Pontificia Universidad Católica Argentina
"SANTA MARIA DE LOS BUENOS AIRES"
Facultad de Ciencias Fisicomatemáticas e Ingeniería

TRABAJO FINAL

Seguimiento y estimación de tiempo de llegada para transporte público

Alumno: Juan Sebastián Gregori
Tutor: Juan Alberto Poey
Fecha de presentación: 19JUL11

INDICE

INTRODUCCIÓN

Este trabajo tiene como objetivo bosquejar el desarrollo de un sistema para hacer el seguimiento del transporte público, específicamente de los colectivos, y poder brindarles información a los usuarios en tiempo real, como es el caso de la estimación del tiempo de llegada del próximo colectivo a la estación.

En la Ciudad de Buenos Aires, como en otras grandes ciudades en donde se hace un uso intensivo del transporte público sería de utilidad poder brindarles a los usuarios mayor información ya sea a través de Internet, en las mismas paradas de colectivos o incluso utilizando los teléfonos móviles. Este tipo de sistema también le es de gran utilidad a los gobiernos ya que la recopilación de estos datos se puede utilizar para tomar mejores decisiones en cuanto al control del tráfico, a la elección de los trayectos utilizados o controlar el límite de velocidad.

En este trabajo se propone desarrollar un sistema que identifique la posición en la que se encuentran los colectivos y la envié a un control central para luego procesar la información y reenviarla a las estaciones para mostrar el tiempo estimado del próximo colectivo y mostrar mediante un display de leds la estación en la que se encuentra.

Con el objetivo de desarrollar el sistema se secciono el trabajo en etapas. En la primera instancia se plantean el alcance del sistema y se lo pasa a diseñar. Luego se comienza con la implementación. En esta sección se divide el trabajo en dos partes, la primera trata el dispositivo que se ubicará en los colectivos para determinar la posición del colectivo y enviarla a la central. En la segunda parte se desarrolla la central. Por último se hacen las pruebas para mostrar cómo funciona el sistema, y se analizan los resultados.

.

RESUMEN

El siguiente trabajo expone el diseño e implementación de un sistema para realizar el seguimiento del transporte público y poder brindarles a los usuarios (pasajeros) información sobre el servicio. En un principio se planteo un sistema muy ambicioso, que luego se redujo debido a la dificultad que planteaba. Entre las modificaciones más grandes que se hicieron a este primer diseño, fue el de utilizar el servicio SMS para enviar información al de GPRS. Esta modificación no solo se debió a la dificultad que plantea el uso de GPRS, sino que también se presentaban otros inconvenientes como el direccionamiento IP. Como la red celular utiliza direcciones privadas hay utilizar un servicio aparte para poder comunicarse desde internet a dispositivos que estén en esta red.

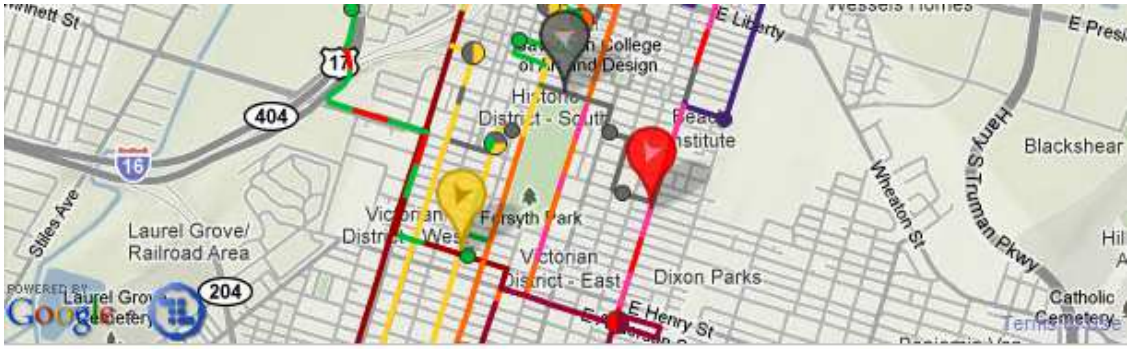
Luego del diseño del sistema, se pasó a la implementación. En esta etapa se selecciono material disponible en el mercado y se le hicieron algunas modificaciones o agregados para satisfacer las necesidades tanto de la UT (Unidad de Transmisión) como de la UR (Unidad de Recepción). Durante la implementación se presentaron algunos inconvenientes, entre ellos la implementación de una interface SPI y el uso de una batería para la alimentación de la UT (Unidad de Transmisión). En el caso de la interface SPI, fue problemático el lograr que los dos dispositivos que se pretendía comunicar lo hicieran. Los problemas estaban relacionados a la falta de una tierra en común y a la sincronización. En el caso de la batería, esta no suministraba la potencia necesaria ante los picos de corriente que genera el módulo GSM (SIM900).

Luego de la implementación, se pasó a hacer las pruebas para corroborar el correcto comportamiento del sistema diseñado. En esta oportunidad se determinó que el intervalo de tiempo utilizado para el envió de información de la UT a la UR era muy grande, y que podía generar que perdida de información en el display. Esta particularidad, se debe a la simplicidad que tiene el sistema, la solución más fácil es reducir el tiempo del intervalo para enviar información, aunque también se le podría dar más inteligencia a la unidad de recepción, y que haga la estimación correspondiente con la información que tiene, o sea, a partir de la velocidad promedio determinar cuánto avanza, y mostrar en qué parada se encuentra. Con cada recepción de nueva información se debería volver a hacer la estimación.

Por último, a modo de detalle, se decidió hacer una gráfica común tanto para la Unidad de Transmisión como para la Unidad de Recepción.

ANTECEDENTES Y TENDENCIA

El seguimiento del trasporte público es un tema que se viene desarrollando hace tiempo en las grandes ciudades de los países desarrollados. En Estados Unidos hay varios sistemas implementados para hacer el seguimiento del trasporte público, como es el caso del servicio TransLoc – Transit Visualization System[1] que permite visualizar la posición de los colectivos y brindar información como la estimación del próximo colectivo accediendo tanto por internet, utilizando una app para teléfonos móviles o incluso en las paradas. Este servicio está enfocado específicamente al transporte de universidades y al de los campus de instituciones.



En el estado de Massachusetts (EEUU), las autoridades de transporte[2] permitieron el acceso a la información del transporte público (colectivos, trenes, subtes y barcos), lo que favoreció a que varios desarrolladores de software implementaran sus propias aplicaciones para teléfonos móviles con el objetivo de visualizar esta información. La mayoría de las app también brindan alertas sobre el servicio para indicar por ejemplo interrupciones en el servicio, cambio de rutas o avisos de las líneas que un pasajero suele utilizar.

[1] http://www.transloc.com/
[2] http://www.mbta.com/rider_tools/apps/

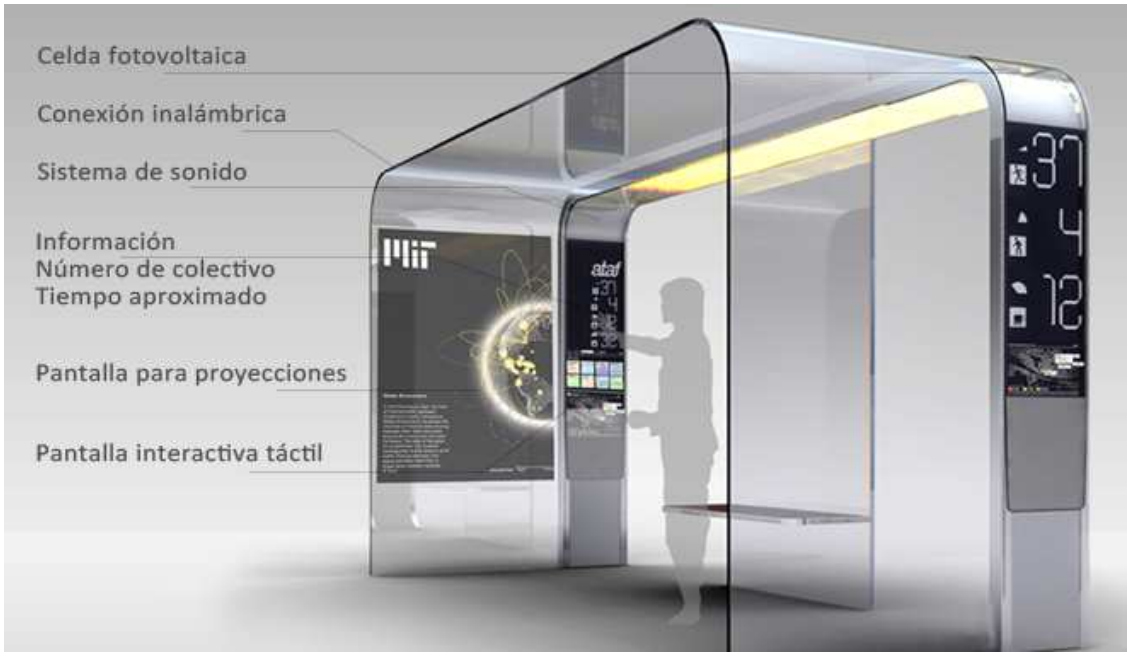
Nextime

Nextime is a real-time transit app, featuring the unique TransitTrack feature, which monitors your bus—in the background—notifying you when you need to leave.

Made By : ReeD

Platform : iOS (iPhone)

Otro desarrollo interesante es el que realizó el Instituto de Tecnología de Massachusetts (MIT) identificado como EyeStop con el objetivo de explorar la próxima generación inteligente de mobiliario urbano; su propósito es enriquecer la ciudad con el arte del censado utilizando las nuevas tecnologías y brindar servicios interactivos, información de la comunidad y entretenimiento. El proyecto está parcialmente cubierto con sensores sensibles al tacto y pantallas.



En Argentina el 17 de Agosto de 2010 se publicó una nota[3] en el diario La Nación en la que se enunciaba que la Agencia Nacional de Seguridad Vial (ANSV), la Comisión Nacional de Regulación del Transporte (CNRT) y de Transporte de la Ciudad de Buenos Aires habían resuelto que los colectivos deberían ser monitoreados haciendo uso de la tecnología GPS. El encuentro con los diferentes entes de gobierno estuvo influenciado por la escalada de

[3] http://www.lanacion.com.ar/nota.asp?nota_id=1295475

accidentes en el que fueron protagonistas esos vehículos en la Ciudad de Buenos Aires en ese periodo. Al tiempo se publicó otra nota[4] el 8 de Octubre de 2010 donde se informaba que el gobierno había dispuesto la utilización de un sistema de posicionamiento geográfico para controlar el funcionamiento de trenes y subtes del área metropolitana, con el objetivo de hacer más eficiente el servicio.

El 31 de Mayo de 2011[5] el gobierno de la Ciudad de Buenos Aires inauguró el MetroBus que está compuesto por líneas existentes que tenían en su trayecto la Av. Juan B. Justo, y se incorporó a los colectivos GPSs, para poder brindar a los pasajeros en las estaciones el tiempo estimado de llegada del próximo colectivo[6].

Se puede visualizar en Argentina, al menos en Buenos Aires, el gran interés que hay en recopilar información de posicionamiento del transporte público. Es probable que en los próximos años esta información se ponga a disposición de los usuarios para que la puedan aprovechar, como en el caso del MetroBus, y no solo sirva como un método para controlar el sistema de transporte.

A futuro pareciera que la tendencia en el censado y recopilación de información va a involucrar al mismo usuario. Hoy en día se hace evidente los beneficios de la colaboración, un ejemplo claro es wikipedia que es una fuente de conocimientos de todo tipo realizada por los mismos usuarios.

Los teléfonos móviles actuales ya permiten reconocer la posición geográfica y otros datos que podrían ser recopilados y procesados para obtener o incluso analizar información. Un ejemplo que demuestra está tendencia es la aplicación Roadify[7] para iPhone, en la cual las personas en la calle brindan y obtienen actualizaciones que les ayuda a viajar más fácilmente. Rodify ayuda a encontrar estacionamiento, información de los colectivos y trenes, a través de proveer acceso simple y en tiempo real a información de otros viajeros y de fuentes de datos de transito como el departamento de transporte de New York, Google Transit, etc. Roadify mantiene una base de datos automatizada de plazas de aparcamiento, autobuses y horarios de los trenes que se puede acceder y actualizar por cualquier teléfono que pueda enviar un mensaje de texto.

[4] http://www.lanacion.com.ar/nota.asp?nota_id=1312945
[5] http://www.lanacion.com.ar/1377682-comenzo-a-funcionar-el-metrobus-sobre-avenida-juan-b-justo
[6] http://movilidad.buenosaires.gob.ar/metrobus/%C2%BFpor-que-metrobus/
[7] http://www.roadify.com/index.php

DISEÑO DEL SISTEMA

Para poder obtener información acerca del transporte público, es necesario algún tipo de dispositivo que tenga la capacidad de determinar la posición de cada colectivo y transferirla a un control central, donde sería almacenada y procesada. Para ello se propone la utilización de una placa con módulo GPS (del ingles Global Positioning System), y con conectividad a la red celular. Debido a la amplia cobertura que tiene la red celular, esta es óptima para establecer las conexiones entre las unidades y el control central.
De los servicios prestados por las compañías de telefonía móvil, GPRS (del ingles General Packet Radio Service) sería la mejor opción a utilizar para transferir información, ya que su costo es menor que si se utilizara el servicio de SMS (del ingles Short Message Service).
Recopilada la información, se podría poner a disposición de los usuarios ya sea a través de Internet, haciendo consultas por medio de mensajes SMS o incluso mostrando la información en las mismas paradas de colectivos. A continuación se expone un diagrama general del sistema.

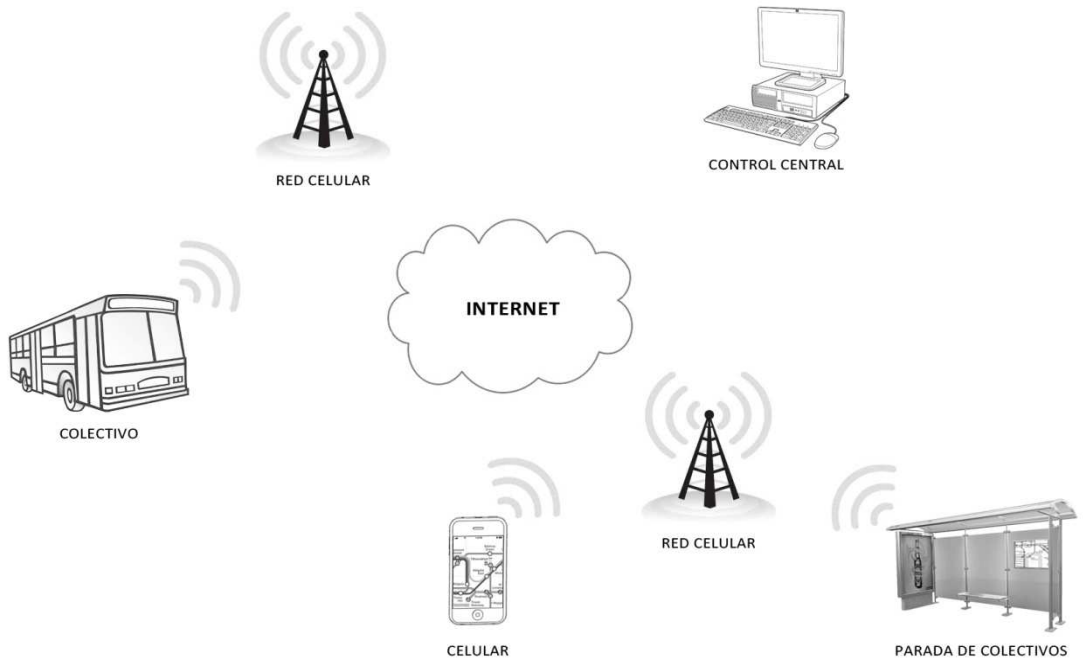


Diagrama general del sistema

En cuanto a los datos transferidos desde cada unidad al control central estos deberían ser: identificador de colectivo, línea, latitud, velocidad y distancia recorrida.
Este sistema se podría integrar al Sistema Único de Boleto Electrónico (SUBE), y así llevar también un registro del número de pasajeros en cada colectivo, e incluso el nivel de ocupación. A partir de estos datos también se podría llegar a obtener el nivel de tráfico en los recorridos de las líneas de colectivos en función del tiempo que emplean en llegar de una parada a otra.

COLECTIVOS

Como ya se expuso anteriormente los colectivos deben estar equipados con un dispositivo capaz de determinar la posición geográfica en la que se encuentran y transmitirla al control central. Para cumplir con este requerimiento se hace necesario utilizar una placa que posea un módulo GPS y conectividad con la red celular. Con tal fin se optó por elegir la placa TrackMe producida por mcelectronics[8] que además de estar equipada con los módulos GPS y GSM, incluye microSD, conexión USB y cargador de batería, que pueden ser útiles en el desarrollo del dispositivo. Más adelante se pasará a explicar cada sección de la placa.

CONTROL CENTRAL

El control central que contaría en un servidor conectado a Internet tendrá como objetivo recibir y almacenar los datos enviados desde cada colectivo, para luego de procesada esta información, ponerla a disposición de los usuarios.
La información recopilada también puede ser utilizada para controlar si los conductores han excedido el límite de velocidad, o incluso para determinar el nivel de congestión de tráfico existente en el trayecto analizando los tiempos empleados de una parada a otra.

037A03	37	09/05/11	14:33	-34.601542	-58.417852	55.5	6280	60
ID	LINEA	FECHA	HORA	LATITUD	LONGITUD	VELOCIDAD	DISTANCIA RECORRIDA	OCUPACIÓN

Ejemplo de datos registrados en el control central

PARADAS DE COLECTIVOS

A modo de brindarle más información a los usuarios se podría equipar las paradas de colectivos con un mapa mostrando el recorrido de cada línea e indicado la ubicación en la que se encuentran los colectivos. También se podría indicar el tiempo estimado de llegada de los próximos colectivos.
A modo de ejemplo, en este trabajo se diseñó un mapa indicando el recorrido de la línea de colectivos número 37, y se colocó un LED por cada estación. También se agregó un display numérico para indicar el tiempo estimado del próximo colectivo. Más adelantes se expondrá el circuito implementado para controlar los LEDs y el display numérico.

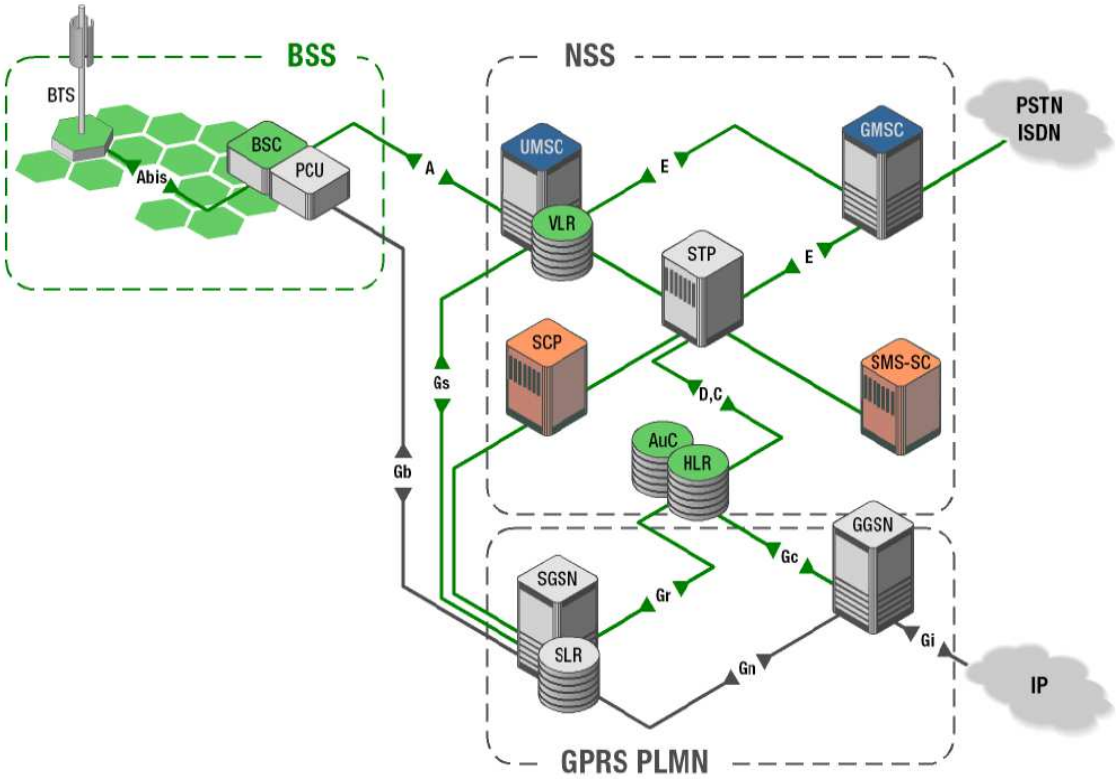
[8] http://www.mcelectronics.com.ar/

Mapa con el recorrido de la línea 37

DISPOSITIVOS MÓVILES

Con los dispositivos móviles de hoy, se podría ingresar a Internet para consultar información sobre el transporte público, como podría ser el tiempo estimado de llegada del próximo colectivo, su nivel de ocupación, o consultar cómo llegar de un lugar a otro que es algo que ya está disponible.
En el caso de no poseer Internet, se podría hacer uso del servicio de SMS para brindar esta información, aunque esta sería más reducida.

PANORÁMA DE LA RED CELULAR



Base Station Subsystem (BSS)

El BSS es responsable de la administración de la red inalámbrica, y es controlado por un MSC. Usualmente, un MSC contiene varios BSSs. Un BSS en si mismo puede cubrir un área geográfica considerablemente grande de varias celdas (una celda se refiere a un área cubierta por uno o más recursos de frecuencia). El BSS contiene de los siguientes elementos:

- BSC (del ingles Base Station Controller)
- BTS (del ingles Base Transceiver Station)
- TC (Transcoder)

Algunas de las tareas más importantes del BSS se listan a continuación:

- Control del enlace de radio: en la red GSM, el BSS tiene como tareas de red el aprovechamiento de los recursos de red, lo que implica la asignación de canales de radio y la calidad de las conexiones
- Sincronización: el BSS utiliza sincronización jerárquica, lo que significa que el MSC se sincroniza con el BSC, y el BSC pasa a sincronizar los BTSs asociados con el particular BSC. Dentro del BSS, la sincronización es controlada por el BSC. La sincronización es tema crítico en la red GSM debido a la naturaleza de la información que se transfiere. Si la sincronización no funciona correctamente, las llamadas puede ser interrumpidas, o la calidad puede ser insuficiente. En última instancia, puede que se haga imposible establecer la llamada.
- Aire e interface de señalización A: con el objetivo de establecer una llamada, el MS (del ingles Mobile Station) debe establecer una conexión a través del BSS.
- Establecimiento de la conexión entre el MS y el NSS[9]: el BSS se encuentra entre dos interfaces, el aire y la interface A. El MS debe establecer una conexión a través de estas dos interfaces antes de poder establecer una llamada. Generalmente, esta conexiónpuede ser tanto una conexión de señalización o una conexión de tráfico (voz, datos).
- Gestión de de movilidad y transcodificación de audio: la gestión de movilidad cubre diferentes casos de handovers.

Base Transceiver Station (BTS)

El BTS es elemento de red responsable de mantener las interfaces por aire y minimizar los problemas de transmisión (la interface por aire es muy sensible a perturbaciones). Esta tarea se logra con la ayuda de 120 parámetros. Estos parámetros definen exactamente qué tipo de BTS es en cuestión y cómo una estación móvil (MS) ve la red mientras se desplata por el área del BTS.

[9] El NSS (del ingles Network Switching Subsystem) se explica más adelante.

Los parámetros del BTS operan los siguientes principales ítems: qué tipo de handovers (dónde y cuándo), organización de paginación, control de la potencia de radio e identificación del BTS. El BTS tiene varias importantes tareas, algunas de las cuelas son presentadas a continuación:

- Señalización en la interface aérea: muchas señalizaciones de llamadas y de no-llamadas deben ser realizadas en orden por el sistema para funcionar. Un ejemplo es cuando un MS es encendido por primera vez, este necesita enviar y recibir una gran cantidad de información con la red (más precisamente con el VLR[10]) antes de poder comenzar a recibir e iniciar llamadas telefónicas. Otro ejemplo es la señalización requerida para configurar tanto la estación que origina la llamada como la que la termina. Una tercera señalización es muy importante en redes móviles es la necesidad de informar a un MS cuando un handover está por ser realizado (y después cuando un MS envía un mensaje en la dirección uplink confirmándole a la red que el handover se completó).

- Cifrado: tanto el BTS como el MS deben ser capaces de cifrar y descifrar información con el objeto de proteger la información transmitida por la interface aérea.

- Procesamiento de audio: se refiere a todas las funciones que el BTS realiza con objeto de garantizar una conexión sin errores entre el MS y el BTS. Esto incluye tareas como codificación de audio (digital a analógico), codificación del canal (para la protección de errores), interpolación (para permitir transmisiones seguras), y formato del burst (agrega información a la codificación de audio/data con el fin de lograr una transmisión segura y organizada)

Base Station Controller (BSC)

El BSC es el elemento central de la red en el BSS y controla la red de radio. Tiene varias importantes tareas, de las cuales algunas se presentan a continuación:
Establecimiento de conexione entre los MS y el NSS: todas las llamadas al y desde el MS son establecidas a través del grupo de switches del BSC (GSWB).
Recopilación de datos estadísticos: la información de los BTSs, Transcoders, y BSCs es recolectada en el BSC y retransmitida por medio de la DCN (del ingles Data Communications Network) al NMS (del ingles Network Management Subsystem), donde son procesados y presentados, permitiendo obtener el estado y la calidad de la red.
Soporte de señalización: como la interface aérea utiliza un protocolo diferente a la interface A que conecta al MSC con el BSC, este debe llegar acabo la conversión entre protocolos y señalización. El BSC también habilita la conexión de señalización virtual entre el MSC/VLR y los MS.

Packet Control Unit (PCU)

El PCU es un elemento que se ha incorporado al estándar de GSM. Lleva acabo algunas de las tareas de procesamiento del BSC, pero para paquetes de datos. La asignación de canales entre

[10] El VLR (del ingles Visitor Location Register) se explica más adelante.

voz y datos es contralada por BSC, pero una vez que un canal es asignado al PCU, este toma el control total sobre el canal.

Network Switching Subsystem (NSS)

El NSS contiene los elementos de red MSC, VLR, HLR, AC y EIR. Las funciones principales del NSS son:

- Control de llamada: permite la identificación del subscriptor, el establecimiento de la llamada, y cierra las conexiones después de que la conversación finalizó.
- Cobranza: recoge información para cobrar el uso del servicio (el número que inició la llamada y el de destino, el tiempo y el tipo de transacción, etc.) y lo transfiere al centro de facturación.
- Gestión de movilidad: mantiene información de la localización del los subscriptores.
- Señalización: esto aplica a las interfaces con el BSS y la PSTN.

Manejo de la información del subscriptor: esta función hace referencia a la información permanente que se almacena en el HLR y a la información relevante que se almacena temporalmente en el VLR.

Mobile Switching Center (MSC)

Un gran número de BSCs están conectados al MSC a través de una interface A. El MSC es muy similar a una central telefónica digital regular, y puede ser accedido por redes externas de la misma forma, a través de un GMSC. El MSC es responsable de controlar las llamadas en la red móvil. Identifica el origen y el destino de una llamada (estación móvil o teléfono fijo), como también el tipo de llamada. Es responsable de varias tareas importantes como puede ser:

- Control de llamada: el MSC identifica el tipo de llamada, el destino, y el origen de la llamada. También configurar, supervisa y termina las conexiones.
- Inicialización de paging: paging es el proceso de localizar a una estación móvil en particular en el caso de que la llamada termine (una llamada a una estación móvil).
- Recopilación de datos de facturación

Gateway Mobile Switching Center (GMSC)

El GMSC es la puerta de enlace (Gateway) para comunicarse con la PSTN (del ingles Public Switched Telephone Network) o la ISDN (del ingles Integrated Services Digital Network).

Visitor Location Register (VLR)

EL VLR fue ideado para que el HLR no fuera sobrecargado con consultas de datos sobre los subscriptores. Tal como el HLR, un VLR contiene los datos de los subscriptores, pero solo parte de los datos del HLR y solo mientras un subscriptor en particular se encuentra en el área de la cual el VLR se encarga. Cuando el subscriptor sale del área del VLR, el HLR peticiona que se remueva la información relacionada a ese subscriptor del VLR. El área geográfica del VLR

consiste en el área cubierta por aquellos BTSs que están relacionados con el MCSs para el cual el VLR proporciona servicios. Entre la información que almacena se encuentra: número de identificación del abonado, información de seguridad para autentificar la SIM, servicios a los que está subscripto y dirección del HLR al que pertenece.

Home Location Register (HLR)

El HLR mantiene un registro permanente de los abonados, como la identidad del abonado y los servicios a los que está subscripto. Además de los datos fijos, el HLR también realiza un seguimiento de la ubicación actual de cada cliente. Un HLR puede ser considerado como una gran base de datos que administra los datos de literalmente cientos de miles de subscriptores. Cada PLMN requiere al menos un HLR.

Authentication Center (AuC)

El AuC siempre es implementado como parte integral del HLR. La única función principal asignada al AuC es calcular y proveer tres autentificaciones, que son, indicación de respuesta (SRES), número aleatorio (RAND) y Kc. Estos parámetros son transferidos al HLR y luego este los reenvía al VLR, que los utiliza como parámetros de entrada para autentificación y cifrado.

Signal Transfer Point (STP)

Un STP es un router que transmite mensajes SS7 entre SPs (del ingles Signaling Points) a través de enlácese de señalización. La función principal de un STP es identificar el mejor camino para comunicar dos SPs. En algunas aplicaciones los SPs se encuentran directamente conectados a través de un enlace de señalización. Normalmente está implementación tiene como propósito mejorar la robustez o performance entre SPs críticos. En estos casos no es necesario utilizar STPs.

Service Control Point (SCP)

Un SCP es un componente estándar de una red inteligente (IN) en un sistema de telefonía, que es utilizado para llevar el control de servicios. Estoy pueden ser implementados utilizando tecnología SS7, Sigtran o SIP. Los SCPs proveen tanto servicios tradicionales en una IN como también servicios de próxima generación. Permite la operación exitosa y manejo todo tipo de sesiones y eventos incluido voz, SMS, datos y servicios de valor agregado para todos los tipos de abonados, tanto en la red como en roaming. Puede proveer control en tiempo real para facturación en tiempo real, servicios de números 800, servicios [*], traducción de servicios, portabilidad numérica, verificación de llamadas etc.

Short Message Service – Service Center (SMS-SC)

En un elemento de la red de telefonía móvil cuya función es la de enviar y recibir mensajes SMS. En el momento que un usuario envía un mensaje de texto (SMS) a otro este es trasferido en primera instancia al SMS-SC correspondiente al operador del usuario remitente. El SMS-SC guarda el mensaje y lo entrega a su destinatario cuando este se encuentra en cobertura. Por lo general el SMS-SC, dentro de los cientos de parámetros configurables que posee, dispone de

un tiempo máximo durante el cual el mensaje es guardado, si en ese tiempo el destinatario no es localizado, el mensaje es desestimado.

General Packet Radio Service – Public Land Mobile Network

Se identifica como GPRS-PLMN a la sección de la red que se encarga del tráfico de datos. Está compuesta por el SGSN, el SLR y el GGSN.

Service GPRS Support Node (SGSN)

Un GSSN es responsable de la entrega de paquetes de datos desde y hasta las estaciones móviles (MS) dentro de su área geográfica de servicio. Sus tareas incluyen ruteo de paquetes y transferencia, gestión de movilidad (adjuntar/separar y gestión de ubicación), gestión de enlaces lógicos, autentificación y funciones de facturación. El registro de ubicación del SGSN almacena información acerca de la ubicación (como celda actual, VLR actual) y los perfiles de usuario (como IMSI, dirección utilizada en la red de paquetes de datos) de todos los usuarios GPRS registrados con un SGSN.

Gateway GPRS Support Node – GGSN

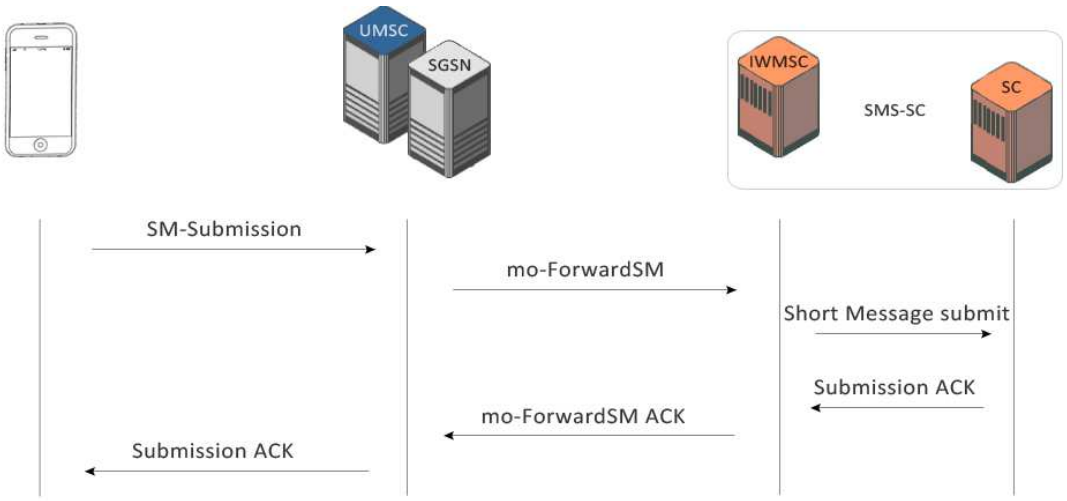
El GGSN es un componente primario de las redes celulares que emplean GPRS. El GGSN actúa como la interface a redes de paquetes IP externas, accediendo a funciones ISP externas tales como routers y acceso remoto a servidores de servicios de usuario dial-in (RADIUS). En términos de la red externa IP, el GGSN rutea las direcciones IP a los abonados de la red GPRS, intercambiado información de ruteo con redes externas.

SGSN Location Register (SLR)

Almacena información relacionada con el abonado similar a la información almacenada en el VLR por el MSC. SGSN/SLR se podría considerar como el análogo al MSC/VLR.

Procesamiento de mensajes SMS

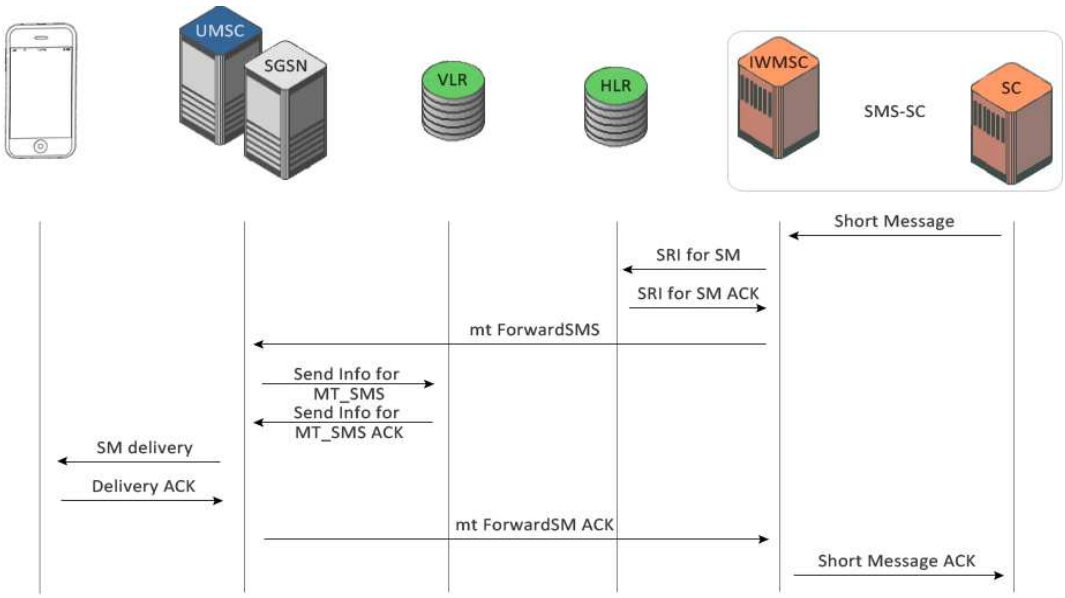
Cuando un abonado envía un mensaje SMS, el teléfono envía el texto del mensaje al UMSC/SGSN. Con el texto del mensaje también se adjunta dirección de destino y la dirección del SMS-SC. El UMSC/SGSN invoca el servicio de paquetes MAP para enviar el mensaje al IWMSC cuya dirección se especificó por el teléfono. El IWMSC al recibir el mensaje MAP mo-ForwardSM, pasa el mensaje al SC correspondiente para ser almacenado. En respuesta se envía un mensaje ACK (del ingles Acknowledge), para confirmar que la entrega se realizó con éxito. La siguiente figura muestra un flujo simplificado de una entrega exitosa para un mensaje SMS.



Cuando el SMS-SC determina que se debe intentar entregar un mensaje a su destinatario, este envía un mensaje SMS-PP APDU que contiene el texto del mensaje, la dirección del teléfono de destino y otros datos al GMSC. El GMSC, al recibir el mensaje, debe determinar la localización del destinatario con el fin de poder entregar correctamente el mensaje a su destinatario. Para ello, el GSMC invoca el servicio de paquetes MAP, el cual envía un mensaje SRI-for-SM (SendRoutingInfo for SM) al HLR del destinatario consultando la ubicación actual. El mensaje SRI-forSM puede ser enviado a un HLR en la misma red en la que se encuentra el SMS-SC o a un HLR en otra PLMN, dependiendo a la red a la cual el destinatario pertenece.

El HLR realiza una búsqueda en su base de datos para localizar al destinatario, y devuelve la información en un mensaje de ACK al GSMC del SMS-SC. La ubicación actual podrá ser la dirección del UMSC en el cual el abonado se encuentra operando, la del SGSN o ambas.

El HLR también puede llegar a responder con un mensaje de error, en el caso de considerar que el destinatario es inalcanzable o no puede recibir mensajes. Obtenida la información de ruteo desde el HLR, el GSMC intentará entregar el mensaje a su destinatario. Esto se realiza invocando el servicio MAP_MT_FORWARD_SHORT_MESSAGE, el cual envía un mensaje MAP mt-ForwardSM a la dirección devuelta por el HLR, sin tener en cuenta si es el UMSC o el SGSN. El UMSC consultará al VLR para obtener la información necesaria con el fin de realizar la entrega. El VLR hará una búsqueda del abonado para obtener el número MSISDN (Mobile Subscriber ISND Number) y se lo transferirá al UMSC, para que este entregue el mensaje. Si la entrega es exitosa el UMSC envía un mensaje de ACK al SMS-SC. El GSMC pasa el resultado de entrega exitosa al SC. En el caso de una entrega exitosa, el mensaje será eliminado del SC. En caso de que la entrega no haya sido exitosa el SMS-SC intentará periódicamente entregar el mensaje; adicionalmente, puede llegar a registrarse con el HLR para recibir una notificación cuando el destinatario se encuentre disponible y realizar la entrega. La siguiente figura muestra un flujo simplificado en la entrega de un mensaje SMS.


UMSC
SGSN
VLR
HLR
IWMSC
SMS-SC
SC
Short Message
SRI for SM
SRI for SM ACK
mt ForwardSMS
Send Info for MT_SMS
Send Info for MT_SMS ACK
SM delivery
Delivery ACK
mt ForwardSM ACK
Short Message ACK

SISTEMA GPS

El Sistema de Posicionamiento Global (GPS) es una utilidad de propiedad estadounidense que proporciona a los usuarios posicionamiento, navegación y servicios de cronometría (PNT). Este sistema consta de tres segmentos: el segmento espacial, segmento de control, y el segmento de usuario. La Fuerza Aérea de EE.UU. desarrolla, mantiene y opera los segmentos espacial y de control.

Funcionamiento

Los satélites del Sistema de Posicionamiento Global transmiten señales a los equipos sobre la tierra. Los receptores GPS reciben pasivamente las señales de los satélites. Los receptores GPS requieren de una vista despejada del cielo, por lo que sólo se utilizan al aire libre y con frecuencia no funcionan bien dentro de las áreas boscosas o cerca de edificios altos. Las operaciones de GPS dependen de una referencia temporal muy precisa, que es proporcionada por los relojes atómicos del Observatorio Naval de los EE.UU.

Cada satélite GPS transmite datos indicando su ubicación y la hora actual. Las operaciones de los satélites GPS se encuentras sincronizadas con el fin de que estas señales se transmitan en el mismo instante. Las señales, que se transmiten a la velocidad de la luz, llegan a un receptor GPS en momentos ligeramente diferentes ya que algunos satélites están más lejos que otros. La distancia a los satélites GPS se puede determinar mediante la estimación del tiempo que se requiere para que la señal llegue al receptor. Cuando el receptor calcula la distancia de al menos cuatro satélites GPS, se puede calcular su posición en tres dimensiones.

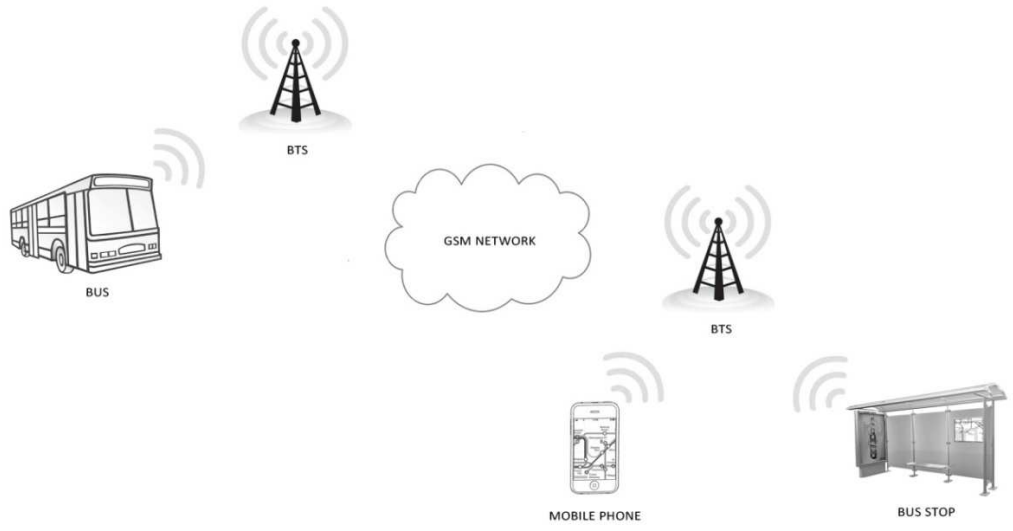
Hay por lo menos 24 satélites operativos en todo momento. Los satélites, controlados por la Fuerza Aérea de los EE.UU., órbitan con un período de 12 horas. En las estaciones de control de la tierra se lleva un seguimiento constante para determinar con precisión la órbita de cada satélite.

Para determinar la posición, un receptor GPS necesita determinar la ubicación de los satélites, información que se incluye en las transmisiones que estos realizan. Al estimar la distancia a un satélite, el receptor también "sabe" que se encuentra en algún lugar de la superficie de una esfera imaginaria centrada en el satélite. A continuación, determina el tamaño de las diversas esferas, una para cada satélite. El receptor se encontrará en el lugar donde estas esferas se cruzan.

La precisión de una determinada posición con el GPS depende del tipo de receptor. La mayoría de los equipos de mano GPS tiene una precisión de 10 a 20 metros. Otros tipos de receptores utilizan un método llamado GPS diferencial (DGPS) para obtener una precisión mucho mayor. DGPS requiere de un receptor adicional fijo en un lugar conocido cercano. Las observaciones realizadas por el receptor fijo se utilizan para corregir posiciones registradas por las unidades itinerantes, produciendo una precisión superior a 1 metro.

ALCANCE DEL TRABAJO

Debido a la complejidad que presenta realizar un sistema de esta magnitud, se optó por hacer una versión reducida. Se omitirá el paso de la información por un servidor a través de internet y en cambio se la enviará directamente a la parada de colectivo, que al recibirla mostrará en un mapa la posición del colectivo utilizando LEDs y con un display numérico el tiempo estimado de llegada del próximo colectivo. Por otro lado en vez de utilizar GPRS se utilizará el servicio de SMS para transferir información de las unidades a las paradas. En principio el trabajo estaba orientado a hacer uso del servicio GPRS, pero debido a la complejidad que tiene, habría que invertir más tiempo para desarrollar el programa que haga uso del servicio. Además la placa que se eligió para hacer de terminal y procesar la información tiene muy pocos recursos de memoria (Program Memory 16K) con lo cual tampoco se podría desarrollar el programa. A continuación se muestra un esquema del sistema reducido.



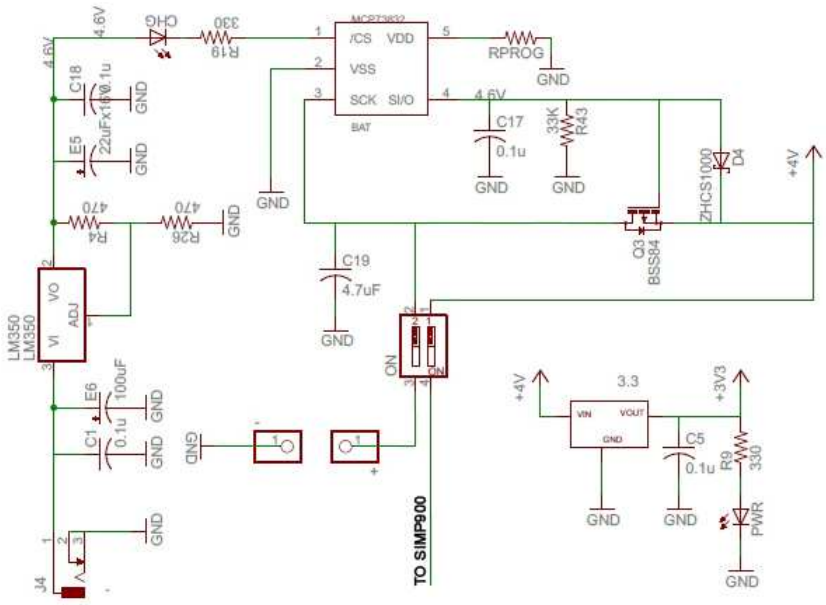
Sistema reducido

UNIDAD DE TRANSMISIÓN

HARDWARE

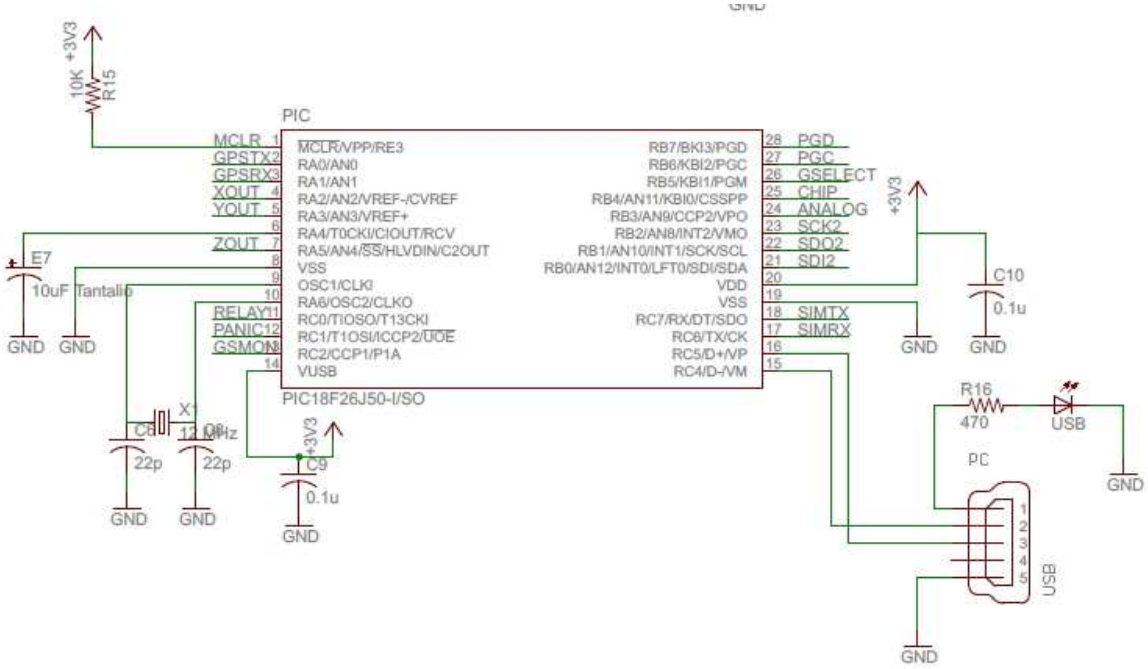
Fuente de alimentación

La tensión de alimentación necesaria se obtiene haciendo uso de un regulador LM350, que alimenta a un controlador lineal (MCP73832) para cargar baterías de Li-Ion en el caso de estar conectada. El controlador disminuye la tensión hasta 4V y alimenta al módulo GSM (SIM900). Para alimentar al PIC18F26J50, se utiliza un regulador de 3.3V que tiene como entrada los 4V de la salida del controlador. La placa puede ser alimentada tanto con corriente continua o con una batería.



PIC

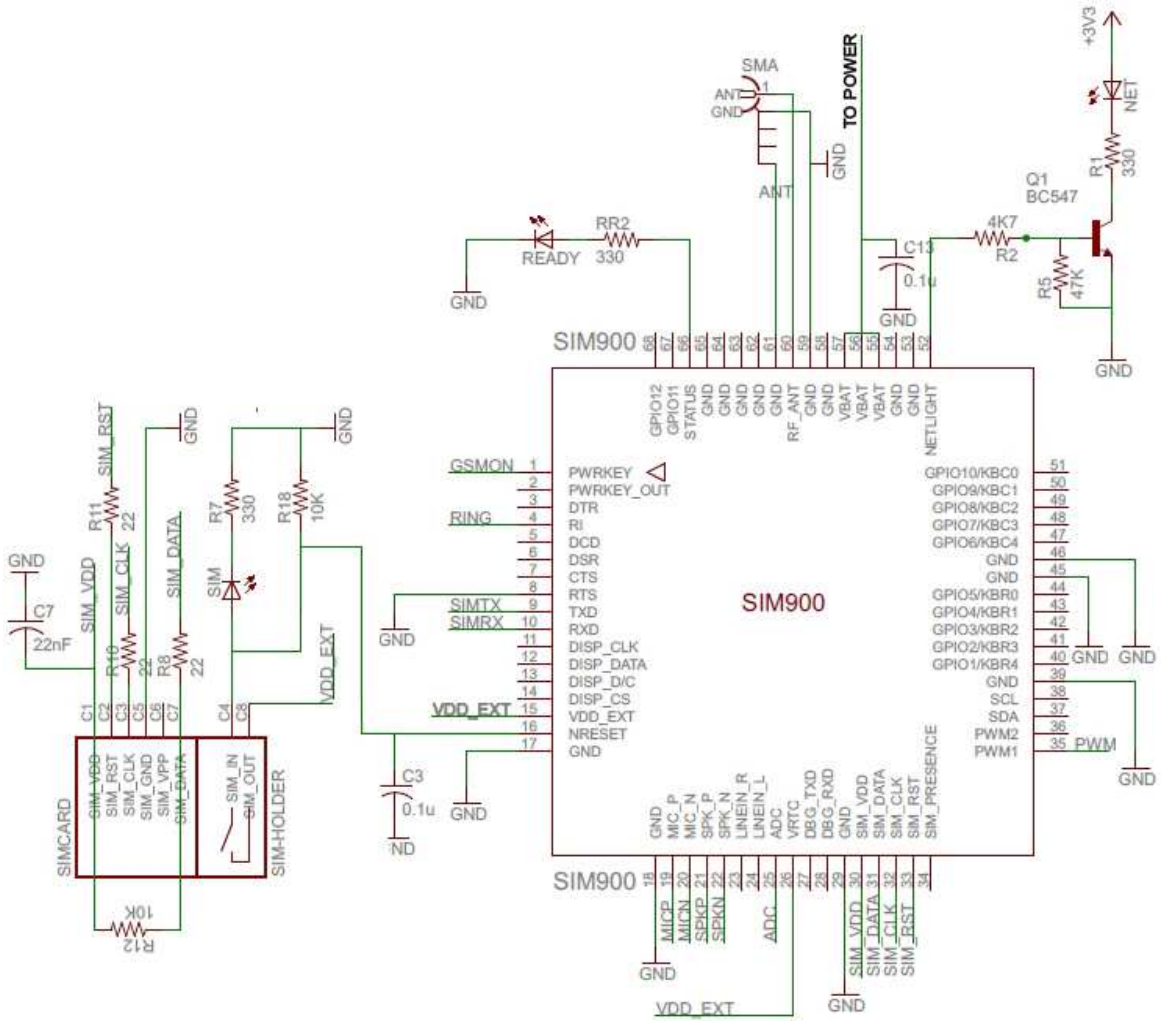
La placa tiene incorporado el PIC18F26J50 en donde se aloja el programa desarrollado para este trabajo. El PIC tiene pines dedicados para comunicarse por medio de la USART con el módulo GSM y con un puerto USB. También tiene pines se utilizan para adquirir la señal de los ejes del acelerómetro que posee la placa.



GSM Y SIM

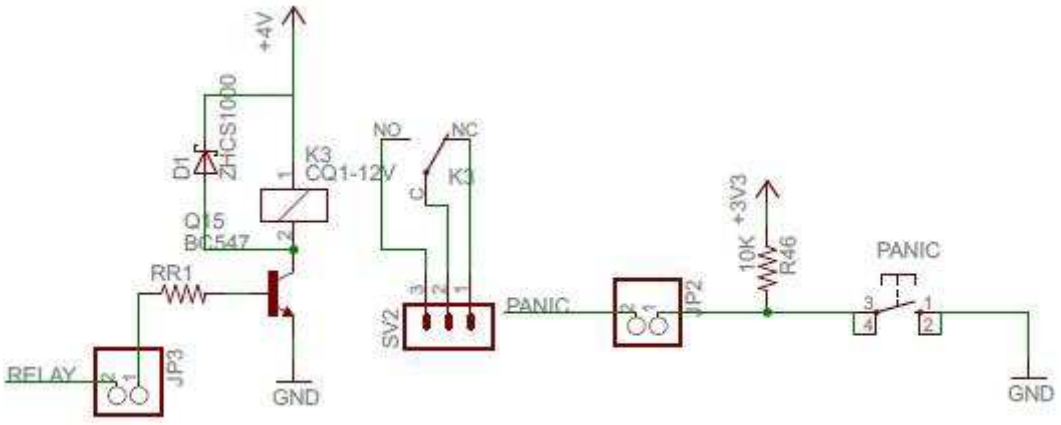
En el esquema se ve la conexión del módulo GSM. Se comunica con la USART[11] del PIC a través de SIMTX y SIMRX. El GSM se alimenta con 4.0 V, que se obtienen a través de un regulador LM350 y del controlador (MCP73832). Es posible alimentar la placa con una batería recargable conectándola en los pines BAT. Para encender el módulo el programa del PIC pone en cero el PIN RC2 (PWRKEY). En este momento se enciende el LED STATUS. Si la tarjeta SIM está en el zócalo el módulo debe registrarse en la red celular. En este caso el LED NET debe titilar cada 3 segundos.

[11] (del ingles Universal Synchronous/Asynchronous Receiver/Transmitter)



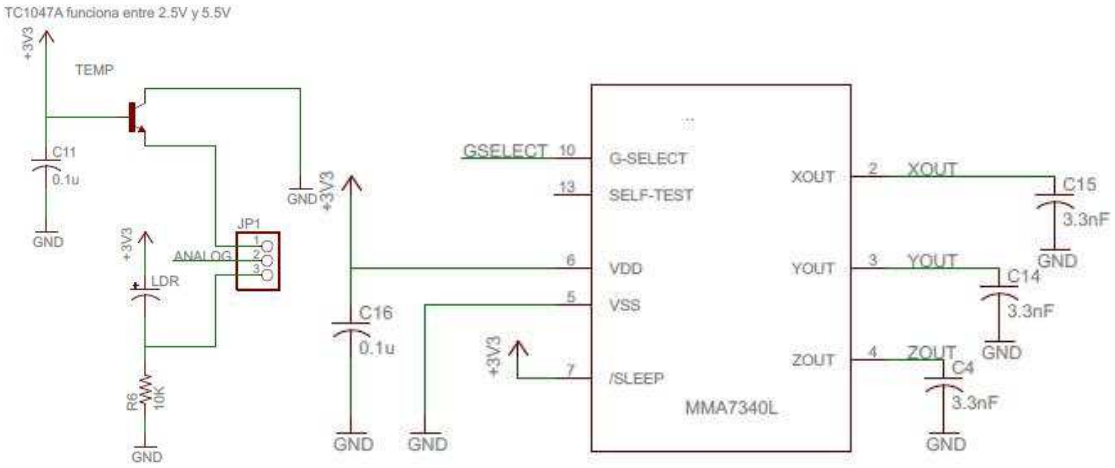
Pulsador y Relay

El pulsador de PANIC entrega un reposo de un uno lógico. También posee un Relé que se puede activar a través del puerto RC0. Los pines normalmente abierto (NA) y normalmente cerrado (NC) están indicados en la serigrafía de la placa. Para este trabajo el Relé no fue utilizado.



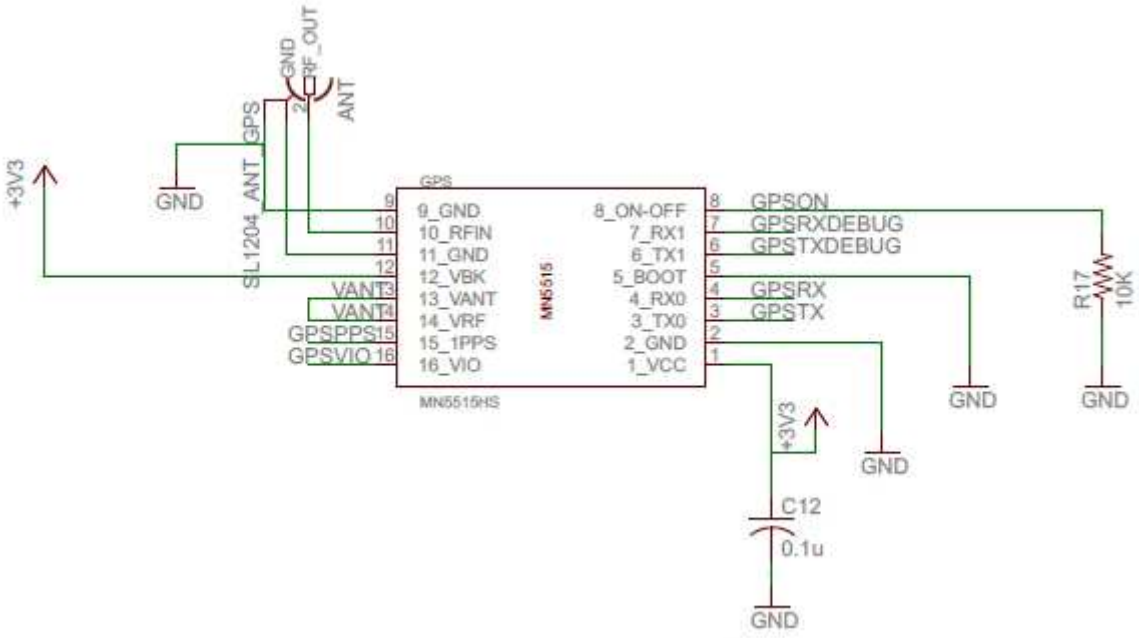
Sensor de temperatura y acelerómetro

La placa incluye un sensor de temperatura que se conecta al PIC a través de AN9. También incluye un acelerómetro al que se accede a través de los puertos AN2 (XOUT), AN3 (YOUT), AN4 (ZOUT) y RB5 (GSELECT) por donde se puede seleccionar la sensibilidad. Lamentablemente por problemas de fabricación es imposible hacer uso del acelerómetro, el cual hubiera sido muy últil para poder calcular la distancia recorrida.



GPS

Uno de los principales componentes que incluye la placa es el GPS (MN5515) que es accesible a través de los pines RA0 y RA1 correspondientes a GPSTX y GPSRX respectivamente. Para poder utilizar estos pines y establecer una conexión a través de la USART, lo pines deben ser remapeados por software.



MicroSD

Por último la placa posee la posibilidad de utilizar una MicroSD. Para acceder a la memoria se debe utilizar el módulo SPI que tiene el PIC y configurarlo como maestro. La conexión se establece por medio de los pines RB0 (SDI2), RB1 (SDO2), RB2 (SCK2) y RB4 (CHIP).

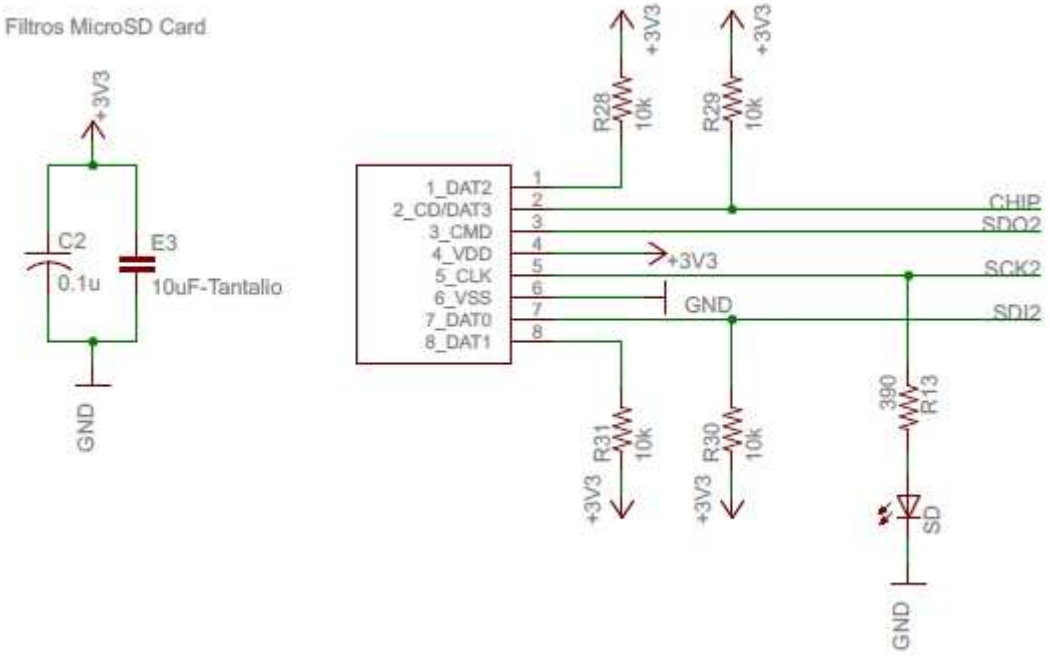
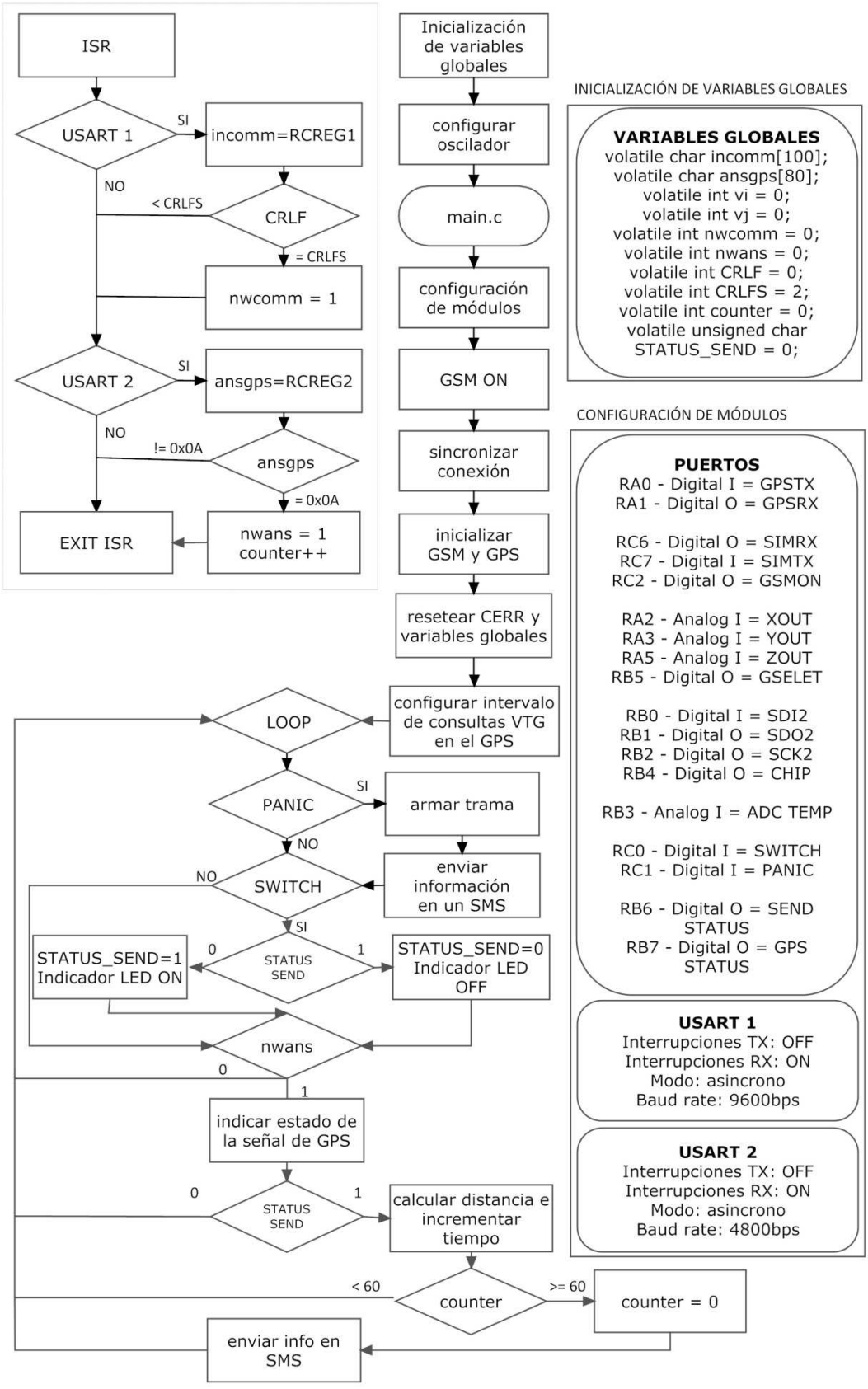


DIAGRAMA DE FLUJO DEL SOFTWARE



TRAMAS

Se definen dos tramas que pueden ser enviados desde la unidad de transmisión ubicada en los colectivos a la central. La primera trama se utiliza para informar la posición, el espacio recorrido y el tiempo empleado desde que se inició el recorrido.

BUS,37,24,LAT,-34.601542,LON,-58.417852,D,27000.00,T,1620
LINEA UNIDAD — LATITUD — LONGITUD — DISTANCIA [m] — TIEMPO [s]

La otra trama definida, se utiliza para informar la posición en el caso de presentarse la necesidad de asistencia mecánica, o de algún otro tipo de inconveniente.

SOS,37,24,LAT,-34.601542,LON,-58.417852
LINEA UNIDAD — LATITUD — LONGITUD

SOFTWARE[12]

```
#include <p18f26j50.h>
#include <usart.h>
#include <timers.h>
#include <delays.h>
#include <string.h>
#include <stdlib.h>
#include <stdio.h>
#include <math.h>

#include "global.h"
#include "gsmf.h"
#include "gpsf.h"
#include "genf.h"
#include "fftoa.h"

#include "gfunc.h"

void low_isr(void);                 // Function declaration - ISR_LOW
void high_isr(void);                // Function declaration - ISR_HIGH
int _user_putc (char c);            // Function declaration - USART2

#pragma config XINST = OFF, STVREN = ON, PLLDIV = 3, WDTEN = OFF                    // CONFIG1L
#pragma config CP0 = OFF, CPUDIV = OSC1                                             // CONFIG1H
#pragma config IESO = ON, FCMEN = OFF, LPT1OSC = OFF, T1DIG = OFF, OSC = HS         // CONFIG2L
#pragma config WDTPS = 32768                                                        // CONFIG2H
#pragma config DSWDTPS = 2048, DSWDTEN = OFF, DSBOREN = OFF                         // CONFIG3L_1
#pragma config RTCOSC = INTOSCREF, DSWDTOSC = INTOSCREF                             // CONFIG3L_2
#pragma config MSSP7B_EN = MSK7, IOL1WAY = OFF                                      // CONFIG3H
#pragma config WPCFG = ON, WPEND = PAGE_0, WPFP = PAGE_1                            // CONFIG4L
#pragma config WPDIS = ON                                                           // CONFIG4H


#define GSMON    PORTCbits.RC2  // GSMON
//#define RELAY  PORTCbits.RC0  // RELAY
#define PANIC    PORTCbits.RC1  // PANIC
#define SWITCH   PORTCbits.RC0  // AUX_SWITCH
#define LED_01   PORTBbits.RB6  // AUX_LED_01 - STATUS ON/OFF CAPTURE
#define LED_02   PORTBbits.RB7  // AUX_LED_02 - STATUS GPS
```

[12] El software explicado solo comprende el main.c. El resto del software está en el proyecto de MPLAB que se encuentra en el CD (4. Sistema/2. UT/2. Software/UT_10JUL11/).

```
#pragma code low_vector=0x18          // LOW ISR
void interrupt_at_low_vector(void)
{
_asm GOTO low_isr _endasm
}
#pragma code                                        /* return to the default code section */

#pragma interruptlow low_isr
void low_isr (void)                   // LOW ISR Function
{

}

#pragma code high_vector=0x08         // HIGH ISR
void interrupt_at_high_vector(void)
{
_asm GOTO high_isr _endasm
}
#pragma code                                        /* return to the default code section */

#pragma interrupt high_isr
void high_isr (void)                  // HIGH ISR Function
{
        if(PIR1bits.RC1IF == 1){                    // ¿Interrumpió la USART2?
                incomm[vi] = RCREG1;                // Leer byte recibido
                if(incomm[vi] == 0x0A){             // Si el byte recibido es <LF> entonces
                        CRLF++;                     // Incrementar el contador de <LF>
                }
                vi++;                               // Incrementar la posición del arreglo
                if(CRLF == CRLFS){                  // ¿Es este <LF> el delimitador de fin?
                        incomm[vi] = '\0';          // Delimitar el fin del arreglo
                        vi = 0;                     // Resetear posición del arreglo
                        CRLF = 0;                   // Resetear el contador de <LF>
                        CRLFS = 2;                  // El segundo <LF> es delimitador de fin
                        nwcomm = 1;                 // Nuevo comando
                }
        }
        if(PIR3bits.RC2IF == 1){                    // ¿Interrumpió la USART2?
                ansgps[vj] = RCREG2;                // Leer byte recibido
                vj++;                               // Incrementar la posición del arreglo
                if(ansgps[vj - 1] == 0x0A){         // Chequear si es el fin del mensaje
                        ansgps[vj] = '\0';          // Delimitar el fin del arreglo
                        vj = 0;                     // Resetear posición del arreglo
                        nwans = 1;                  // Nuevo mensaje
                        counter++;
                }
        }
}


void main (void){

        float distance = 0.0;
                                                    // Definición de variables locales
        float lat1 = 0.0;
        float lon1 = 0.0;
        float time = 0.0;

        char lon[12] = "";
        char lat[12] = "";
        char SMS[60] = "";
        char aux[12] = "";

        LED_01 = 0;                                 // Apagar LED_01
        LED_02 = 0;                                 // Apagar LED_02
        STATUS_SEND = 0;                            // Deshabilitar el envio de informacion

        config_pam();                               // Configuracion de puertos y modulos

        DI();                                       // Deshabilitar interrupciones

        gsm_on();                                   // Encender modulo GSM
        synchronize();                              // Synchronize USART
        initialize_GSM();                           // Configurar modulo GSM
        Delay10KTCYx(2000);                         // Retardo
```

```
initialize_GPS();                       // Configurar modulo GPS

clear_OERR();                           // Clear Overrun Error caused by GPS module
clear_isrv();                           // Resetear variables utilizadas en la rutina de interrupcion

queryrc_VTG();                          // Fijar el tiempo entre consultas VTG en el GPS

EI();

while(1){                               // LOOP PRINCIPAL

        if(PANIC == 0){                         // Se presiono el boton SOS?
                if(gll_GPS(&lat1, &lon1)){      // Consultar posicion geografica
                        ftoa(lat1, lat,6,'f');  // Transformar a caracter
                        ftoa(lon1, lon,6,'f');  // Transformar a caracter
                                                // Generar trama
                        strcpypgm2ram(SMS,(char*)"SOS,152,24,LAT,");
                        strcat(SMS, lat);       // Incluir latitude
                                                // Generar separador
                        strcpypgm2ram(aux,(char*)",LON,");
                        strcat(SMS, aux);       // Incluir separador
                        strcat(SMS, lon);       // Incluir longitud
                                                // Enviar informacion en SMS
                        send_SMS(SMS, "1558181786");
                }
                Delay10KTCYx(1000);             // Retardo
        }

        if(SWITCH == 0){                        // Cambio de estado?
                if(STATUS_SEND == 0){           // Si esta deshabilidado
                        STATUS_SEND = 1;        // Habilitar
                        counter = 0;            // Resetear contador (intervalo entre envios de
                                                info)
                        LED_01 = 1;             // Encender indicador LED
                }
                else{                           // Si esta habilidado
                        STATUS_SEND = 0;        // Deshabilitar
                        LED_01 = 0;             // Apagar indicador LED
                }
                Delay10KTCYx(1000);             // Retardo
        }


        if(nwans == 1){                         // Nuevo comando?
                nwans = 0;                      // eliminar flag de nuevo comando
                                                // Indicar si hay señal de GPS
                if(GPS_STATUS()){LED_02 = 1;}else{LED_02 = 0;}

                if(STATUS_SEND == 1){           // Está habilitado el envio de info?
                                                // Calcular distancia
                        distance = distance + vtg_GPS() * (1.0/3600.0);
                        time++;                 // Incrementar tiempo
                        if(counter >= 60){      // Ya pasó un segundo?

                                counter = 0;

                                if(gll_GPS(&lat1, &lon1)){      // Consultar posición geográfica
                                        ftoa(lat1, lat,6,'f');  // Transformar a caracter
                                        ftoa(lon1, lon,6,'f');  // Transformar a caracter
                                                                // Generar trama
                                        strcpypgm2ram(SMS, (char*)"BUS,37,24,LAT,");
                                        strcat(SMS, lat);       // Agragar la latitude
                                                                // Generar separador
                                        strcpypgm2ram(aux, (char*)",LON,");
                                        strcat(SMS, aux);       // Agregar separador
                                        strcat(SMS, lon);       // Agregar longitude
                                                                // Generar separador
                                        strcpypgm2ram(aux, (char*)",D,");
                                        strcat(SMS, aux);       // Agregar separador
                                                                // Convertir distancia a caracteres
                                        ftoa((distance * 1000.0), aux,2,'f');
                                        strcat(SMS, aux);       // Agregar distancia
                                                                // Generar separador
                                        strcpypgm2ram(aux, (char*)",T,");
                                        strcat(SMS, aux);       // Agregar separador
```

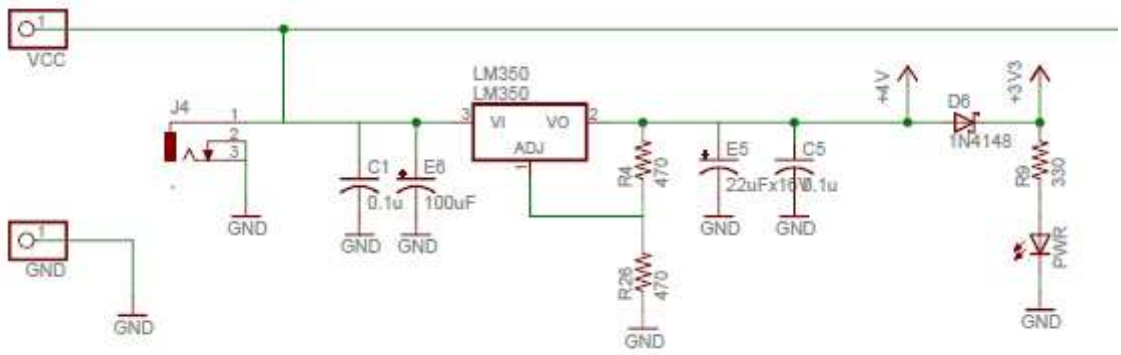
```
                                                    ftoa(time, aux,0,'f'); // Convertir tiempo a catarcteres
                                                    strcat(SMS, aux);    // Agregar tiempo
                                                                         // Enviar información en SMS
                                                    send_SMS(SMS, "1554872672");
                                                }
                                            }
                                        }
                                    }
                                }
                            }
}

int _user_putc (char c){
                                                                         // Definición para la USART2
            while (Busy2USART());
            Write2USART(c);
            return(c);
}
```

UNIDAD DE RECEPCIÓN

HARDWARE

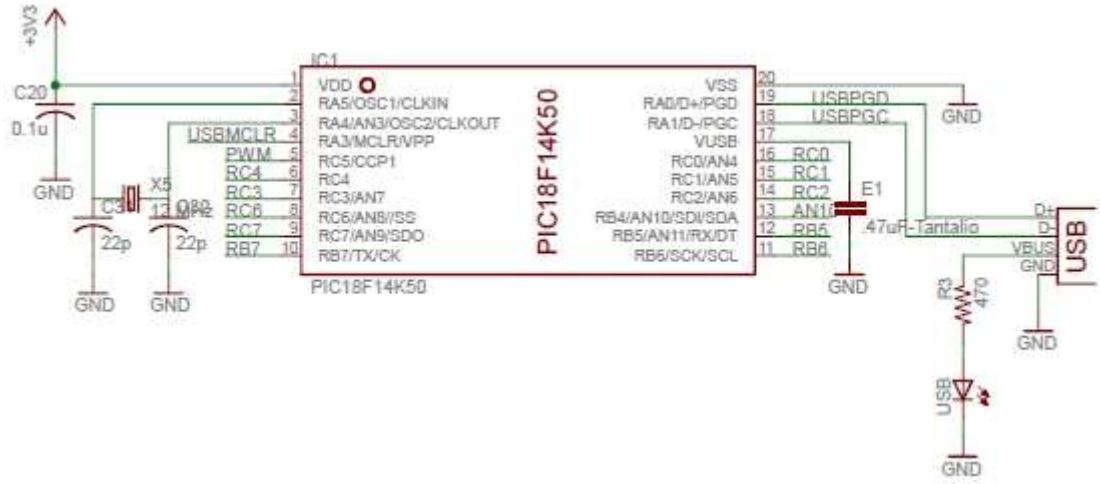
Fuente de alimentación

El módulo GSM se alimenta con 4V a través del regulador LM350. Luego, se llega a los 3.3V del PIC con una caída de 0.7V en D6. Debe alimentar la placa con una tensión entre 6 y 12V con positivo al centro.



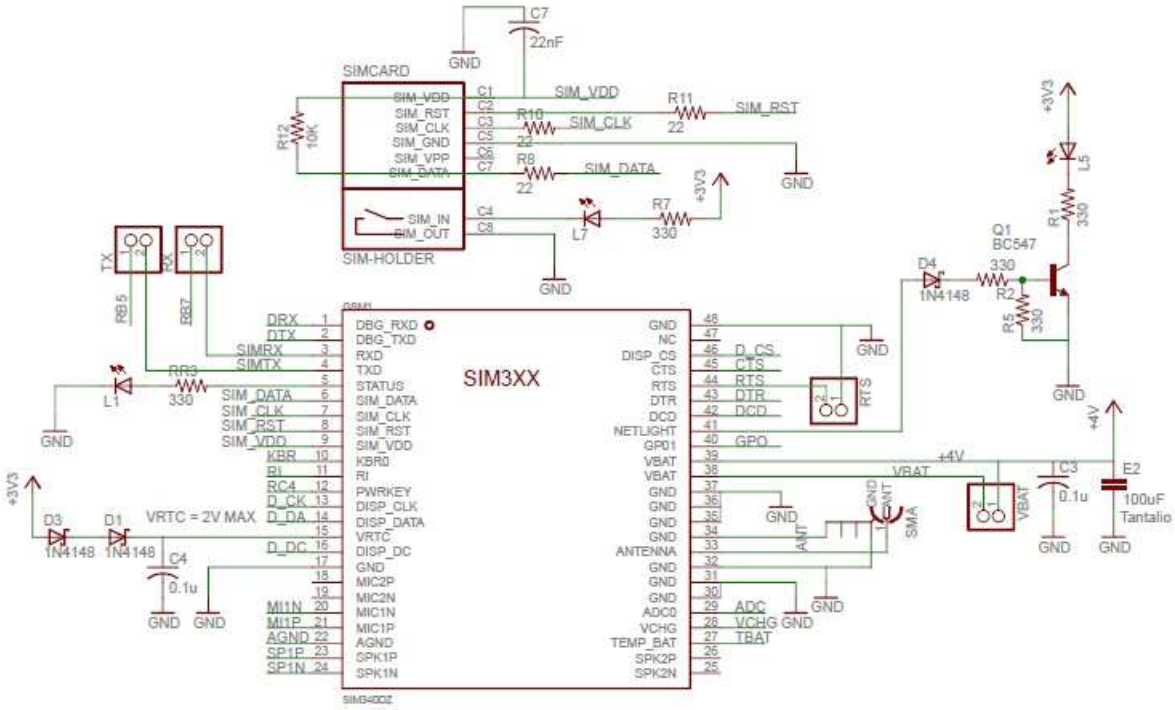
PIC

Se utiliza el PIC18F14K50 en donde se aloja el programa desarrollado. Por medio de los jumpers TX y RX es posible acceder directamente a los pines de transmisión y recepción del GSM.



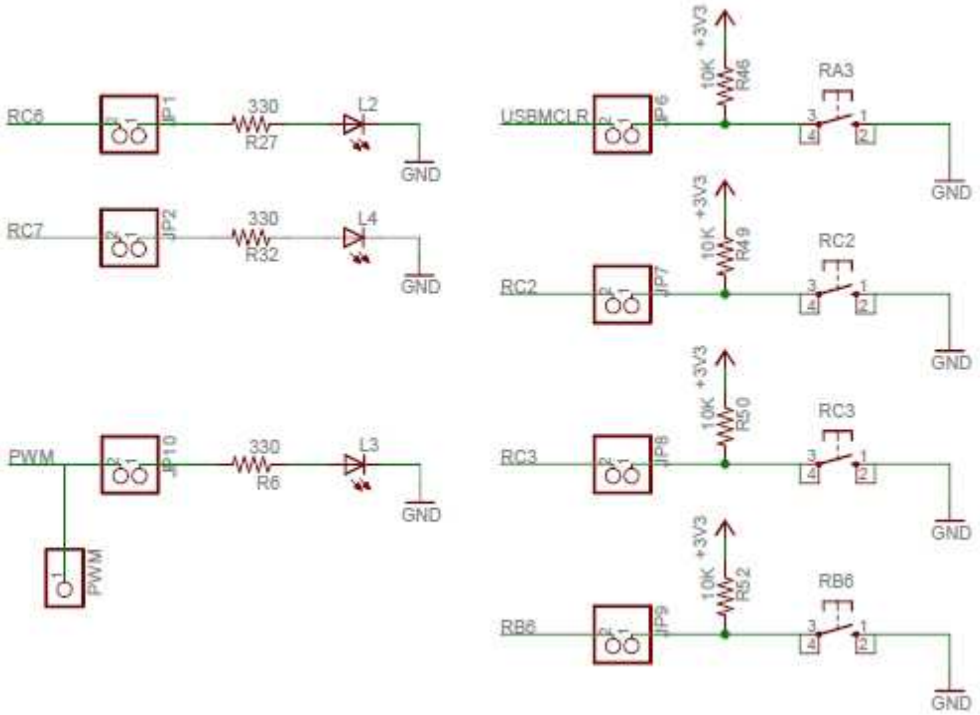
GSM y SIM

En el esquema se ve la conexión del módulo GSM. Se comunica con la USART del PIC a través de SIMTX y SIMRX. El GSM se alimenta con 4.0 V, que se obtienen a través de un regulador LM350. Es posible alimentar la placa con una batería recargable conectándola en VBAT. Para encender el módulo el programa del PIC pone en cero el PIN RC4 (PWRKEY). En este momento se enciende el LED STATUS. Si la tarjeta SIM está en el zócalo el módulo debe registrarse en la red celular. En este caso el LED NET debe titilar cada 3 segundos.



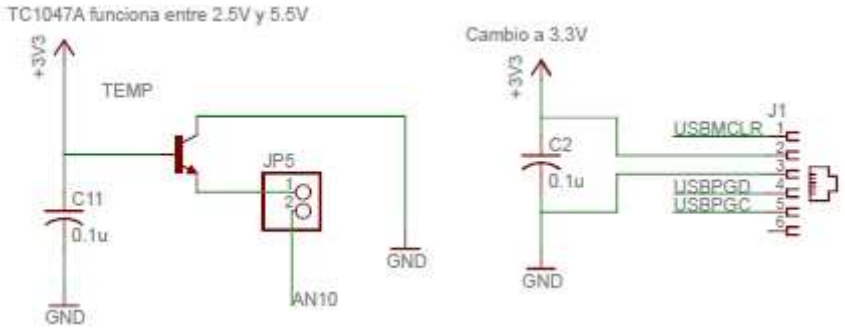
Pulsadores, LEDs y PWM

Los cuatro pulsadores entregan en reposo un ***uno lógico***. Además cuenta con dos LEDs para simular salidas digitales (se activan con 3.3v) y una salida para el PWM. Como fue necesario hacer uso del módulo SPI los jumpers JP1, JP2 y JP9 fueron retirados, y se configuraron los puertos RC6 como entrada SS, el puerto RB6 como entrada para el SCK y el RC7 como salida SDO.



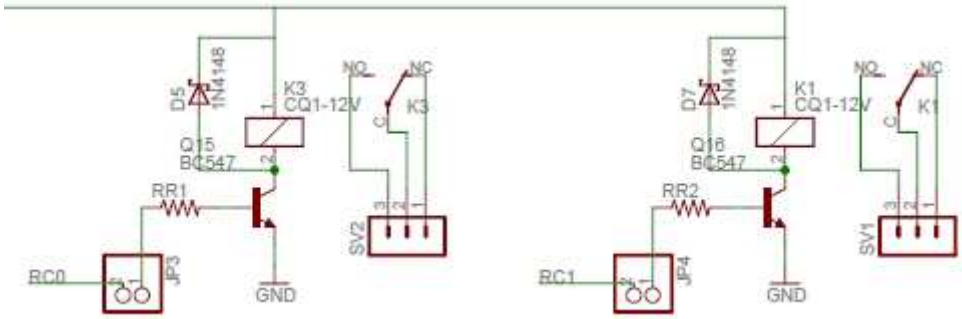
Sensor de temperatura y conector ICSP

La placa incluye un sensor de temperatura que se conecta al PIC a través de AN10. Para programar el micro es necesario un programador ICD3 o compatible, el conector RJ1 se muestra en la figura 2. Como se expuso anteriormente, fue necesario hacer uso del módulo SPI. Como la entrada SDI (del SPI) se encuentra en el puerto RB4, asignado en un principio al puerto analógico AN10 para adquirir la temperatura, se retiro el jumper JP5 y se lo reconfiguro como digital.



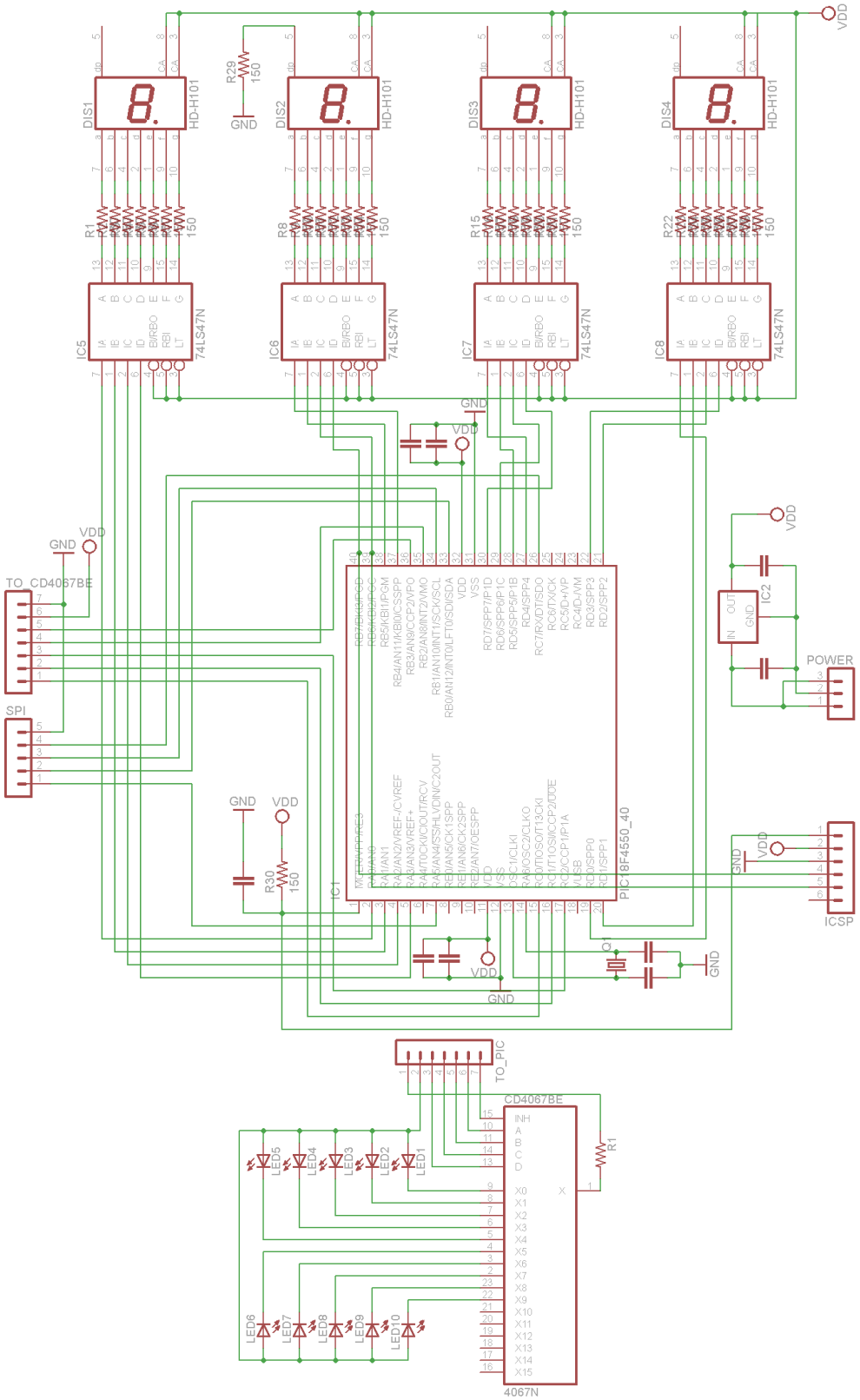
Salida por Relay

La placa cuenta con 2 Relé que podemos activar a través de RC0 y RC1, y que no fueron utilizados en este trabajo. Los pines normalmente abierto (NA) y normalmente cerrado (NC) están indicados en la serigrafía de la placa.



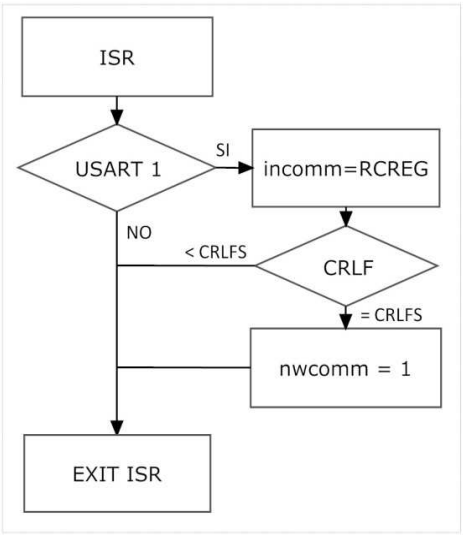
Indícadores (Display)

La placa para mostrar la información está compuesta por 4 displays de siete segmentos que se encuentran conectados a un PIC18f4550 a través de un driver (74LS47N). También posee 10 leds que son manejados a través de multiplexador (CD4067BE). Por último tiene acceso a los puertos del ICSP para su programación y a los puertos designados para establecer la comunicación por SPI.



Esquemático del Display

DIAGRAMA DE FLUJO DEL SOFTWARE



INICIALIZACIÓN DE VARIABLES GLOBALES

VARIABLES GLOBALES
volatile char incomm[150];
volatile int vi = 0;
volatile int nwcomm = 0;
volatile int CRLF = 0;
volatile int CRLFS = 2;
volatile char aux_SMS[60];

CONFIGURACIÓN DE MÓDULOS

PUERTOS
RB4 - Digital I = SDI
RC7 - Digital O = SDO
RB6 - Digital O = SCK
RC6 - Digital O = SS

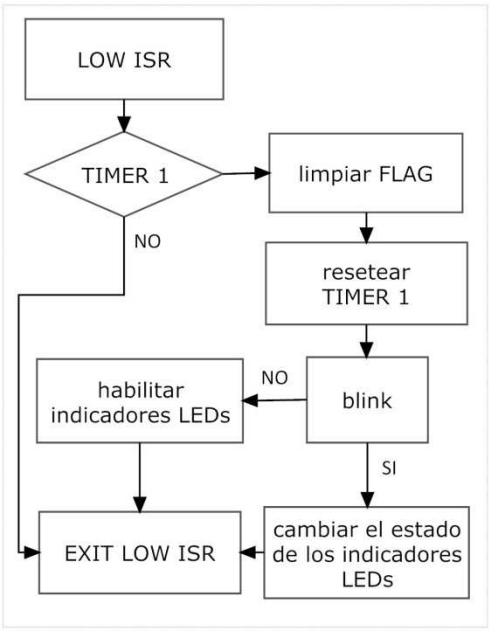
RA3 - Digital I = MCLR
RC0 - Digital O = Relay
RC1 - Digital O = Relay
RC2 - Digital I = Switch
RC3 - Digital I = Switch
RC4 - Digital O = PWRKEY
RC5 - Digital O = PWM

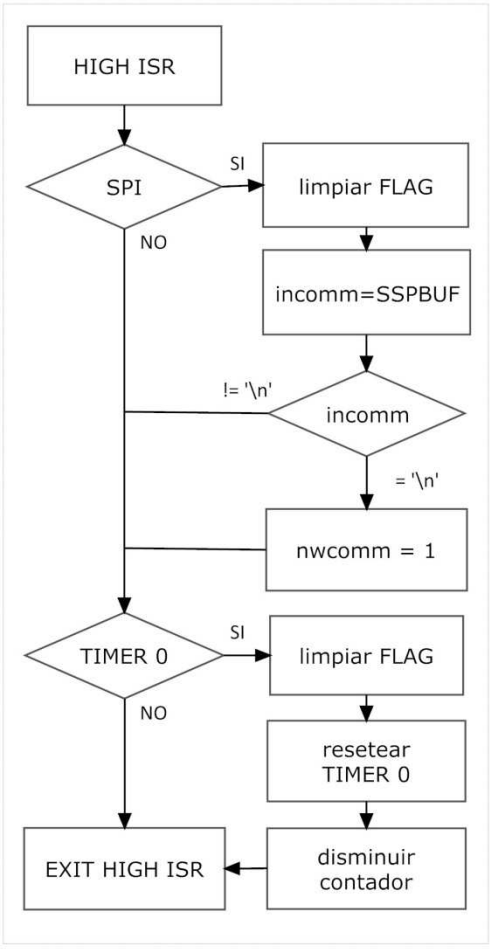
RB5 - Digital I = SIMTX
RB7 - Digital O = SIMRX

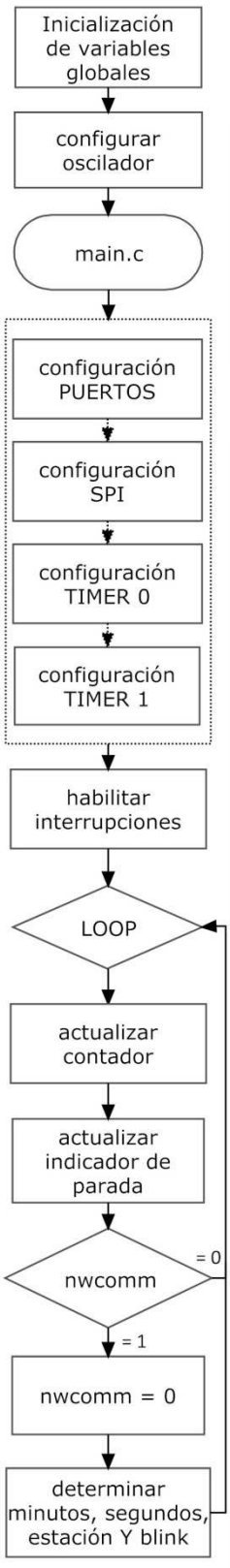
USART
Interrupciones TX: OFF
Interrupciones RX: ON
Modo: asincrono
Baud rate: 9600bps

TIMER 0
Interrupciones: OFF
Modo: 16bits
Oscilador: interno
Prescaler: 128

SPI
Modo: Master
Interrupciones: OFF
Baud Rate: 480







INICIALIZACIÓN DE VARIABLES GLOBALES

VARIABLES GLOBALES
volatile int seconds;
volatile int minutes;
volatile char updown;
volatile int blink = 0;
volatile int vi = 0;
volatile int nwcomm = 0;
volatile char incomm[20];

CONFIGURACIÓN DE MÓDULOS

PUERTOS
RD0 - Digital O = SSD01/0
RD1 - Digital O = SSD01/1
RD2 - Digital O = SSD01/2
RD3 - Digital O = SSD01/3

RD4 - Digital O = SSD02/0
RD5 - Digital O = SSD02/1
RD6 - Digital O = SSD02/2
RD7 - Digital O = SSD02/3

RB4 - Digital O = SSD03/0
RB5 - Digital O = SSD03/1
RB6 - Digital O = SSD03/2
RB7 - Digital O = SSD03/3

RA0 - Digital O = SSD04/0
RA1 - Digital O = SSD04/1
RA2 - Digital O = SSD04/2
RA3 - Digital O = SSD04/3

RB2 - Digital O = LEDD/0
RB3 - Digital O = LEDD/1
RC0 - Digital O = LEDD/2
RC1 - Digital O = LEDD/3

RC7 - Digital O = SDO
RB0 - Digital I = SDI
RB1 - DIgital I = SCK
RA5 - Digital I = SS

SPI
Modo: slave
Interrupciones RX: ON

TIMER 0
Interrupciones: ON
Modo: 16bits
Oscilador: interno
Prescaler: 64

TIMER 1
Interrupciones: ON
Modo: 16bits
Oscilador: interno
Prescaler: 8

COMANDO "BUS"

La unidad de recepción no está orientada a enviar información, pero debido a la reducción que se realizó al sistema original, se decidió que esta unidad se ocupara de las consultas que los usuarios (pasajeros) pudieran hacer desde sus teléfonos móviles haciendo uso del servicio de SMS. Sin embargo, la función principal de la unidad receptora sigue siendo la de recibir los datos generados por la UT (Unidad de Transmisión).
Para realizar una consulta a la UR (Unidad de Recepción) se debe enviar un mensaje SMS con el comando "BUS" al número asignado de la parada de colectivo, y a continuación se recibirá la respuesta con la información requerida.
Si bien en el sistema (sistema reducido) diseñado no hay una central común de datos, de implementarse, habría que tener dos consideraciones:

1. De utilizarse el servició de mensajería SMS, se debería especificar un código para indicar la parada en la que uno se encuentra y el colectivo que se espera.

2. En el caso de utilizar teléfonos inteligentes se podría hacer uso del GPS para determinar la parada en la que uno se encuentra y ver el tiempo estimado de llegada de los próximos colectivos a esa parada.

BUS

COMANDO ENVIADO POR SMS

Colectivo: 37
Tiempo estimado: 12:23
http://maps.google.com/maps?q=-34.601542,-58.417852

RESPUESTA ANTE COMANDO "BUS"

SOFTWARE

Software UR (Unidad de Recepción[13]):

```
#include <p18f14k50.h>
#include <usart.h>
#include <timers.h>
#include <delays.h>
#include <string.h>
#include <stdio.h>
#include <spi.h>

#include "global.h"
#include "genf.h"
#include "gsmf.h"

void low_isr(void);                    // Function declaration - ISR_LOW
void high_isr(void);                   // Function declaration - ISR_HIGH

                                       // CONFIGURATION BITS
```

[13] El software explicado solo comprende el main.c. El resto del software está en el proyecto de MPLAB que se encuentra en el CD (4. Sistema/2. UR/2. Software/UR_31JUL11/).

```
#pragma config CPUDIV = NOCLKDIV, USBDIV = OFF                                  // CONFIG1L
#pragma config FOSC = HS, FCMEN = OFF,  PLLEN = OFF,PCLKEN = OFF, IESO = OFF    // CONFIG1H
#pragma config PWRTEN = OFF, BOREN = OFF, BORV = 30                             // CONFIG2L
#pragma config WDTEN = OFF, WDTPS = 32768                                       // CONFIG2H
#pragma config MCLRE = ON, HFOFST = OFF                                         // CONFIG3H
#pragma config STVREN = ON, LVP = OFF,BBSIZ = OFF, XINST = OFF                  // CONFIG4L
#pragma config CP0 = OFF, CP1 = OFF                                             // CONFIG5L
#pragma config CPB = OFF, CPD = OFF                                             // CONFIG5H
#pragma config WRT0 = OFF, WRT1 = OFF                                           // CONFIG6L
#pragma config WRTB = OFF, WRTC = OFF, WRTD = OFF                               // CONFIG6H
#pragma config EBTR0 = OFF, EBTR1 = OFF                                         // CONFIG7L
#pragma config EBTRB = OFF                                                      // CONFIG7H


#define PWRKEY   PORTCbits.RC4          // PWRKEY
//#define LED2     PORTCbits.RC6        // LED2
#define SS         PORTCbits.RC6        // SS
#define LED4       PORTCbits.RC7        // LED4
#define RELAY0     PORTCbits.RC0        // RELAY0
#define RELAY1     PORTCbits.RC1        // RELAY1
#define SWITCH2    PORTCbits.RC2        // SWITCH2
#define SWITCH3    PORTCbits.RC3        // SWITCH3
#define SWITCH6    PORTBbits.RB6        // SWITCH6


#pragma code low_vector=0x18            // LOW ISR
void interrupt_at_low_vector(void)
{
_asm GOTO low_isr _endasm
}
#pragma code /* return to the default code section */

#pragma interruptlow low_isr
void low_isr (void)                     // LOW ISR Function
{

}

#pragma code high_vector=0x08           // HIGH ISR
void interrupt_at_high_vector(void)
{
_asm GOTO high_isr _endasm
}
#pragma code /* return to the default code section */

#pragma interrupt high_isr
void high_isr (void)                    // HIGH ISR Function
{
        incomm[vi] = RCREG;             // Read incoming byte
        if(incomm[vi] == 0x0A){         // Is the byte = to <LF>
                CRLF++;                 // Number of <LF> arrived
        }
        vi++;                           // Next position in the array
        if(CRLF == CRLFS){              // Is the last <LF>?
                incomm[vi] = '\0';      // Delimiter array
                vi = 0;                 // Reset the position in the array
                CRLF = 0;               // Reset the <LF> counter
                CRLFS = 2;              // The last <LF> is the second
                nwcomm = 1;             // New command
        }
}

void main (void){                       // MAIN PROGRAM

        char lat[14] = "-34.601542";    // Variables locales
        char lon[14] = "-58.417852";
        float distance = 0.0;
        float time_counter = 0.0;

        char phone[15];
        char date[10];
        char time[15];

        char aux[5];
```

```
        config_pam();                       // Configuraciòn de Puertos y Módulos

        DI();

        gsm_on();                           // Prender módulo GSM
        synchronize();                      // Synchronize USART

        clear_OERR();                       // Eliminar posible Error de Overrun

        EI();                               // Enable Global Interrupt

        while(!initialize_GSM());           // Set up GSM module
        reset_isrv();                       // Reset variables de la ISR

        while(1){                           // Loop

                if(nwcomm == 1){            // CONTINUAR SI: llegó un SMS
                        nwcomm = 0;         // Elilinar el indicador de nuevo SMS

                        if(read_CMD()){     // Interpretar el comando

                                analize_SMS(phone, date, time, aux_SMS);        // Analizar el mensaje
                                        // CONTINUAR SI: SMS == 'BUS'
                                if(strcmppgm2ram(aux_SMS, (char*)"BUS") == 0){
                                        // Crear link a la ubicación
                                        strcpypgm2ram(aux_SMS,(char*)"http://maps.google.com/maps?q=");
                                        // Agregar la latitud al link
                                        strcat(aux_SMS, lat);
                                        // Copiar ',' a una variable auxiliar
                                        strcpypgm2ram(aux,(char*)",");
                                        // Agregar ',' al link
                                        strcat(aux_SMS, aux);
                                        // Agregar la longitud al link
                                        strcat(aux_SMS, lon);
                                        // Enviar la ubicación al remitente
                                        send_SMS(aux_SMS, phone);
                                }
                                else{       // CASO CONTRARIO
                                        // Analizar el mensaje
                                        get_lld(aux_SMS, lat, lon, &distance, &time_counter);
                                        // Mostrar información
                                        stop_indicator(&distance, &time_counter);

                                }
                        }
                }
        }
}
```

Display[14]:

```
#include <p18f4550.h>
#include <delays.h>
#include <timers.h>
#include <string.h>
#include <stdlib.h>

void SSDisplay_01(int n);
void SSDisplay_02(int n);
void SSDisplay_03(int n);
void SSDisplay_04(int n);
void LEDDisplay(int n);
void clock(int minutes, int seconds);
void set_mss(int* bus_stop);

void low_isr(void);                         // Function declaration - ISR_LOW
void high_isr(void);                        // Function declaration - ISR_HIGH
```

[14] El software explicado solo comprende el main.c. El resto del software está en el proyecto de MPLAB que se encuentra en el CD (4. Sistema/2. UR/2. Software/MI_13JUL11/).

```
                                                // CONFIGURATION BITS

#pragma config CPUDIV = OSC1_PLL2, USBDIV = 1, PLLDIV = 1                   // CONFIG1L
#pragma config FOSC = HS, FCMEN = OFF, IESO = OFF                           // CONFIG1H
#pragma config PWRT = OFF, BOR = OFF                                        // CONFIG2L
#pragma config WDT = OFF                                                    // CONFIG2H
#pragma config MCLRE = ON, LPT1OSC = OFF, PBADEN = OFF, CCP2MX = OFF        // CONFIG3H
#pragma config STVREN = ON, LVP = OFF, DEBUG = OFF, XINST = OFF             // CONFIG4L
#pragma config CP0 = OFF, CP1 = OFF, CP2 = OFF, CP3 = OFF                   // CONFIG5L
#pragma config CPB = OFF, CPD = OFF                                         // CONFIG5H
#pragma config WRT0 = OFF, WRT1 = OFF, WRT2 = OFF, WRT3 = OFF               // CONFIG6L
#pragma config WRTB = OFF, WRTC = OFF, WRTD = OFF                           // CONFIG6H
#pragma config EBTR0 = OFF, EBTR1 = OFF, EBTR2 = OFF, EBTR3 = OFF           // CONFIG7L
#pragma config EBTRB = OFF                                                  // CONFIG7H

volatile int seconds;
volatile int minutes;
volatile char updown;
volatile int vi = 0;
volatile int blink = 0;
volatile int nwcomm = 0;
volatile char incomm[20];

#pragma code low_vector=0x18
                                                // LOW ISR
void interrupt_at_low_vector(void)
{
_asm GOTO low_isr _endasm
}
#pragma code /* return to the default code section */

#pragma interruptlow low_isr
void low_isr (void)                             // LOW ISR Function
{
        if(PIR1bits.TMR1IF == 1){               // Interrupción del TIMER1
                PIR1bits.TMR1IF = 0;            // Eliminar Flag
                TMR1H = 0x34;                   // Cargar timer
                TMR1L = 0x8D;
                if(blink == 1){                 // Si blink = 1 cambiar estado del LED
                        PORTCbits.RC2 = ~PORTCbits.RC2;
                }
                else{                           // Caso contrario mantener encendido
                        PORTCbits.RC2 = 0;
                }
        }
}

#pragma code high_vector=0x08                   // HIGH ISR
void interrupt_at_high_vector(void)
{
_asm GOTO high_isr _endasm
}
#pragma code /* return to the default code section */

#pragma interrupt high_isr
void high_isr (void)                            // HIGH ISR Function
{
        if(PIR1bits.SSPIF == 1){                // Interrupción del SPI?
                PIR1bits.SSPIF = 0;             // Eliminar flag
                incomm[vi] = SSPBUF;            // Cargar caracter recibido en incomm
                vi++;                           // Incrementar posición del arreglo
                if(incomm[vi - 1] == '\n'){     // Si el caracter es igual a '\n'
                        incomm[vi - 1] = '\0';// Reemplazarlo por '\0'
                        vi = 0;                 // Reiniciar posición del arreglo
                        nwcomm = 1;             // Indicar de un nuevo comando
                }
        }

        if(INTCONbits.TMR0IF == 1){             // Interrupción del TIMER0?
                INTCONbits.TMR0IF = 0;          // Eliminar flag
                TMR0H = 0x67;                   // Cargar timer
                TMR0L = 0x84;
                if(updown == 1){                // Determinar si se incrementa/decrementa el contador
                        seconds++;              // Incrementar segundos
```

```
                        if(seconds == 60){  // Si pasaron 60 segundos
                                            // Segundos es 0
                            seconds = 0;
                                            // Incrementar minutos
                            minutes++;
                        }
                    }
                    else{
                                            // Si el tiempo no se termino
                        if((seconds != 0)||(minutes != 0)){
                                            // Decrementar segundos
                            seconds--;
                                            // Si se decremento un minuto entonces
                            if(seconds == -1){
                                            // Poner en 59 los segundos
                                            seconds = 59;
                                            // Decrementar minutos
                                            minutes--;
                            }
                        }
                    }
            }

}

void main (void){
                                            // MAIN PROGRAM

            int bus_stop;
            char buffer;
            int i = 0;

            ADCON1 = 0b00001111;            // Configuración de puertos

            PORTC = 0b00000000;
            TRISC = 0b11111000;

            PORTD = 0b00000000;
            TRISD = 0b00000000;

            PORTB = 0b00000000;
            TRISB = 0b00000011;

            UCONbits.USBEN = 0;             // USB module and supporting circuitry disabled (device detached)
            UCFGbits.UTRDIS = 1;            // On-chip transceiver disabled; digital transceiver interface enabled

            PORTA = 0b00000000;
            TRISA = 0b11110000;

// ********************** SPI  SET UP  ********************** //

            TRISCbits.TRISC7 = 0;           // SDO
            TRISBbits.TRISB0 = 1;           // SDI
            TRISBbits.TRISB1 = 1;           // SCK
            TRISAbits.TRISA5 = 1;           // SS

            SSPSTATbits.SMP = 0;            // Slave mode
            SSPSTATbits.CKE = 1;            // Transmit occurs on transition from Idle to active clock state

            SSPCON1bits.WCOL = 0;           // No collision
            SSPCON1bits.SSPOV = 0;          // No overflow
            SSPCON1bits.SSPEN = 1;          // Enables serial port and configures SCK, SDO, SDI and SS as serial
                                            port pins
            SSPCON1bits.CKP = 0;            // Idle state for clock is a low level
            SSPCON1bits.SSPM0 = 0;          // SPI Slave mode, clock = SCK pin, SS pin control disabled, SS can
                                            be used as I/O pin
            SSPCON1bits.SSPM1 = 0;
            SSPCON1bits.SSPM2 = 1;
            SSPCON1bits.SSPM3 = 0;

            IPR1bits.SSPIP = 1;             // High priority
            PIE1bits.SSPIE = 1;             // Enables the MSSP interrupt

// ********************** TIMER SET UP ********************** //
```

```
	OpenTimer0(                     // Configuración del TIMER 0
		TIMER_INT_ON &          // Interrupcciones habilitadas
		T0_16BIT &              // 16 Bits
		T0_SOURCE_INT &         // Oscilador interno
		T0_PS_1_64              // Prescaler 1:64
	);
	WriteTimer0(26500);             // 1 Seconds

	OpenTimer1(                     // Configuración del TIMER 1
		TIMER_INT_ON &          // Interrupcciones habilitadas
		T1_16BIT_RW &           // 16 Bits
		T1_SOURCE_INT &         // Oscilador interno
		T1_PS_1_8 &             // Prescaler 1:8
		T1_OSC1EN_OFF &         // Deshabilitar oscilador externo
		T1_SYNC_EXT_OFF         // Sincro externa: OFF
	);

	WriteTimer1(13453);             // 0.5/3 Seconds


	INTCONbits.GIE = 1;             // Enables Global Interrupt
	INTCONbits.PEIE = 1;            // Enables Peripheral Interrupt

	RCONbits.IPEN = 1;              // Enable priority levels on interrupt
	INTCON2bits.TMR0IP = 1;         // TMR0 Overflow Interrup  High priorit
	IPR1bits.TMR1IP = 0;            // TMR1 Overflow Interrupt Low priorit

	INTCONbits.TMR0IE = 1;          // Enables the TMR0 overflow interrupt
	PIE1bits.TMR1IE = 1;            // Enables the TMR1 overflow interrupt


// ********************************************************** //

	clock(0,0);                     // Poner en 0 el display
	updown = 0;                     // Decrementar contador
	minutes = 0;                    // Minutos = 0
	seconds = 0;                    // Segundos = 0
	bus_stop = 10;                  // Parada fuera de vista
	vi = 0;                         //
	blink = 1;                      // Deshabilitar indicadores LEDs

	while(1){

		clock(minutes, seconds);// Actualizar Display numérico
		LEDDisplay(bus_stop);   // Mostar parada
		if(nwcomm == 1){        // Consultar si se recibió un comando
			nwcomm = 0;     // Eliminar flag
                                        // Fijar parada y tiempo estimado
			set_mss(&bus_stop);
		}
	}
}

//-------------------------------------------------------------------------//

                                        // Driver para controlar el display numérico y los indicadores LEDs
void SSDisplay_01(int n){
	PORTDbits.RD0 = (n & 0x000000001);
	PORTDbits.RD1 = (n & 0x000000002) >> 1;
	PORTDbits.RD2 = (n & 0x000000004) >> 2;
	PORTDbits.RD3 = (n & 0x000000008) >> 3;
}

void SSDisplay_02(int n){
	PORTDbits.RD4 = (n & 0x000000001);
	PORTDbits.RD5 = (n & 0x000000002) >> 1;
	PORTDbits.RD6 = (n & 0x000000004) >> 2;
	PORTDbits.RD7 = (n & 0x000000008) >> 3;
}

void SSDisplay_03(int n){
	PORTBbits.RB4 = (n & 0x000000001);
	PORTBbits.RB5 = (n & 0x000000002) >> 1;
	PORTBbits.RB6 = (n & 0x000000004) >> 2;
	PORTBbits.RB7 = (n & 0x000000008) >> 3;
```

```
}

void SSDisplay_04(int n){
        PORTAbits.RA0 = (n & 0x000000001);
        PORTAbits.RA1 = (n & 0x000000002) >> 1;
        PORTAbits.RA2 = (n & 0x000000004) >> 2;
        PORTAbits.RA3 = (n & 0x000000008) >> 3;
}

void LEDDisplay(int n){
        PORTBbits.RB2 = (n & 0x000000001);                      // A
        PORTBbits.RB3 = (n & 0x000000002) >> 1;                 // B
        PORTCbits.RC0 = (n & 0x000000004) >> 2;                 // C
        PORTCbits.RC1 = (n & 0x000000008) >> 3;                 // D
        //PORTCbits.RC2 = 0;                                    // INHIBIT
}

void clock(int minutes, int seconds){           // Fijar contador

        int aux = 0;                            // Variable auxiliar

        aux = seconds/10;                       // Determinar el primer digito de los segundos
        aux = seconds - aux*10;
        SSDisplay_01(aux);                      // Mostrar primer digito
        SSDisplay_02(seconds/10);               // Mostrar segundo digito

        aux = minutes/10;                       // Determinar el primer digito de los minutos
        aux = minutes - aux*10;

        SSDisplay_03(aux);                      // Mostrar primer digito
        SSDisplay_04(minutes/10);               // Mostrar segundo digito

}

void set_mss(int* bus_stop){                    // Determinar parada
        float ttttssb = 0.0;                    // Variables auxiliares
        long t_estimado = 0;

                                                // De no recibirse DUMMY
        if(strcmppgm2ram(incomm, (char*)"DUMMYI") != 0){
                ttttssb = atof(incomm);         // Convertir de caracter a float
                                                // Extraer tiempo estimado
                t_estimado = (long)(ttttssb/1000);
                                                // Determinar minutos
                minutes = (int)(t_estimado / 60);
                                                // Determinar segundos
                seconds = (int)t_estimado - minutes*60;
                                                // Determinar parada
                *bus_stop = (int)(ttttssb - t_estimado * 1000)/10;
                                                // Determinar si se titila
                blink = (int)(ttttssb - t_estimado * 1000) - *bus_stop * 10;
        }
}
//-------------------------------------------------------------------------//
```

CONCLUSIONES

Con este trabajo tuve la oportunidad de poner en práctica los conocimientos adquiridos en diferentes materias de la facultad, como Computación para ingeniería, Circuitos eléctricos, Microprocesadores y periféricos o Sistemas y servicios de comunicaciones, entre otras.
Por otro lado tuve el desafió de hacer una implementación completa, empezando por el diseño, desarrollando el equipamiento necesario y por último haciendo las pruebas para corroborar su correcto comportamiento.

La multiplicidad de aéreas y temas involucrados en el trabajo me llevaron a adquirir nuevos conocimientos, sobre todo a lo que respecta al uso de los módulos GPS y GSM, ambos de gran utilidad y con un gran campo de implementación en el mercado.

En cuanto al sistema en sí, a mi criterio, va a ser de gran utilidad en un futuro cercano, ya que los inconvenientes que presentan las grandes ciudades, sobre todo en cuanto al tema de movilidad, ha llevado a la comunidad a pensar cómo solucionarlos, y entre las opciones se encuentra dotar a la mayor parte de la infraestructura de las ciudades de inteligencia. Hacer el seguimiento del trasporte público, controlarlo, y tomar decisiones en función de los datos adquiridos en una forma de darle inteligencia a este servicio.

Aunque el sistema propuesto en este trabajo es muy rudimentario y simplista, la idea estaba enfocada en desarrollar y presentar una idea. Espero haber podido cumplir con el objetivo.