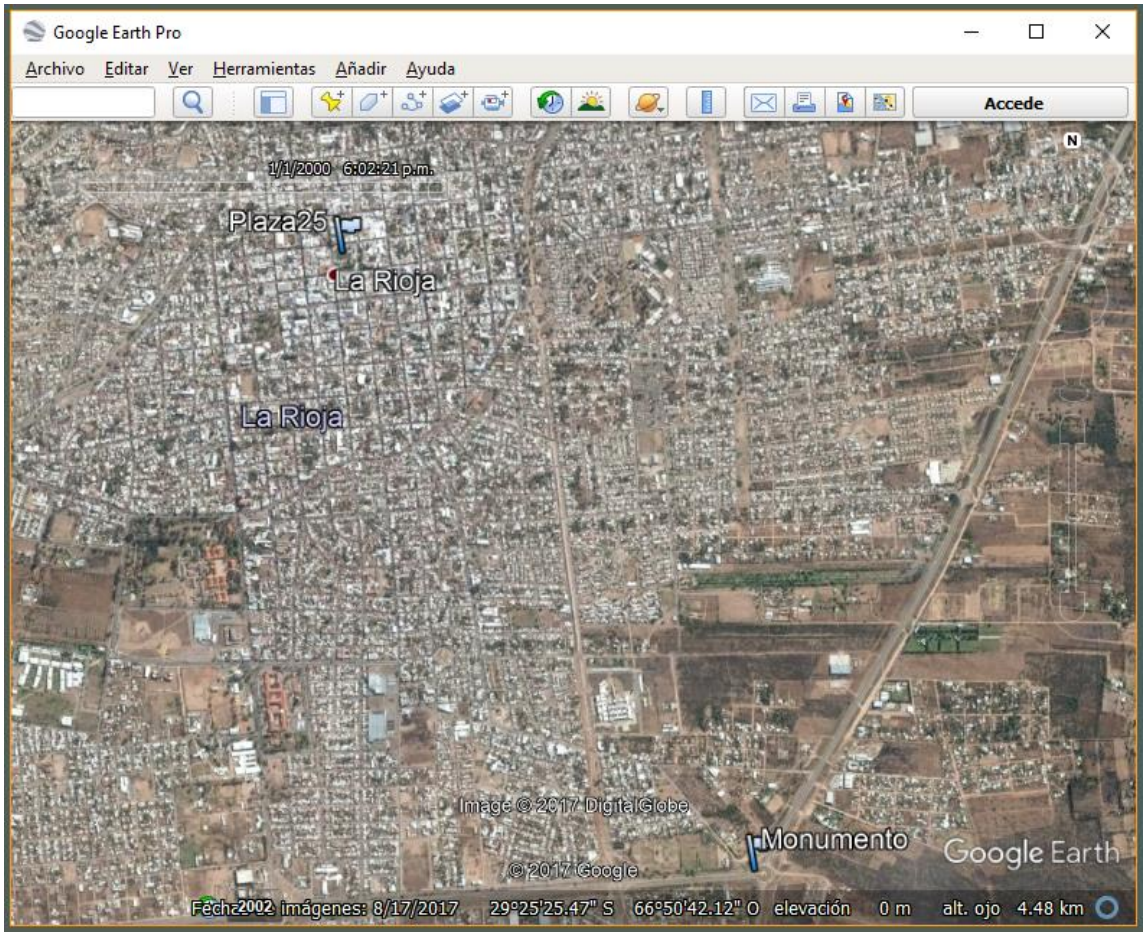

Fig. 56: Archivo GPX visualizado en el programa Google Earth.

6.4 Ensayos web

Los ensayos web se realizaron simplemente accediendo al servidor desde las dos redes disponibles. Para el caso del punto de acceso, se conectó una computadora personal a la red generada por el modulo, mientras que para la conexión restante se utilizó un *Router* con acceso a internet. En ambos casos, se verifico la conexión con el modulo a través de la herramienta *ping* del conjunto de protocolos TCP/IP.

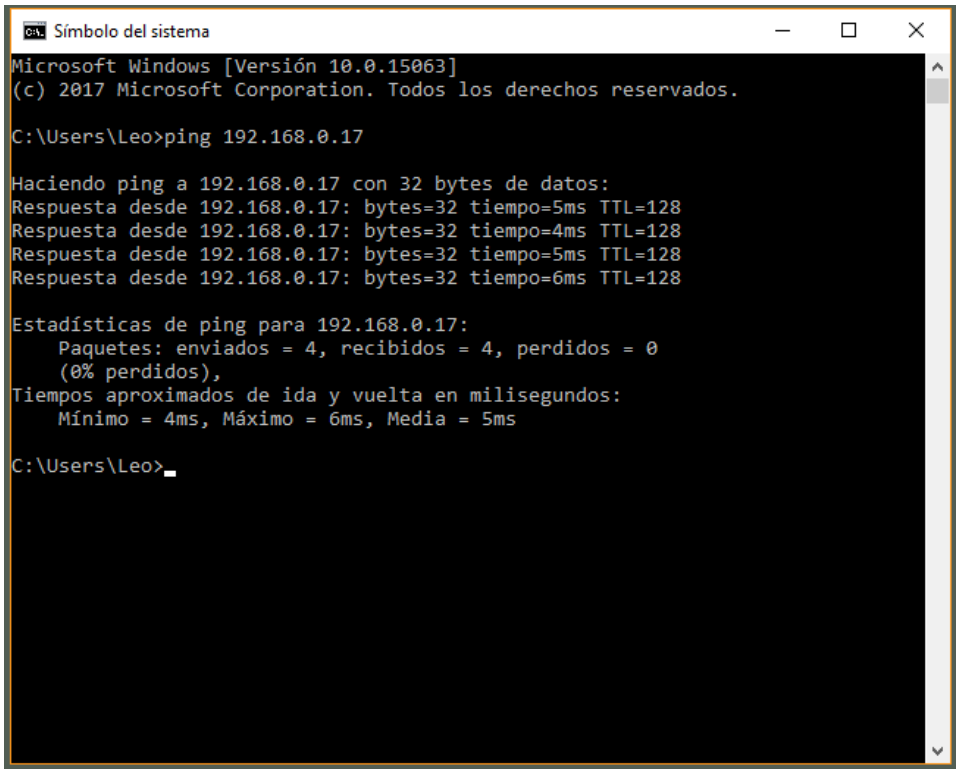

Fig. 57: Ping a la dirección IP del módulo wi-fi desde el CMD de Windows.

El funcionamiento del servidor y la correcta transferencia de los distintos archivos web se comprobaron directamente con el navegador, mientras que las acciones realizadas por cada una de las páginas de configuración fueron evaluadas a través de la UART de depuración.

Fig. 58: Descarga del archivo GPX desde el navegador web Chrome.

6.5 Ensayos de mensajes de texto

La evaluación de estas funciones fue realizada enviado diferentes mensajes de texto desde distintos números celulares, verificando a través de la UART de depuración que el sistema solo reaccionara a las raíces gramaticales definidas cuando son recibidas desde el número del administrador, modificando en cada caso el valor del modo de funcionamiento.

La función del envío de mensajes fue evaluada de manera conjunta, debido a que todo mensaje recibido desde el número del administrador genera una respuesta automática.

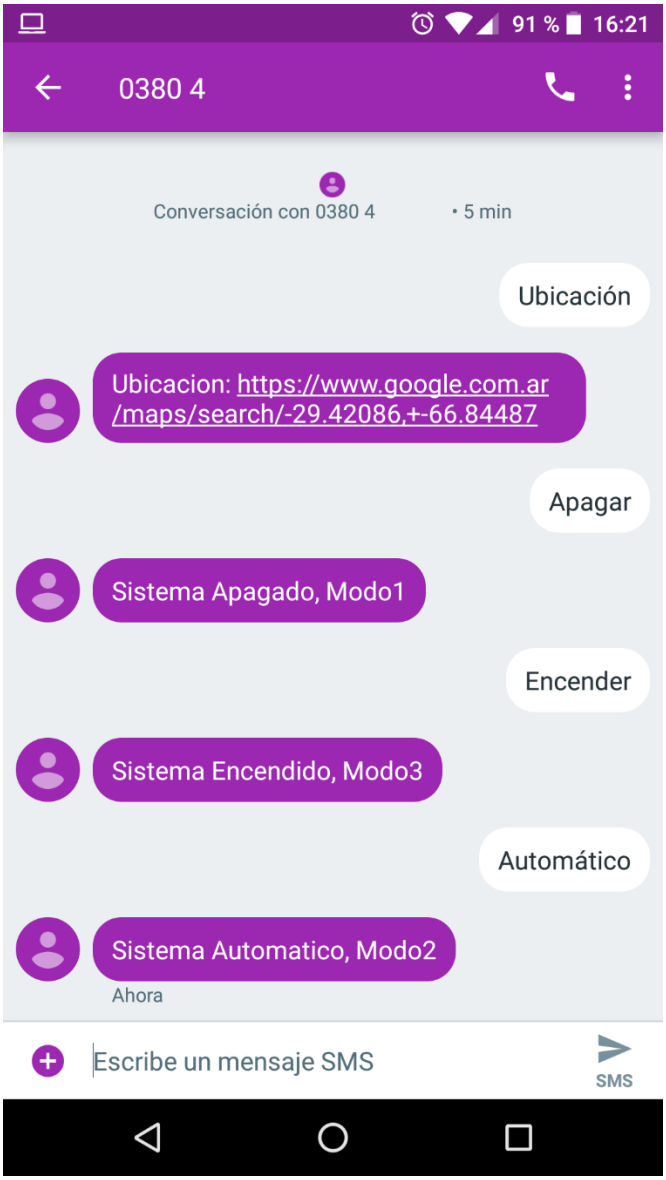

Fig. 59: Captura de pantalla de los mensajes de texto en el teléfono del administrador.

Por último, se probó que el sistema bloquee todas las llamadas entrantes, independientemente del número de celular del cual se realicen.

6.6 Ensayos del reloj de tiempo real

Debido a que durante los ensayos realizados en las funciones de envío y recepción de mensajes de texto se verifico el correcto funcionamiento del módulo GSM, la evaluación del reloj en tiempo real se realizó simplemente configurando los valores utilizados por el mismo a partir de las funciones implementadas y corroborando que la fecha y la hora permanecieran actualizada durante largos periodos de tiempo.

6.7 Ensayos de comunicación CAN

Para realizar el ensayo de las comunicaciones a través del bus CAN, se construyó una placa adicional a partir de un microcontrolador dsPIC30F4013 y un transceptor Sn65hvd230.

En el caso de las funciones implementadas en el Arduino Due, para verificar el envío y la recepción de la información transmitida se utilizó a la UART de depuración, enviado el contenido de cada mensaje recibido directamente a una computadora.

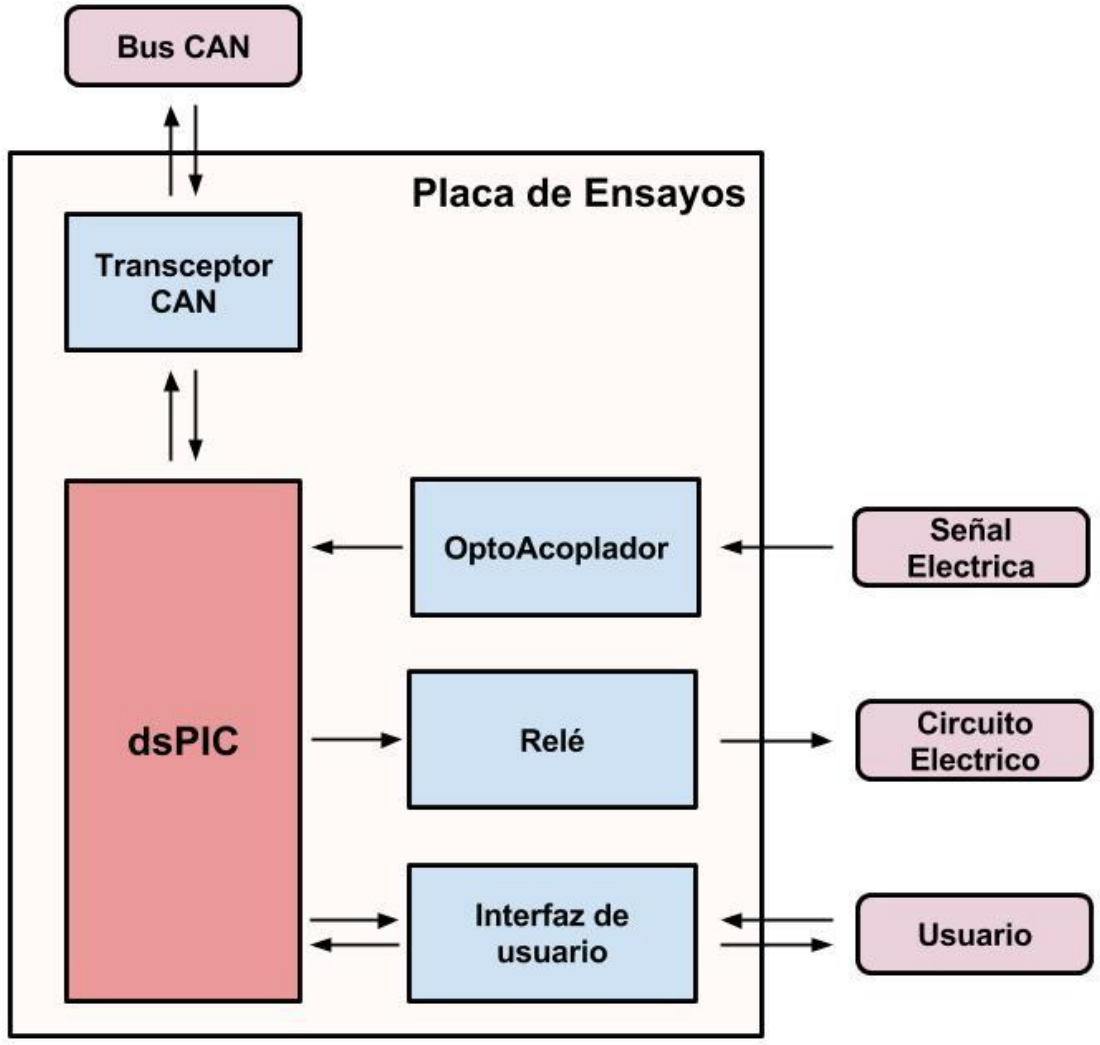

Fig. 60: Esquemático de la placa de ensayos utilizada para la comunicación CAN.

6.7.1 Entradas

La placa cuenta con dos tipos de entradas utilizadas para generar estímulos o bifurcaciones en el funcionamiento del programa. Los botones, que forman parte de la interfaz de usuario y las entradas opto acopladas, utilizadas para ingresar en el sistema señales del tipo eléctricas.

6.7.2 Salidas

La placa también cuenta con dos tipos de salidas, las luminosas utilizadas como interfaz de usuario y las salidas comandadas por los relés.

6.7.3 Funcionamiento

El funcionamiento del dispositivo se encuentra diseñado para permitir una simulación de conexión al sistema de encendido de un automóvil, brindando la capacidad de bloquear el encendido del motor a partir del control de los relés, midiendo el estado del motor y entregando al usuario una serie de botones de funcionalidad variable. Todo esto a partir de la realización de una serie de acciones.

6.7.3.1 Atender los mensajes entrantes desde el bus CAN

Los mensajes entrantes por el bus determinan el modo de funcionamiento e indican si es que el sistema principal se encuentra conectado.

En caso de la llegada de un mensaje IDinforma con un identificador de placa 0, es decir un mensaje proveniente del sistema principal, la placa responderá un IDinforma con un identificador de placa igual al valor 42.

6.7.3.2 Atender el estado de una entrada opto acoplada

En caso de que entrada cambie a un estado alto, solicitar por un mensaje CAN que el servidor web sea desactivado. En el momento en el que la entrada vuelva a un estado bajo, el servidor debe ser encendido y el valor de la velocidad máxima debe ser configurado en 10 Km/h.

Este funcionamiento permite simular la acción de una conexión con el sistema eléctrico del automóvil utilizada para indicar si el motor se encuentra apagado o encendido, por lo que los cambios en la señal pueden ser evaluados como:

- Cambio al estado alto: Motor encendido, vehículo a punto de desplazarse. En esta situación el servidor web no es necesario.

- Cambio al estado bajo: Motor apagado, vehículo detenido. En esta situación el servidor web es necesario y la velocidad máxima se configura a un valor que permita saber si el automóvil es remolcado del lugar.

6.7.3.3 Evaluar el valor del modo de funcionamiento

En caso de que la señal opto acoplada se encuentre en estado bajo, el funcionamiento del programa depende del valor contenido en la variable utilizada para almacenar el modo de funcionamiento.

Debido a que el funcionamiento de los relés puede utilizarse para abrir o cerrar un circuito eléctrico, los mismos podrían ser utilizados para bloquear el encendido del automóvil. De esta manera, en el ensayo se supone que cuando los relés se encuentran en estado abierto el encendido del vehículo estará desactivado.

A partir de lo anterior, los modos de funcionamiento definen el siguiente comportamiento:

- Modo1: Sistema de encendido desbloqueado.

- Modo2: Sistema de encendido desbloqueado durante 5 segundos luego de pulsar el botón A.

- Modo3: Sistema de encendido bloqueado.

6.7.3.4 Atender la pulsación de los botones

La pulsación de cada botón generara el envío de una cadena de caracteres diferentes al sistema principal:

- Botón A: Atendido solamente durante el Modo2 con la señal opto acoplada en estado bajo. Envía un mensaje IDenvia con los caracteres “P42: Boton A”.

- Botón B: Atendido durante todo el desarrollo normal del programa. Envía un mensaje IDalmacena con los caracteres “P42: Boton B”.

6.7.3.5 Definir el estado de los indicadores luminosos

Los indicadores luminosos permiten al usuario conocer el estado del dispositivo:

- Led_desbloqueo: Indica que el encendido del automóvil esta desbloqueado.

- Led_EntradaAux: Indica que la entrada opto acoplada está en estado alto.

- Led_sistema: Indica que el sistema se encuentra encendido. Titilara en el caso de que una pulsación de un botón sea atendida por el dispositivo.

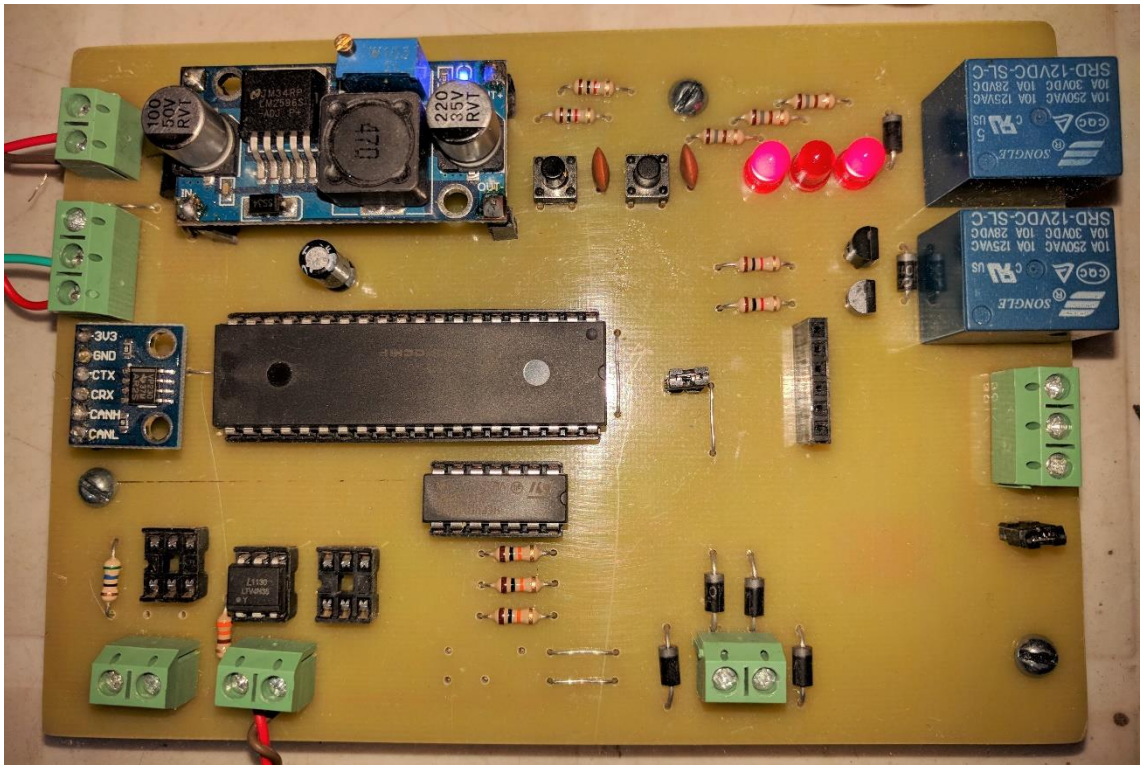

Fig. 61: Placa de ensayos de la comunicación CAN.

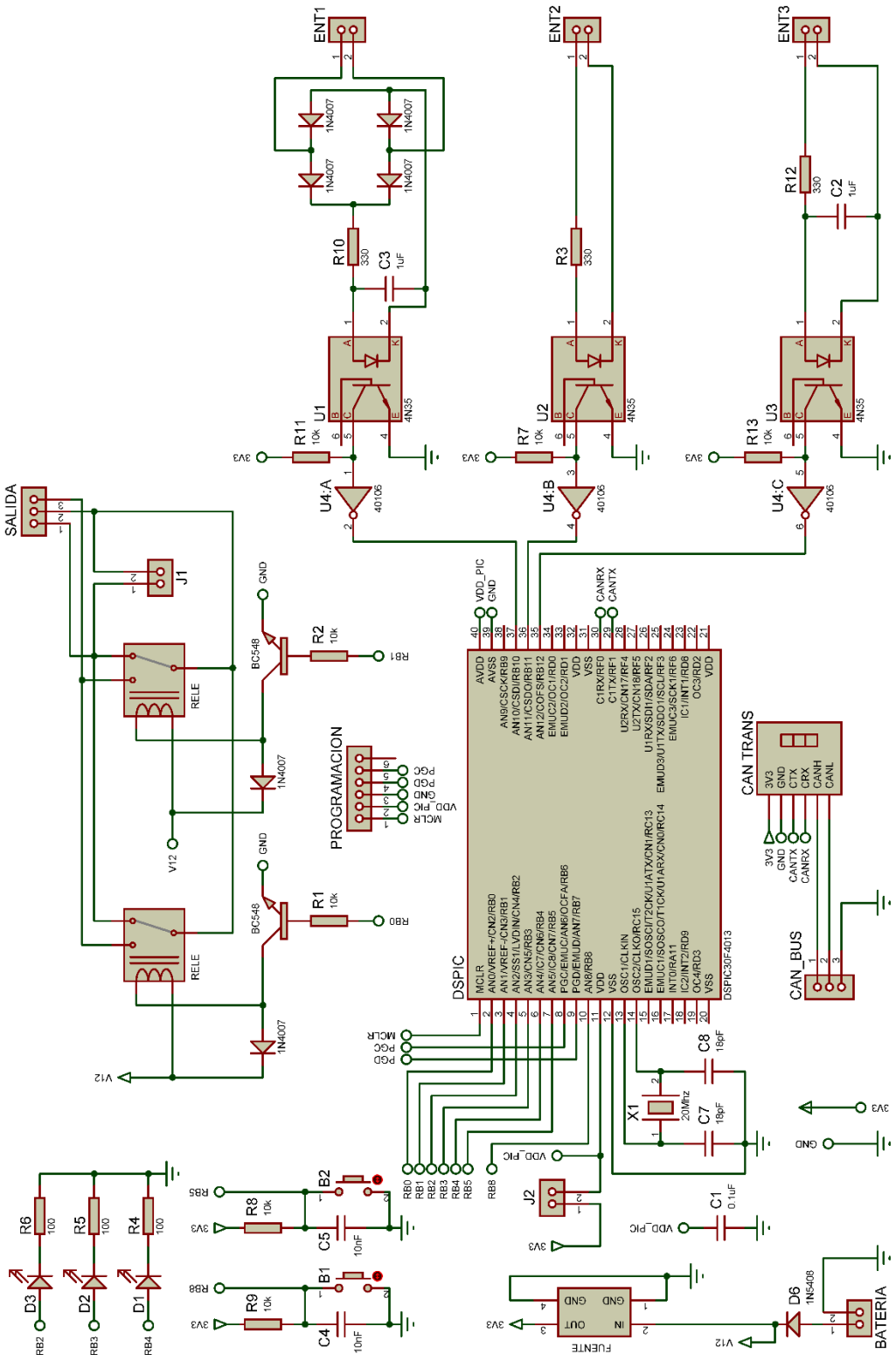

Fig. 62: Esquemático de la Placa de ensayos de la comunicación CAN.

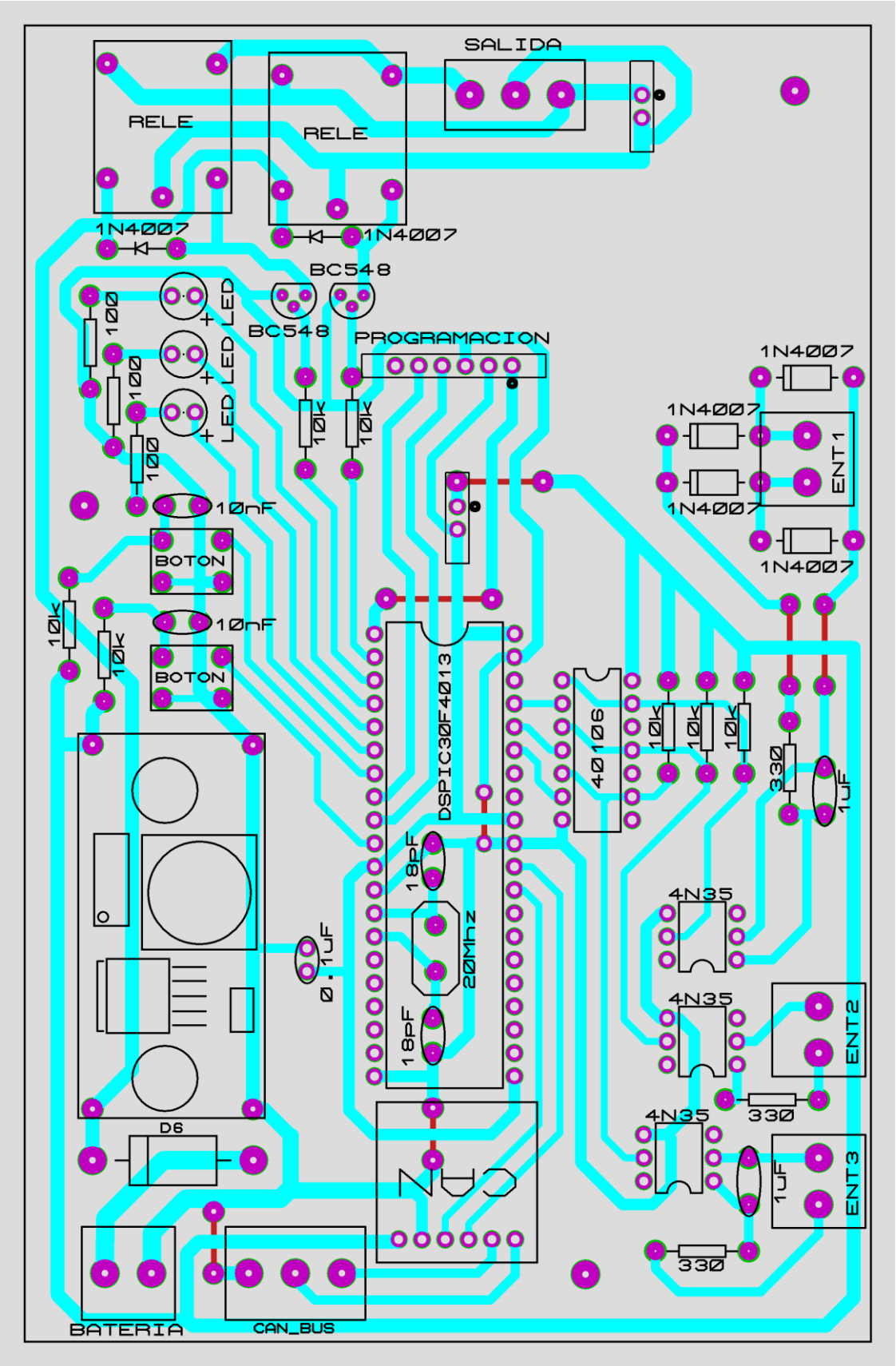

Fig. 63: PCB de Placa de ensayos de la comunicación CAN.

6.8 Ensayos de la interfaz de usuario

Las pruebas de la interfaz de usuario se realizaron primero en la interfaz de salida, debido a que la misma depende solo del estado del sistema y de la interacción de las distintas funciones previamente evaluadas.

En el caso de la UART adicional, debido a que se trata de la misma utilizada para la depuración, los mensajes entregados por el sistema durante su funcionamiento fueron visualizados sin inconvenientes en la computadora personal.

La señal sonora funciono de manera correcta durante los ensayos de ubicación y velocidad. Mientras que el indicador luminoso LEN se comportó adecuadamente, titilando de forma constante durante una prueba de encendido sin la tarjeta SD.

El ensayo del visualizador se realizó en dos etapas separas. En la primera, los mensajes fueron entregados desde una computadora a través de un conversor USB a UART, logrando observar los resultados esperados en cada uno. La segunda se realizó directamente con el visualizador conectado al resto del sistema, llamando a la función generadora de los mensajes desde la UART de depuración.

Fig. 64: Mensajes en el dispositivo visualizador.

En el caso de la interfaz de entrada, las pruebas realizadas son del tipo integradoras, ya que las funciones utilizadas por las mismas abarcan a casi la totalidad de las que fueron definidas para el sistema.

El botón PANIC realizo el envío de los mensajes correctamente, agregando la información de la ubicación cuando la misma estaba disponible. Mientras que el botón RQST entrego toda la información solicitada a través de la UART y envió los caracteres correspondientes al visualizador de la pantalla.

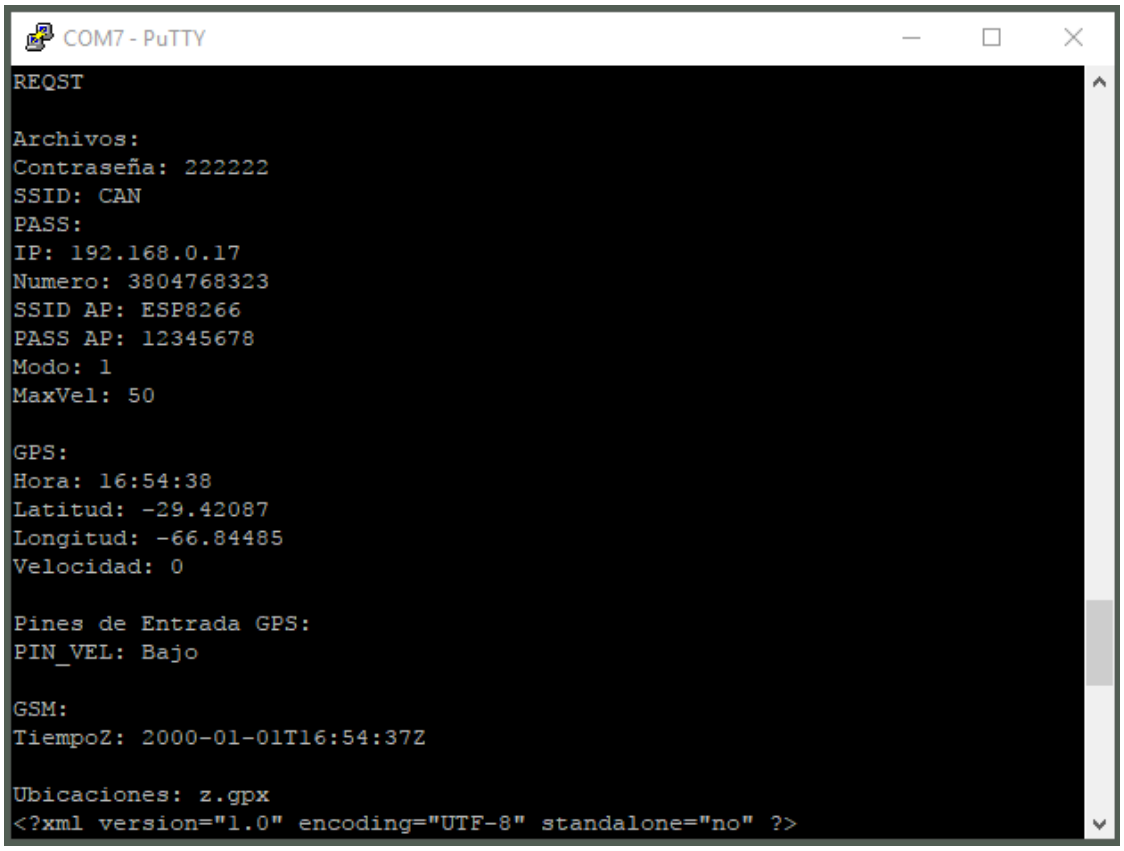

Fig. 65: Información enviada por la UART al pulsar el botón RQST, visualizada en la terminal del programa Putty.

6.9 Informes de errores

Durante la realización de todos ensayos del sistema, los informes de errores fueron enviados por la UART adicional, informando correctamente el nombre de las funciones en las que el problema estaba ocurriendo, con lo que se logró corregir de manera adecuada a las líneas de código correspondientes.

6.10 Ensayos de campo

Luego de realizar todas las pruebas a cada una de las partes del sistema, verificando el correcto funcionamiento de las mismas, se procedió a ensayar el módulo GPS.

El ensayo del módulo consistió en la realización de dos recorridos por trayectorias previamente determinadas, en un rango de velocidades permitidas por las legislaciones

vigentes[36] [37], para verificar que los datos entregados por el GPS fueran correctos. El vehículo utilizado durante las pruebas fue un automóvil Fiat Palio modelo 2000.

En el primer recorrido, los datos obtenidos se almacenaron en una computadora, para verificar mediante el programa U-Center que la información relacionada con la ubicación fuera correcta.

Tabla 48: Características del primer recorrido.

Primer recorrido	
Inicio	Ruta nacional número 38, intersección ruta provincial número 5.
Final	Ruta nacional número 38, intersección avenida Malvinas Argentinas.
Trayectoria	Ruta nacional número 38, dirección sur.
Observaciones	A lo largo del recorrido la velocidad se mantuvo dentro de los límites establecidos por las normas de tránsito.

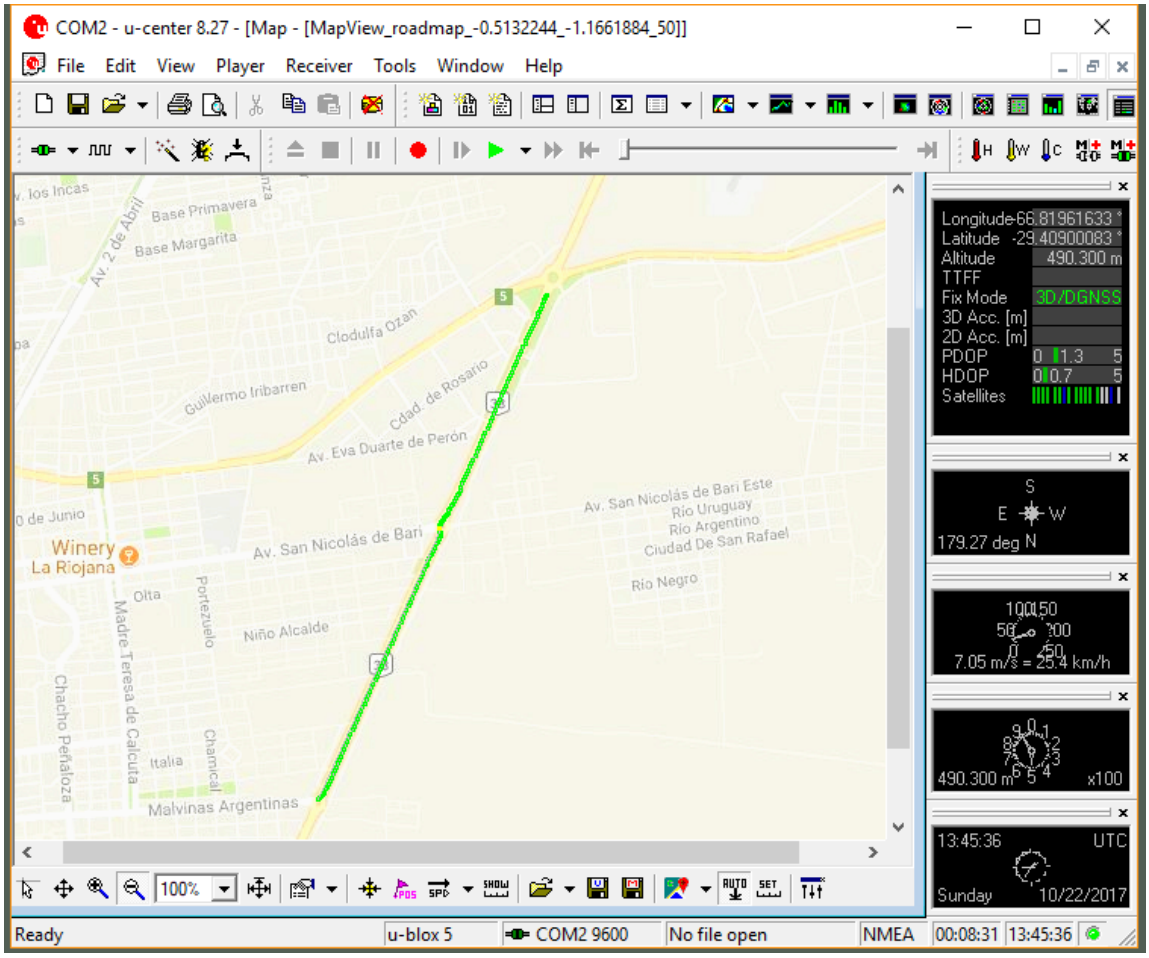

Fig. 66: Captura de pantalla del recorrido realizado visualizado en el programa U-center.

[36] Ley nacional de tránsito de la República Argentina (N° 24.449). Disponible en internet: http://servicios.infoleg.gob.ar/infolegInternet/anexos/0-4999/818/texact.htm

[37] Ley N° 6.168 de la provincia de La Rioja. Disponible en internet: https://www.argentina.gob.ar/sites/default/files/ley_24449_la_rioja.pdf

El segundo recorrido se dividió en varias etapas diferentes, realizando una comparación entre los valores de la velocidad observada en el tablero del vehículo con la entregada por el visualizador de la Placa de Usuario.

Tabla 49: Características del segundo recorrido.

Primera etapa:	
Rango de velocidades [km/h]	20 - 40
Ubicación	Calle 320, Barrio Islas Malvinas
Valores observados	
Velocidad en tablero	Velocidad en visualizador
20	19
31	30
41	40
Segunda etapa:	
Rango de velocidades [km/h]	50 - 60
Ubicación	Ruta nacional 38, zona urbana
Valores observados	
Velocidad en tablero	Velocidad en visualizador
50	50
60	60
Tercera etapa:	
Rango de velocidades [km/h]	70 - 80
Ubicación	Ruta nacional 38, zona rural
Valores observados	
Velocidad en tablero	Velocidad en visualizador
70	70
80	80

Fig. 67: Tablero del vehículo indicando una velocidad de 20 km/h.

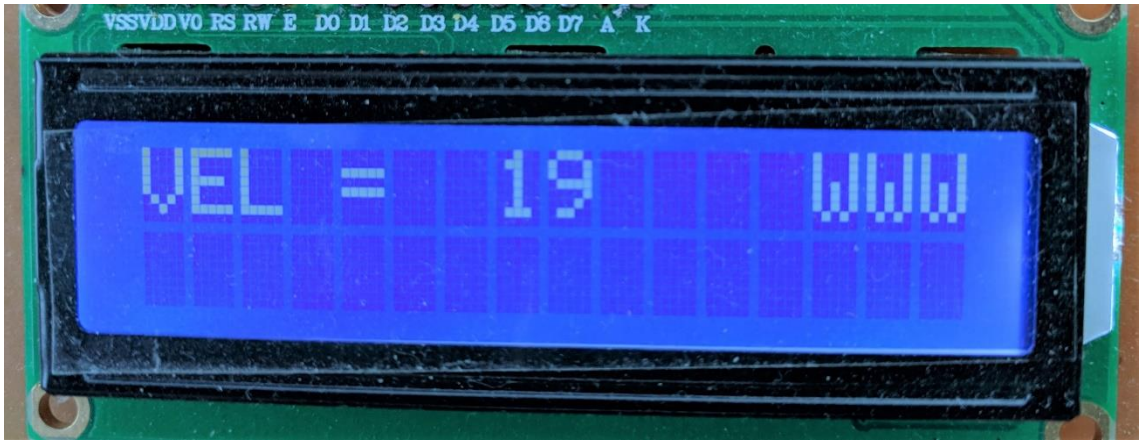

Fig. 68: Visualizador del sistema indicando una velocidad de 19 km/h.

Fig. 69: Tablero del vehículo indicando una velocidad de 50 km/h.

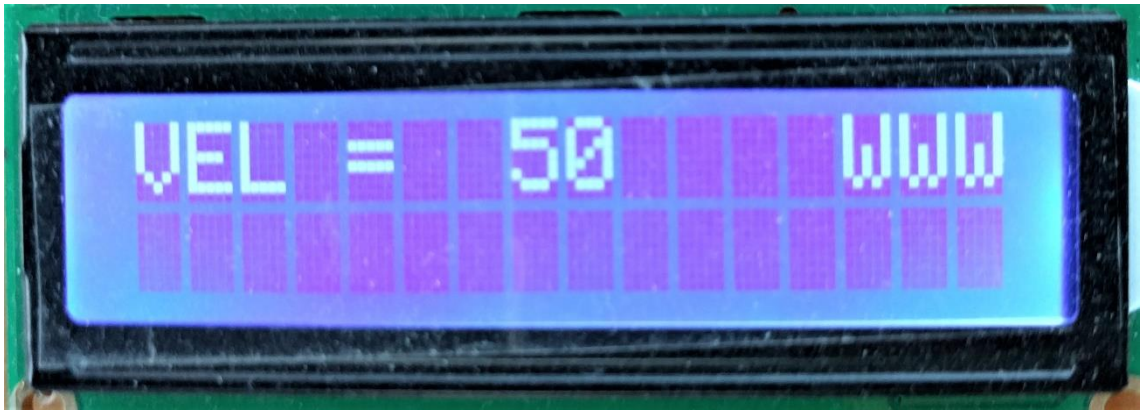

Fig. 70: Visualizador del sistema indicando una velocidad de 50 km/h.

Fig. 71: Tablero del vehículo indicando una velocidad de 80 km/h.

Fig. 72: Visualizador del sistema indicando una velocidad de 80 km/h.

De esta manera, considerando las posibles variaciones del valor medido en el tablero del automóvil debidas a cambios en la calibración de los neumáticos durante el recorrido, se concluye que el modulo GPS utilizado tiene la capacidad de determinar correctamente los parámetros de ubicación y velocidad con la precisión requerida por las funciones implementadas.

6.11 Ensayo final

El ensayo final consistió en dejar al sistema encendido un periodo de tiempo de 48 horas, durante el cual, se le aplicaron estímulos a todas las entradas, simulando para cada caso parámetros de ubicación y velocidad, observando en ellos un correcto desempeño de todas las funciones implementadas.

De esta manera, el ensayo final demostró que el funcionamiento del sistema es independiente de las horas de uso y dio por terminado al conjunto de pruebas realizadas.

7 Costos del proyecto

7.1 Introducción

Los costos del proyecto involucran al valor monetario de todos los elementos utilizados en sus distintas etapas.

A lo largo del capítulo se tendrán en cuenta las siguientes aclaraciones:

- Los valores de los precios serán expresados en dólares estadounidenses (USD[38]) con IVA incluido (consumidor final).

- En caso de que el valor de la potencia disipable de un resistor no este aclarado, el mismo deberá tomarse como un ¼ W (0.25 Vatios).

- Las tiras de pines utilizadas tienen una distancia entre cada terminal de 2.54 mm.

- Los circuitos integrados utilizados en el proyecto poseen encapsulado del tipo DIL (también conocido como DIP).

7.2 Costos de desarrollo

Los costos de desarrollo están definidos por el precio de las distintas herramientas y el de los insumos utilizados.

7.2.1 Herramientas de Software

Cantidad	Medida	Articulo	Valor
1	Unidad	CCS Compiler - PCDIDE IDE Compiler	350,00
1	Unidad	Proteus PCB Design Starter Kit	248,00
1	Unidad	Microsoft Office 365 - Personal	67,75
		Total	665,75

7.2.2 Herramientas de Hardware

Cantidad	Medida	Articulo	Valor
3	Unidad	*Protoboard*	13,72
1	Paquete 40 unid.	Cable para *protoboard* de 20 cm, terminación macho-hembra	2,85
1	Paquete 40 unid.	Cable para *protoboard* de 20 cm, terminación macho-macho	2,85
1	Paquete 40 unid.	Cable para *protoboard* de 20 cm, terminación hembra-hembra	2,85
2	Unidad	Conversor USB a UART basado en Cp2102	19,44
1	Unidad	Modulo Bluetooth HC-05 con adaptador	7,72

[38] ISO 4217. Disponible en internet: https://www.iso.org/iso-4217-currency-codes.html

1	Unidad	Extensor de cable USB	1,71
1	Unidad	Fuente de alimentación de 12 V 2 A	12,01
2	Metros	Cable bipolar tipo taller de 2.5 mm	2,85
		Total	66,00

7.2.3 Insumos

Cantidad	Medida	Articulo	Valor
2	Litros	Cloruro Férrico concentrado	14,86
1	Unidad	Tubo de estaño 60/40 de 0.7 mm	2,85
2	Metros	Cable rojo de 0.25 mm	0,57
2	Metros	Cable negro de 0.25 mm	0,57
2	Metros	Cable verde de 0.25 mm	0,57
		Total	19,42

7.3 Costos del sistema

7.3.1 Placa Principal

Cantidad	Medida	Articulo	Valor
1	Unidad	Transistor BC548	0,17
1	Unidad	Botón	0,14
2	Unidad	Bornera de 3 pines con tornillos, distancia 5mm	0,68
1	Unidad	Modulo transceptor CAN basado en Sn65hvd230	5,71
1	Unidad	Conector KF2510 macho (*header*) de 6 pines en ángulo	0,04
1	Unidad	Conector KF2510 macho (*header*) de 8 pines en ángulo	0,05
1	Unidad	Microcontrolador dSPIC30F4013	16,58
1	Unidad	Arduino Due	31,45
1	Unidad	Sócalo para DSPIC30F4013	0,28
1	Unidad	Módulo wi-fi ESP8266	5,71
1	Unidad	Modulo GPS basado en U-blox Neo-6m	25,73
1	Unidad	Cable conector U.FL a SMA hembra	6,86
1	Unidad	Antena GPS activa de 28dB, IP67, con conector SMA.	19,10
1	Unidad	Modulo adaptador de tarjetas micro SD a SPI	2,85
1	Unidad	Tarjeta micro SD de 8 GB	6,86
1	Unidad	Módulo GSM basado en SIM800L con antena PCB	13,15
1	Unidad	Tarjeta SIM	1,43
1	Unidad	Tira de pines macho	0,57
5	Unidad	Resistor de 1 kΩ	0,18
7	Unidad	Capacitor cerámicos de 0.1 µF	0,64
2	Unidad	Capacitor cerámicos de 10 nF	0,18
1	Unidad	Placa virgen Pertinax de 20 cm x20 cm	5,43
		Total	143,79

7.3.2 Placa de Alimentación

Cantidad	Medida	Articulo	Valor
3	Unidad	Modulo fuente de alimentación basada en Lm2569	7,72
1	Unidad	Bornera de 2 pines con tornillos, distancia 5 mm	0,28
1	Unidad	Conector KF2510 macho (*header*) de 8 pines en ángulo	0,04
1	Unidad	Jumper	0,02
2	Unidad	Clip porta fusible, perfil H de 5mm	0,22
1	Unidad	Fusible de 0.5 A, 20 mm	0,11
1	Unidad	Placa virgen Pertinax de 10 cm x10 cm	2,00
		Total	10,39

7.3.3 Placa de Usuario

Cantidad	Medida	Articulo	Valor
2	Unidad	Bornera de 2 pines con tornillos, distancia 5mm	0,57
2	Unidad	Botón	0,28
1	Unidad	Jumper	0,02
1	Unidad	Modulo Pantalla LCD16x2 basado en HD44780	4,00
1	Unidad	Conector KF2510 macho (*header*) de 6 pines en ángulo	0,04
1	Unidad	Cristal 20 MHz	0,85
2	Unidad	LED Rojo Estándar 5 mm	0,22
1	Unidad	Opto acoplador 4n35	0,57
1	Unidad	Sócalo para 4n35	0,17
1	Unidad	Regulador LM7805	0,74
1	Unidad	Disipador de calor adhesivo de cobre	1,25
1	Unidad	Microcontrolador PIC18F2550	8,00
1	Unidad	Sócalo para 18F2550	0,40
1	Unidad	Resistor de 330 Ω	0,03
3	Unidad	Resistor de 10 kΩ	0,11
1	Unidad	Resistor de 1 kΩ	0,03
1	Unidad	Resistor de 330 kΩ	0,03
1	Unidad	Resistor de 330 Ω, 1W	0,10
3	Unidad	Capacitor cerámicos de 0.1 µF	0,27
1	Unidad	Capacitor electrolítico de 0.47 µF	0,05
2	Unidad	Capacitor cerámicos de 18 pF	0,11
1	Unidad	Tira de pines hembra	0,57
1	Unidad	Placa virgen Pertinax de 10cm x20cm	2,85
		Total	21,26

7.3.4 Placa generadora de señal sonora

Cantidad	Medida	Articulo	Valor
1	Unidad	Bornera de 3 pines con tornillos, distancia 5mm	0,34
1	Unidad	Jumper	0,03

1	Unidad	Resistor de 10 kΩ	0,03
1	Unidad	Transistor BC558	0,17
1	Unidad	*Buzzer* Activo 12V	1,72
1	Unidad	Diodo 1n4007	0,06
		Total	2,35

7.3.5 Interconexión de placas

Cantidad	Medida	Articulo	Valor
2	Metros	Cable rojo de 0.25mm	0,57
1	Metros	Cable negro de 0.25mm	0,28
1	Metros	Cable verde de 0.25mm	0,28
2	Unidad	Conector KF2510 hembra (*housing*) de 8 pines	0,11
2	Unidad	Conector KF2510 hembra (*housing*) de 6 pines	0,10
28	Unidad	Terminales para KF2550	0,02
		Total	1,36

7.3.6 Estructuras y soportes

Cantidad	Medida	Articulo	Valor
1	Unidad	Placa de acrílico de 20 cm x70 cm	25,73
24	Unidad	Tornillos	0,82
48	Unidad	Tuercas	2,05
6	Unidad	Bases de goma	1,03
		Total	29,63

7.4 Costos de ensayo

7.4.1 Placa de Ensayos

Cantidad	Medida	Articulo	Valor
2	Unidad	Botón	0,28
4	Unidad	Bornera de 2 pines con tornillos, distancia 5mm	1,14
2	Unidad	Bornera de 3 pines con tornillos, distancia 5mm	0,68
1	Unidad	Modulo transceptor CAN basado en Sn65hvd230	5,71
1	Unidad	Microcontrolador dSPIC30F4013	16,58
1	Unidad	Sócalo para DSPIC30F4013	0,28
1	Unidad	Modulo fuente de alimentación basada en Lm2569	2,57
2	Unidad	Jumper	0,05
2	Unidad	Relé de 12V	2,28
1	Unidad	Cristal 20 MHz	0,85
3	Unidad	LED Rojo Estándar 5 mm	0,34
6	Unidad	Diodo 1n4007	0,34
1	Unidad	Diodo 1n5408	0,17
2	Unidad	Transistor BC548	0,34
1	Unidad	Inversor Cd40106	0,80
1	Unidad	Sócalo para Cd40106	0,22
3	Unidad	Opto acoplador 4n35	1,71
3	Unidad	Sócalo para 4n35	0,51

3	Unidad	Resistor de 100 Ω	0,11
3	Unidad	Resistor de 330 Ω	0,11
7	Unidad	Resistor de 10 kΩ	0,26
2	Unidad	Capacitor cerámicos de 18 pF	0,11
2	Unidad	Capacitor cerámicos de 10 nF	0,18
2	Unidad	Capacitor electrolítico de 1 µF	0,11
1	Unidad	Capacitor cerámicos de 0.1 µF	0,09
1	Unidad	Tira de pines hembra	0,57
1	Unidad	Placa virgen Epoxi de 10cm x20cm	6,29
		Total	42,68

7.4.2 Conexión con el sistema

Cantidad	Medida	Articulo	Valor
1	Metros	Cable negro de 0.25mm	0,28
		Total	0,28

7.4.3 Insumos

Cantidad	Medida	Articulo	Valor
5	Litros	Combustible para automóvil, nafta súper	5,69
300	ARS	Crédito celular	17,15
		Total	22,84

7.5 Costos de desarrollo de ingeniería

Si bien el trabajo realizado corre por cuenta del alumno responsable del proyecto, se agregaron los costos de desarrollo de ingeniería para contar con una referencia.

Cantidad	Medida	Articulo	Valor
360	Horas	Desarrollo de ingeniería	17.280,00
		Total	17.280,00

7.6 Costos totales

Etapa	Valor
Costos de desarrollo	751,17
Costos del sistema	208,78
Costos de ensayo	65,80
Costos de desarrollo de ingeniería	17.280,00
Total	18305,75

8 Conclusión

8.1 Conclusiones del proyecto

Durante el desarrollo del proyecto se pudieron afianzar los conocimientos adquiridos en la carrera de Ingeniería Electrónica, llevando a la práctica los conceptos teóricos aprendidos y adquiriendo una valiosa experiencia de trabajo. Además, se logró profundizar en nuevas áreas de conocimiento, permitiendo la incorporación de herramientas utilizadas para abordar los temas planteados.

La construcción satisfactoria del prototipo planteando determina que una vigilancia continua de la velocidad del automóvil obliga al conductor a mantenerse dentro de los márgenes establecidos por el dueño o administrador del vehículo y por lo tanto posibilita contrarrestar una de las principales causas de accidentes viales.

Además, también se obtuvieron resultados satisfactorios a la hora de implementar las distintas interfaces de comunicación utilizando módulos electrónicos económicos, disponibles en el mercado local. Lo que facilita notablemente la interconexión de dispositivos adicionales, brindando nuevas funcionalidades capaces de contribuir a la eliminación del resto de las causas de los accidentes de tránsito.

8.2 Futuros desarrollos

Como en todo prototipo, antes de construir un dispositivo final, se pueden realizar múltiples mejoras en el hardware del sistema e implementar numerosas funciones adicionales.

Dentro de las mejoras del hardware, la principal es la reducción del tamaño de las distintas placas del sistema a partir del montaje de los módulos y microcontroladores del Arduino Due directamente en un PCB multicapa, al igual que la utilización de encapsulados más pequeños para el resto de los microcontroladores y componentes SMD. Otra mejora importante consistirá en el diseño de gabinetes con soportes adecuados para tolerar las vibraciones generadas en el interior de un automóvil.

Una de las principales mejoras en cuanto a las funcionalidades del sistema consiste en la utilización de un servidor en la nube, para generar una comunicación continua a través de la red de datos móviles, implementación que no requiere de modificaciones en el hardware actual y que posee una programación basada en el contenido del Anexo B.

Una mejora genérica que puede realizarse consiste en el desarrollo de una aplicación para computadoras de escritorio que integre la capacidad de formatear la tarjeta de memoria utilizada y almacenar en ella los archivos necesarios, junto a la posibilidad de gestionar mediante la conexión wi-fi varios dispositivos al mismo tiempo.

Anexo A: Archivos auxiliares y de programación

En el presente anexo se incluye a la programación utilizada para cada uno de los microcontroladores del sistema, a la programación de los archivos escritos en Python, a las líneas de descripción de las distintas páginas web y a la imagen original utilizada para crear el favicon.

A.1 Programación de microcontroladores

Las líneas de código utilizadas para la programación de los microcontroladores se clasificaran en cinco tipos distintos:

- Directivas de preprocesador.

- Funciones.

- Interrupciones.

- Configuraciones iniciales.

- Bucle principal.

Según el microcontrolador, la estructura del programa varía ligeramente:

Arduino

#Directivas de preprocesador
Definición de variables globales;
Setup()
{
  Configuraciones iniciales;
}
 	
Loop()
{
  Bucle principal;
}
 	
Funciones()

PICs
#Directivas de preprocesador Definición de variables globales; #Interrupciones Funciones()

```
Main()
{
    Configuraciones iniciales;

    While(true)
    {
        Bucle principal;
    }
}
```

A.1.1 Arduino Due

A.1.1.1 Directivas de preprocesador

Dentro de las directivas, cabe destacar a la utilización de la librería *due_can*[39], debido a que la misma no forma parte de las librerías estándar de Arduino.

```
#include "due_can.h"
#include <SPI.h>
#include <SD.h>

//UART
#define PC   Serial
#define GPS  Serial1
#define SIM  Serial2
#define WIFI Serial3

//Interfaz de usuario ubicada en Placa Principal
#define REQST 25

//Interfaz de usuario ubicada en Placa de Usuario
#define PANIC 27
#define LEN 29

//Pines conectados al dsPIC
#define PIN_CAL 49
#define PIN_VEL 47
#define PIN_CLK 45
#define LWW     43
#define ARD_RST  53 //Reset del dsPIC

//Mensajes CAN
#define IDmodo     0
#define IDmodoRQST  1
```

[39] Collin Kidder. Disponible en internet: https://github.com/collin80/due_can

```
#define IDservidor  2
#define IDvelocidad 4
#define IDpantalla  5
#define IDalmacena  6
#define IDenvia     7
#define IDfinal     8
#define IDAux       9
#define IDinforma   10
#define DATAplaca   0
```

A.1.1.2 Funciones

Funciones utilizadas para los archivos de configuración

```
String Leer_Archivo (String Archivo){

  String Archivo0;
  Archivo = "config/" + Archivo;

  File myFile = SD.open(Archivo);

   if(myFile)
   {
    while (myFile.available())
    {
     char aux=myFile.read();
     Archivo0 = Archivo0 + String( aux );
    }
    myFile.close();
   }
   else
   {
    Archivo0="";

    while(true){
     PC.println("\n\r-ERR: Leer_Archivo "+ Archivo);
     delay(500);
     digitalWrite(LEN, LOW);
     PC.println("\n\rReiniciar Sistema.");
     delay(500);
     digitalWrite(LEN, HIGH);
    }
   }
  return (Archivo0);
}

void Escribir_Archivo (String Archivo, String Datos){
```

```
  Archivo = "config/" + Archivo;
  File myFile = SD.open(Archivo,FILE_WRITE);

  if(myFile)
  {
   myFile.print(Datos);
   myFile.close();
  }
  else PC.println("\n\r-ERR: Escribir_Archivo "+ Archivo);
}

void Eliminar_Archivo(String Archivo){

  Archivo = "config/" + Archivo;

  SD.remove(Archivo);
  File myFile = SD.open(Archivo,FILE_WRITE);

  if(myFile)
  {
   myFile.close();
  }
  else PC.println("\n\r-ERR: Eliminar_Archivo "+ Archivo);
}
```

Funciones utilizadas para generar los archivos de ubicación

```
void Eliminar_YZ( void ){
 SD.remove("ubic/z.txt");
 File myFile = SD.open("ubic/z.txt",FILE_WRITE);

 if(myFile)
 {
  myFile.println("<?xml version=\"1.0\" encoding=\"UTF-8\" standalone=\"no\"
?>\n\r<gpx>"); //
  myFile.print("</gpx>");
  myFile.close();
 }
 else PC.println("\n\r-ERR: Eliminar_YZ Z");

 SD.remove("ubic/y.txt");
 myFile = SD.open("ubic/y.txt",FILE_WRITE);

 if(myFile)
 {
```

```
   myFile.close();
  }
  else PC.println("\n\r-ERR: Eliminar_YZ Y");
}

void Escribir_YZ(String Name, String Lat , String Lon ){
  File myFile = SD.open("ubic/z.txt",FILE_WRITE);
  const String FINAL =  "</gpx>";

  String Fecha = Consultar_Tiempo();

  String Data = "<wpt lat=\"" + Lat + "\" lon=\"" + Lon + "\"><time>" + Fecha +
"</time><name>" + Name + "</name></wpt>";

  if(myFile)
  {
   myFile.seek(myFile.position()- FINAL.length());
   myFile.println(Data);
   myFile.print(FINAL);
   myFile.close();
  }
  else PC.println("\n\r-ERR: Escribir_YZ Z");

  myFile = SD.open("ubic/y.txt",FILE_WRITE);

  Data = "['" + Name + "'," + Lat + "," + Lon + "]";

  if(myFile)
  {
   if(myFile.position()!= 0) myFile.println(",");
   myFile.print(Data);
   myFile.close();
  }
  else PC.println("\n\r-ERR: Escribir_YZ Y");
}
```

Funciones utilizadas para la configuración del módulo wi-fi

```
void Config_WIFI_HOST( void ){

  PC.println("Configuracion HOST: Iniciada");
  ApagarServidorWIFI();
  BorrarEnviarEsperarWIFI("AT+CWQAP",'O');  //Desconecta de redes
  BorrarEnviarEsperarWIFI("AT+CIPMUX=1",'K'); //Permite múltiple conexión
  BorrarEnviarEsperarWIFI("AT+CIPSTA=\"" + IX + "\"",'K');  //Selecciona IP
  BorrarEnviarEsperarWIFI("AT+CWJAP=\"" + SX + "\",\"" + PX + "\"",'K');   //Conectar a Red
```

```
  BorrarEnviarEsperarWIFI("AT",'O');  //Sincronismo
  EncenderServidorWIFI();
  PC.println("Configuracion HOST: Terminada\n\r");
}

void Config_WIFI_AP( void ){

  PC.println("Configuracion AP: Iniciada");
  BorrarEnviarEsperarWIFI("ATE0",'K'); //No ECHO
  BorrarEnviarEsperarWIFI("AT",'O');  //Sincronismo
  ApagarServidorWIFI();
  BorrarEnviarEsperarWIFI("AT+CWMODE=3",'K'); //Enciende modo AP + host
  BorrarEnviarEsperarWIFI("AT+CWSAP=\"" + SP + "\",\"" + PP + "\",5,3",'K'); //Crear Red
  EncenderServidorWIFI();
  PC.println("Configuracion AP: Terminada\n\r");
}
```

Funciones utilizadas para enviar las páginas web solicitadas

```
void Enviar_Pagina(char host, String Archivo){

String Host = String(host);

if(Archivo != "s") {

  if(Archivo == "favicon"){
    Archivo=String("f");
  }
  else{
    if(Archivo != "a" && Archivo != "w" && Archivo != "k" && Archivo != "c" && Archivo != "g"
&& Archivo != "m" && Archivo != "z" && Archivo != "i" && Archivo != "v" && Archivo != "t"
&& Archivo != "p" && Archivo != "q" && Archivo != "r")
    {
      Archivo=String("e");
    }
  }

  PC.print("\n\rEnviar_Pagina: Archivo " + Archivo);
  PC.println(", Host " + Host);

  if(Archivo == "z"){
    Archivo = String("ubic/") + Archivo + String(".txt");}
  else {
    Archivo = String("web/") + Archivo + String(".txt");
  }
```

```
File auxFile;
if(Archivo == "web/m.txt"){
 Archivo = "ubic/y.txt";
 auxFile = SD.open("web/m.txt");

 if (auxFile){
  BorrarEnviarEsperarWIFI("AT+CIPSEND="+Host+",354", '>');

  VaciarBufferWIFI();

  int j = 0;
  while (auxFile.available() && j<354){
   WIFI.write(auxFile.read());
   j++;
  }

  EsperaPorWIFIC('K');

 } else PC.println("\n\r-ERR: Enviar_Pagina m.html");
}

File myFile = SD.open(Archivo);

if (myFile) {
 int siz = myFile.size();

 while(0<siz)
 {
  VaciarBufferWIFI();

  WIFI.print("AT+CIPSEND="+Host);
  WIFI.print(",");

  if(siz < 2000)
  {WIFI.println(siz);}
  else
  {WIFI.println("2000");}

  EsperaPorWIFIC('>');


  VaciarBufferWIFI();
  int i = 0;
  while (myFile.available() && i<2000){
   WIFI.write(myFile.read());
```

```
      i++;
      siz--;
     }

     EsperaPorWIFIC('K');
   }

   myFile.close();

  } else PC.println("\n\r-ERR: Enviar_Pagina "+Archivo);


  if(Archivo == "ubic/y.txt")
  {
   if (auxFile)
   {

     BorrarEnviarEsperarWIFI("AT+CIPSEND="+Host+",996", '>');

     auxFile.seek(484);
     while (auxFile.available()){
      WIFI.write(auxFile.read());
     }

     EsperaPorWIFIC('K');

     auxFile.close();
   } else PC.println("\n\r-ERR: Enviar_Pagina y.html");
  }

  BorrarEnviarEsperarWIFI("AT+CIPCLOSE="+Host,'K');
}
else Enviar_Pagina_Estado(Host);
}

void Enviar_Pagina_Estado( String Host ){

  PC.println("\n\rEnviar_Pagina_Estado: Host " + Host);

  File myFile = SD.open("web/s.txt");

  if (myFile) {

   String AuxGPS;
   String AuxSIM;
```

```
  String AuxFecha;

  if(digitalRead(PIN_CAL)==HIGH)
  {
   AuxGPS="Satelites encontrados";
  }
  else
  {
   AuxGPS="Satelites no encontrados";
  }

  String fecha = Consultar_Tiempo();
  AuxFecha = fecha.substring(11,19)+" "+ fecha.substring(8,10)+ "/" + fecha.substring(5,7) +
"/" + fecha.substring(0,4);


  if(BorrarEnviarEsperarSIM("AT", 'K')) { AuxSIM="Conectado"; }
  else { AuxSIM="No Conectado"; }

  int siz = myFile.size() + MV.length() + MD.length() + NX.length() + SX.length() + IX.length()
+SP.length() -6 + AuxGPS.length() + AuxSIM.length() + AuxFecha.length()-3;

  VaciarBufferWIFI();

  BorrarEnviarEsperarWIFI("AT+CIPSEND="+Host+","+siz, '>');

  char auxFch = 'a';

  //Enviar valor de la Velocidad Máxima
  while (myFile.available()){
     auxFch=myFile.read();
     if(auxFch == '?') break;
     WIFI.write(auxFch);
   }
  WIFI.print(MV);

  //Enviar valor del Modo de funcionamiento
  while (myFile.available()){
     auxFch=myFile.read();
     if(auxFch == '?') break;
     WIFI.write(auxFch);
   }
  WIFI.print(MD);

  //Enviar Numero de Teléfono del Administrador
```

```
  while (myFile.available()){
    auxFch=myFile.read();
    if(auxFch == '?') break;
    WIFI.write(auxFch);
   }
  WIFI.print(NX);

  //Enviar Nombre de Red WIFI
  while (myFile.available()){
    auxFch=myFile.read();
    if(auxFch == '?') break;
    WIFI.write(auxFch);
   }
  WIFI.print(SX);

  //Enviar IP de Red WIFI
  while (myFile.available()){
    auxFch=myFile.read();
    if(auxFch == '?') break;
    WIFI.write(auxFch);
   }
  WIFI.print(IX);

  //Enviar Nombre de Punto de Acceso WIFI
  while (myFile.available()){
    auxFch=myFile.read();
    if(auxFch == '?') break;
    WIFI.write(auxFch);
   }
  WIFI.print(SP);

  //GPS satélites conectados
  while (myFile.available()){
    auxFch=myFile.read();
    if(auxFch == '?') break;
    WIFI.write(auxFch);
   }
  WIFI.print(AuxGPS);

  //SIM conectado
  while (myFile.available()){
    auxFch=myFile.read();
    if(auxFch == '?') break;
    WIFI.write(auxFch);
   }
```

```
    WIFI.print(AuxSIM);


    //Mostrar Fecha del Sistema
    while (myFile.available()){
       auxFch=myFile.read();
       if(auxFch == '?') break;
       WIFI.write(auxFch);
     }
    WIFI.print(AuxFecha);


    while (myFile.available()){
       WIFI.write(myFile.read());
     }

    EsperaPorWIFIC('K');

    myFile.close();

  } else PC.println("\n\r-ERR: Enviar_Pagina_Estado");

  BorrarEnviarEsperarWIFI("AT+CIPCLOSE="+Host,'K');

}
```

Funciones del módulo GSM

```
void Config_SIM(void){
  PC.println("Configuracion SIM: Iniciada");
  BorrarEnviarEsperarSIM("AT", 'O'); //Sincronismo
  BorrarEnviarEsperarSIM("AT+CMEE=0", 'K'); //Errores solo muestra ERROR
  BorrarEnviarEsperarSIM("ATE0", 'K'); //No echo
  BorrarEnviarEsperarSIM("AT+GSMBUSY=1", 'K'); //No recibir llamadas
  BorrarEnviarEsperarSIM("AT+CMGF=1", 'K'); //Modo texto
  BorrarEnviarEsperarSIM("AT+CMGDA=\"DEL ALL\"", 'O'); //Eliminar todos los mensajes
  BorrarEnviarEsperarSIM("AT+CNMI=,1", 'K'); //Notificaciones Activas
  BorrarEnviarEsperarSIM("AT+IPR=115200", 'K'); //Baudrate
  Definir_Tiempo("00/01/01");
  PC.println("Configuracion SIM: Terminada\n\r");
}

void Enviar_Mensaje(String Mensaje){
   PC.println("Enviar_Mensaje: " + Mensaje);
   BorrarEnviarEsperarSIM("AT+CMGS=\"" + NX + "\"", '>');
   SIM.print(Mensaje);
```

```
  SIM.print((char)26);
  delay(100);
  SIM.println("AT");
}

void Definir_Tiempo(String Fecha){
  String Hora = Consultar_GPS("Hor");
  if(Hora[1]==':') Hora = "0" + Hora;
  if( Hora == "0" ) {
   Hora = "00:00:00";
   PC.println("\n\r-ERR: Definir_Tiempo Consulta Hora");
  }

  String ATC = "AT+CCLK=\"" + Fecha + "," +Hora+ "-03\"";
  BorrarEnviarEsperarSIM(ATC, 'K');
}

String Consultar_Tiempo( void ){

  String Fecha;
  int t=0;
  String Anno = "";
  String Mess = "";
  String Diaa = "";
  String Hora = "";

  while(SIM.available()>0) SIM.read(); //Vaciar Buffer GPS

  SIM.print("AT+CCLK?\r"); //Consultar

  while(t<1000) //Esperar
  {
   if(SIM.available()>0)
   {
    if(SIM.find("LK: \"")>0)
    {
     Anno = SIM.readStringUntil('/');
     Mess = SIM.readStringUntil('/');
     Diaa = SIM.readStringUntil(',');
     Hora = SIM.readStringUntil('-');
     if(Hora[5]==':'){ t=1002; }
    }
   }
   t++;
   delay(2);
```

```
  }

  if(Hora.length()==8)
  {
    Fecha = "20"+Anno+"-"+Mess+"-"+Diaa+"T"+Hora+"Z";
  }
  else
  {
    PC.println("\n\r-ERR: Consultar_Tiempo");
    Fecha="2000-01-01T00:00:01Z";
  }
 return (Fecha);
}
```

Funciones para la comunicación con el dsPIC

```
String Consultar_GPS(String DATA){
 int t=0;
 String auxF = "";
 char auxC = '0';

 while(GPS.available()>0) GPS.read(); //Vaciar Buffer GPS

 GPS.print(DATA+"\r"); //Consultar

 while(t<3000) //Esperar
 {
  if(GPS.available()>0)
  {
   auxC = GPS.read();
   if(auxC!=13){ auxF = auxF + auxC;}
   else t=3002;
  }
  t++;
  delay(2);
 }

 if(auxC!=13)
 {
  PC.println("\n\r-ERR: Consultar_GPS "+DATA);
  auxF="0";
 }

 return (auxF);
}
```

```
void Definir_MV(String xS){
  int U = 0;
  for(int x=0; x<xS.length(); x++)
  {
    U = U*10+xS[x]-48;
  }

  if(U<0) U=0;
  else if(U>255) U=255;

  while(GPS.available()>0) GPS.read(); //Vaciar Buffer GPS

  GPS.print("MV");
  GPS.write(U);
  GPS.write(13);

  char auxF=0;
  int t = 0;
  while(auxF!='K' && t<400) //Esperar
  {
    if(GPS.available()>0)
    {
      auxF = GPS.read();
    }
    t++;
    delay(10);
  }

  MostrarPantalla("Vel max: "+xS+"Km/h");

  if( auxF!='K' ){ PC.println("\n\r-ERR: Definir_MV "+xS);}
}

void MostrarPantalla(String DATA){
  if(DATA.length()>16){
    DATA=DATA.substring(0,16);
  }

  //Vaciar buffer
  while(GPS.available()>0)
  {
    GPS.read();
  }

  //Avisar que los caracteres serán enviadon
```

```
  GPS.print("ST\r");

  //Esperar por una respuesta durante 2 segundos
  char auxC = 0;
  int t=0;

  while(auxC!='>' && t<200)
  {
    if(GPS.available()>0)
    {
      auxC = GPS.read();
    }
    t++;
    delay(10);
  }

  //Si no hubo respuesta, mostrar mensaje de error
  if(t==1000)
  {
    PC.println("\n\r-ERR: MostrarPantalla "+DATA);
  }
  else
  {
    GPS.print(DATA+"\r");
  }

}
```

Funciones de la comunicación CAN

```
void Enviar_CAN ( int CanID , int CanData ){

  PC.print("Sal_CAN: ");

  if(CanID==IDmodo)      PC.print("IDmodo ");
  else if(CanID==IDpantalla)    PC.print("IDpantalla ");
  else if(CanID==IDalmacena)  PC.print("IDalmacena ");
  else if(CanID==IDenvia)       PC.print("IDenvia ");
  else if(CanID==IDfinal)       PC.print("IDfinal ");
  else if(CanID==IDAux)         PC.print("IDAux ");
  else if(CanID==IDinforma)    PC.print("IDinforma ");

  if(CanData=='1' || CanData == '2' || CanData =='3'){
    PC.print("Modo");
    PC.write(CanData);
  }
```

```
  else{
    PC.print(String(CanData));
  }

  PC.print("\n\r");

  mensaje.id=CanID;
  mensaje.data.bytes[0]=CanData;
  Can0.sendFrame(mensaje);
}
```

Funciones auxiliares

```
boolean Buscar(String Frase, String Fragmento) {
  boolean resultado = false;
  int x = Frase.length() - Fragmento.length();
  for (int i = 0; i <= x; i++) {
    if (Frase.substring(i,Fragmento.length()+i) == Fragmento) {
      resultado = true;
      break;
    }
  }
  return resultado;
}

void VaciarBufferWIFI(void ){
  while(WIFI.available()>0)
  {
    char auxC = WIFI.read();
  }
}

boolean EsperaPorWIFIC(char dX){
  char auxF='0';
  int t = 0;
  while(auxF!=dX && t<1000)
  {
    if(WIFI.available()>0)
    {
      auxF = WIFI.read();
    }
    t++;
    delay(10);
  }

  VaciarBufferWIFI();
```

```
  boolean res = false;
  if(auxF != dX) res = true;
  return res;
}

void BorrarEnviarEsperarWIFI(String xS, char xC){
  VaciarBufferWIFI();
  WIFI.println(xS);
  if(EsperaPorWIFIC(xC)) { PC.print("\n\r-ERR: BEEW " + xS +" "); PC.write(xC); PC.println(""); }
}

boolean EsperaPorSIMC(char dX) {
  char auxF=127;
  int t = 0;
  while(auxF!=dX && t<1000)
  {
    if(SIM.available()>0)
    {
      auxF = SIM.read();
    }
    t++;
    delay(10);
  }

  while(SIM.available()>0) SIM.read();

  boolean res = false;
  if(auxF != dX) res = true;
  return res;
}

boolean BorrarEnviarEsperarSIM(String xS, char xC){
  boolean aux = true;

  while(SIM.available()>0) SIM.read();
  SIM.println(xS);
  if(EsperaPorSIMC(xC))
  {
    PC.print("\n\r-ERR: BEES " + xS +" "); PC.write(xC); PC.println("");
    aux = false;
  }
  return aux;
}
```

```
void ApagarServidorWIFI( void ){
  if(ORDENdeAPAGADO==false){
    BorrarEnviarEsperarWIFI("AT+CIPSERVER=0",'O'); digitalWrite(LWW, LOW);
  }
}

void EncenderServidorWIFI( void ){
  if(ORDENdeAPAGADO==false){
    BorrarEnviarEsperarWIFI("AT+CIPSERVER=1,80",'O'); digitalWrite(LWW, HIGH);
  }
}
```

A.1.1.3 Configuraciones iniciales

Definición de las variables globales

```
//Archivos de configuración en RAM
String CX; //Contraseña de Sistema
String SX; //Nombre de Red WIFI
String PX; //PASS de Red WIFI
String IX; //IP de Red WIFI
String NX; //Numero celular administrador
String SP; //Nombre de Red AP
String PP; //PASS de Red AP
String MD; //Modo de Funcionamiento
String MV; //Máxima velocidad

//Mensaje del botón PANIC solo una vez por cada pulsación
boolean enbPANIC=true;

//Temporización de 5 segundos
boolean MAINCLK = false;
boolean enbCLK = false;

//Mensaje de Velocidad Máxima solo durante cambio a estado alto
boolean enbPIN_VEL=true;

//Utilizadas por CAN
CAN_FRAME mensaje;
boolean ORDENdeAPAGADO = false; //Encender o apagar el servidor web
boolean enbMD = false; //Enviar Modo de Funcionamiento al Bus
boolean enbCAM_VEL=true; //Determina el valor de la velocidad máxima
```

Configuraciones de Arduino

```
  pinMode(LEN, OUTPUT);
  pinMode(LWW, OUTPUT);
```

```
  pinMode(REQST, INPUT);
  pinMode(PANIC, INPUT);

  pinMode(PIN_CAL, INPUT);
  pinMode(PIN_VEL, INPUT);
  pinMode(PIN_CLK, INPUT);
  pinMode(ARD_RST, OUTPUT);

  digitalWrite(LEN, HIGH);

  //Configuración UART Principal
  PC.begin(115200);
  while (!PC) {;}
  delay(200);

  digitalWrite(LEN, LOW);
  PC.println("Iniciando Sistema\n\r");
```

Lectura inicial de los archivos de configuración

```
if (!SD.begin(4)) {
   while(true)
   {
     PC.println("\n\r-ERR: SD");
     delay(500);
     digitalWrite(LEN, LOW);
     PC.println("\n\rReiniciar Sistema.");
     delay(500);
     digitalWrite(LEN, HIGH);
   }
  }
  else{
    CX  =Leer_Archivo("CX.log");
    SX  =Leer_Archivo("SX.log");
    PX  =Leer_Archivo("PX.log");
    IX   =Leer_Archivo("IX.log");
    NX =Leer_Archivo("NX.log");
    SP  =Leer_Archivo("SP.log");
    PP  =Leer_Archivo("PP.log");
    MD =Leer_Archivo("MD.log");
    MV =Leer_Archivo("MV.log");
  }
```

Configuración del dsPIC

```
  digitalWrite(ARD_RST, HIGH);
  delay(2000); //Permite a los módulos tener tiempo de encender
```

```
GPS.begin(115200);
while (!GPS) {;}
PC.println("Configuracion GPS: Iniciada");
Consultar_GPS("ATE");
delay(500);
Definir_MV(MV);
PC.println("Configuracion GPS: Terminada\n\r");
delay(5000); //Permite a los módulos tener tiempo de encender
```

Configuraciones del módulo SIM

```
SIM.begin(115200);
while (!SIM) {;}
Config_SIM();
```

Configuraciones del módulo wi-fi

```
WIFI.begin(115200);
while (!WIFI) {;}
Config_WIFI_AP();
Config_WIFI_HOST();
```

Configuraciones de la comunicación CAN

```
PC.println("Configuracion CAN: Iniciada");
Can0.begin(CAN_BPS_125K);
Can0.watchFor();
mensaje.extended=1;
mensaje.priority=0;
mensaje.length=1;
PC.println("Configuracion CAN: Terminada\n\r");
```

Indicadores del final de las configuraciones

```
MostrarPantalla("Sistema Listo");
PC.println("Configuraciones Terminadas\n\r");

//Informa a las demás placas que la principal está lista para recibir mensajes
Enviar_CAN ( IDinforma , DATAplaca);

digitalWrite(LEN, HIGH);
```

A.1.1.4 Bucle principal

Código para atender a los mensajes provenientes del bus CAN

```
if (Can0.available() > 0){
  //Lectura del mensaje y almacenamiento en el objeto CAN
  Can0.read(mensaje);
  int CanID = mensaje.id;
```

```
  int CanData=mensaje.data.bytes[0];

  //Mensaje solicitud de modo
  if(CanID == IDmodoRQST){
   PC.println("Ent_CAN: IDmodoRQST "+ String(CanData));

   char auxMD = MD[0];
   Enviar_CAN ( IDmodo , auxMD);
  }
  //Mensaje para encender o apagar el servidor
  else if(CanID == IDservidor){
   PC.println("Ent_CAN: IDservidor "+ String(CanData));

   if(CanData == '1' && ORDENdeAPAGADO==true)
   { //Encender servidor
     ORDENdeAPAGADO = false;
     Config_WIFI_HOST(); //Inicia una configuración, luego enciende servidor
   }
   else if(CanData == '0' && ORDENdeAPAGADO==false)
   { //Apagar servidor
     BorrarEnviarEsperarWIFI("AT+CIPSERVER=0",'K');
     digitalWrite(LWW, LOW);
     ORDENdeAPAGADO = true;
   }

   Enviar_CAN ( IDfinal , IDservidor); //Informar que la operación finalizo
  }
  //Mensaje para cambiar la velocidad máxima a 10Km/h
  else if(CanID == IDvelocidad){
   PC.println("Ent_CAN: IDvelocidad "+ String(CanData));

   if(CanData == '1' && enbCAM_VEL == true)
   { //Configurar 10Km/h
     Definir_MV("10");
     enbCAM_VEL=false;
   }
   else if(CanData == '0' && enbCAM_VEL == false)
   { //Configurar velocidad normal
     Definir_MV(MV);
     enbCAM_VEL=true;
   }

   Enviar_CAN ( IDfinal , IDvelocidad); // Informar que la operación finalizo
  }
  //Mensajes formados por las cadenas de caracteres
```

```
else if(CanID == IDpantalla || CanID == IDalmacena || CanID == IDenvia){
 PC.println("Ent_CAN: Buzon "+ String(CanData));

 char KanID = CanID;
 char KanData = CanData;

 ApagarServidorWIFI();

 Enviar_CAN ( KanID , KanData); // Informar al bus que podemos recibir la cadena

 String buzon ="";
 while(!(Can0.available() > 0)){;} //Espera por la cadena

 //Lectura de la cadena en dos mensajes de 8bytes de longitud
 int i=0;

 Can0.read(mensaje);
 for(i=0;i<8;i++)
 {
  if(mensaje.data.bytes[i] != 0)
   buzon=buzon+char(mensaje.data.bytes[i]);
  else break;
 }

 while(!(Can0.available() > 0)){;}
 Can0.read(mensaje);
 for(i=0;i<8;i++)
 {
  if(mensaje.data.bytes[i] != 0)
   buzon=buzon+char(mensaje.data.bytes[i]);
  else break;
 }

 PC.println("StringCAN: "+ buzon);

 MostrarPantalla(buzon);

 if(KanID != IDpantalla){
  String Data = buzon + ". ";
  String LAT = "0";
  String LON = "0";

  if(digitalRead(PIN_CAL)==HIGH){
   LAT = Consultar_GPS("Lat");
   LON = Consultar_GPS("Lon");
```

```
      Data = Data + "Ubicacion: https://www.google.com.ar/maps/search/" + LAT + ",+" +
LON;
     }
     else{
      Data = Data + "Ubicacion no disponible.";
     }

     Escribir_YZ(buzon,LAT, LON);

     if( KanID == IDenvia ) Enviar_Mensaje(Data);
    }

    Enviar_CAN ( IDfinal , KanID); //Informar que la operación finalizo

    EncenderServidorWIFI();
   }
  }
  else if(enbMD) { //Si MODO cambio en algún punto del programa, enviarlo
    char auxMD = MD[0];
    Enviar_CAN ( IDmodo , auxMD);
    enbMD=false;
    MostrarPantalla("Cambio: Modo"+MD);
  }
```

Código para actualizar funciones cada cinco segundos

```
if(digitalRead(PIN_CLK)==HIGH && enbCLK==false) {
    enbCLK=true;
    MAINCLK=true;  //Sera puesta en bajo por la función que necesite de la temporización
  }
  else if(enbCLK==true){
    enbCLK=false;
    MAINCLK=true;  //Sera puesta en bajo por la función que necesite de la temporización
  }
```

Código para atender al botón RQST

```
if(digitalRead(REQST)==LOW){
    PC.println("\n\rREQST");

    int aux_time =0;
    while(digitalRead(REQST)==LOW && aux_time<10)
    {
     delay(20);
     aux_time++;
    }
```

```
  MostrarPantalla("REQST");

  PC.println("\n\rArchivos:");
  PC.println("Contraseña: "+ Leer_Archivo("CX.log"));
  PC.println("SSID: "+        Leer_Archivo("SX.log"));
  PC.println("PASS: "+        Leer_Archivo("PX.log"));
  PC.println("IP: "+          Leer_Archivo("IX.log"));
  PC.println("Numero: "+      Leer_Archivo("NX.log"));
  PC.println("SSID AP: "+     Leer_Archivo("SP.log"));
  PC.println("PASS AP: "+     Leer_Archivo("PP.log"));
  PC.println("Modo: "+        Leer_Archivo("MD.log"));
  PC.println("MaxVel: "+      Leer_Archivo("MV.log"));

  PC.println("\n\rGPS:");
  if(digitalRead(PIN_CAL)==HIGH)
  {
   PC.println("Hora: " + Consultar_GPS("Hor"));
   delay(10);
   PC.println("Latitud: " + Consultar_GPS("Lat"));
   delay(10);
   PC.println("Longitud: " + Consultar_GPS("Lon"));
   delay(10);
   PC.println("Velocidad: " + Consultar_GPS("VRP"));
  }
  else PC.println("Pin Calidad en estado bajo");


  PC.println("\n\rPines de Entrada GPS:");
  PC.print("PIN_VEL: ");
  if(digitalRead(PIN_VEL)==HIGH) { PC.println("Alto"); }
  else { PC.println("Bajo"); }

  PC.println("\n\rGSM:");
  PC.println("TiempoZ: " + Consultar_Tiempo());

  PC.println("\n\rUbicaciones: z.gpx");
  File myFile = SD.open("ubic/z.txt");
  if (myFile)
  {
   while (myFile.available())
   {
     PC.write(myFile.read());
   }

   myFile.close();
```

```
      PC.print("\n\r");
    }
    else PC.println("Error al abrir");

    PC.println("\n\rVariables de Control:");
    PC.print("enbPIN_VEL: ");
    if(enbPIN_VEL) { PC.println("Alto"); }
    else { PC.println("Bajo"); }

    PC.print("ORDENdeAPAGADO: ");
    if(ORDENdeAPAGADO) { PC.println("Alto"); }
    else { PC.println("Bajo"); }

    PC.print("enbCAM_VEL: ");
    if(enbCAM_VEL) { PC.println("Alto"); }
    else { PC.println("Bajo"); }

    PC.print("enbMD: ");
    if(enbMD) { PC.println("Alto"); }
    else { PC.println("Bajo"); }
  }
```

Código utilizado para determinar la página solicitada y almacenar la información de los formularios

```
if(WIFI.available() > 0){

   char host = '0'; //Dispositivo desde donde se realizó la petición
   char R;          //Define si se trata de un pedido de la página principal o de una secundaria
   char GorP;       //Permite definir si se trata de una petición GET o POST

   if (WIFI.find("+IPD,"))
   {
    delay(10);
    host = WIFI.read();

    if(WIFI.find(":"))
    {
     delay(10);
     GorP = WIFI.read();

     String Aux;
     if ( WIFI.find(" /") )
     {
      delay(10);
      R = WIFI.peek();
```

```
if (R == ' ')
{
  Aux=String("a");
}
else
{
  Aux = WIFI.readStringUntil('.');
}

String CX_AUX; //Almacena contraseña ingresada
String H;          //Almacena la orden principal contenida en el método POST
String Fecha;    //Almacena la información de la FECHA de configuración

if (GorP == 'P')
{
  if ( WIFI.find("CX=") ){
    CX_AUX = WIFI.readStringUntil('&');
    if (CX_AUX==CX && WIFI.find("H=") )
    {
      Aux=String("q");
      H = WIFI.readStringUntil('&');

          if(H=="w"){ //Pagina de configuración de WIFI HOST
        if ( WIFI.find("SX=") )
        {
          SX = WIFI.readStringUntil('&');

          if ( WIFI.find("PX=") )
          {
            PX = WIFI.readStringUntil('&');

              if ( WIFI.find("IX=") )
              {
                IX = WIFI.readStringUntil('&');
              }
          }
        }
      }
      else if(H=="i"){ //Pagina de configuración del Modo de funcionamiento
        if ( WIFI.find("MD=") )
        {
          MD = WIFI.readStringUntil('&');
        }
      }
```

```
      else if(H=="v"){ //Pagina de configuración de la velocidad máxima
       if ( WIFI.find("MV=") )
       {
        MV = WIFI.readStringUntil('&');
       }
      }
      else if(H=="t"){ //Pagina de configuración de WIFI AP
       if ( WIFI.find("SP=") )
       {
        SP = WIFI.readStringUntil('&');

        if ( WIFI.find("PP=") )
        {
         PP = WIFI.readStringUntil('&');
        }
       }
      }
      else if(H=="p"){ //Pagina utilizada para cambiar la contraseña del administrador
       if ( WIFI.find("QX=") )
       {
        CX = WIFI.readStringUntil('&');
       }
      }
      else if(H=="k"){ //Pagina utilizada para cambiar la FECHA
       if ( WIFI.find("DD=20") )
       {
        String anno= WIFI.readStringUntil('-');
        String mess= WIFI.readStringUntil('-');
        String diaa= WIFI.readStringUntil('&');

        Fecha = anno+ "/" + mess + "/" + diaa;
       }
      }
      else if(H=="c"){//Pagina utilizada para cambiar el numero del administrador
       if ( WIFI.find("NX=") )
        {
         NX = WIFI.readStringUntil('&');
        }
       }
      }
      else
      {
       Aux=String("r");
      }
     }
```

```
    }

    Enviar_Pagina(host,Aux ); //Enviamos la página solicitada al dispositivo host

    //Realizar los cambios que la página solicito
       if ( H == "w" ){
     Eliminar_Archivo("SX.log");
     Escribir_Archivo("SX.log", SX);

     Eliminar_Archivo("PX.log");
     Escribir_Archivo("PX.log", PX);

     Eliminar_Archivo("IX.log");
     Escribir_Archivo("IX.log", IX);

     Config_WIFI_HOST();
    }
    else if ( H == "t" ){
     Eliminar_Archivo("SP.log");
     Escribir_Archivo("SP.log", SP);

     Eliminar_Archivo("PP.log");
     Escribir_Archivo("PP.log", PP);

     Config_WIFI_AP();
    }
    else if ( H == "c" ){
     Eliminar_Archivo("NX.log");
     Escribir_Archivo("NX.log", NX);
    }
    else if ( H == "v" ){
     Eliminar_Archivo("MV.log");
     Escribir_Archivo("MV.log", MV);
     if(enbCAM_VEL) Definir_MV(MV);
    }
    else if ( H == "p" ){
     Eliminar_Archivo("CX.log");
     Escribir_Archivo("CX.log", CX);
    }
    else if ( H == "k" ){
     Definir_Tiempo(Fecha);
    }
    else if ( H == "i" ){
     Eliminar_Archivo("MD.log");
     Escribir_Archivo("MD.log", MD);
```

```
          enbMD=true;
        }
        else if ( H == "g" ){
          Eliminar_YZ();
        }
      }
    }
  }
}
```

Código utilizado para atender los mensajes de texto entrantes

```
if(SIM.available() > 0){
  if(SIM.find("+CMTI"))
  {
   SIM.println("AT+CMGL=\"REC UNREAD\""); //Solicitar al mensaje de texto
   delay(20);
   if (SIM.find("+54")){

    String Aux_N = SIM.readStringUntil('"');

    if (Aux_N==NX && SIM.find(13))
    {
     String Aux_T = SIM.readStringUntil(13);
     ApagarServidorWIFI(); //Apagar servidor WIFI para dedicarle tiempo al GSM

     Aux_T.toUpperCase();
     BorrarEnviarEsperarSIM("AT+CMGDA=\"DEL ALL\"", 'K');  //Borrar mensajes

     String LAT ="0";
     String LON ="0";
     String auxD2M = "";
     String auxD2YZ ="";
     boolean auxaux = false;

     if(digitalRead(PIN_CAL)==HIGH)
      {
       LAT = Consultar_GPS("Lat");
       LON = Consultar_GPS("Lon");
       auxaux=true;
      }
     else PC.println("PIN_CAL bajo");

      if(Buscar(Aux_T , "UBI")){
      if(auxaux) //Si la lectura del GPS pudo realizarse
      {
```

```
          auxD2M="Ubicacion: https://www.google.com.ar/maps/search/" + LAT + ",+" +
LON;
         }
         else
         {
          auxD2M="Ubicacion momentaneamente no disponible";
         }

         auxD2YZ="SIM: Ubic";
        }
        else if(Buscar(Aux_T , "APA")){
         auxD2M="Sistema Apagado, Modo1";
         MD = "1";
         Eliminar_Archivo("MD.log");
         Escribir_Archivo("MD.log", MD);
         auxD2YZ="SIM: Modo1";
         enbMD=true;
        }
        else if(Buscar(Aux_T , "AUT")){
         auxD2M="Sistema Automatico, Modo2";
         MD = "2";
         Eliminar_Archivo("MD.log");
         Escribir_Archivo("MD.log", MD);
         auxD2YZ="SIM: Modo2";
         enbMD=true;
        }
        else if(Buscar(Aux_T , "ENC")){
         auxD2M="Sistema Encendido, Modo3";
         MD = "3";
         Eliminar_Archivo("MD.log");
         Escribir_Archivo("MD.log", MD);
         auxD2YZ="SIM: Modo3";
         enbMD=true;
        }
        else{
         auxD2M="Comando no reconocido";
         auxD2YZ="SIM: NR";
        }

        Escribir_YZ(auxD2YZ,LAT,LON);
        EncenderServidorWIFI();
        Enviar_Mensaje(auxD2M);

      }
     }
```

```
  }
 }
```

Código para atender al botón PANIC

```
if(digitalRead(PANIC)==LOW){
  if(enbPANIC)
  {
    digitalWrite(LEN, LOW);
    ApagarServidorWIFI();
    delay(100);
    digitalWrite(LEN, HIGH);

    enbPANIC = false;

    String Data = "Boton Antipanico Presionado.";

    if(digitalRead(PIN_CAL)==HIGH)
     {
       String LAT = Consultar_GPS("Lat");
       String LON = Consultar_GPS("Lon");
       Data = Data + "Ubicacion: https://www.google.com.ar/maps/search/" + LAT + ",+" +
LON;
       Escribir_YZ("PANIC",LAT, LON);
     }
    else
     {
       Data = Data + "Ubicacion no disponible";
     }

    Enviar_Mensaje(Data);

    EncenderServidorWIFI();
  }
 }
 else { enbPANIC = true; }
```

Código para atender el estado alto del PIN_VEL

```
if(digitalRead(PIN_VEL)==HIGH && MAINCLK){

    MAINCLK = false;

    String VRP = Consultar_GPS("VRP");
    String Data = "Limite de Velocidad Superado. Velocidad: " + VRP +"Km/h. ";
    String LAT = "0";
    String LON = "0";
```

```
    if(digitalRead(PIN_CAL)==HIGH)
    {
      LAT = Consultar_GPS("Lat");
      LON = Consultar_GPS("Lon");
      Data = Data + "Ubicacion: https://www.google.com.ar/maps/search/" + LAT + ",+" +
LON;
    }
    else
    {
      Data = Data + "Ubicacion no disponible";
    }

    Escribir_YZ("VEL: " + VRP , LAT, LON);

    if(enbPIN_VEL == true) { Enviar_Mensaje(Data); }

    enbPIN_VEL = false;

   }
  else { if(digitalRead(PIN_VEL)==LOW){ enbPIN_VEL = true;}}
```

A.1.2 DsPIC de la Placa Principal

A.1.2.1 Directivas del preprocesador

```
#include <30F4013.h>
#FUSES NOWDT              //No Watch Dog Timer
#FUSES CKSFSM             //Clock Switching is enabled, fail Safe clock monitor is enabled
#FUSES BORV27             //Brownout reset at 2.7V
#use delay(clock=120MHz,internal=7.5MHz)

//////////////////////////Serial////////////////////////////////////////////
#use rs232(UART1, baud=115200, stream=DUE)
#use rs232(UART2, baud=9600, stream=GPS)
#define U2STA 0x0218
#BIT OERR = U2STA.1

//////////////////////////Entrada-Salida////////////////////////////////////
#use FIXED_IO( B_outputs= PIN_B2, PIN_B1, PIN_B0)
#use FIXED_IO( D_outputs= PIN_D2)

#define PIN_ON  output_high
#define PIN_OFF output_low

#define PIN_CAL   PIN_B0
#define PIN_VEL   PIN_B1
```

```
#define PIN_CLK   PIN_B2

#define PIN_ZUM   PIN_D2

#define INWWW input(PIN_B3)
```

A.1.2.2 Interrupciones

```
#INT_TIMER5
void  timer5_isr(void)
{
    segundos++;

    if(segundos == 5)
    {
     pin_on(PIN_CLK);
    }
    else if(segundos == 10 )
    {
     pin_off(PIN_CLK);
     segundos=0;
    }
}
```

A.1.2.3 Configuraciones iniciales

Definición de las variables globales
unsigned int8 segundos = 0;
Definición de variables locales y configuraciones del microcontrolador

```
  ////////////////////////Pantalla/////////////////////////////////////////////////
      int1 INWWWAnt = 1;

  ////////////////////////Due/////////////////////////////////////////////////
  const char ATVelRPM[]     = {"VRP"};
  const char ATE[]          = {"ATE"};
  const char ATHora[]       = {"Hor"};
  const char ATLongitud[]   = {"Lon"};
  const char ATLatitud[]    = {"Lat"};
  const char ATMaxCon[]     = {"Xmv"}; //Consulta velocidad máxima
  const char ATMaxVel[]     = {"MV"};   //Define velocidad máxima
  const char ATStr[]        = {"ST"};  //Cadena de caracteres

        char  AT[]          = {"xxx"};
        char  Str[]         = {"0123456789abcdfe"};
```

```
  ////////////////////////GPS/////////////////////////////////////////////////
      int  i              =   0 ;
      char msj            =  'x';
      char Calidad        =  '0';

  const char GPGGA[]       = {"GPGGA"};
  const char GPVTG[]       = {"GPVTG"};

      char GPXXX[]        = {"GPXXX"};

      char Hora[]         = {"000000.00"};
      char Latitud[]      = {"0000.00000"};
      char NS             =   'N';
      char Longitud[]     = {"00000.00000"};
      char EW             =   'E';

      char strvel[]       = {"SAT"};
      char strvelAnt[]    = {"SAT"};

  ////////////////////////Variables de Velocidad//////////////////////////////
unsigned int8      MaxVel   = 180; //Velocidad Máxima por defecto: 180Km/h
unsigned int8    velocidad   = 0;
unsigned int8 velocidadAnt   = 0;

  ////////////////////////Otros/////////////////////////////////////////////////
  setup_timer4(TMR_INTERNAL | TMR_DIV_BY_1 | TMR_32_BIT, 30000000);
  enable_interrupts(INT_TIMER5);
  enable_interrupts(INTR_GLOBAL);
```

A.1.2.4 Bucle principal

Caracteres recibidos desde el Arduino Due

```
if(kbhit(DUE))
      {
        gets(AT,DUE);

      ////////////////////////ATE/////////////////////////////////////////////
         if(AT[0]==ATE[0] && AT[1]==ATE[1] && AT[2]==ATE[2])
        {
          fprintf(DUE,"OK\r");
        }
      ////////////////////////Hor/////////////////////////////////////////////
      else if(AT[0]==ATHora[0] && AT[1]==ATHora[1] && AT[2]==ATHora[2])
        {

          int8 auxHor = (Hora[0]-48)*10 + Hora[1]-48;
```

```
          if(auxHor > 2) auxHor = auxHor - 3; //Conversion a Hora de Argentina GTM-3
          else auxHor = 21 + auxHor;

          fprintf(DUE,"%d:%c%c:%c%c\r", auxHor,Hora[2],Hora[3],Hora[4],Hora[5]);
        }
      ////////////////////////MaxVel/////////////////////////////////////////
      else if(AT[0]==ATMaxCon[0] && AT[1]==ATMaxCon[1] && AT[2]==ATMaxCon[2])
        {
          fprintf(DUE,"%u\r",MaxVel);
        }
      ////////////////////////Lon/////////////////////////////////////////
      else if(AT[0]==ATLongitud[0] && AT[1]==ATLongitud[1] && AT[2]==ATLongitud[2])
        {
          float Longitud_dec=(Longitud[3]-48)*10+(Longitud[4]-48)+(Longitud[6]-
48)*0.1+(Longitud[7]-48)*0.01+(Longitud[8]-48)*0.001+(Longitud[9]-
48)*0.0001+(Longitud[10]-48)*0.00001;
          Longitud_dec=Longitud_dec/60+(Longitud[0]-48)*100+(Longitud[1]-
48)*10+(Longitud[2]-48);

          if(EW=='W') { Longitud_dec=-Longitud_dec;}

          fprintf(DUE,"%.5f\r",Longitud_dec);
        }
      ////////////////////////Lat/////////////////////////////////////////
      else if(AT[0]==ATLatitud[0] && AT[1]==ATLatitud[1] && AT[2]==ATLatitud[2])
        {
          float Latitud_dec=(Latitud[2]-48)*10+(Latitud[3]-48)+(Latitud[5]-
48)*0.1+(Latitud[6]-48)*0.01+(Latitud[7]-48)*0.001+(Latitud[8]-48)*0.0001+(Latitud[8]-
48)*0.00001;
          Latitud_dec=Latitud_dec/60+(Latitud[0]-48)*10+(Latitud[1]-48);

          if(NS=='S') { Latitud_dec=-Latitud_dec;}

          fprintf(DUE,"%.5f\r",Latitud_dec);
        }
      ////////////////////////VGP/////////////////////////////////////////
      else if(AT[0]==ATMaxVel[0] && AT[1]==ATMaxVel[1])
        {
          MaxVel=AT[2];
          fprintf(DUE,"OK\r");
        }
      ////////////////////////VRP/////////////////////////////////////////
      else if(AT[0]==ATVelRPM[0] && AT[1]==ATVelRPM[1] && AT[2]==ATVelRPM[2])
        {
          fprintf(DUE,"%u\r",Velocidad);
```

```
      }
    ////////////////////////ST/////////////////////////////////////////
    else if(AT[0]==ATStr[0] && AT[1]==ATStr[1])
    {
      for(int j=0; j<16; j++)
      {  Str[j]=0;   }

      fprintf(DUE,">"); //Listo para recibir el String

      gets(Str,DUE);
      fprintf(GPS,">%s\r",Str);
    }

  }
```

Caracteres recibidos desde el GPS

```
    if(kbhit(GPS) || OERR)
    {
     if(OERR) OERR=0; //Eliminamos el error generado por el OverRun

     msj=getch(GPS);
     if(msj=='$')
       {
         i=0;
         msj=getch(GPS);
         while(msj!=',')
           {
             GPXXX[i]=msj;
             msj=getch(GPS);
             i++;
           }

           if(GPXXX[2]==GPGGA[2] && GPXXX[3]==GPGGA[3] && GPXXX[4]==GPGGA[4])
           {
           ////////////////////////Hora////////////////////////////////
             i=0;
             msj=getch(GPS);
             while(msj!=',')
             {
               Hora[i]=msj;
               msj=getch(GPS);
               i++;
             }
           ////////////////////////Latitud/////////////////////////////
             i=0;
```

```
        msj=getch(GPS);
        while(msj!=',')
        {
          Latitud[i]=msj;
          msj=getch(GPS);
          i++;
        }
      ////////////////////////NS/////////////////////////////////
        msj=getch(GPS);
        if(msj!=',')
        {
          NS=msj;
          msj=getch(GPS);
        }
      ////////////////////////Longitud/////////////////////////////
        i=0;
        msj=getch(GPS);
        while(msj!=',')
        {
          Longitud[i]=msj;
          msj=getch(GPS);
          i++;
        }
      ////////////////////////EO/////////////////////////////////
        msj=getch(GPS);
        if(msj!=',')
        {
          EW=msj;
          msj=getch(GPS);
        }
      ////////////////////////Calidad/////////////////////////////
        msj=getch(GPS);
        if(msj!=',')
        {
          Calidad=msj;
          msj=getch(GPS);
        }

      }

    else if(GPXXX[2]==GPVTG[2] && GPXXX[3]==GPVTG[3] && GPXXX[4]==GPVTG[4])
      {
      ////////////////////////IgnorarComas/////////////////////////
        i=0;
        while(i<6)
```

```
    {
      if(kbhit(GPS) && getch(GPS)==',')
      {
        i++;
      }
    }
  ////////////////////////AlmacenarVelocidad/////////////////
    msj=getch(GPS);

    velocidadAnt=velocidad;

    strvelAnt[2]=strvel[2];
    strvelAnt[1]=strvel[1];
    strvelAnt[0]=strvel[0];

    velocidad=0;
    i=0;
    while(msj!=',' && msj!='.')
    {
      strvel[i]=msj;
      i++;
      velocidad=(msj-'0')+velocidad*10;
      msj=getch(GPS);
    }

      if(i==0)
      {
        strvel="SAT";
      }
      else if(i==1)
      {
        strvel[2]=strvel[0];
        strvel[1]=' ';
        strvel[0]=' ';
      }
      else if(i==2)
      {
        strvel[2]=strvel[1];
        strvel[1]=strvel[0];
        strvel[0]=' ';
      }
      ////////////////////////Comparar////////////////////
      if(strvel[2]!=strvelAnt[2] || strvel[1]!=strvelAnt[1] || strvel[0]!=strvelAnt[0])
      {
        fprintf(GPS,"#%s\r",strvel);
```

```
            }
          }
        }
      }
```

Acciones realizadas en caso de que la información entregada por el GPS sea correcta

```
if(Calidad == '1' || Calidad == '2')
    {
      pin_on(PIN_CAL);

      if(velocidad>MaxVel && velocidadAnt>MaxVel)
      {
        pin_on(PIN_VEL);
        pin_on(PIN_ZUM);

      }
      else
      {
        pin_off(PIN_VEL);
        pin_off(PIN_ZUM);
      }
    }
    else
    {
      pin_off(PIN_CAL);
      pin_off(PIN_VEL);
      pin_off(PIN_ZUM);
    }
```

Código para atender el cambio de estado del PIN INWWW

```
if(INWWW && !INWWWAnt)
   {
     INWWWAnt=1;
     fprintf(GPS,"$S\r");
   }
   else if(!INWWW && INWWWAnt)
   {
     INWWWAnt=0;
     fprintf(GPS,"$N\r");
   }
```

A.1.3 PIC18F2550

A.1.3.1 Directivas del preprocesador

```
#include <18F2550.h>
#FUSES NOWDT                 //No Watch Dog Timer
#FUSES WDT128                //Watch Dog Timer uses 1:128 Postscale
#FUSES BROWNOUT              //Brownout reset
#FUSES NOLVP                 //No low voltage prgming, B3(PIC16) or B5(PIC18) used for I/O
#FUSES NOXINST               //Extended set extension and Indexed Addressing mode disabled
#FUSES NOMCLR
#FUSES PUT                   //Power Up Timer
#use delay(crystal=20000000)

//////////////////////////Serial/////////////////////////////////////////////
#use rs232(UART1, baud=9600, stream=dsPIC)
#word RCSTA = 0xFAB
#bit  OERR  = RCSTA.1
#bit  CREN  = RCSTA.4

//////////////////////////LCD/////////////////////////////////////////////
#define LCD_RS_PIN     PIN_B0
#define LCD_RW_PIN     PIN_B1
#define LCD_ENABLE_PIN  PIN_B2
#define LCD_DATA4      PIN_B3
#define LCD_DATA5      PIN_B4
#define LCD_DATA6      PIN_B5
#define LCD_DATA7      PIN_B6

#include <lcd.c>

//////////////////////////Entrada-Salida/////////////////////////////////////
#define IN  PIN_B7

//////////////////////////USB/////////////////////////////////////////////
#byte UCON = 0xF6D
#bit USBEN = UCON.3
#byte UCFG = 0xF6F
#bit USBTRANS = UCFG.3

//////////////////////////TMR1/////////////////////////////////////////////
////////////////////// Dt1=4*Pre*(2^16-Carga)/fosc ////////////////////////////
////////////////////// Carga=2^16-Dt1*fosc/(4*Pre) ////////////////////////////
////////////////////// Dt1=0.1s  ->   Carga=3036       ////////////////////////////
////////////////////// Dt1=0.01s ->   Carga=59286    ////////////////////////////
#define carga 3036
```

A.1.3.2 Interrupciones

```
#INT_TIMER1
void TIMER1_isr(void)
{
  set_timer1(carga);
  decimas++;
  fdecimas++; //Temporizador de funciones

  if(enableSL==1 && decimas>50)
  {
    enableSL=0;
    enable=1;
    for(int j=0; j<16; j++)
    {
      Cadenas[j]=0;
    }
  }
}
```

A.1.3.3 Configuraciones iniciales

Definición de las variables globales

```
char Cadenas[]     = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
char Velocidad[]   = {"SAT"};
char Servidor[]    = {"   "};

int8 decimas   = 0;
int8 fdecimas  = 0;
int1 enable    = 0;
int1 enableSL  = 0;
```

Definición de variables locales y configuraciones del microcontrolador

```
  //Modulo USB desactivado
  USBEN =0;
  USBTRANS =1;

  delay_ms(2000);

  //LCD
  lcd_init();
  printf(LCD_PUTC, "\f");
  printf(LCD_PUTC, "Bienvenidos");

  int  i          =   0 ;
  char msj        =   'x';
```

```
//Temporizador
setup_timer_1(T1_INTERNAL|T1_DIV_BY_8);
set_timer1(carga);

//Interrupciones
enable_interrupts(INT_TIMER1);
enable_interrupts(GLOBAL);

delay_ms(2000);

if(OERR) { CREN=0; CREN=1; }
while(kbhit(dsPIC)) getch();
```

A.1.3.4 Bucle principal

```
if((kbhit(dsPIC) || OERR))
   {
     if(OERR) { CREN=0; CREN=1; } //Eliminar el error generado por el OverRun
     msj=getch(dsPIC);

     if(msj=='#') //Dato de velocidad
     {
       msj=getch(dsPIC);
       fdecimas=0;
       i=0;
       while(i<3 && fdecimas<5)
       {
         Velocidad[i]=msj;
         msj=getch(dsPIC);
         i++;
       }
       enable=1;
     }
     else if(msj=='$') //Dato de estado del servidor
     {
       msj=getch(dsPIC);
       if(msj=='S')
       {
         Servidor="WWW";
       }
       else if(msj=='N')
       {
         Servidor="   ";
       }
       enable=1;
     }
```

```
    else if(msj=='>') //Cadena de caracteres
    {
      msj=getch(dsPIC);

      decimas=0;

      fdecimas=0;
      i=0;
      while(msj!=13 && i<16 && fdecimas<5)
      {
        Cadenas[i]=msj;
        msj=getch(dsPIC);
        i++;
      }
      while(i<16)
      {
        Cadenas[i]=0;
        i++;
      }
      enable=1;
      enableSL=1;
    }
  }

  if(enable)
  {
    enable=0;
    printf(LCD_PUTC, "\f");
    printf(LCD_PUTC, "VEL = %s    %s\n",Velocidad,Servidor);
    printf(LCD_PUTC, "%s",Cadenas);
  }
```

A.1.4 DsPIC de la Placa de Ensayos

A.1.4.1 Directivas del preprocesador

```
#include <30F4013.h>
#FUSES NOWDT              //No Watch Dog Timer
#FUSES CKSFSM             //Clock Switching is enabled, fail Safe clock monitor is enabled
#FUSES BORV27             //Brownout reset at 2.7V
#FUSES NOMCLR
#use delay(crystal=20MHz)

////////////////////////////CAN////////////////////////////////////////////////
#include <can-dsPIC30.c>
#define IDmodo     0
#define IDmodoRQST  1
```

```
#define IDservidor  2
#define IDvelocidad 4
#define IDpantalla  5
#define IDalmacena  6
#define IDenvia     7
#define IDfinal     8
#define IDAux       9
#define IDinforma   10
#define DATAplaca   42

//////////////////////////Entrada-Salida//////////////////////////////////////
#use FIXED_IO( B_outputs=PIN_B4, PIN_B3, PIN_B2, PIN_B1, PIN_B0 )

#define PIN_ON  output_high
#define PIN_OFF output_low
#define R1              PIN_B0
#define R2              PIN_B1
#define Boton_A     PIN_B5
#define Boton_B     PIN_B8
#define led_desbloqueado   PIN_B2  //LED de Encendido desbloqueado
#define led_EntradaAux      PIN_B3  //LED de Señal Aux
#define led_sistema         PIN_B4  //LED de ENCENDIDO
#define EntradaAux          PIN_B11 //Señal Aux
```

A.1.4.2 Interrupción

```
#INT_TIMER3
void  timer3_isr(void)
{
  segundos++;
}
```

A.1.4.3 Funciones

Función para enviar la mitad de la cadena de caracteres por el bus CAN

```
void Enviar_Cadena(int8 ID, char Cadena[])
{
  char AuxP[8];
  char AuxS[8];
  int32 IDx = (int32) ID;

  int i;
  for(i=0; i<8; i++)
  {
    AuxP[i]=0;
```

```
    AuxS[i]=0;
  }

  i=0;
  while (Cadena[i]!=0 && i<8)
  {
    AuxP[i]=Cadena[i];
    i++;
  }

  while (Cadena[i]!=0 && i<16)
  {
    AuxS[i-8]=Cadena[i];
    i++;
  }

  while ( !can_tbe() )
  {
    ;
  }

  can_putd(IDx, &AuxP, 8, 1, 1, 0);

  while ( !can_tbe() )
  {
    ;
  }

  can_putd(IDx, &AuxS, 8, 1, 1, 0);

}
```

Función para enviar un carácter por el bus CAN

```
void Enviar_Caracter( int8 ID, int8 DATOS)
{
  int32 IDx = (int32) ID;

  while ( !can_tbe() )
  {
    ;
  }

  can_putd(IDx, &DATOS, 1, 1, 1, 0);
}
```

A.1.4.4 Configuraciones iniciales

Definición de las variables globales

```
int8 segundos = 0;
```

Definición de variables locales y configuraciones del microcontrolador

```
//Constantes de mensajes CAN entrantes
  struct rx_stat rxstat;
  int8 rx_len;
  int32 rx_id;
  int8 rx_data[8];

  //Inicializa el controlador CAN
  can_init();

  //Modo de Funcionamiento
  char MODO = "2";
  int1 enable = 1;
  pin_off(led_EntradaAux);
  pin_on(R1);
  pin_on(R2);
  pin_off(led_desbloqueado);

  ////////////////////////Otros/////////////////////////////////////////////
setup_timer2(TMR_INTERNAL | TMR_DIV_BY_1 | TMR_32_BIT, 5000000);
  enable_interrupts(INT_TIMER3);
  enable_interrupts(INTR_GLOBAL);

  //Informa a los sistemas que esta placa se encuentra conectada
  Enviar_Caracter( IDinforma, DATAplaca);

  //Solicita Modo de funcionamiento a placa de comunicaciones
  Enviar_Caracter( IDmodoRQST, DATAplaca);

  pin_on(led_sistema);
```

A.1.4.5 Bucle principal

Mensajes recibidos por el bus CAN

```
if ( can_kbhit() )
    {
      if(can_getd(rx_id, &rx_data[0], rx_len, rxstat))
      {
        if(rx_id == IDmodo)
        {//Modo de funcionamiento
          MODO = rx_data[0];
        }
```

```
        if(rx_id == IDinforma && rx_data[0] == 0)
        {//Mensaje enviado por placa de comunicaciones al encender
          Enviar_Caracter(IDinforma,DATAplaca);
          delay_ms(1000);
          Enviar_Caracter(IDmodoRQST,DATAplaca);
        }
      }
    }
```

Código para atender al botón B

```
if(input(Boton_B)==0) //Presionamos botón B
    {
      pin_off(led_sistema);
      delay_ms(300);
      pin_on(led_sistema);

      //Enviar una solicitud de trasferencia de la cadena de caracteres
      Enviar_Caracter(IDalmacena,DATAplaca); //Placa 42 requiere atención

      //Esperar respuesta
      char caracter_esperado=0;
      segundos=0;
      while(caracter_esperado!=DATAplaca && segundos<2)
      {
        if(can_getd(rx_id, &rx_data[0], rx_len, rxstat))
        {
          if(rx_id == IDalmacena)
          {
            caracter_esperado = rx_data[0];
          }
        }
      }

      //Enviar Cadena de caracteres
      if(caracter_esperado==DATAplaca)
      {
        Enviar_Cadena(IDalmacena,(char*)"P42: Boton B");
      }
    }
```

Código para atender un cambio en el estado de la Entrada Auxiliar

```
    if(input(EntradaAux)==1)
    {
      if(enable)
      {
```

```
      pin_on(led_EntradaAux);
      pin_off(R1);
      pin_off(R2);
      pin_off(led_desbloqueado);

      Enviar_Caracter(IDservidor,'0');   //Ordena Apagar el servidor
      Enviar_Caracter(IDvelocidad,'1'); //Ordena Velocidad 10km/h
      enable=0;
    }
  }
  else
  {
    if(enable==0)
    {
      pin_off(led_EntradaAux);
      Enviar_Caracter(IDservidor,'1');   //Ordena Encender el servidor
      Enviar_Caracter(IDvelocidad,'0'); //Ordena Apagar Velocidad 10km/h
      enable=1;
    }

      if(MODO == '1') //Sistema Desbloqueado
    {
      pin_off(R1);
      pin_off(R2);
      pin_on(led_desbloqueado);
    }
    else if(MODO == '2') //Sistema Automático, enviar a CAN al presionar botón
    {
      pin_on(R1);
      pin_on(R2);
      pin_off(led_desbloqueado);

      if(input(Boton_A)==0) //Presionamos botón A
      {
        pin_off(led_sistema);
        delay_ms(300);
        pin_on(led_sistema);

        pin_off(R1);
        pin_off(R2);
        pin_on(led_desbloqueado);

        //Enviar una solicitud de trasferencia de la cadena de caracteres
        Enviar_Caracter(IDenvia,DATAplaca); //Placa 42 requiere
```

```
        //Esperar respuesta
        char caracter_esperado=0;
        segundos=0;
        while(caracter_esperado!=DATAplaca && segundos<2)
        {
          if(can_getd(rx_id, &rx_data[0], rx_len, rxstat))
          {
            if(rx_id == IDenvia)
            {
              caracter_esperado = rx_data[0];
            }
          }
        }

        //Enviar Cadena de caracteres
        if(caracter_esperado==DATAplaca)
        {
          Enviar_Cadena(IDenvia,(char*)"P42: Boton A");
        }

        //Esperar 5 segundos
        delay_ms(5000);
      }

    }
    else if(MODO == '3') //Sistema Bloqueado
    {
      pin_on(R1);
      pin_on(R2);
      pin_off(led_desbloqueado);
    }
  }
```

A.2 Programación en Python

Debido a que el lenguaje Python es del tipo interpretado, la programacion puede realizarse directamente con un editor de texto.

```
#3.x python print
from __future__ import print_function

#Librerías utilizadas
import msvcrt
import serial
```

```
import glob
import time

#Función utilizada para mantener los valores acotados
def limites (valor, vector):
		longitud=len(vector)

		if valor >= longitud :
			valor = 0
		elif valor<0 :
			valor = longitud-1
		return valor

Archivo = 0

#Vector con la información de todos los archivos txt en el directorio de trabajo
ArchivosTexto = glob.glob("*.txt")

#Si el vector contiene elementos
if len(ArchivosTexto)>0:
	numArchivo=0
	Archivo = ArchivosTexto[0]

	print ("Archivos de texto disponibles:")
	print (ArchivosTexto)
	print ("\nTeclas de seleccion: Flechas y Enter")
	print ("Abrir archivo: ", end="")
	print (ArchivosTexto [numArchivo], end="\r")

	#Selección del archivo mediante teclas
	while True:
		if msvcrt.kbhit():
			tecla = msvcrt.getch().decode('utf-8','ignore')
			if tecla == '\x48' or tecla == '\x4D': #Arriba H M
				numArchivo=limites(numArchivo+1,ArchivosTexto)
				print ("                                                ", end="\r")
				print ("Abrir archivo: ", end="")
				print (ArchivosTexto [numArchivo], end="\r")
			elif tecla == '\x50' or tecla == '\x4B': #Abajo P K
				numArchivo=limites(numArchivo-1,ArchivosTexto)
				print ("                                                ", end="\r")
				print ("Abrir archivo: ", end="")
				print (ArchivosTexto [numArchivo], end="\r")
			elif tecla == '\x0d': #Enter
				Archivo = ArchivosTexto [numArchivo]
```

```
                                        break
                                elif tecla == '\x1b': #Esc
                                        print ("Fin")
                                        exit()
else:
        print ("No hay archivos disponibles")

#Vector con la información de todos los puertos COM disponibles
ports = ['COM%s' % (i + 1) for i in range(256)]
result = []
for port in ports:
   try:
      s = serial.Serial(port)
      s.close()
      result.append(port)
   except (OSError, serial.SerialException):
      pass

puerto=0
#SI el vector contiene elementos
if len(result) >= 1:

        numport=0
        puerto = result [0]

        print ("\n\nPuertos disponibles:")
        print (result)
        print ("\nTeclas de seleccion: Flechas y Enter")
        print ("Abrir puerto: ", end="")
        print (result [numport], end="\r")

        #Seleccionar el Puerto COM mediante teclas
        while True:
                if msvcrt.kbhit():
                        tecla = msvcrt.getch().decode('utf-8','ignore')
                        if tecla == '\x48' or tecla == '\x4D': #Arriba H M
                                numport=limites(numport+1,result)
                                print ("                    ", end="\r")
                                print ("Abrir puerto: ", end="")
                                print (result [numport], end="\r")
                        elif tecla == '\x50' or tecla == '\x4B': #Abajo P K
                                numport=limites(numport-1,result)
                                print ("                    ", end="\r")
                                print ("Abrir puerto: ", end="")
                                print (result [numport], end="\r")
```

```
                        elif tecla == '\x0d': #Enter
                                puerto = result [numport]
                                break
                        elif tecla == '\x1b': #Esc
                                print ("Fin")
                                exit()
else :
        print ("\n\nNo hay puertos disponibles")


#Si el archivo y el puerto pudieron ser seleccionados
if Archivo!=0 and puerto != 0:

        Bauds=["4800","9600","19200","38400","57600","115200"]

        numbaud=1
        velocidad = Bauds[1]
        print ("\n\nVelocidades disponibles:")
        print (Bauds)
        print ("\nTeclas de seleccion: Flechas y Enter")
        print ("Baudrate: ", end="")
        print (Bauds [numbaud], end="\r")

        #Seleccionar la velocidad de transferencia mediante teclas
        while True:
                if msvcrt.kbhit():
                        tecla = msvcrt.getch().decode('utf-8','ignore')
                        if tecla == '\x48' or tecla == '\x4D': #Arriba H M
                                numbaud=limites(numbaud+1,Bauds)
                                print ("                ", end="\r")
                                print ("Baudrate: ", end="")
                                print (Bauds [numbaud], end="\r")
                        elif tecla == '\x50' or tecla == '\x4B': #Abajo P K
                                numbaud=limites(numbaud-1,Bauds)
                                print ("                ", end="\r")
                                print ("Baudrate: ", end="")
                                print (Bauds [numbaud], end="\r")
                        elif tecla == '\x0d': #Enter
                                velocidad = Bauds [numbaud]
                                break
                        elif tecla == '\x1b': #Esc
                                print ("Fin")
                                exit()
        #Abrir puerto
        ser = serial.Serial(
```

```
                port=puerto,
                baudrate=velocidad,
                parity=serial.PARITY_ODD,
                stopbits=serial.STOPBITS_TWO,
                bytesize=serial.EIGHTBITS
        )
        #Abrir archivo
        archivo = open(Archivo,"r")

        #Si el puerto se encuentra abierto, enviar el archivo, tomando 8 líneas a la vez cada
un segundo
        if ser.isOpen():
                print ("\n\nEnviando a "+ Archivo + " por " + puerto + " a " + velocidad)
                print ("\n--------------------------------------------------------------\n");

                while True:
                        cantline =0
                        for line in archivo:
                                if msvcrt.kbhit() and msvcrt.getch().decode('utf-8','ignore')
== '\x1b': #Esc
                                        print ("Fin")
                                        exit()
                                print(line, end="")
                                ser.write(line)
                                if cantline==7:
                                        cantline=0
                                        print("\n")
                                        time.sleep(1)
                                else:
                                        cantline=cantline+1;
                        archivo.seek(0)

                print ("\n\n")

                ser.close()
                print ("\n\n--------------------------------------------------------------\n");

        archivo.close()

print ("\nFin")
exit()
```

A.3 Descripción de páginas web

Al igual que en el lenguaje Python, la descripción de los elementos de las distintas paginas puede realizarse directamente con un editor de texto.

A.3.1 Página Principal

```
<!DOCTYPE html>
<html lang=es>
<head>
	<meta charset="utf-8" />
	<meta name="viewport" content="width=device-width, initial-scale=1.0">
	<title>Pagina Principal</title>

  <style>
		body{
			background: #F0FAD1;
		}

		fieldset{
			border-radius: 10px;
			background: #FEFDF1;
		}

		legend{
			border-radius: 3px;
			background-color: #EA7B00;
			color: #FFFFFF;
		}

		a{
			color: blue;
		}
	</style>
</head>

<body>
	<fieldset>
		<legend>Estado</legend>

		<ul>
			<li><a href="s.html">Estado del sistema</a></li>
		</ul>
	</fieldset>

	<br>

	<fieldset>
```

```
                <legend>Sistema</legend>

                <ul>
                        <li><a href="v.html">Configurar velocidad maxima</a></li><br>
                        <li><a href="i.html">Configurar modos de funcionamiento</a></li>

                </ul>

        </fieldset>

        <br>

        <fieldset>
                <legend>Conectividad</legend>

                <ul>
                        <li><a href="w.html">Configurar datos de red wifi</a></li><br>
                        <li><a href="t.html">Configurar punto de acceso wifi</a></li><br>
                        <li><a href="c.html">Modificar numero de administrador</a></li>
                </ul>

        </fieldset>

        <br>

        <fieldset>
                <legend>Gestion de datos</legend>

                <ul>
                        <li><a href="k.html">Modificar fecha del sistema</a></li><br>
                        <li><a href="p.html">Modificar contraseña de
administrador</a></li><br>
                        <li><a href="g.html">Descargar o eliminar ubicaciones
almacenadas</a></li>
                </ul>

        </fieldset>

  <br>

        <fieldset>
                <legend>Visualizacion de Mapas</legend>

    <p><em> Para realizar esta accion se requiere de una conexion a internet</em></p>
```

```
                <ul>
                        <li><a href="m.html">Mapas</a></li>
                </ul>
        </fieldset>

</body>
</html>
```

A.3.2 Página de configuración del número del administrador

```
<!DOCTYPE html>
<html lang=es>
<head>
        <meta charset="utf-8" />
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>Numero del administrador</title>

        <style>
                input[type=submit]{
                        background-color: #EA7B00;
                        color: #FFFFFF;
                        border: none;
                        padding: 8px 18px;
                        margin: 2px 2px;
                        cursor: pointer;
                        border-radius: 5px;
                }

                body{
                        background: #F0FAD1;
                }

                fieldset{
                        border-radius: 10px;
                        background: #FEFDF1;
                }

                legend{
                        border-radius: 3px;
                        background-color: #EA7B00;
                        color: #FFFFFF;
                }
        </style>
</head>

<body>
```

```
        <h1>Numero del administrador</h1>

        <FORM method=POST autocomplete="on">

                <p>
                        <label for="CX">Contraseña:</label>
                        <input
                                        type="password"
                                        id="CX"
                                        name="CX"
                                        placeholder="123456"
                                        pattern="[0-9]{6}"
                                        size="6"
                                        maxlength="6"
                                        required
                                        autofocus
                        >
                </p>

                <input type="hidden" name="H" value="c" />

                <fieldset>
                        <legend>Numero de telefono</legend>

                        <p>

                                <input
                                                type="text"
                                                name="NX"
                                                id="NX"
                                                placeholder="3804256398"
                                                pattern="[0-9]{3}4[0-9]{6}"
                                                size="10"
                                                maxlength="10"
                                                required
                                >
                        </p>
                </fieldset>

                <p>
                        <INPUT type="submit" name="E" id="E" value="Guardar Cambios">
                </p>
        </FORM>
</body>
</html>
```

A.3.3 Página de error

```
<!DOCTYPE html>
<html lang=es>
<head>
	<meta charset="utf-8" />
	<meta name="viewport" content="width=device-width, initial-scale=1.0">
	<title>Error</title>
	<style>
		body{
			background: #F0FAD1;
		}
	</style>
</head>
<body>
	<h1>Error</h1>

  <p><em> La Pagina solicitada no existe.</em></p>

</body>
</html>
```

A.3.4 Página de gestión de archivos

```
<!DOCTYPE html>
<html lang=es>
<head>
	<meta charset="utf-8" />
	<meta name="viewport" content="width=device-width, initial-scale=1.0">
	<title>Gestion de Archivos</title>

	<style>
		input[type=submit]{
			background-color: #EA7B00;
			color: #FFFFFF;
			border: none;
			padding: 8px 18px;
			margin: 2px 2px;
			cursor: pointer;
			border-radius: 5px;
		}

		body{
			background: #F0FAD1;
		}
```

```
                fieldset{
                        border-radius: 10px;
                        background: #FEFDF1;
                }

                legend{
                        border-radius: 3px;
                        background-color: #EA7B00;
                        color: #FFFFFF;
                }

                a{
                        color: blue;
                }
        </style>
</head>

<body>
        <h1>Gestion de Archivos</h1>

        <fieldset>
                <legend>Descarga</legend>

                <p><em> Para realizar esta accion se recomienda el uso de Chrome, Firefox
u Opera.</em></p>

                <ul>
                        <li><a href="z.gpx" download="z.gpx">Archivo de
Ubicaciones.</a><br></li>
                </ul>
        </fieldset>
        <br>
        <FORM method=POST autocomplete="on">

                <fieldset>
                        <legend>Opciones</legend>

                        <p>
                                <label for="CX">Contraseña:</label>
                                <input
                                                type="password"
                                                id="CX"
                                                name="CX"
                                                placeholder="123456"
```

```
                                        pattern="[0-9]{6}"
                                        size="6"
                                        maxlength="6"
                                        required
                                >
                        </p>
                        <p>
                        <input type="hidden" name="H" value="g" />


                                <input type="submit" name="E" id="E" value="Eliminar
Archivo">
                        </p>

                </fieldset>

        </FORM>

</body>
</html>
```

A.3.5 Configuración de los modos de funcionamiento

```
<!DOCTYPE html>
<html lang=es>
<head>
        <meta charset="utf-8" />
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>Modos de Funcionamiento</title>

        <style>
                input[type=submit]{
                        background-color: #EA7B00;
                        color: #FFFFFF;
                        border: none;
                        padding: 8px 18px;
                        margin: 2px 2px;
                        cursor: pointer;
                        border-radius: 5px;
                }

                body{
                        background: #F0FAD1;
                }

                fieldset{
```

```
                        border-radius: 10px;
                        background: #FEFDF1;
                }

                legend{
                        border-radius: 3px;
                        background-color: #EA7B00;
                        color: #FFFFFF;
                }
        </style>
</head>

<body>
        <h1>Modos de Funcionamiento</h1>

        <FORM method=POST autocomplete="on">

                <p>
                        <label for="CX">Contraseña:</label>
                        <input
                                type="password"
                                id="CX"
                                name="CX"
                                placeholder="123456"
                                pattern="[0-9]{6}"
                                size="6"
                                maxlength="6"
                                required
                                autofocus
                        >

                </p>

                <input type="hidden" name="H" value="i" />

                <fieldset>
                        <legend>Opciones</legend>

                        <input type="radio" name="MD" value="1"><b>Modo 1:</b>
Sistema Apagado.<br><br>
                        <input type="radio" name="MD" value="2" checked><b>Modo
2:</b> Sistema Automatico.<br><br>
                        <input type="radio" name="MD" value="3"><b>Modo 3:</b>
Sistema Encendido.<br>
```

```
                </fieldset>

                <p>
                        <input type="submit" name="E" id="E" value="Guardar Cambios">
                </p>

        </FORM>
</body>
</html>
```

A.3.6 Configuración de fecha

```
<!DOCTYPE html>
<html lang=es>
<head>
        <meta charset="utf-8" />
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>Fecha</title>

        <style>
                input[type=submit]{
                        background-color: #EA7B00;
                        color: #FFFFFF;
                        border: none;
                        padding: 8px 18px;
                        margin: 2px 2px;
                        cursor: pointer;
                        border-radius: 5px;
                }

                body{
                        background: #F0FAD1;
                }

                fieldset{
                        border-radius: 10px;
                        background: #FEFDF1;
                }

                legend{
                        border-radius: 3px;
                        background-color: #EA7B00;
                        color: #FFFFFF;
                }
        </style>
</head>
```

```
<body>
	<h1>Fecha</h1>

	<FORM method=POST>

		<p>
			<label for="CX">Contraseña:</label>
			<input
				type="password"
				id="CX"
				name="CX"
				placeholder="123456"
				pattern="[0-9]{6}"
				size="6"
				maxlength="6"
				required
				autofocus
				autocomplete="on"
			>
		</p>

		<input type="hidden" name="H" value="k" />

		<fieldset>
			<legend>Fecha Actual</legend>
			<p>

				<input
					type="date"
					name="DD"
					id="DD"
					max="2099-12-31"
					min="2017-01-01"
					size="10"
					maxlength="10"
					required
				>
			</p>
		</fieldset>

		<p>
			<INPUT type="submit" name="E" id="E" value="Guardar Cambios">
		</p>
	</FORM>
```

```
</body>
</html>
```

A.3.7 Página de mapas

La página de mapas utiliza una API de Google Maps para generar el contenido y requiere de una clave personal configurada para el proyecto[40].

Clave utilizada en el proyecto (a nombre del responsable del mismo)	
GOOGLE_API_CLAVE	AIzaSyDtDbN0UFCeMIV_YKygjNpTMcGW9Bksoq0

```
<!DOCTYPE html>
<html lang=es>
<head>
        <meta charset="utf-8" />
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Mapas</title>

                <style>
                        #map {
                                height: 100%;
                                }

                        html, body {
                                height: 100%;
                                margin: 0;
                                padding: 0;
                                }
                </style>

                <script>
                        var marcadores = [

                                ['Leo', -29.420833, -66.844860],
                                ['Cristian', -29.404800, -66.858225],
                                ['UTN', -29.410139, -66.832183]

                                ];


                        function initMap()
                        {
```

[40] Obtener una clave o autenticación de Google Maps JavaScript API: https://developers.google.com/maps/documentation/javascript/get-api-key?hl=ES

```
                                var map = new
google.maps.Map(document.getElementById('map'),
                                        {
                                                center: {lat: -29.420833, lng: -66.844860},
                                                zoom: 14
                                        });

                                var infowindow = new google.maps.InfoWindow();


                                for (var i = 0; i < marcadores.length; i++)
                                {
                                        var marker = new google.maps.Marker(
                                                {
                                                position: new
google.maps.LatLng(marcadores[i][1], marcadores[i][2]),
                                                map: map
                                                });

                                                google.maps.event.addListener(marker, 'click',
(function(marker, i)
                                                {
                                                return function()
                                                        {

        infowindow.setContent(marcadores[i][0]);
                                                                infowindow.open(map, marker);
                                                                }
                                                })(marker, i));
                                }

                                marker.setMap(map);
                        }
                </script>

                <script
src="https://maps.googleapis.com/maps/api/js?key=GOOGLE_API_CLAVE&callback=initMap
" async defer>
                </script>
</head>

<body>
        <div id="map"></div>
</body>
</html>
```

A.3.8 Página para cambiar contraseña

```
<!DOCTYPE html>
<html lang=es>
<head>
	<meta charset="utf-8" />
	<meta name="viewport" content="width=device-width, initial-scale=1.0">
	<title>Cambiar Contraseña</title>

	<style>
		input[type=submit]{
			background-color: #EA7B00;
			color: #FFFFFF;
			border: none;
			padding: 8px 18px;
			margin: 2px 2px;
			cursor: pointer;
			border-radius: 5px;
		}

		body{
			background: #F0FAD1;
		}

		fieldset{
			border-radius: 10px;
			background: #FEFDF1;
		}

		legend{
			border-radius: 3px;
			background-color: #EA7B00;
			color: #FFFFFF;
		}
	</style>
</head>

<body>
	<h1>Cambiar Contraseña</h1>

	<FORM method=POST autocomplete="on">

		<fieldset>
			<legend>Nueva contraseña del Administrador</legend>
```

```
                        <p><em> La contraseña debe estar formada por un numero de seis digitos.</em></p>

                        <p>
                                <label for="CX">Actual:</label>
                                <input
                                        type="password"
                                        id="CX"
                                        name="CX"
                                        placeholder="123456"
                                        pattern="[0-9]{6}"
                                        size="6"
                                        maxlength="6"
                                        required
                                        autofocus
                                >
                        </p>

                        <input type="hidden" name="H" value="p" />

                        <p>
                                <label for="QX">Nueva:</label>
                                <input
                                        type="password"
                                        id="QX"
                                        name="QX"
                                        placeholder="123456"
                                        pattern="[0-9]{6}"
                                        size="6"
                                        maxlength="6"
                                        required
                                >
                        </p>
                </fieldset>

                <p>
                        <INPUT type="submit" name="E" id="E" value="Guardar Cambios">
                </p>
        </FORM>
</body>
</html>
```

A.3.9 Página para indicar que los cambios fueron almacenados

```
<!DOCTYPE html>
<html lang=es>
```

```
<head>
        <meta charset="utf-8" />
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>Cambios Guardados</title>
        <style>
                body{
                        background: #F0FAD1;
                }
        </style>
</head>
<body>
        <h1>Cambios Guardados</h1>

   <p><em> La informacion se almacenó satisfactoriamente.</em></p>

</body>
</html>
```

A.3.10 Página para indicar que la contraseña ingresada no es la correcta

```
<!DOCTYPE html>
<html lang=es>
<head>
        <meta charset="utf-8" />
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>Contraseña Incorrecta</title>
        <style>
                body{
                        background: #F0FAD1;
                }
        </style>
</head>
<body>
        <h1>Contraseña Incorrecta</h1>

   <p><em> Ingrese la contraseña del Administrador para poder realizar cambios en el
sistema.</em></p>

</body>
</html>
```

A.3.11 Página de estado

```
<!DOCTYPE html>
<html lang=es>
```

```
<head>
        <meta charset="utf-8" />
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>Estado del Sistema</title>

  <style>
                body{
                        background: #F0FAD1;
                }

                fieldset{
                        border-radius: 10px;
                        background: #FEFDF1;
                }

                legend{
                        border-radius: 3px;
                        background-color: #EA7B00;
                        color: #FFFFFF;
                }
        </style>
</head>

<body>
<h1> Estado del Sistema </h1>
        <fieldset>
                <legend>Sistema</legend>

                <ul>
                        <li><b>Velocidad maxima:</b> ?</li><br>
                        <li><b>Modo de funcionamiento:</b> Modo ?</li>

                </ul>

        </fieldset>

        <br>

        <fieldset>
                <legend>Conectividad</legend>

                <ul>
                        <li><b>Numero de Administrador:</b> ?</li><br>

                        <li><b>Datos de red WIFI</b></li><br>
```

```
                        <ul>
                                <li><b>Nombre:</b> ?</li><br>
                                <li><b>IP:</b> ?</li><br>
                        </ul>

                        <li><b>Datos del Punto de Acceso WIFI</b></li><br>
                        <ul>
                                <li><b>Nombre:</b> ?</li><br>
                                <li><b>IP:</b> 192.168.4.1</li><br>
                        </ul>

                        <li><b>Modulos</b></li><br>
                        <ul>
                                <li><b>GPS:</b> ?</li><br>
                                <li><b>SIM:</b> ?</li>
                        </ul>
                </ul>

        </fieldset>

        <p align="right"><b> ?</b></p>
</body>
</html>
```

A.3.12 Configuración del punto de acceso

```
<!DOCTYPE html>
<html lang=es>
<head>
        <meta charset="utf-8" />
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>Punto de acceso WIFI</title>

        <style>
                input[type=submit]{
                        background-color: #EA7B00;
                        color: #FFFFFF;
                        border: none;
                        padding: 8px 18px;
                        margin: 2px 2px;
                        cursor: pointer;
                        border-radius: 5px;
                }

                body{
```

```
                background: #F0FAD1;
        }

        fieldset{
                border-radius: 10px;
                background: #FEFDF1;
        }

        legend{
                border-radius: 3px;
                background-color: #EA7B00;
                color: #FFFFFF;
        }
    </style>
</head>

<body>
    <h1>Punto de acceso WIFI</h1>

    <FORM method=POST autocomplete="on">

        <p>
            <label for="CX">Contraseña:</label>
            <input
                    type="password"
                    id="CX"
                    name="CX"
                    placeholder="123456"
                    pattern="[0-9]{6}"
                    size="6"
                    maxlength="6"
                    required
                    autofocus
            >
        </p>

        <input type="hidden" name="H" value="t" />

        <fieldset>
            <legend>Datos de Red</legend>

            <p>
                <label for="SP">SSID</label>
                <input
                        type="text"
```

```
                                        name="SP"
                                        id="SP"
                                        placeholder="Red35"
                                        required
                                >
                        </p>

                        <p>
                                <label for="PP">PASS</label>
                                <input
                                        type="password"
                                        id="PP"
                                        name="PP"
                                        placeholder="********"
                                >
                        </p>

                </fieldset>

                <p>
                        <INPUT type="submit" name="E" id="E" value="Guardar Cambios">
                </p>
        </FORM>
</body>
</html>
```

A.3.13 Página para la configuración de la velocidad máxima

```
<!DOCTYPE html>
<html lang=es>
<head>
        <meta charset="utf-8" />
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>Velocidad Maxima</title>

        <style>
                input[type=submit]{
                        background-color: #EA7B00;
                        color: #FFFFFF;
                        border: none;
                        padding: 8px 18px;
                        margin: 2px 2px;
                        cursor: pointer;
                        border-radius: 5px;
                }
```

```
                body{
                        background: #F0FAD1;
                }

                fieldset{
                        border-radius: 10px;
                        background: #FEFDF1;
                }

                legend{
                        border-radius: 3px;
                        background-color: #EA7B00;
                        color: #FFFFFF;
                }
        </style>
</head>

<body>
        <h1>Velocidad Maxima</h1>

        <FORM method=POST autocomplete="on">

                <p>
                        <label for="CX">Contraseña:</label>
                        <input
                                        type="password"
                                        id="CX"
                                        name="CX"
                                        placeholder="123456"
                                        pattern="[0-9]{6}"
                                        size="6"
                                        maxlength="6"
                                        required
                                        autofocus
                        >
                </p>

                <input type="hidden" name="H" value="v" />

                <fieldset>
                        <legend>Limite de Velocidad</legend>
                        <p><em> El valor ingresado esta expresado en kilometros por
hora.</em></p>
                        <p>
```

```
                        <input
                                type="number"
                                name="MV"
                                id="MV"
                                placeholder="120"
                                min="50"
                                max="180"
                                step="5"
                                size="3"
                                maxlength="3"
                                required
                        >
                    </p>
                </fieldset>

                <p>
                    <INPUT type="submit" name="E" id="E" value="Guardar Cambios">
                </p>
            </FORM>
</body>
</html>
```

A.3.14 Pagina de configuración de la red wi-fi

```
<!DOCTYPE html>
<html lang=es>
<head>
        <meta charset="utf-8" />
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>Configuracion de red WIFI</title>

        <style>
                input[type=submit]{
                        background-color: #EA7B00;
                        color: #FFFFFF;
                        border: none;
                        padding: 8px 18px;
                        margin: 2px 2px;
                        cursor: pointer;
                        border-radius: 5px;
                }

                body{
                        background: #F0FAD1;
                }
```

```
                fieldset{
                        border-radius: 10px;
                        background: #FEFDF1;
                }

                legend{
                        border-radius: 3px;
                        background-color: #EA7B00;
                        color: #FFFFFF;
                }
        </style>
</head>

<body>
        <h1>Configuracion de red WIFI</h1>

        <FORM method=POST autocomplete="on">

                <p>
                        <label for="CX">Contraseña:</label>
                        <input
                                        type="password"
                                        id="CX"
                                        name="CX"
                                        placeholder="123456"
                                        pattern="[0-9]{6}"
                                        size="6"
                                        maxlength="6"
                                        required
                                        autofocus
                        >
                </p>

                <input type="hidden" name="H" value="w" />

                <fieldset>
                        <legend>Datos de Red</legend>

                        <p>
                                <label for="SX">SSID</label>
                                <input
                                                type="text"
                                                name="SX"
                                                id="SX"
```

```
                                        placeholder="Red2"
                                        required
                                >
                        </p>

                        <p>
                                <label for="PX">PASS</label>
                                <input
                                        type="password"
                                        id="PX"
                                        name="PX"
                                        placeholder="********"
                                >
                        </p>

                        <p>
                                <label for="IX">IP</label>
                                <input
                                        type="text"
                                        name="IX"
                                        id="IX"
                                        placeholder="192.168.0.2"
                                        pattern="[0-9]{1,2,3}.[0-9]{1,2,3}.[0-
9]{1,2,3}.[0-9]{1,2,3}"
                                        size="23"
                                        maxlength="15"
                                        required
                                >
                        </p>
                </fieldset>

                <p>
                        <INPUT type="submit" name="E" id="E" value="Guardar Cambios">
                </p>
        </FORM>
</body>
</html>
```

A.4 Imagen favicon

Imagen original utilizada para la creación del archivo favicon.

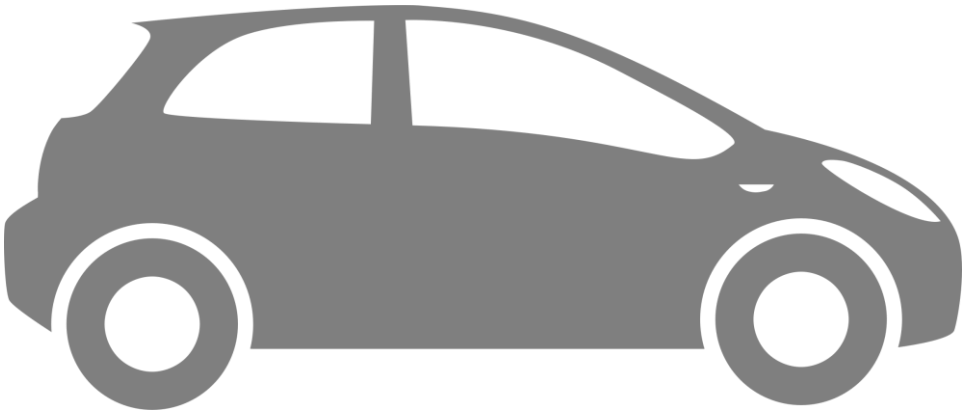

Fig. 73: Imagen del Archivo favicon[41].

[41] Cheeseness. Disponible en internet: https://openclipart.org/detail/275131/car

Anexo B: Internet móvil

En el presente anexo se describen los comandos utilizados para generar una comunicación HTTP mediante internet móvil de la red celular con un servidor web en la nube. Además, se incluye la programación PHP del servidor que utiliza una base de datos MySQL para almacenar información.

B.1 Comandos del módulo SIM900

Comandos utilizados para la configuración de una conexión GPRS.

Comando	Descripción
AT+SAPBR=3,1,"CONTYPE","GPRS"	Define método.
AT+SAPBR=3,1,"APN","*APNEmpresa*"	Define el APN de la conexión, su valor depende de la empresa que provee los servicios.
AT+SAPBR=3,1,"USER","*Usuario*"	Define el *Usuario* de la conexión, su valor depende de la empresa que provee los servicios.
AT+SAPBR=3,1,"PWD","*Contraseña*"	Define el *Contraseña* de la conexión, su valor depende de la empresa que provee los servicios.
AT+SAPBR=1,1	Realiza la conexión a la red de datos. Demora algunos segundos.

Comandos utilizados para consultar la dirección IP del módulo en la red de la empresa.

Comandos	Descripción	Respuesta
AT+SAPBR=2,1	Conexión exitosa	+SAPBR: 1,1,"100.86.220.128" OK
	Conexión no establecida	+SAPBR: 1,3,"0.0.0.0" OK

Comandos utilizados para establecer una conexión HTTP.

Comandos	Descripción
AT+HTTPINIT	Inicia el servicio HTTP.
AT+HTTPPARA="CID",1	Identificador de Portador.
AT+HTTPPARA="URL","*URL*"	URL del servidor.
AT+HTTPPARA="CONTENT","*Content*-Type "	*Content*-Type
AT+HTTPDATA= Content-Length,*TiempEsp*	Content-Lenght: Numero de caracteres de Datos. TiempEsp: Tiempo de espera por Datos.
Datos	Información enviada.
AT+HTTPACTION=*Método*	Método: - 0: GET. - 1: POST. - 2: HEAD.
AT+HTTPTERM	Finaliza el servicio HTTP.

B.2 Ejemplo de conexión

En los ejemplos se detallan los comandos utilizados para enviar información de la ubicación del vehículo, a una URL que direcciona a un servidor web en la nube, para almacenar información en una tabla de una base de datos.

Configuración de una conexión GPRS para la compañía CLARO Argentina.

Comando	Respuesta
AT+SAPBR=3,1,"CONTYPE","GPRS"	OK
AT+SAPBR=3,1,"APN","igprs.claro.com.ar"	OK
AT+SAPBR=3,1,"USER","clarogprs"	OK
AT+SAPBR=3,1,"PWD","clarogprs999"	OK
AT+SAPBR=1,1	OK

Configuración de una conexión GPRS para la compañía Personal Argentina.

Comando	Respuesta
AT+SAPBR=3,1,"CONTYPE","GPRS"	OK
AT+SAPBR=3,1,"APN"," datos.personal.com"	OK
AT+SAPBR=3,1,"USER","datos"	OK
AT+SAPBR=3,1,"PWD","datos"	OK
AT+SAPBR=1,1	OK

Consulta de la dirección IP del módulo para determinar si la conexión ha sido exitosa.

Comando	Respuesta
AT+SAPBR=2,1	+SAPBR: 1,1,"100.86.220.128" OK

Conexión HTTP utilizando el método POST para enviar parámetros de ubicación a la URL de almacenamiento llamada *guardar.php*.

Comando	Respuesta
AT+HTTPINIT	OK
AT+HTTPPARA="CID",1	OK
AT+HTTPPARA="URL","*URLSERVIDOR*/guardar.php"	OK
AT+HTTPPARA="CONTENT","application/x-www-form-urlencoded"	OK
AT+HTTPDATA=39,10000	DOWNLOAD
Nom=Plaza25&Lat=-29.41308&Lon=-66.85591	OK
AT+HTTPACTION=1	OK +HTTPACTION:1,200,31
AT+HTTPTERM	OK

B.3 Programación del lado del servidor

Programación en PHP que permite almacenar los datos recibidos mediante el método POST en la URL del servidor llamada *guardar.php*. La información de la base de datos no se encuentra especificada.

```
<?php

// Leer la información que llega al servidor
$VAR_Nom      = $_POST['Nom'];
$VAR_Lat      = $_POST['Lat'];
$VAR_Lon      = $_POST['Lon'];

// Datos de la base de datos utilizada
$servername = "URLdelServidordeBaseDatos";
$username    = "UsuariodeBaseDatos";
$password    = "ContraseñadeBaseDatos";
$dbname      = "NombredeBaseDatos";

// Si la información está completa
if($VAR_Nom and $VAR_Lat and $VAR_Lon){

        // Realizar conexión
        $conn = new mysqli($servername, $username, $password, $dbname);
        // Verificar conexión
        if ($conn->connect_error) {
                die("Conexión fallo: " . $conn->connect_error);
        }

        // Almacenar valores en la tabla mapas de la base de datos
        $sql = "INSERT INTO mapas(Nom, Lat, Lon)
```

```
        VALUES('".$VAR_Nom."','".$VAR_Lat."','".$VAR_Lon."')";

        // Verificar que los datos se hayan almacenado
        if ($conn->query($sql) === TRUE) {
                echo "Creado";
        }
        else {
                echo "Error: " . $sql . "<br>" . $conn->error;
        }

        // Cierra conexión
        $conn->close();
}

?>
```

Programación en PHP para generar una página de mapas con la información almacenada.

```
<?php

// Datos de la base de datos utilizada
$servername = "URLdelServidordeBaseDatos";
$username   = "UsuariodeBaseDatos";
$password   = "ContraseñadeBaseDatos";
$dbname     = "NombredeBaseDatos";

// La función echo genera el contenido especificado entre comillas dobles
        echo "
                        <!DOCTYPE html>
                        <html lang=es>
                        <head>
                        <meta charset=\"utf-8\" />
                        <meta name=\"viewport\" content=\"width=device-width, initial-
scale=1.0\">
                        <title>Mapas</title>
                ";

        // Realizar conexión
        $conn = new mysqli($servername, $username, $password, $dbname);
        // Verificar conexión
        if ($conn->connect_error) {
                echo "</head><body>";
                die("Conexión fallo: " . $conn->connect_error);
                echo "</body></html>";
        }
        else {
                // Leer información almacenada en la tabla
                $sql = "SELECT Nom, Lat, Lon FROM mapas";
                $result = $conn->query($sql);
```

```
                echo "
                        <style>
                                #map {
                                height: 100%;
                                }

                                html, body {
                                        height: 100%;
                                        margin: 0;
                                        padding: 0;
                                        }
                        </style>

                        <script>
                        var marcadores = [";

                // Generar el vector de ubicación con la información leída
                if ($result->num_rows > 0) {
                        while($row = $result->fetch_assoc()) {
                                echo "   ['" . $row["Nom"]. "', " . $row["Lat"]. ", " .
$row["Lon"]. "], ";
                        }
                }
                // En caso de que la tabla este vacía, generar un valor por defecto
                else {
                        echo "['NO_DATA', -29.410139, -66.832183]";
                }

                $conn->close();

                // Generar el contenido del script de mapas
                echo "
                                        ];

function initMap()
{
        var map = new google.maps.Map(document.getElementById('map'),
        {
                center: {lat: -29.420833, lng: -66.844860},
                zoom: 14
        });

        var infowindow = new google.maps.InfoWindow();

        for (var i = 0; i < marcadores.length; i++)
        {
                var marker = new google.maps.Marker(
                {
```

```
                        position: new google.maps.LatLng(marcadores[i][1],
marcadores[i][2]),
map: map
                });

                google.maps.event.addListener(marker, 'click', (function(marker, i)
                {
                        return function()
                        {
                        infowindow.setContent(marcadores[i][0]);
                        infowindow.open(map, marker);
                        }
                        })(marker, i));
                }
                marker.setMap(map);
        }
                                </script>

                                <script
src=\"https://maps.googleapis.com/maps/api/js?key=GOOGLE_API_CLAVE&callback=initMa
p\" async defer>
                                </script>

                                </head>

                                <body>
                                        <div id=\"map\"></div>
                                </body>
                                </html>
                        ";
        }
?>
```