ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN
UNIVERSIDAD POLITÉCNICA DE CARTAGENA

Proyecto Fin de Carrera

SAUMI

AUTOR: Daniel Norberto López Cánovas
DIRECTOR: José Fernando Cerdán Cartagena
CODIRECTOR: Diego García Sánchez

JUNIO / 2013

Autor	Daniel Norberto López Cánovas
E-mail del Autor	Daniel.lopez.canovas@gmail.com
Director(es)	José Fernando Cerdán Cartagena
E-mail del Director	fernando.cerdan@upct.es
Codirector(es)	Diego García Sánchez
Título del PFC	SAUMI (Sistema de Adquisición para Seguimiento de Usuarios y Monitorización de Itinerarios)
Descriptores	miniPC , Sistema operativo Linux , GPS ,Bluetooth, GPRS, C

Resumen

Con este proyecto se desea crear un sistema para realizar la detección de usuarios anónimos, principalmente de líneas de tren.
El sistema engloba diferentes tecnologías para la consecución del mismo: Bluetooth, posicionamiento GPS y envío de datos a través de GPRS.
Es un sistema completo pero sencillo y que ocupa muy poco espacio, lo que le da una mayor escalabilidad en cuanto a sus diversas aplicaciones futuras.
La tecnología para el desarrollo de la aplicación ha sido la programación en C, ya que es un lenguaje muy extendido, y que utiliza muy pocos recursos del pc cuando se está ejecutando en modo background.

Titulación	Ingeniería Técnica de Telecomunicación especialidad Telemática

Intensificación	-
Departamento	Tecnologías de la Información y las Comunicaciones
Fecha de Presentación	Junio - 2013

INDICE GENERAL

INDICE DE FIGURAS

CAPITULO 1. INTRODUCCIÓN

1. Marco de trabajo

Hoy en día hay mucha información en cualquier ámbito de la vida diaria, mucha información sobre los ámbitos, costumbres, forma de actuar de las personas que no es recopilada y que se pierde. En este proyecto queremos recoger toda esta información y encauzarla hacia una posible utilización de la misma.

Es posible conocer donde suele comprar más la gente, si cambian de hábitos, o que línea de trenes es más imprescindible y cual menos, y todo ello sin necesidad de molestar a la gente con encuestas que muchas veces no nos apetece contestar.

En este panorama surge la idea de realizar este proyecto, crear un sistema pequeño y portátil para estudiar la información que arroja la gente sin interferir en su vida diaria. Para ello nos centraremos en el estudio sobre la utilización de las líneas de trenes, como ejemplo de uno de estos ámbitos de estudio.

2. Objetivo

Este sistema se basa en la tecnología Bluetooth y está pensando para detectar la presencia de usuarios en cualquier espacio físico.

La idoneidad de esta tecnología se debe a que está totalmente integrada en la sociedad gracias a su incorporación en prácticamente la totalidad de los terminales móviles o tablets del momento. Es por ello que se ha optado por esta tecnología ya que nos brinda la posibilidad de obtener multitud de información acerca de los ámbitos de actuación de las personas, y concretamente del uso del transporte ferrovial.

El hecho de conocer las paradas en las que buena parte de los usuarios suben y bajan de los trenes sirve como muestra a partir de la que obtener una buena representación de datos estadísticos como: duración media de cada trayecto, paradas más frecuentadas, líneas en desuso u otros muchos atributos referentes a cualquier combinación de rutas utilizadas por los usuarios.

Por medio de la posición GPS y un perímetro creado por nosotros, podemos estudiar el comportamiento de las personas en diferentes ámbitos: bien en una línea de trenes (como veremos más adelante), en un estadio de futbol, seguimiento del personal de seguridad de una

empresa, realizar un estudio de a dónde va más gente a comprar, etc, ya que las posibilidades son numerosas.

La finalidad de este proyecto, la parte Cliente, es crear una aplicación capaz de realizar las detecciones BT, posicionarlas mediante GPS, procesar y optimizar la información recibida y enviarla vía GPRS al servidor para ser procesada.

CAPITULO 2. TECNOLOGÍAS

1. Introducción

A continuación vamos a introducir brevemente las tecnologías usadas para nuestro proyecto. Como ya hemos indicado nos decidimos por estas tecnologías y no otras por su estandarización, ya que nos ha permitido adaptarlas a nuestras necesidades y encontrar mucha información para realizar la programación de las mismas, sin olvidar su integración en la sociedad.

Todo ello hizo de este proyecto, un proyecto viable a la vez que atractivo.

2. Bluetooth

Ya se ha explicado anteriormente que se escogió esta tecnología porque estaba en el cien por cien de los dispositivos móviles actualmente, y este fue el principal hándicap.

Bluetooth es un estándar de comunicaciones que identifica un conjunto de protocolos, los cuales facilitan la comunicación inalámbrica entre diferentes dispositivos electrónicos. El origen de la palabra Bluetooth proviene del rey Vikingo, Harald Bluetooth (Harald Diente Azul), que gracias a su habilidad para la comunicación y el tratar con las personas consiguió unificar Dinamarca. Esto es lo que se pretende con Bluetooth, unificar los estándares y las tecnologías diferentes mediante un dispositivo universal para la interconexión de todo tipo de periféricos.

Es una tecnología reciente, de especificación abierta, que define una forma de comunicación inalámbrica, la cual posibilita la transmisión de voz y datos entre diferentes equipos mediante un enlace por radiofrecuencia de corto alcance.

La especificación abierta hace posible que los fabricantes puedan, libremente, implementar sus propios protocolos de aplicación sobre los protocolos específicos de Bluetooth, permitiendo así el desarrollo de nuevas aplicaciones que fomentan el desarrollo de las capacidades tecnológicas de Bluetooth.

Los principales objetivos que se pretende conseguir con esta norma son:

- Facilitar las comunicaciones entre equipos móviles y fijos.
- Eliminar cables y conectores entre éstos.
- Ofrecer la posibilidad de crear pequeñas redes inalámbricas y facilitar la sincronización de datos entre los equipos personales.

Bluetooth es ideal para el uso en dispositivos móviles ya que cumple las características básicas para poder ser usado en éstos: reducido consumo, pequeño tamaño y un bajo precio, lo que hace que actualmente la inmensa mayoría de móviles que se venden en el mercado ya lleven instalada esta tecnología.

2.1. Topología

Para poder establecer una comunicación entre dispositivos Bluetooth, uno de ellos debe establecer el papel de maestro y otro el de esclavo. El maestro es el encargado de la sincronización de la comunicación. Todos los esclavos conectados al maestro saltarán a la frecuencia establecida por el maestro. El maestro se encargará de controlar cuando un esclavo puede transmitir, reservando ranuras temporales para éste. El papel de maestro es, generalmente, de aquel que comienza la comunicación, aunque este papel puede cambiarse

más tarde.

Un maestro puede establecer comunicación con hasta 7 esclavos a la vez. Cabe destacar que el papel de maestro o esclavo solamente tiene importancia a la hora de realizar el sincronismo a bajo nivel.

2.2. Piconet

Si un equipo se encuentra dentro del radio de cobertura de otro, estos pueden establecer una conexión. En principio sólo son necesarias un par de unidades con las mismas características de hardware para establecer un enlace.

Los participantes en la piconet pueden intercambiar los papeles si un esclavo quisiera asumir el papel de maestro. Sin embargo sólo puede haber un maestro en la piconet al mismo tiempo cada unidad de la piconet utiliza su identidad maestra y el reloj nativo para seguir en el canal de salto. Cuando se establece la conexión, se añade un ajuste de reloj a la propia frecuencia de reloj nativa de la unidad, para poder sincronizarse con el reloj nativo del maestro.

Las unidades maestras controlan el tráfico del canal. A un esclavo sólo se le permite enviar un slot a un maestro cuando éste se ha dirigido por su dirección MAC (control de acceso al medio) en el procedimiento de slot maestro esclavo.

2.3. Scatternet

Los equipos que comparten un mismo canal sólo pueden utilizar una parte de la capacidad de éste. Aunque los canales tienen un ancho de banda de un 1Mhz, cuantos más usuarios se incorporan a la piconet, disminuye la capacidad hasta unos 10 kbit/s más o menos. Teniendo en cuenta que el ancho de banda medio disponible es de unos 80 MHz éste no puede ser utilizado eficazmente, cuando cada unidad ocupa una parte del mismo canal de salto de 1Mhz. Para poder solucionar éste problema se adoptó una solución de la que nace el concepto de scatternet.

Para crear una scatternet un dispositivo de una de las dos piconets tiene que pertenecer a dos de éstas. Esto se puede hacer de dos formas: ser esclavo de las dos piconets o ser esclavo de una piconet y maestro de la otra.

Las unidades que se encuentran en el mismo radio de cobertura pueden establecer potencialmente comunicaciones entre ellas. Sin embargo, sólo aquellas unidades que realmente quieran intercambiar información comparten un mismo canal creando la piconet.

Éste hecho permite que se creen varias piconets en áreas de cobertura superpuestas. A un grupo de piconets se le llama scatternet. En un conjunto de varias piconets, éstas seleccionan diferentes saltos de frecuencia y están controladas por diferentes maestros, por lo que si un mismo canal de salto es compartido temporalmente por piconets independientes, los paquetes de datos podrán ser distinguidos por el código de acceso que les precede, que es único en cada piconet.

Una unidad al incorporarse a una nueva piconet debe modificar el offset (ajuste interno) de su reloj para minimizar la deriva entre su reloj nativo y el del maestro, por lo que gracias a éste sistema se puede participar en varias piconets realizando cada vez los ajustes correspondientes una vez conocidos los diferentes parámetros de la piconet. Cuando una unidad abandona una piconet, el esclavo informa al maestro actual que ésta no estará disponible por un determinado periodo, que será en el que estará activa en otra piconet. Durante su ausencia, el tráfico en la piconet entre el maestro y otros esclavos continúa igualmente.

De la misma manera que un esclavo puede cambiar de una piconet a otra, un maestro también lo puede hacer, con la diferencia de que el tráfico de la piconet se suspende hasta la vuelta del maestro.

El maestro que entra en una nueva piconet, en principio, lo hace como esclavo, a no ser que posteriormente éste solicite actuar como maestro.

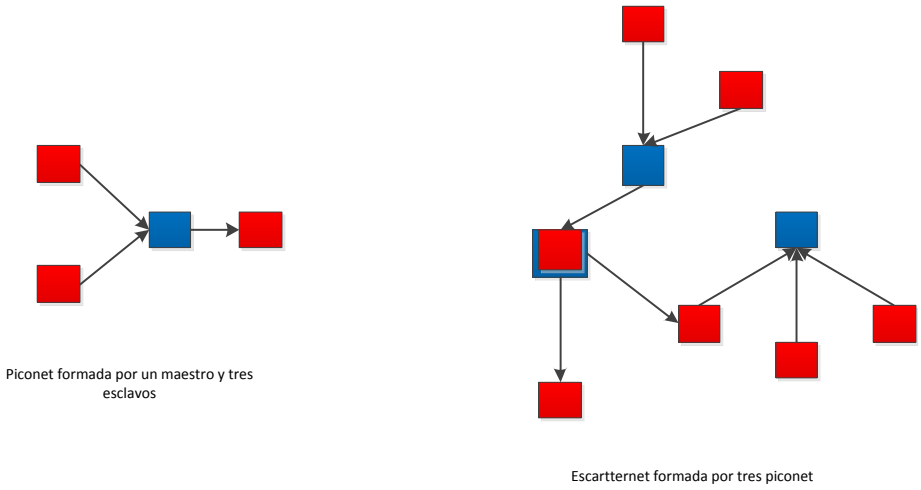


Figura 1 Ejemplo de maestros y esclavos en Picconet y Scartternet

2.4. Aspectos generales

Para poder operar era necesaria una banda de frecuencias abierta, es decir, sin necesidad de licencia independientemente del lugar del planeta donde nos encontremos, para transmitir hasta una distancia de 100 metros a velocidades de transferencia del orden de los 721 Kbps. Solo la banda ISM (médico-científica internacional) a 2,45 GHz cumple estos requisitos, con rangos que van de los 2400 MHz a los 2500 MHz, y sólo con algunas restricciones en países como Francia y Japón.

Un esquema de saltos de frecuencia aleatorios permite a los dispositivos comunicarse incluso en áreas donde existe gran interferencia electromagnética. Bluetooth suministra también mecanismos de encriptación y autentificación, para controlar la conexión y evitar que cualquier dispositivo, no autorizado, pueda acceder a los datos o modificarlos. El manejo de la clave se hace a nivel de aplicación.

Bluetooth se ha diseñado para operar en una ambiente multi-usuario.

Los dispositivos pueden habilitarse para comunicarse entre sí e intercambiar datos de una forma transparente al usuario. Aunque cada enlace es codificado y protegido contra interferencias y pérdidas de enlace, Bluetooth puede considerarse como una red inalámbrica segura. Aun así también es posible es uso de algunas técnicas a nivel de aplicación para poder aumentarla.

2.5. Salto en frecuencia

Debido a que la banda ISM está abierta a cualquiera, el sistema de radio Bluetooth deberá estar preparado para evitar múltiples interferencias que pudieran producirse. Éstas pueden ser evitadas utilizando un sistema que busque una parte no utilizada del espectro o un sistema de salto de frecuencia.

En los sistemas de radio Bluetooth se suele utilizar el método se salto en frecuencia debido a que ésta tecnología puede ser integrada en equipos de baja potencia y bajo coste. Éste sistema divide la banda de frecuencia en varios canales de salto, donde, los transceptores, durante la conexión van cambiando de uno a otro canal de salto, de manera pseudo-aleatoria.

Con esto se consigue que el ancho de banda instantáneo sea muy pequeño y también una propagación efectiva sobre el total de ancho de banda.

El canal de radio de Bluetooth utiliza espectro ensanchado y salto en frecuencia (FHSS, Frecuency Hopping Spread Spectrum), por lo que puede cambiar de estado hasta 1600 veces por segundo sobre 79 frecuencias distintas separadas 1 MHz, empezando en 2,402 GHz y terminando en 2,480 GHz. Para cumplir las normas de banda en cada país, se deja una banda de guardia en los límites

superior e inferior de la banda.

Otro aspecto a destacar es la pequeña longitud de los paquetes, con lo que si otro dispositivo intentara transmitir a la misma frecuencia la probabilidad de colisión también seria baja debido a este motivo.

En la siguiente figura tenemos un ejemplo de dos dispositivos que están intentando transmitir en la misma frecuencia.

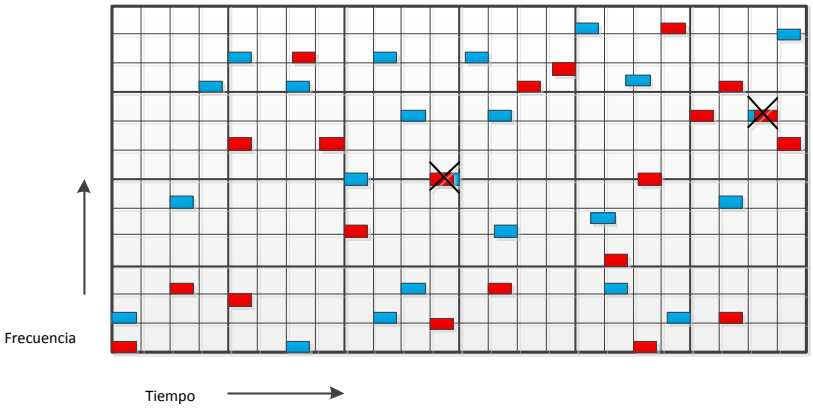


Figura 2 Ejemplo de la técnica de salto en frecuencia

Este caso se refiere a una situación donde hay dos o mas piconets activas transmitiendo a una determinada frecuencia o un dispositivo no Bluetooth usando la misma frecuencia que uno que si lo sea transmitiendo a la vez.

2.6. Definición del canal

El canal queda dividido en intervalos de 625 :s, llamados slots, donde cada salto de frecuencia es ocupado por un slot. Esto da lugar a una frecuencia de salto de 1600 veces por segundo, en la que un paquete de datos ocupa un slot para la emisión y otro para la recepción y que pueden ser usados alternativamente, dando lugar a un esquema de división dúplex en el tiempo(TDD, Time Division Duplex).

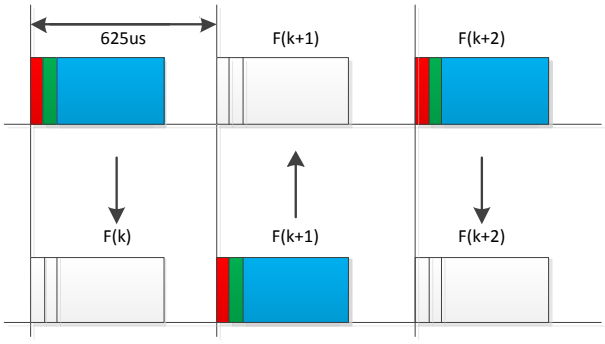


Figura 3 Transmisión por división en el tiempo

2.7. Enlaces

Existen dos tipos de enlaces que permiten soportar incluso aplicaciones multimedia:

- Enlace de sincronización de conexión orientada (SCO, Sinchronous Connection Oriented). Los enlaces SCO soportan conexiones asimétricas, punto a punto, usadas normalmente en conexiones de voz. El enlace es punto a punto entre el esclavo y el maestro de la piconet. Pudiendo tener éste hasta tres enlaces full-duplex simultáneos. La transmisión de voz se realiza sin ningún mecanismo de protección, y los paquetes no se retransmiten nunca.
- Enlace asíncrono de baja conexión (ACL, Assinchronous Connectionless). Los enlaces ACL soportan conmutaciones puntomultipunto (enlace entre el maestro y varis esclavos) típicamente usadas en la transmisión de datos. El maestro puede establecer conexión con cualquier esclavo de la piconet, aunque éste tenga un enlace SCO ya establecido. Permite la retransmisión de paquetes y utiliza los slots del canal que no están siendo utilizados por enlaces SCO.

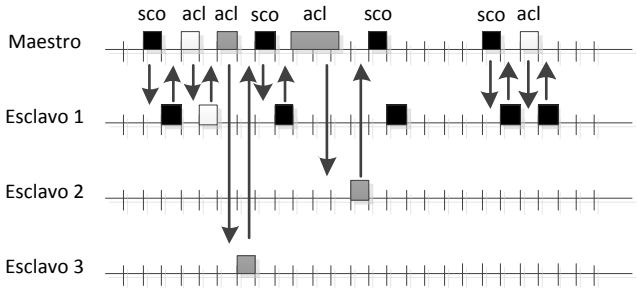


Figura 4 Transmisión por división en el tiempo

2.8. Definición de paquete

La información que se intercambia entre dos unidades Bluetooth se realiza mediante un conjunto de slots que forman un paquete de datos. En una primera clasificación, hay tres tipos de paquetes, los que ocupan 1,3 ó 5 ranuras temporales (slots). Además de las ranuras temporales, los paquetes también se diferencian por los distintos modos de corrección de errores, que son DM (Data Medium Speed) o DH (Data High Speed). También existen paquetes DV (Data Voice), como combinación de datos SCO y ACL en un mismo paquete en una ranura. Por lo que existen nueve tipos de paquetes.

Cada paquete comienza con un código de acceso de 72 bits, que se deriva de la identidad maestra, seguido de un paquete de datos de cabecera de 54 bits. Éste contiene importante información de control, como tres bits de acceso de dirección, tipo de paquete, bits de control de flujo, bits para la retransmisión automática de la pregunta, y chequeo de errores de campos de cabeza.

Finalmente, la parte que contiene la información, de longitud variable según sea el tipo de paquete.

Figura 5 Formato de los paquetes SCO y ACL

2.9. Establecimiento de la conexión

Un dispositivo Bluetooth, al principio, si no conoce nada de los dispositivos del entorno, realizará un proceso de búsqueda, y a continuación el proceso de paginación.

- Proceso de búsqueda: Consiste en descubrir los dispositivos que están en su área de cobertura y determina las direcciones y los relojes de los aparatos. El dispositivo fuente envía los paquetes de la búsqueda y espera a recibir respuesta. En el otro lado, el destino, recibe los paquetes de búsqueda siempre que esté en estado de análisis de la búsqueda, y pasará al estado de respuesta, mandando los paquetes de respuesta de la búsqueda a la fuente.
- Proceso de paginación: Consiste en establecer conexión con el otro dispositivo. Únicamente se necesita la dirección del dispositivo Bluetooth con el que se quiere crear la conexión. En el primer paso, el dispositivo fuente pagina el dispositivo destino, en el otro lado, el destino recibe la paginación, entonces, éste manda una contestación a la fuente. Después, la fuente manda un paquete FHS (Frecuency Hop Synchronisation) al destino y éste manda una segunda contestación a la fuente. Por último, la fuente y el destino intercambian características del canal.

2.10. Modos de conexión

Un dispositivo Bluetooth en estado de conexión pueden entrar en los modos de ahorro de energía en los cuales la actividad del dispositivo es menor.

Existen tres tipos de modos:

- Modo Sniff: En el modo sniff, un dispositivo esclavo escucha a la piconet a una tasa reducida, lo que reduce su ciclo de trabajo, El intervalo sniff es programable y depende de la aplicación. Tiene el mayor ciclo de vida de los tres modos de ahorro de energía.
- Modo Hold: La unidad maestra puede poner unidades esclavas en modo hold, donde únicamente está funcionando un contador interno. Las unidades esclavas también pueden demandar ser puestas en modo hold. La transferencia de datos vuelve a comenzar de forma instantánea cuando las unidades abandonan este modo.

- Modo Park: En este modo, un dispositivo se encuentra sincronizado a la piconet pero no participa en el tráfico. Los dispositivos en el estado park han abandonado sus direcciones MAC y ocasionalmente escuchan el tráfico del maestro para volver a sincronizarse y comprobar los mensajes broadcast. Tiene el ciclo de trabajo más corto de los tres modos de ahorro de energía.

2.11. Perfiles

Un perfil es un conjunto de características funcionales para cada aplicación que Bluetooth soporta. El concepto de perfil se utiliza para segurar la interoperabilidad entre varias unidades Bluetooth que cumplan los mismos perfiles. Cada dispositivo Bluetooth tiene al menos un perfil, es decir, una aplicación para la cual se puede utilizar el dispositivo. Cuando dos equipos quieran comunicarse entre sí, deben tener un perfil compartido.

El concepto de perfil surge debido a que en pequeños dispositivos el implementar toda la pila de protocolos de Bluetooth es algo innecesario, pues éste solo va a hacer uso de servicios muy específicos (ejemplo un ratón inalámbrico). Los perfiles son dinámicos, en el sentido de que cuando aparecen nuevas aplicaciones no se tiene más que añadir un nuevo perfil a la especificación de Bluetooth.

La posibilidad de definir perfiles en Bluetooth marca una diferencia con otras tecnologías inalámbricas que solamente se centran en capas física y enlace.

Los perfiles se clasifican dentro de modelos de uso. Dentro de cada modelo existen uno o más perfiles que definen las diferentes capas del protocolo bluetooth. Hay muchos modelos de uso, y a medida que pasa el tiempo se diseñan nuevos modelos, por lo que es muy difícil enumerarlos todos.

La especificación inicial de Bluetooth incluía 13 perfiles. Para garantizar la interoperatibilidad en algunas áreas de aplicaciones, el SIG añadió 12 nuevos perfiles.

En la siguiente figura observamos los perfiles de Bluetooth mas utilizados, centrandonos en aquellos mas genéricos:

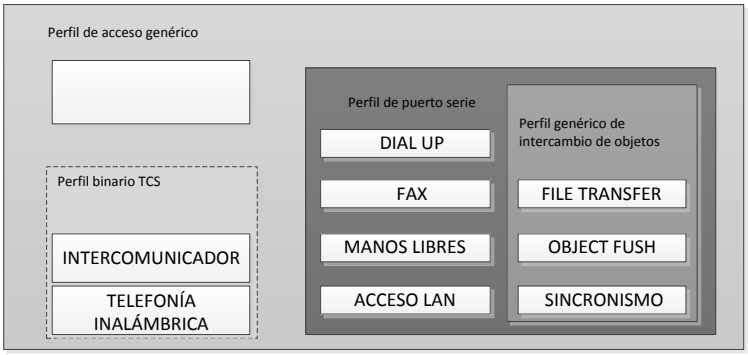


Figura 6 Perfiles de Bluetooth

- Perfil de Acceso Genérico (GAP) Este perfil es el más importante de todos, ya que el resto de perfiles depende de el. Define los procedimientos generales para el descubrimiento y establecimiento de conexión entre dispositivos Bluetooth. El GAP asegura que cualquier dispositivo Bluetooth, sin importar el fabricante, pueda intercambiar información con otro cualquiera, para descubrir que tipo de aplicaciones soportan.
- Perfil de Puerto Serie (SPP) Este perfil especifica los requisitos necesarios para establecer una conexión emulada de cable serie usando RFCOMM entre dos dispositivos similares. Este perfil solamente requiere soporte para paquetes de un slot. Esto significa que pueden ser usadas ráfagas de datos de hasta 128 Kbps. siendo el soporte para ráfagas más altas opcional. RFCOMM es usado para transportar los datos de usuario, señales de control de módem y comandos de configuración. El perfil de puerto serie es dependiente del GAP.
- Perfil de Descubrimiento de Servicio (SDAP) Este perfil concreta los protocolos y procedimientos para una aplicación en un dispositivo Bluetooth donde se desea descubrir y recuperar información relaciona con servicios localizados en otros dispositivos. También depende del GAP.
- Perfil genérico de Intercambios de Objetos (GOEP) Este perfil define protocolos y procedimientos usados por aplicaciones para ofrecer características de intercambio de objetos. Los usos pueden ser, por ejemplo, sincronización, transferencia de archivos o modelo Object Push. Los dispositivos más comunes que usan este modelo son agendas electrónicas, PDAs, teléfonos móviles y teléfonos móviles. El GOEP depende del perfil de puerto serie.

2.12. Pila de protocolos

La pila de protocolos Bluetooth tiene como objetivo que dos dispositivos con la misma pila puedan permitir que las aplicaciones funcionen correctamente entre sí. Cada pila de protocolos usa un enlace de datos y una capa física común.

Los protocolos por debajo de la interfaz del controlador del host (HCI) son los protocolos que necesariamente debe contener un dispositivo Bluetooth ya que componen el núcleo de Bluetooth.

El resto de protocolos, no pertenecientes al núcleo de Bluetooth, sólo serán necesarios si lo requiere la aplicación concreta. La mayor o menor complejidad de la pila de protocolos para una aplicación dada vendrá determinada por los requerimientos de ésta. A continuación se explican se explican las características principales de capa del protocolo:

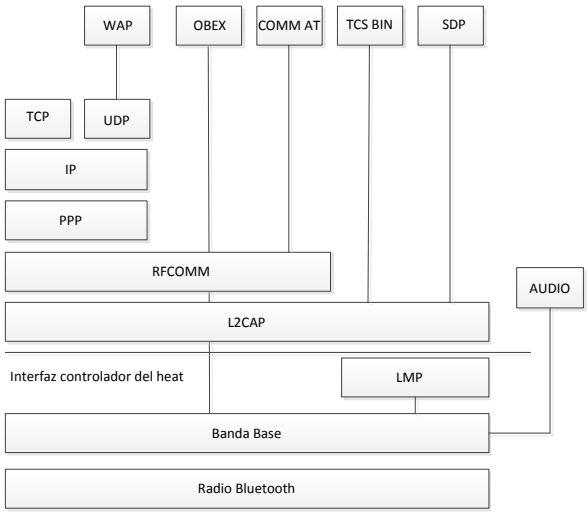


Figura 7 Pila de Protocolos Bluetooth

2.12.1. Arquitectura Hardware

Las capas o protocolos inferiores que se encuentran por debajo de HCI, son implementados por el propio hardware del dispositivo Bluetooth. Estas capas dan una interfaz simple para crear comunicaciones de voz y datos entre dos dispositivos.

- Capa de radio Es la capa más baja definida en la especificación de Bluetooth. Especifica los requisitos del aparato transceptor. Maneja y controla las señales de radiofrecuencia (RF, Radio Feecuency). Un dispositivo Bluetooth se clasifica dentro de 3 tipos de potencia:
    - Nivel de potencia 1: para dispositivos de gran cobertura, 100 metros, con una potencia máxima de 20 dBm.
    - Nivel de potencia 2: para dispositivos de cobertura media, 10 metros, con una potencia máxima de 4 dBm.
    - Nivel de potencia 3: para dispositivos de bajo alcance, 10 centímetros, con una potencia máxima de entorno a 0 dBm.
- Banda base Esta capa se encarga de la corrección de errores, seguridad en Bluetooth y selección de salto. También trata los paquetes, hace la paginación y aplica un esquema de división duplex en el tiempo (TDD, Time Division duplex), para así poder,

alternativamente, ser transmisor y receptor. El maestro empieza su transmisión en los slots pares y el esclavo en los slots impares, para así poder llevar a cabo latransmisión.

- LMP Es el protocolo encargado de administrar los recursos preestablecidos por la capa banda base, realiza transición entre modos de potencia, administra los enlaces y realiza funciones de seguridad. Principalmente, el Protocolo de administración del enlace, se encarga de descubrir otros administradores remotos y comunicarse con ellos.

2.12.2. Arquitectura Software

Ahora explicaremos las capas de las pilas Bluetooth que no pertenecen a la parte Hardware. Estas capas solamente son utilizadas por un dispositivo si las necesita para desarrollar la aplicación para la que ha sido desarrollado.

- L2CAP Es uno de las capas necesarias para los protocolos de niveles superiores, ya que ofrece servicios de datos a estos protocolos. L2CAP puede ser visto como un multiplexor, ya que se encarga de coger los datos de las capas superiores y de las aplicaciones y enviarlas, gracias a la interfaz HCI, a las capas bajas de la pila. También se encarga de la segmentación y reensamblado de los paquetes que permite a las capas superiores transmitir paquetes mayores que los que aceptan las capas bajas de la pila. Esta capa, sólo puede transmitir datos, nunca audio.
- SDP El protocolo de descubrimiento de servicios sirve para ofrecer u obtener información sabre otros dispositivos Bluetooth, con vistas a comunicarse con ellos y utilizar los servicios que ofrecen. El protocolo SDP se utiliza siempre y su objetivo principal es descubrir que servicios ofrecen otros aparatos Bluetooth.
- RFCOMM El protocolo de radio frecuencia orientado a emular puertos serie COM en un PC (RFCOMM, Radio Frecuency oriented of the serial COM port on a PC) proporciona la emulación de puertos sucesivos sobre el protocolo L2CAP. RFCOMM es un protocolo de transporte que modula una comunicación a través de l puerto RS-232 sobre un canal L2CAP. Además, RFCOMM proporciona mecanismos para el control del flujo.

2.12. Interconexión con dispositivos Bluetooth desde Linux

La pila de protocolos Bluetooth para Linux, BlueZ, permite conectar un PC con dispositivos Bluetooth remotos. Las diferentes herramientas que incluye permiten detectar dispositivos Bluetooth cercanos, obtener información básica de los mismos y conectarse a los servicios que soportan.

2.13. Configuración del dispositivo Bluetooth local

El primer paso es conectar al PC el módulo Bluetooth que se va a emplear para la Comunicación con otros dispositivos. Linux debería reconocer automáticamente el dispositivo sin necesidad de instalar drivers. No obstante, es posible que algún módulo Bluetooth requiera la instalación adicional de drivers en el sistema antes de utilizarlo. En tal caso, se debe consultar con el fabricante.

Así mismo, lo más habitual es que Linux monte automáticamente en el interfaz hci0 el módulo Bluetooth conectado, pero en algunas distribuciones puede no suceder. La verificación se realiza mediante la herramienta Hciconfig.

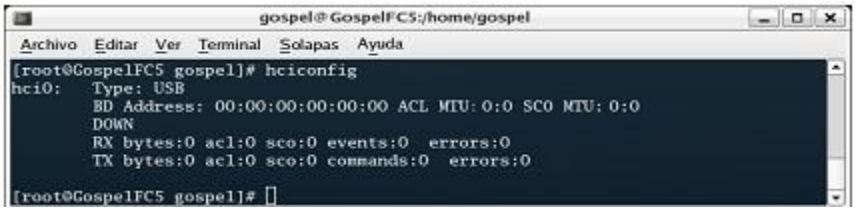


Figura 8 Ejemplo Hciconfig

En caso de que el dispositivo Bluetooth no se haya montado automáticamente, Será necesario montarlo manualmente con la herramienta Hciconfig:

Comandos:

-hciconfig hci0 up

hciconfig -a

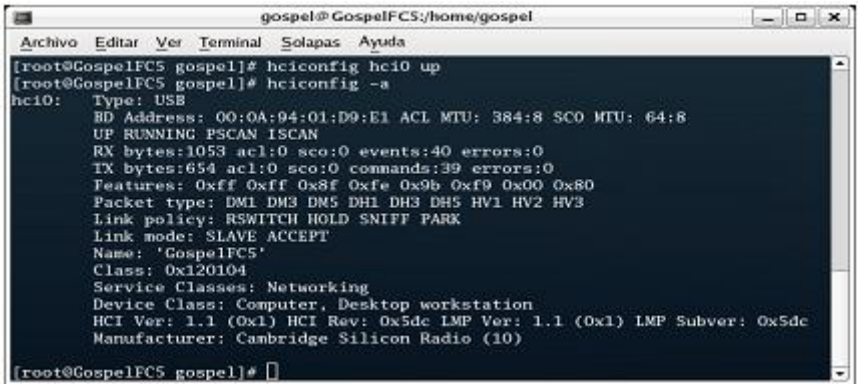


Figura 9 Ejemplos opciones hciconfig

Configuración de opciones del interfaz HCI

Antes de establecer comunicación con otro dispositivo Bluetooth, se debe configurar el fichero de opciones de HCI, localizado en /etc/bluetooth/hcid.conf.

Es recomendable utilizar la siguiente configuración para hcid.conf:

```
# HCId options
options {
# Automatically initialize new devices
autoinit yes;
# Security Manager mode
#
none - Security manager disabled
#
auto - Use local PIN for incoming connections
#
user - Always ask user for a PIN
#
security auto;
# Pairing mode
#
none - Pairing disabled
#
multi - Allow pairing with already paired devices
#
once - Pair once and deny successive attempts
pairing multi;
# PIN helper
pin_helper /usr/bin/bluepin;
}
# Default settings for HCI devices
device {
# Local device name
#
%d - device id
#
%h - host name
name "GospelFC5";
```

```
# Local device class
class 0x120104;
# Inquiry and Page scan
iscan enable; pscan enable;
# Default link mode
#
none
- no specific policy
#
accept - always accept incoming connections
#
master - become master on incoming connections,
#
deny role switch on outgoing connections
lm accept;
# Default link policy
#
none
- no specific policy
#
rswitch - allow role switch
#
hold
- allow hold mode
#
sniff
- allow sniff mode
#
park
- allow park mode
lp rswitch,hold,sniff,park;
# Authentication and Encryption (Security Mode 3)
#auth enable;
#encrypt enable;
```

Tras aplicar los cambios, procede a la re inicialización de los servicios Bluetooth.

Comando: /etc/init.d/bluetooth restart

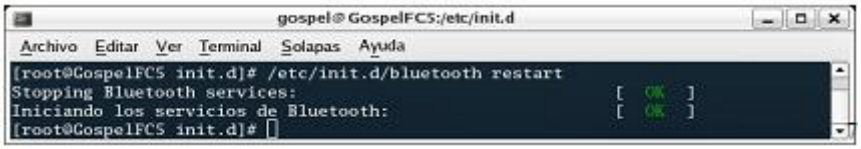


Figura 10 Ejemplo reinicio demonio Bluetooth

2.14. Detección de dispositivos Bluetooth con Hcitool

La herramienta Hcitool permite enviar paquetes inquiry para detectar la existencia de dispositivos Bluetooth cercanos.

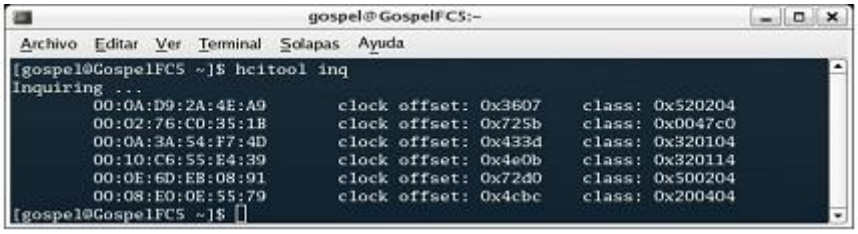


Figura 11 Ejemplo herramienta hcitool

Así mismo, también es posible obtener cierta información sobre los dispositivos detectados, como su Class of Device/Service y su nombre de dispositivo.

Figura 12 Ejemplo herramienta hcitool opción scan

2.15. Descubrimiento de servicios Bluetooth con Sdptool

La herramienta Sdptool permite identificar los perfiles disponibles en un dispositivo Bluetooth detectado, por ejemplo, un teléfono móvil.

Figura 13 Ejemplo herramienta hcitool opción scan

2.16. BlueZScanner: el escáner de dispositivos Bluetooth

BlueZScanner es un sencillo escáner de dispositivos Bluetooth que utiliza BlueZ y está desarrollado en lenguaje C.

Implementa las siguientes funciones:

- Detección de dispositivos Bluetooth cercanos.
- Resolución de nombre de dispositivo.
- Obtención del fabricante del chip Bluetooth incorporado en el dispositivo.
- Análisis del campo Device Class, que identifica la naturaleza del
- dispositivo.
- Análisis de los campos Service Classes, que identifican los servicios
- ofrecidos por el dispositivo.
- Descubrimiento de Perfiles Bluetooth disponibles en el dispositivo.

El código fuente y la herramienta BlueZScanner están disponibles para descarga en la siguiente dirección: http://gospel.endorasoft.es/

El código fuente de BlueZScanner se distribuye libremente bajo licencia GNU. Se necesita instalar previamente el paquete de librerías bluez libs devel.

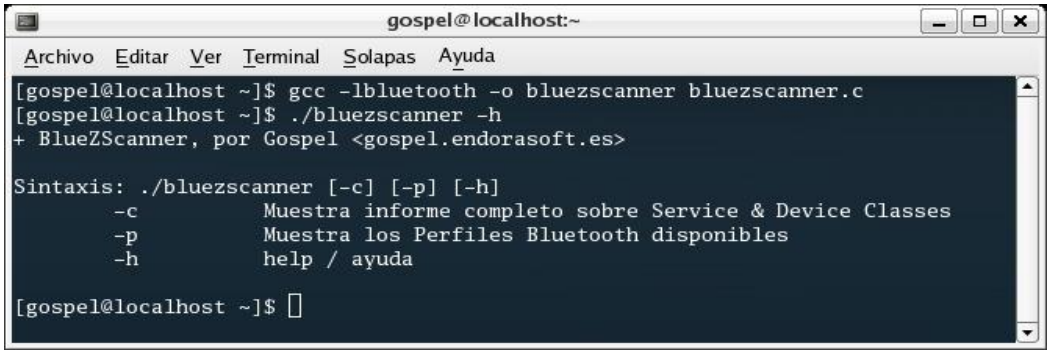


Figura 14 Ejemplo aplicación BluezScanner

2.17. Detección de dispositivos en modo visible o discoverable

El proceso de detección de dispositivos Bluetooth forma parte de las funciones de la capa HCI.

o Inicialmente, el dispositivo origen envía paquetes inquiry y se mantiene en espera de recibir respuestas de otros dispositivos presentes en su zona de cobertura.

o Si los dispositivos destino están configurados en modo visible (discoverable) se encontrarán en estado inquiry_scan y en predisposición de atender estos paquetes. En este caso, al recibir un paquete inquiry cambiarán a estado inquiry_response y enviarán una respuesta al dispositivo origen con sus direcciones MAC y otros parámetros.

o Los dispositivos que estén configurados en modo no visible (nondiscoverable) se encontrarán en modo inquiry_response y, por tanto, no responderán al dispositivo origen y permanecerán ocultos.

3. GPS

Esta tecnología la utilizamos, como es evidente, para posicionar las detecciones bluetooth que se van obteniendo. Permite realizar un posicionamiento exacto, o como en nuestro caso, gracias a varias coordenadas encuadrar dentro de un marco, las detecciones.

Vamos a hacer una breve introducción de la tecnología:

3.1. Componentes del Sistema Global de Navegación por Satélite

Figura 15 Satelistes GPS

- Sistema de satélites. Está formado por 24 unidades con trayectorias sincronizadas para cubrir toda la superficie del globo terráqueo. Más concretamente, repartidos en 6 planos orbitales de 4 satélites cada uno. La energía eléctrica que requieren para su funcionamiento la adquieren a partir de dos paneles compuestos de celdas solares adosados a sus costados.

- Estaciones terrestres. Envían información de control a los satélites para controlar las órbitas y realizar el mantenimiento de toda la constelación.

- Terminales receptores: Indican la posición en la que están; conocidas también como Unidades GPS, son las que podemos adquirir en las tiendas especializadas.

3.2. Funcionamiento GPS

El sistema GPS funciona en cinco pasos lógicos: Triangulación, Medición de distancia, Tiempo, Posición y Corrección.

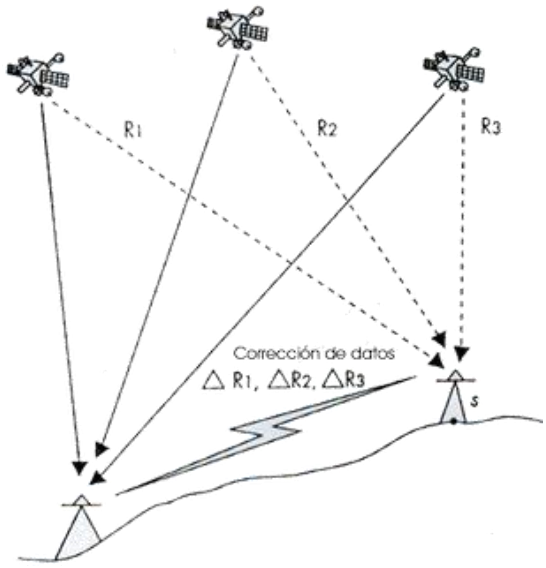


Figura 16 Funcionamiento GPS

Triangulación

Nuestra posición se calcula en base a la medición de las distancias a los satélites Matemáticamente se necesitan cuatro mediciones de distancia a los satélites para determinar la posición exacta. En la práctica se resuelve nuestra posición con solo tres mediciones si podemos descartar respuestas ridículas o utilizamos ciertos trucos. Se requiere de todos modos una cuarta medición por razones técnicas que luego veremos.

Midiendo la distancia

La distancia al satélite se determina midiendo el tiempo que tarda una señal de radio, emitida por el mismo, en alcanzar nuestro receptor de GPS. Para efectuar dicha medición asumimos que ambos, nuestro receptor GPS y el satélite, están generando el mismo Código Pseudo Aleatorio en exactamente el mismo momento. Comparando cuanto retardo existe entre la llegada del Código Pseudo Aleatorio proveniente del satélite y la generación del código de nuestro receptor de GPS, podemos determinar cuanto tiempo le llevó a dicha señal llegar hasta nosotros. Multiplicamos dicho tiempo de viaje por la velocidad de la luz y obtenemos la distancia al satélite.

Obtener un Timing Perfecto

Un timing muy preciso es clave para medir la distancia a los satélites. Los satélites son exactos porque llevan un reloj atómico a bordo. Los relojes de los receptores GPS no necesitan ser tan

exactos porque la medición de un rango a un satélite adicional permite corregir los errores de medición.

Posición Además de la distancia, el GPS necesita conocer exactamente donde se encuentran los satélites en el espacio. Orbitas de mucha altura y cuidadoso monitoreo, le permiten hacerlo.

Corrección. Finalmente el GPS debe corregir cualquier demora en el tiempo de viaje de la señal que esta pueda sufrir mientras atraviesa la atmósfera.

3.3. Lenguaje del GPS: El protocolo NMEA

La NMEA en realidad es una institución (National Marine Electronic Asociation, pero al protocolo también se le conoce por sus siglas) dedicada a establecer un estándar para la comunicación de dispositivos periféricos marinos, ya que hace algunas décadas atrás, cada marca manejaba su propia forma de comunicarse (protocolo de datos), a fin de hacer más fácil la comunicación entre diversos dispositivos (tanto a nivel del protocolo de datos como de la interfaces eléctrica), se creó esta institución en la cual participan fabricantes, distribuidores, instituciones educacionales y otros interesados en equipos periféricos marinos, esta institución no tiene ánimo de lucro, la última versión del protocolo NMEA es la 0183.

Este protocolo se caracteriza por que transmite sentencias, cada una de las sentencias comienza con "$", y termina con (CR: Carriage Retun, LF: Line Feed), los 2 caracteres que preceden a "$" son los que identifican el equipo (Para los GPS "GP"), los siguientes 3 caracteres indican la sentencia que se está enviando, hay 3 tipos de sentencias que son de consulta (Query Sentences), de origen del equipo (Proprietary Sentences) y de envío (Talker Sentences).

4. GPRS

Escogimos esta tecnología porque nos permitía de una forma rápida y sencilla el envío de la información captada al servidor ftp, gracias a los nuevos modem usb 3G.

4.1. Introducción

GPRS es una nueva tecnología que comparte el rango de frecuencias de la red GSM utilizando una transmisión de datos por medio de 'paquetes'. La conmutación de paquetes es un procedimiento más adecuado para transmitir datos, hasta ahora los datos se habían transmitido mediante conmutación de circuitos, procedimiento más adecuado para la transmisión de voz.

Los canales se comparten entre los diferentes usuarios.

En GSM, cuando se realiza una llamada se asigna un canal de comunicación al usuario, que permanecerá asignado aunque no se envíen datos. En GPRS los canales de comunicación se comparten entre los distintos usuarios dinámicamente, de modo que un usuario sólo tiene asignado un canal cuando se está realmente transmitiendo datos. Para utilizar GPRS se precisa un teléfono que soporte esta tecnología. La mayoría de estos terminales soportarán también GSM, por lo que podrá realizar sus llamadas de voz utilizando la red GSM de modo habitual y sus llamadas de datos (conexión a internet, WAP,...) tanto con GSM como con GPRS.

La tecnología GPRS, o generación 2.5, representa un paso más hacia los sistemas inalámbricos de Tercera Generación o UMTS. Su principal baza radica en la posibilidad de disponer de un terminal permanentemente conectado, tarificando únicamente por el volumen de datos transferidos (enviados y recibidos) y no por el tiempo de conexión como hemos podido observar en un punto anterior.

Obtiene mayor velocidad y mejor eficiencia de la red.

Tradicionalmente la transmisión de datos inalámbrica se ha venido realizando utilizando un canal dedicado GSM a una velocidad máxima de 9.6 Kbps. Con el GPRS no sólo la velocidad de transmisión de datos se ve aumentada hasta un mínimo 40 Kbps y un máximo de 115 Kbps por comunicación, sino que además la tecnología utilizada permite compartir cada canal por varios usuarios, mejorando así la eficiencia en la utilización de los recursos de red.

La tecnología GPRS permite proporcionar servicios de transmisión de datos de una forma más eficiente a como se venía haciendo hasta el momento.

GPRS es una evolución no traumática de la actual red GSM: no conlleva grandes inversiones y reutiliza parte de las infraestructuras actuales de GSM. Por este motivo, GPRS tendrá, desde sus

inicios, la misma cobertura que la actual red GSM. GPRS (Global Packet Radio Service) es una tecnología que subsana las deficiencias de GSM

4.2. CARACTERÍSTICAS DE GPRS

Velocidad de transferencia de hasta 144 Kbps.

Conexión permanente. Tiempo de establecimiento de conexión inferior al segundo.

Pago por cantidad de información transmitida, no por tiempo de conexión. Veamos unos ejemplos de los tamaños de información que descargaríamos:

Envío de un e-mail de 5 líneas de texto con un anexo (documento tipo de Word de 4 páginas), consumiría alrededor de 95 kbytes.

Acceder a un buscador, buscar un término (ej. viajes) y recibir una pantalla de respuesta podría ocupar 100 kbytes aproximadamente.

Recibir una hoja de cálculo (documento tipo Excel de 5 hojas), consumiría aproximadamente 250 kbytes.

Bajarse una presentación (documento tipo PowerPoint de 20 diapositivas y con fotos) equivale a unos 1.000 kbytes.

Como vemos estas características se amoldan mucho mejor para la transmisión de datos que el tradicional sistema GSM.

4.3. FUNCIONAMIENTO DE GPRS

El GPRS es la tecnología que se ha decidido utilizar para la transmisión de datos entre el sistema y el servidor. Se ha utilizado porque hoy en día está muy estandarizada y hay pequeños dispositivos 3G que se puede colocar en cualquier sitio, y por tamaños de espacio junto a que es una tecnología muy práctica para el envío de datos, se decidió utilizarla.

4.3.1. *Pila de protocolos del plano de transmisión.*

El plano de transmisión es el encargado de proveer la transmisión de los datos del usuario y su señalización para el control de flujo, detección de errores y la corrección de los mismos.

GTP: GPRS Tunneling Protocol. Es el encargado de transportar los paquetes del usuario y sus señales relacionadas entre los nodos de soporte de GPRS (GSN). Los paquetes GTP contiene los

paquetes IP o X.25 del usuario. Por debajo de él, los protocolos estándares TCP o UDP se encargan de transportar los paquetes por la red. Resumiendo, en el Backbone del GPRS tenemos una arquitectura de transporte IP/X.25-sobre-GTP-sobre-UDP/TCP-sobre IP.

SNDCP: Subnetwork Dependent Convergence Protocol. Es el encargado de transferir los paquetes de datos entre los SGSN (nodo responsable de la entrega de paquetes al terminal móvil) y la estación móvil. Las funciones que desempeña:

Multiplexación de diversas conexiones de la capa de red en una conexión lógica virtual de la capa LLC.

Compresión y descompresión de los datos e información redundante de cabecera.

AIR INTERFACE: Concierne a las comunicaciones entre la estación móvil y la BSS en los protocolos de las capas física, MAC, y RLC.

Las subcapas RLC/MAC permiten una eficiente multiplexación multiusuario en los canales de paquetes de datos compartidos, y utiliza un protocolo ARQ selectivo para transmisiones seguras a través del interfaz aire. El canal físico dedicado para tráfico en modo paquete se llama PDCH(Packet Data Channel).

En adelante se considerará la capa de enlace de datos (Data Link Layer) y la capa física (Physical Layer) como parte del Interfaz Aire Um.

DATA LINK LAYER: Capa de enlace de datos. Se encuentra entre la estación móvil (el móvil GPRS en sí) y la red.

Se subdivide en:

- La capa LLC (entre MS-SGSN): Provee una enlace altamente fiable y esta basado en el protocolo DIC e incluye control de secuencia, entrega en orden, control de flujo, detección de errores de transmisión y retransmisión. Es básicamente una adaptación del protocolo LAPDm usado en GSM.
- La capa RLC/MAC (entre MS-BSS): Incluye dos funciones. El principal propósito de la capa de Control de Radio Enlace (RLC) es la de establecer un enlace fiable. Esto incluye la segmentación y re ensamblado de las tramas LLC en bloques de datos RLC y ARQ (peticiones de retransmisión) de códigos incorregibles. La capa MAC controla los intentos de acceder de un MS a un canal de radio compartido por varios MS. Emplea algoritmos de resolución de contenciones, multiplexación de multiusuarios y prioridades según la QoS contratada.

PHYSICAL LAYER: Capa física entre MS y BSS. También se subdivide en dos subcapas.

- La capa del enlace físico (PLL) provee un canal físico. Sus tareas incluyen la codificación del canal (detección de errores de transmisión, corrección adelantada (FEC), indicación de códigos incorregibles), interleaving y la detección de congestión del enlace físico.
- La capa de enlace de radio frecuencia (RFL) trabaja por debajo de la PLL e incluye la modulación y la demodulación.

INTERFAZ BSS-SGSN: El protocolo de aplicación BSS GPRS (BSSGP) se encarga del enrutado y lo relativo a la información de la QoS entre BSS y SGSN. El servicio de red (NS) esta basado en el protocolo de Frame Relay.

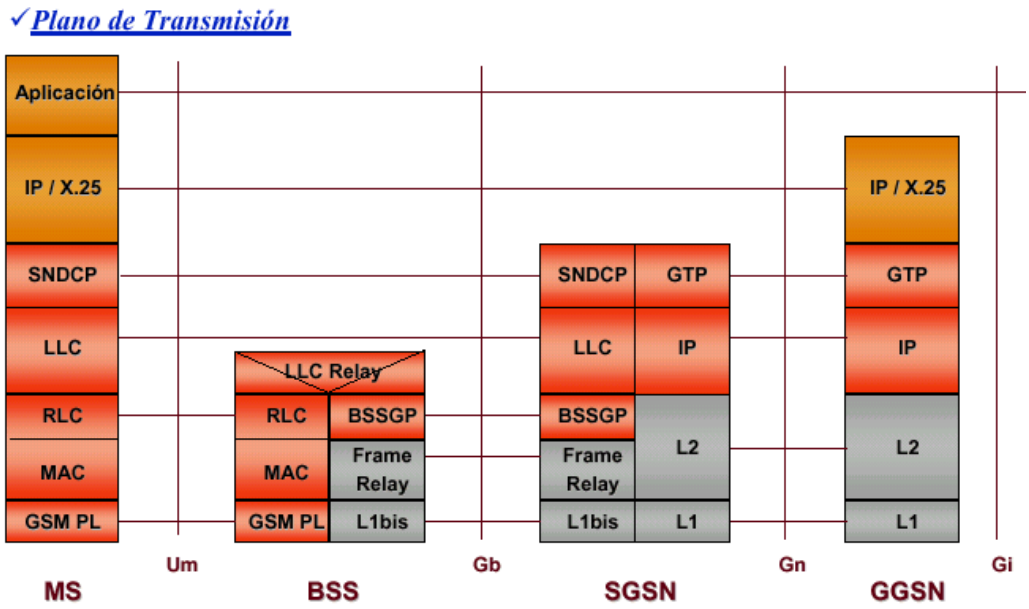


Figura 17 Plano de Transmisión GPRS

4.3.2. *Pila de protocolos del plano de SEÑALIZACIÓN.*

Se incluye en esta pila de protocolos aquellos encargados del control y mantenimiento de las funciones del plano de transmisión, conexión desconexión, activación de contexto, control de caminos de routing y localización de los recursos de la red.

GMM/SM: GPRS MOBILITY MANAGEMENT/SESSION MANAGEMENT. Es el protocolo que se encarga de la movilidad y la gestión de la sesión en momentos de la ejecución de funciones de seguridad, actualizaciones de rutas, etc.

La señalización entre SGSN y los registros HLR, VLR, y EIR utilizan los mismos protocolos que GSM con ciertas funciones ampliadas para el funcionamiento con el GPRS

✓ ***Plano de Señalización***

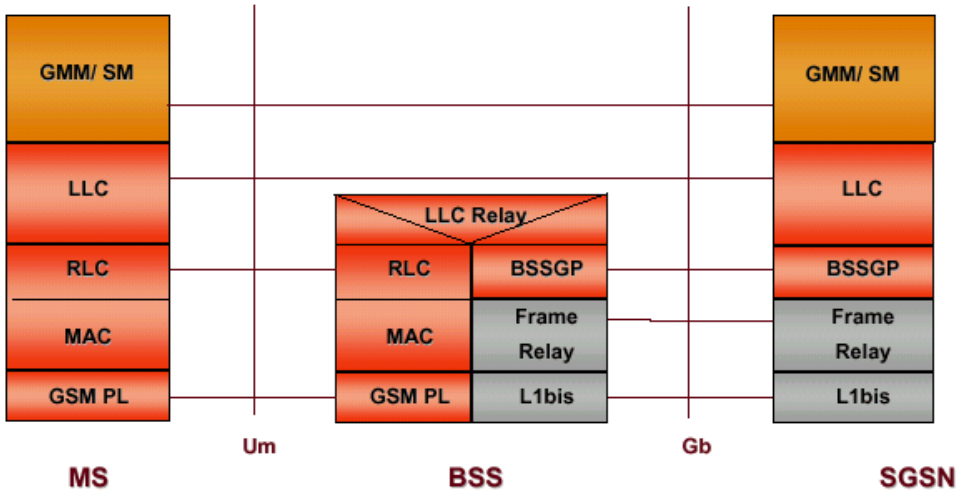


Figura 18 Plano de Señalización GPRS

4.3.3. *Concepto Maestro-Esclavo.*

Como se ha comentado en el apartado del interfaz aire, el canal físico dedicado para el tráfico en modo paquete se llama PDCH (Packet Data Channel).

Al menos 1 PDCH actúa como maestro denominado MPDCH (Master Packet Data Channel), y puede servir como PCCCH (Packet Common Control Channel), el cual lleva toda la señalización de control de necesaria para iniciar la transmisión de paquetes. Si no sirve como tal se encargará de una señalización dedicada o datos de usuario.

El resto actúan como esclavos y solo son usados para transmitir datos de usuario, en dicho caso estaremos hablando de un canal SPDCH (Slave Packet Data Channel). Se introduce el concepto de Capacity on demand; según el cual el operador puede decidir si dedica algún PDCH para trafico GPRS, y puede incrementar o disminuir el número según la demanda.

Nombre	Sentido	Función
PRACH	Ascendente	para iniciar la transferencia de datos desde el móvil
PPCH	Descendente	para informar al móvil de la entrega de paquetes.
PPRCH	Ascendente	de uso exclusivo por el móvil para responder a un paging (búsqueda)
PAGCH	Descendente	para enviar al móvil información sobre reserva de canales.
PNC	Descendente	de uso para notificaciones. MULTICAST
PBCCH	Descendente	para difundir información específica sobre GPRS. BROADCAST.

CANALES QUE COMPONEN EL SPDCH

Nombre	Sentido	Función
PDTCH	Ambos	para transferir datos desde / hacia el móvil
PACCH	Ambos	Para transportar información de señalización.
PDBCH	Descendente	Para enviar en modo de difusión, datos de usuario.

Grupo	Nombre	Dirección	Función	Maestro/ Esclavo
PBCH	PBCCH	De Bajada	Difusión	Maestro
	PDBCH	De Bajada	Difusión	Esclavo
PCCCH	PRACH	De Subida	Acceso aleatorio	Maestro
	PPCH	De Bajada	Búsqueda	Maestro
	PNCH	De Bajada	Multicast	Maestro
	PAGCH	De Bajada	Reserva	Maestro
PTCH	PDTCH	Ambos sentidos	Datos	Esclavo
	PACCH	Ambos sentidos	Control Asociado	Esclavo

Figura 19 Tabla resumen de los Canales lógicos

NOTA:

PBCH (Packet Broadcast Control Channel): Transmite información de sistema a todos los terminales GPRS en una célula.

PTCH (Packet Traffic Channels)

4.4. FLUJO DE DATOS

La unidad de datos del protocolo de la capa de red, denominada N-PDU o paquete, es recibida de la capa de red y es transmitida a través del interfaz de aire entre la estación móvil y el SGSN usando el protocolo LLC.

Primero el SNDCP transforma los paquetes en tramas LLC, el proceso incluye opcionalmente la compresión de la cabecera de datos, segmentación y encriptado.

Una trama trama LLC es segmentada en bloques de datos RLC, que son formados en la capa física, cada bloque consta de 4 ráfagas normales que son similares a las de TDMA.

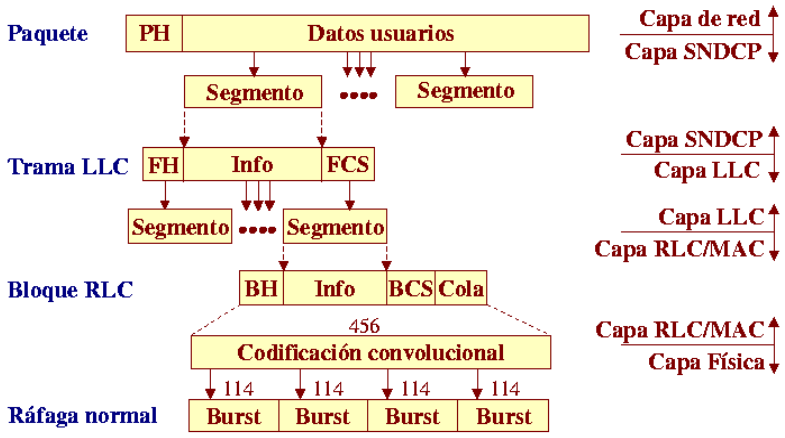


Figura 20 Esquema flujo de datos

4.5. MULTIPLEXADO DE CANALES LÓGICOS

Hay una serie de indicadores para poder hacer el multiplexado de canales lógicos y poder aprovechar al máximo las capacidades de la red.

Cuando las tramas LLC son segmentadas se asigna un TFI en la cabecera de los paquetes RLC que es único dentro de la celda, para permitir la implementación del protocolo de petición (ARQ) selectivo. Permite el multiplexado downlink.

TBF: permite identificar 1 o varias tramas LLC pertenecientes a 1 mismo usuario.

USF: permite el multiplexado uplink. Consta de 3 bits por lo que tiene 8 valores diferentes. Cada bloque RLC del downlink lleva el indicador, si el USF recibido en el downlink es igual al suyo, el usuario puede usar el siguiente bloque uplink; si es igual a FREE, el siguiente bloque es un slot

destinado al proceso de acceso (PRACH); los otros siete valores se utilizan para reservar el uplink para diferentes estaciones móviles.

4.6. CODIFICACIÓN.

Existen 4 tipos de codificación en GPRS cada una con sus características, tanto de carga útil que se codifica como el número de bits codificados. Todos los tipos siguen prácticamente los mismos pasos:

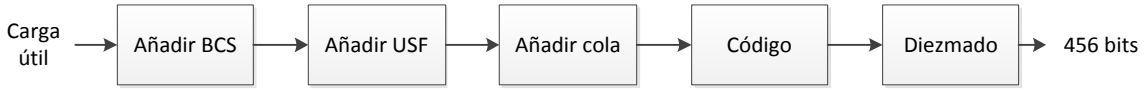


Figura 21 Esquema pasos de codificación

Las dos etapas iniciales añaden información a la carga útil:

BCS: secuencia de chequeo de bloque.

USF: Uplink state flag, ya comentada en el punto anterior.

Una vez obtenida la codificación se puede hacer el diezmado que son bits que se quitan de forma no arbitraria.

Las 4 formas de codificación de GPRS son:

- El CS-1 coincide con el SDCCH de GSM.
- El 2 y 3 son versiones perforadas del 1º.
- El 4 no utiliza código convolucional.

Tipo	Tasa código	Carga útil	BCS	USFp	Cola	Bits codif.	Bits diezm.	Tasa datos (Kbps)
CS-1	1/2	181	40	3	4	456	0	9,05
CS-2	2/3	268	16	6	4	588	132	13,4
CS-3	3/4	312	16	6	4	676	220	15,6
CS-4	1	428	16	12	0	456	0	21,4

Figura 22 Tabla formas de codificación

4.6.1. Transferencia de datos (UP-LINK).

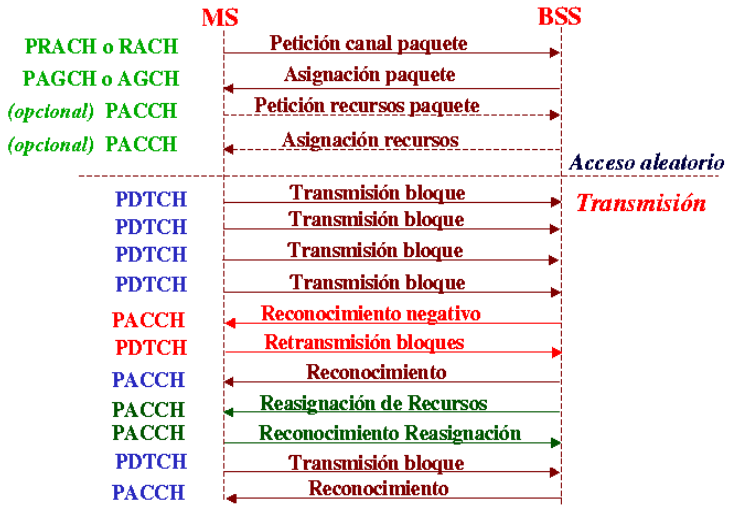


Figura 23 Esquema Transferencia de datos UP-LINK

Una estación móvil inicia una transferencia de paquetes haciendo una petición de canal de paquete en el PRACH.

La red responde en PAGCH con una o dos fases de accesos:

-1 acceso: la red responde con la asignación de paquete, que reserva los recursos en PDCH para transferir ascendentemente un nº de bloques de radio.

-2 accesos: la red responde con la asignación de paquete, que reserva los recursos ascendentes para transmitir la petición de recursos de paquete; a lo que la red responde con la asignación de recursos.

En la transmisión se realizan reconocimientos, si se recibe un reconocimiento negativo o erróneo se repite la transmisión del paquete.

4.6.2. Transferencia de datos (DOWN-LINK).

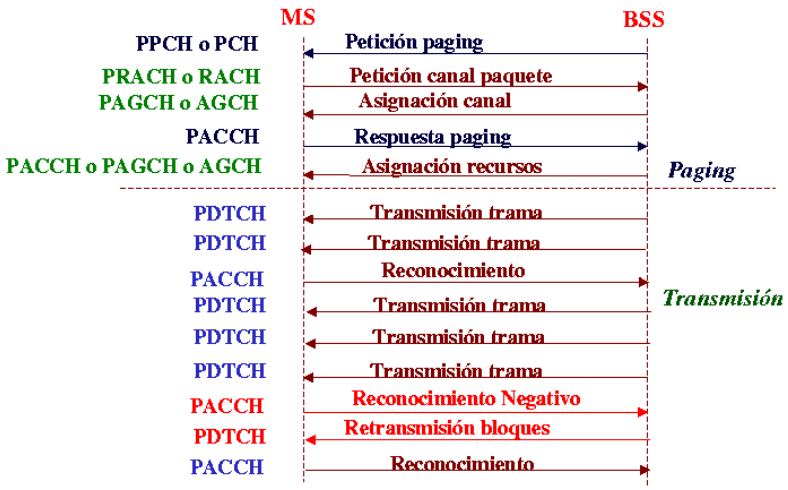


Figura 24 Esquema transferencia de datos DOWN-LINK

Una BSS inicia una transferencia de paquetes enviando una petición de paging (búsqueda) en el PPCH.

La estación móvil responde de forma muy parecida a la del acceso al paquete descrita en el punto anterior.

En la asignación de recursos se envía una trama con la lista de PDCH que son usados.

Si se recibe un reconocimiento negativo solo se retransmite los bloques erróneos.

4.7. DISCIPLINAS DE SERVICIO.

Podemos encontrar gran variedad de disciplinas de servicio, desde las más rudimentarias y poco efectivas, como son FIFO y Round Robin, hasta las más desarrolladas como MED.

Las desarrolladas en el entorno GPRS a día de hoy son las siguientes:

4.7.3. SIN PRIORIDAD.

- FIFO: Se garantiza una QoS de hasta un 30% de carga, sin embargo presenta retardos muy variables.

No existe protección entre diferentes aplicaciones de usuarios móviles.

- RR: Los paquetes se clasifican y envían a N colas garantizando una QoS de hasta un 70% de carga. A pesar de tener también retrasos variables, son inferiores al de FIFO y es más equitativo.

Los dos sistemas, sin aplicar ningún tipo de prioridad arrojan buenos resultados en condiciones de poca carga.

Sin embargo tienen problemas evidentes, como por ejemplo el caso de que FIFO no protege contra usuarios o aplicaciones abusivas que consuman mucho ancho de banda.

RR se comporta mejor por el hecho de separar los paquetes en diferentes colas.

4.7.4. CON PRIORIDAD.

Cada una tiene sus características, pero en cierto modo todos se dirigen a, en caso de congestión, evitar en mayor grado su efecto sobre los usuarios. Aunque para ello se deben definir prioridades o pesos a priori, o basándose en variaciones del tráfico.

- WRR: diferentes pesos para cada cola.
- DRR: el peso de cada cola oscila alrededor de un “deficit”.
- ARR: adopta prioridades hacia colas Round Robin.
- SJN: escoge los paquetes según su tamaño. Los paquetes pequeños se envían antes.
- SPS: una cola de cierta prioridad no se servirá hasta que todas las colas de pioridad superior están vacías.
- WPQ: igual que SPS pero ahora se limita el número de paquetes procesados para evitar la desatención de las colas menos prioritarias.

4.7.5. GARANTIZANDO QoS.

Finalmente encontramos los sistemas basados en asegurar la calidad de servicio (retardo). Para ello cada paquete entrante en el sistema recibe un “Timestamp” o un “Deadline”, que no son más que controladores de la situación del paquete dentro del sistema, indicando cuanto como máximo se puede quedar en las colas. Básicamente se diferencian en la manera de gestionar los paquetes, mientras que Virtual Clock busca el paquete y lo transmite, MED lo busca y lo envía hacia una

segunda cola de QoS. Estas disciplinas de servicios son las que mejores resultados arrojan, incluso que las “Best Effort” con prioridad, como SPS o WPQ.

Virtual Clock: garantiza el ancho de banda por conexión. A cada paquete se le asocia un “Timestamp” y en cada cola se selecciona con menor “Timestamp”.

MED: Aquí a cada paquete se le asigna un “Deadline” y si se cumple dicho valor, este se pone en su cola de QoS.

5. Conclusión

Lo que se ha pretendido en este capítulo ha sido realizar una breve recapitulación de las tecnologías utilizadas, junto con una descripción de las mismas para situar el marco de desarrollo de este proyecto y poder entenderse mejor porque se han utilizado estas tecnologías y no otras para afrontar la problemática para la que ha sido creado este proyecto.

CAPÍTULO 3. ENTORNO DE DESARROLLO

1. Introducción

El entorno de desarrollo que se ha usado para esta parte del proyecto consta de una máquina Linux donde se guardan todos los procesos de la aplicación cliente. Al ser una máquina en Linux está basado en Unix como sistema de la máquina por lo que no hemos usado ningún framework para desarrollar nuestros procesos tales como Netbeans, Eclipse, etc. ya que con un editor de texto es suficiente, en nuestro caso usamos el gedit, ya que incluye un corrector ortográfico multilenguaje y un flexible sistema de plugins que permite añadir características a la aplicación

Porqué elegir Linux y no Windows es bien sencillo, Linux es un sistema mucho más robusto gracias a su estabilidad, su gran nivel de seguridad y el cumplimiento de estándares, especialmente en lo que se refiere a redes, y orientado al desarrollo de software, y el sistema Unix permite múltiples tareas, lo que significa que permite que en un único equipo se ejecuten simultáneamente varios programas a cargo de un único usuario, esta característica nos ha sido muy útil para ejecutar varios script a la vez, como ya explicaremos más adelante y también aprovechamos que está escrito en C lo que le hacía más atractivo ya que nuestra aplicación se ha creado utilizando este lenguaje.

El procedimiento a seguir en estos casos es crear el proceso y después el ejecutable con una herramienta de compilación de Unix RETOCAR make es una herramienta de generación o automatización de código, muy usada en los sistemas operativos tipo Unix/Linux creado también por nosotros.

A continuación haremos un breve repaso por los diferentes componentes usados, aunque ya sean conocidos por todo el mundo.

2. Gedit

Gedit es un editor de textos compatible con UTF-8 para GNU/Linux, Mac OS X y Microsoft Windows. Diseñado como un editor de textos de propósito general, gedit enfatiza la simplicidad y facilidad de uso. Incluye herramientas para la edición de código fuente y textos estructurados, como lenguajes de marcado. Es el editor predeterminado de GNOME.

Distribuido bajo las condiciones de la licencia GPL, gedit es software libre.

2.1. Características principales

Además de las funcionalidades básicas que son habituales en un editor de texto, como copiar, cortar y pegar texto, imprimir, etc., gedit incorpora coloreado de sintaxis para diversos lenguajes de programación y marcado. gedit también posee pestañas en su interfaz para editar múltiples archivos a la vez. Puede editar archivos de manera remota usando la biblioteca GVFS. Otras características orientadas al código incluyen numeración de líneas, resaltado de la línea actual, indentación automática y copiado de seguridad del archivo.[2]

Además, gedit incluye un corrector ortográfico multilenguaje y un flexible sistema de plugins que permite añadir características a la aplicación.[2] Además de los complementos incluidos en gedit, hay más disponibles para descargar

3. Linux

Es un sistema operativo de tiempo compartido, controla los recursos de una computadora y los asigna entre los usuarios. Permite a los usuarios correr sus programas. Controla los dispositivos de periféricos conectados a la máquina.

3.1. Características Principales

Posee las siguientes características:

- Es un sistema operativo multiusuario, con capacidad de simular multiprocesamiento y procesamiento no interactivo.

- Está escrito en un lenguaje de alto nivel C.

- Dispone de un lenguaje de control programable llamado SHELL.

- Ofrece facilidades para la creación de programas y sistemas y el ambiente adecuado para las tareas de diseños de software.

- Emplea manejo dinámico de memoria por intercambio o paginación.

- Tiene capacidad de interconexión de procesos.

- Permite comunicación entre procesos.

- Emplea un sistema jerárquico de archivos, con facilidades de protección de archivos, cuentas y procesos.

- Tiene facilidad para redireccionamiento de Entradas/Salidas.

- Garantiza un alto grado de portabilidad.

3.2. Sistema

El sistema se basa en un Núcleo llamado Kernel, que reside permanentemente en la memoria, y que atiende a todas las llamadas del sistema, administra el acceso a los archivos y el inicio o la suspención de las tareas de los usuarios.

3.3. Comunicación con el Sistema

La comunación con el sistema UNIX se da mediante un programa de control llamado SHELL. Este es un lenguaje de control, un intérprete, y un lenguaje de programación, cuyas características lo hacen sumamente flexible para las tareas de un centro de cómputo. Como lenguaje de programación abarca los siguientes aspectos:

- Ofrece las estructuras de control normales: secuenciación, iteración condicional, selección y otras.

- Paso de parámetros.

- Sustitución textual de variables y Cadenas.

- Comunicación bidireccional entre órdenes de shell.

El shell permite modificar en forma dinámica las características con que se ejecutan los programas en UNIX:

Las entradas y salidas pueden ser redireccionadas o redirigidas hacia archivos, procesos y dispositivos;

Es posible interconectar procesos entre sí.

Diferentes usuarios pueden "ver" versiones distintas del sistema operativo debido a la capacidad del shell para configurar diversos ambientes de ejecución. Por ejemplo, se puede hacer que un usuario entre directamente a su sección, ejecute un programa en particular y salga automáticamente del sistema al terminar de usarlo.

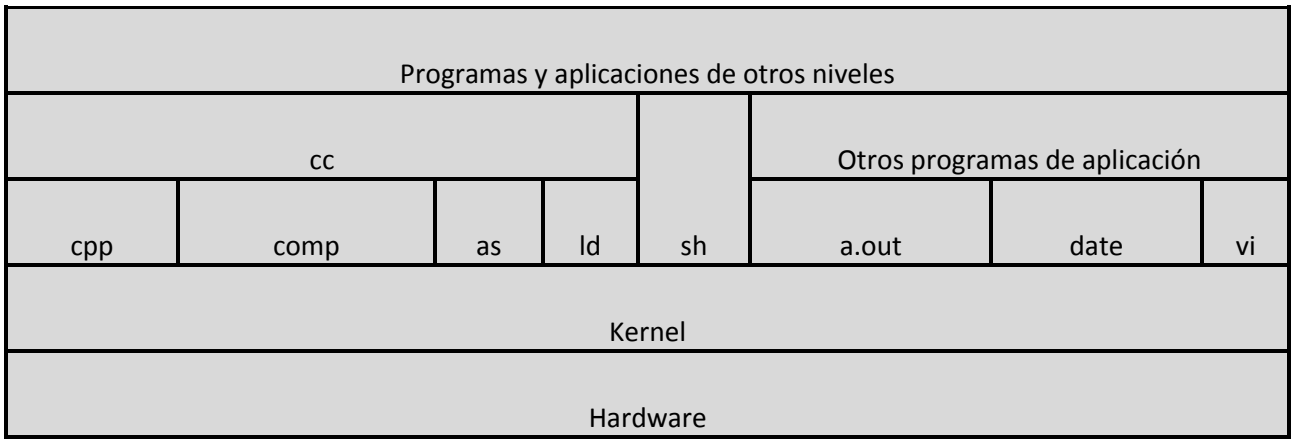


Figura 25 Arquitectura del Sistema Unix

El sistema está formado por varias aplicaciones que corren en paralelo o son llamadas por la aplicación principal.

4. Lenguaje C

C es un lenguaje de programación creado en 1972 por Dennis M. Ritchie en los Laboratorios Bell como evolución del anterior lenguaje B, a su vez basado en BCPL.

Al igual que B, es un lenguaje orientado a la implementación de Sistemas Operativos, concretamente Unix. C es apreciado por la eficiencia del código que produce y es el lenguaje de programación más popular para crear software de sistemas, aunque también se utiliza para crear aplicaciones.

Se trata de un lenguaje débilmente tipificado de medio nivel pero con muchas características de bajo nivel. Dispone de las estructuras típicas de los lenguajes de alto nivel pero, a su vez, dispone de construcciones del lenguaje que permiten un control a muy bajo nivel. Los compiladores suelen ofrecer extensiones al lenguaje que posibilitan mezclar código en ensamblador con código C o acceder directamente a memoria o dispositivos periféricos.

La primera estandarización del lenguaje C fue en ANSI, con el estándar X3.159-1989. El lenguaje que define este estándar fue conocido vulgarmente como ANSI C. Posteriormente, en 1990, fue ratificado como estándar ISO (ISO/IEC 9899:1990). La adopción de este estándar es muy amplia por lo que, si los programas creados lo siguen, el código es portátil entre plataformas y/o arquitecturas.

4.1. Propiedades

Un núcleo del lenguaje simple, con funcionalidades añadidas importantes, como funciones matemáticas y de manejo de archivos, proporcionadas por bibliotecas.

- Es un lenguaje muy flexible que permite programar con múltiples estilos. Uno de los más empleados es el estructurado "no llevado al extremo" (permitiendo ciertas licencias de ruptura).
- Un sistema de tipos que impide operaciones sin sentido.
- Usa un lenguaje de preprocesado, el preprocesador de C, para tareas como definir macros e incluir múltiples archivos de código fuente.
- Acceso a memoria de bajo nivel mediante el uso de punteros.
- Interrupciones al procesador con uniones.
- Un conjunto reducido de palabras clave.
- Por defecto, el paso de parámetros a una función se realiza por valor. El paso por referencia se consigue pasando explícitamente a las funciones las direcciones de memoria de dichos parámetros.
- Punteros a funciones y variables estáticas, que permiten una forma rudimentaria de encapsulado y polimorfismo.
- Tipos de datos agregados (struct) que permiten que datos relacionados (como un empleado, que tiene un id, un nombre y un salario) se combinen y se manipulen como un todo (en una única variable "empleado").

4.2. Carencias

- Recolección de basura nativa, sin embargo se encuentran a tal efecto bibliotecas como la "libgc" desarrollada por Sun Microsystems, o el Recolector de basura de Boehm.
- Soporte para programación orientada a objetos, aunque la implementación original de C++ fue un preprocesador que traducía código fuente de C++ a C.
- Funciones anidadas, aunque GCC tiene esta característica como extensión.
- Soporte nativo para programación multihilo.

Aunque la lista de las características útiles de las que carece C es larga, este factor ha sido importante para su aceptación, porque escribir rápidamente nuevos compiladores para nuevas plataformas, mantiene lo que realmente hace el programa bajo el control directo del programador, y permite implementar la solución más natural para cada plataforma. Ésta es la causa de que a menudo C sea más eficiente que otros lenguajes. Típicamente, sólo la programación cuidadosa en lenguaje ensamblador produce un código más rápido, pues da control total sobre la máquina, aunque los avances en los compiladores de C y la complejidad creciente de los microprocesadores modernos han reducido gradualmente esta diferencia

4.3. Proceso de Compilación

La compilación de un programa C se realiza en varias fases que normalmente son automatizadas y ocultadas por los entornos de desarrollo:

Preprocesado consistente en modificar el código fuente en C según una serie de instrucciones (denominadas directivas de preprocesado) simplificando de esta forma el trabajo del compilador. Por ejemplo, una de las acciones más importantes es la modificación de las inclusiones (`#include`) por las declaraciones reales existentes en el archivo indicado.

Compilación que genera el código objeto a partir del código ya preprocesado.

Enlazado que une los códigos objeto de los distintos módulos y bibliotecas externas (como las bibliotecas del sistema) para generar el programa ejecutable final.

4.4. Herramientas de Programación

Al programar en C, es habitual usar algunas herramientas de programación de uso muy extendido, sobre todo en entorno de tipo unix:

make: Herramienta para automatizar el proceso de compilación, enlazado, etc.

lint: Herramienta utilizada para detectar código sospechoso, confuso o incompatible entre distintas arquitecturas

valgrind: Herramienta utilizada para detectar posibles fugas de memoria.

gdb : Debugger de GNU utilizado para seguir la ejecución del programa.

dbx : Debugger que suele venir instalado con todos los UNIX.

ddd : Interfaz gráfico para el depurador gdb o dbx.

4.5. Aplicabilidad

Hecho principalmente para la fluidez de programación en sistemas UNIX. Se usa también para el desarrollo de otros sistemas operativos como Windows o GNU/Linux. Igualmente para aplicaciones de escritorio como GIMP, cuyo principal lenguaje de programación es C.

De la misma forma, es muy usado en aplicaciones científicas (para experimentos informáticos, físicos, químicos, matemáticos, entre otros, parte de ellos conocidos como modelos ysimuladores), industriales (industria robótica, cibernética, sistemas de información y base de datos para la industria petrolera y petroquímica. Predominan también todo

lo que se refiere a simulación de máquinas de manufactura), simulaciones de vuelo (es la más delicada, ya que se tienen que usar demasiados recursos tanto de hardware como de software para desarrollar aplicaciones que permitan simular el vuelo real de una aeronave. Se aplica por tanto, en diversas áreas desconocidas por gran parte de los usuarios noveles.

Los ordenadores de finales de los 90 son varios órdenes de magnitud más potentes que las máquinas en que C se desarrolló originalmente. Programas escritos en lenguajes de tipo dinámico y fácil codificación (Ruby, Python, Perl...) que antaño hubieran resultado demasiado lentos, son lo bastante rápidos como para desplazar en uso a C. Aun así, se puede seguir encontrando código C en grandes desarrollos de animaciones, modelados y escenas en 3D en películas y otras aplicaciones multimedia.

Actualmente, los grandes proyectos de software se dividen en partes, dentro de un equipo de desarrollo. Aquellas partes que son más "burocráticas" o "de gestión" con los recursos del sistema, se suelen realizar en lenguajes de tipo dinámico o de guión (*script*), mientras que aquellas partes "críticas", por su necesidad de rapidez de ejecución, se realizan en un lenguaje de tipo compilado, como C o C++. Si, después de hacer la división, las partes críticas no superan un cierto porcentaje del total (aproximadamente el 10%) entonces todo el desarrollo se realiza con lenguajes dinámicos. Si la parte crítica no llega a cumplir las expectativas del proyecto, se comparan las alternativas de una inversión en nuevo hardware frente a invertir en el coste de un programador para que reescriba dicha parte crítica.

4.6. Bibliotecas

Una biblioteca de C es una colección de funciones utilizadas en el lenguaje de programación C. Las bibliotecas más comunes son la biblioteca estándar de C y la biblioteca del estándar ANSI C, la cual provee las especificaciones de los estándares que son ampliamente compartidas entre bibliotecas. La biblioteca **ANSI C** estándar, incluye funciones para la entrada y salida de archivos, alojamiento de memoria y operaciones con datos comunes: funciones matemáticas, funciones de manejo de cadenas de texto y funciones de hora y fecha.

Otras bibliotecas C son aquellas utilizadas para desarrollar sistemas Unix, las cuales proveen interfaces hacia el núcleo. Estas funciones son detalladas en varios estándares tales como **POSIX** y el **Single UNIX Specification.**

Ya que muchos programas han sido escritos en el lenguaje C existe una gran variedad de bibliotecas disponibles. Muchas bibliotecas son escritas en C debido a que C genera código objeto rápido; los programadores luego generan interfaces a la biblioteca para que las rutinas puedan ser utilizadas desde lenguajes de mayor nivel, tales como Java, Perl y Python.

5. Conclusión

En este capítulo se ha expuesto el entorno de desarrollo a utilizar y las herramientas necesarias para desarrollarlo.

Como hemos visto nos hemos decantado por un entorno Unix (Linux) y un lenguaje muy estandarizado en el ámbito de los procesos como es el lenguaje C. Que este lenguaje siga siendo a día de hoy el más utilizado para procesos y no haya sido desbancado por otros más modernos como Java, es debido a su mejor utilización de los recursos al no estar enfocado a una ámbito gráfico y centrarse en realizar una comunicación rápida y eficaz de los procesos.

CAPÍTULO 4. SISTEMA

1. Introducción

Esta parte del Sistema SAUMI consta como ya se ha dicho de una estación de captación que irá empotrada dentro del tren.

Esto requería que el empotrado fuese de dimensiones reducidas, pero el principal problema al que me enfrentaba era que tenía que soportar Linux ya que nuestra aplicación estaba creada para este sistema, por lo que después de hacer una búsqueda por Internet encontramos un modelo entre otros del mercado que cubría a la perfección nuestras necesidades la mini-ITX Jetway NC94FL.

2. Placa Mini ITX Board

Figura 26 CPU del Sistema

La mini-ITX Jetway NC94FL es la primera placa madre Intel Atom con una ranura de expansión PCI Express x16. Debido a limitaciones del chipset, la ranura de expansión tiene x4 carriles eléctricos, pero el NC94FL le da flexibilidad al tener una ranura x16 por lo que se puede instalar cualquier tarjeta de expansión que desee. Esta flexibilidad permite que la expansión NC94FL para ser utilizado en una variedad de aplicaciones, ya que permite integrar cualquier tarjeta PCI Express para crear una solución única de bajo consumo y bajo coste. Ideal para tarjetas de expansión PCIe como los controladores RAID para el almacenamiento, las tarjetas de captura de vídeo para vigilancia por video, aceleradores TCPIP o multi-puerto de tarjetas de red para aplicaciones de red, tarjetas de sonido para aplicaciones de gama alta de audio, PBX / VoIP,

tarjetas de vídeo para la señalización digital , imágenes, o aplicaciones financieras. El NC94FL también es compatible con montaje en rack 1U

Jetway NC94FL Mini-ITX	
Procesador	Atom D525 1.66/1.80GHz (antes Pineview-D)
Chipset	Intel NM10 (antes Tiger Point) @ 2.1W
Memoria	2 x DDR2 667/800 DIMM ranura (hasta 4 GB)
Gráficos	Intel Integrated Graphics GMA3150
Audio	Realtek ALC662 5.1 canales de audio
LAN	Intel 82552V Ethernet 10/100Mbit
Almacenamiento	Intel NM10 2 SATA2 de 3 Gb / s puertos

Conectores del panel posterior

- **1 ratón PS / 2**
- **1 PS / 2 Teclado**
- **1 Paralelo**
- **1 Serie**
- **1 VGA (hasta 2048x1536)**
- **4 USB 2.0**
- **1 conector RJ45 LAN (10/100)**
- **3 de audio de 3.5mm**

Conexiones en Placa

- **1 ranura PCI-E x16 (por 4)**
- **1 conector de floppy**
- **1 serie (RS232/RS422/RS485)**
- **2 SATA2 3Gb / s**
- **2 USB con capacidad para 4 puertos USB 2.0 adicionales**
- **1 LVDS de 18 bits de un canal de cabecera LVDS / inversor**
- **1 x 9 pines Audio**
- **1 x 4 pines CD de audio**
- **1 GPIO**
- **1 panel frontal**
- **1 LPC**
- **3 Conectores para ventilador**
- **1 x 20-pines ATX**

Segmentos de mercado compatibles

- **Digital Seguridad y Vigilancia**
- **Digital Signage**
- **Juegos**
- **Automatización Industrial / Control**
- **Médico**
- **Retail (POS / quiosco / ATM)**
- **En los vehículos de información y entretenimiento**
- **Networking / Storage Server / Servidor de correo / servidor de impresión**
- **Thin Client**

- **Nettop**

3. Bluetooth DONGLE

Después de experimentar con varios modelos de bluetooth, me decanté por este modelo de tipo usb por necesidades físicas del sistema, ya que me vi en la tesitura de tener que crear un sistema más pequeño y portátil, y este encajaba por tamaño y cumplía a la perfección con las necesidades técnicas de la aplicación.

Figura 27 Dispositivo Bluetooth

Mini USB 2.0 Bluetooth V2.0 EDR Dongle Wireless Adapter (Black)
Descripción: Este adaptador Bluetooth es un tamaño de una uña, este dispositivo se conecta al ordenador portátil puede o tableta de puerto USB. Este adaptador Bluetooth utiliza el estándar Bluetooth v2.0 + EDR y USB 2.0 para ofrecer menores tiempos de conexión y hasta tres veces más rápida velocidad de transferencia de datos de adaptadores v1.2.
Características principales: Diseño increíble y compacto, forma pequeña que hace lo que se puede dejar en su computadora portátil, incluso cuando no esté en uso. Bluetooth v2.0 con A2DP y con un alcance de hasta 10m. Al mismo tiempo se conecta Bluetooth a otros dispositivos Bluetooth como teléfonos móviles, PDA o PC para la transferencia de datos, redes, acceso telefónico y de fax. Bluetooth compatible con los datos de voz. Convertir cualquier PC sin la tecnología Bluetooth en un PC compatible con Bluetooth.
Ficha técnica: Banda de frecuencia 2.4 GHz. I / O Interface: USB versión 2.0. Compatible con A2DP (Perfil de distribución de Audio Avanzada). Potencia de entrada: DV 5V (USB Power).

Rango de funcionamiento: hasta 30 m (98.4FT.).
Soporte de sistemas operativos: Windows 2000 / XP / Vista.
Spread Spectrum: FHSS (Frequency Hopping Spread Spectrum).
Rango de transmisión: 20m.
Velocidad de datos: Hasta 3Mbps (El rango de operación máximo depende de los factores del medio ambiente.)
Bluetooth estándar: Bluetooth v2.0 + EDR modo de seguridad.

4. GPS HAICOM HI-204 III

Descripción

Como ya se ha comentado en otros puntos tuvimos problemas sobre todo con el GPS, y este fue el modelo que se terminó escogiendo porque es ideal para usarlo en dispositivos PC portátiles (encontramos incompatibilidades con otros equipos para conseguir su correcto funcionamiento) es compatible con Linux (nuestro entorno de desarrollo), se puede usar en una gran variedad de aplicaciones para navegación de vehículos, esto me llamó la atención porque pensaba que sería capaz de funcionar en una aplicación propia, como la nuestra, es ideal para Localización de vehículos, lo que se iba a crear era eso y por último y quizás una de las cosas más importantes es que era compatible al formato NMEA-0183, que se iba a usar para convertirlo a coordenadas UMTS utilizadas por Google.

Figura 28 Dispositivo GPS

PRESTACIONES

- Receptor de 20 canales paralelos
- Soporte SBAS (WAAS, EGNOS)
- Sensibilidad -159dBm
- < 8 segundos arranque en caliente
- < 40 segundos arranque en frío

Elemento incluidos:

- Receptor GPS HI-204III-USBcon conexión DB9 + cable conversor serie RS-232
- Ventosa de sujeción
- Manual de usuario y CD (con programa de pruebas Hai Test y controladores)

Especificaciones técnicas:

Chipset	GSP3F	SiRF StarIII technology
General	Frecuencia	L1, 1575.42 MHz
	C/A code	1.023 MHz chip rate
	Canales	20
Precisión		10 metros, 2D RMS
	Posición	5 metros 2D RMS, con correcciónWAAS
		<5meters(50%), corrección DGPS
	Velocidad	0.1 metros/segundo
	Reloj	1 microsegundo sincronizado con el reloj GPS
Datum	Por defecto	WGS-84
		configurable por software
Tiempo de a aquisición (en cielo abierto)	Actualización	0.1 seg. media
	Arranque inmediato	1 seg. media
	Arranque en caliente	8 seg. media
	Arranque en templado	38 seg. media
	Arranque en frío	42 seg. media
Condiciones dinámicas	Altura	18,000 metros (60,000 pies) max.
	Velocidad	515 metros/seg (1000 nudos) max.
	Aceleración	4g, max.
Alimentación	Tensión de alimentacion	5V CC.
	Consumo	0.38 W
	Intensidad	75mA
	Alimentación Backup	3 V Lithium-Ion recargable
Dimensiones	43mm L x 42mm W x 13mm H	
Interface	USB	
Led indicador	Parpadeo 1 vez por segundo: posicion Fijada	
	Parpadeo 4 vezes por segundo: buscando satélites	

Una vez que teníamos seleccionado el empotrado y los dispositivos nos quedaba la tarea de conectar todos los dispositivos a la CPU y esta a su vez al autobús.

Aquí encontramos otro inconveniente ya que la única salida de alimentación en el tren es a 12V por lo que se ha tenido que crear un transformador a 5V.

Con estas modificaciones y una vez montado el sistema SAUMI quedo de la siguiente forma:

Figura 29 Sistema SAUMI complete, vista trasera

En la foto se puede apreciar el bluetooth usb y el modem.

El MODEM se uso para las pruebas en el laboratorio pero para las pruebas de campo sería sustituido por un usb 3G para el envío de datos al servidor.

Figura 30 Dispositivo GPRS

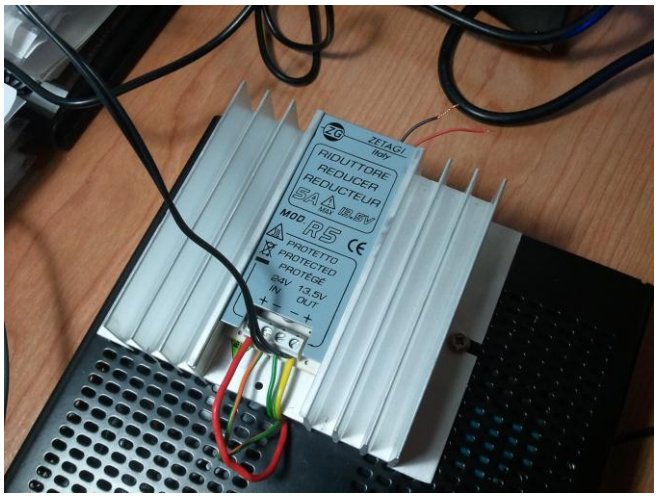


Figura 31 Adaptador externo de alimentación

5. Conclusión

En este capítulo se ha mostrado el sistema físico del proyecto, como ha quedado una vez finalizado, haciendo una breve exposición de los recursos hardware elegidos en base a las problemáticas surgidas con otros dispositivos probados sin éxito.

CAPÍTULO 5. APLICACIÓN

1. Introducción

En este capítulo se realizará una explicación del software creado para la consecución de los objetivos planteados inicialmente.

Veremos una serie de script que se han creado para la automatización de procesos y comprobaciones y explicaremos los 3 principales procesos de que se compone esta aplicación.

2. Diseño funcional

En este proyecto se ha elegido el lenguaje de programación C, debido a que es muy veloz y potente, lo que permite un software efectivo y que consume pocos recursos del ordenador, esto es vital para nuestra aplicación debido a que está ejecutándose continuamente.

La aplicación está corriendo en los empotrados y comienza a realizar detecciones periódicas de dispositivos BT cuando el autobús parte de una parada (se sabrá que parte de una parada por un algoritmo de posicionamiento y por la velocidad del autobús).

Paralelamente se está ejecutando otra aplicación para establecer el posicionamiento GPS de dichas detecciones, y así englobarlas en diferentes ficheros para hacer más fácil su posterior tratamiento.

Tomando como ejemplo la línea de trenes, la información de estas detecciones será almacenada por filas en un fichero de texto con el siguiente formato:

"Núm. tren"+"Pos. GPS"+"Hora"+"MAC"+"Id_detección"

Haciendo una breve descripción del proceso desde su ejecución hasta el envío de la información seguiría la siguiente secuencia:

1. Lanzamiento del sistema (modo background).
2. Detección del usuario.
3. Procesado de la trama.
4. Creación/modificación del fichero con la nueva trama a enviar.
5. Envío del fichero vía GPRS.

3. Partes de la Aplicación

El sistema SAUMI está compuesto una estación de detección instalada en cada autobús.

Estaciones de detección: *Sistema empotrado (aplicación en C)*

- Dispositivo BT.
- Dispositivo GPS.
- Terminal GPRS (envío de las detecciones)

Como ya hemos comentado anteriormente se eligió un empotrado de tamaño muy reducido pero cumpliendo los requerimientos. La ubicación que se pensó fue el puesto de conducción, ya que es ahí donde está la salida de 12v a la que iba a estar enchufada, pero tuvimos que cambiar el lugar para el empotrado debido a que tenía más potencial de señal en el techo del autobús.

4. Desarrollo del software

Para conseguir un funcionamiento efectivo de la aplicación se han creado una serie de script unix, lo que permite que la aplicación se ejecute automáticamente una vez que se enciende el empotrado, esto era primordial para no tener que depender de que una persona gestionara insitu el sistema, sino que es la propia aplicación la que detecta si se ha producido algún fallo, asi si la aplicación falla o pierde algún dato como por ejemplo la posición GPS, se resetea automáticamente esa parte de la aplicación para volver a enlazar la conexión con el satélite. Consiguiendo así un sistema más robusto y eficaz.

Se dispone de una aplicación principal, que se encargará de la detección de dispositivos BT, realizar las pertinentes comprobaciones de las diversas tecnologías, y generación del fichero a enviar con su correcto formato.

Una aplicación secundaria, que se encargará de la lectura de las coordenadas GPS, modificación del fichero a enviar según se trate de un recorrido de ida o de vuelta, y de realizar una serie de comprobaciones necesarias para que la aplicación pueda ejecutarse correctamente (lectura de fichero de paradas de una determinada línea, algoritmo de posicionamiento para detectar cuando estamos en una parada y cuando no, y comprobación de si es la última parada del recorrido)

Procedemos ahora a explicar las distintas partes que conforman nuestra aplicación:

4.1. Script UNIX complementarios

Se han creado varios script para automatizar el sistema y crear un sistema robusto y eficiente, esto se ha ido consiguiendo por medio de diversos fallos que fueron surgiendo a lo largo de la etapa de creación del software, lo que nos llevó a ir realizando modificaciones de la aplicación.

Los script creados son:

- Arrancar Sistema
- Infokern
- Informes
- Paradas
- Scripftp
- Scriplimpieza
- Watchdog

4.1.1. Arrancar Sistema

Es el script principal, se encarga de lanzar la aplicación principal (Cliente), aplicación secundaria (paradas), y la aplicación que recoge las coordenadas GPS y velocidad del vehículo, así como ejecutar los script ftp y de limpieza.

La principal funcionalidad de este script es comprobar si estas aplicaciones están activas o no, y en caso de que no lo estén activarlas.

Esto se consigue por medio de una pequeña comprobación, una llamada al sistema para ver si el demonio del servicio en cuestión está corriendo o no.

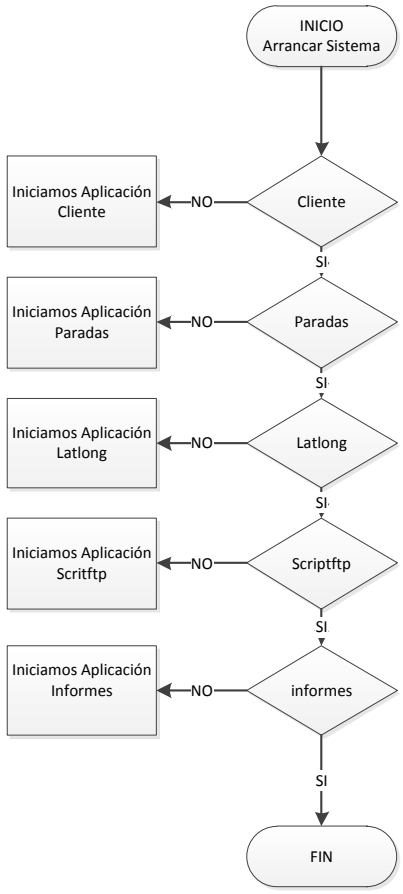


FIGURA 32 Arrancar Sistema

Veamos un ejemplo con la aplicación Cliente:

```
#!/bin/sh
cd …/

# Aplicacion de detecciones BT
SERVICE='Cliente'
if ps ax | grep -v grep | grep $SERVICE > /dev/null
then
echo "El servicio $SERVICE esta ejecutandose"
else
echo "El programa $SERVICE esta detenido voy a arrancarlo"
…/Cliente2  &
echo "arrancado cliente"
fi
…
```

4.1.2. scripftp

Este script se encarga de enviar la información captada comprimida al Servidor vía FTP.

Este proceso se realiza diariamente.

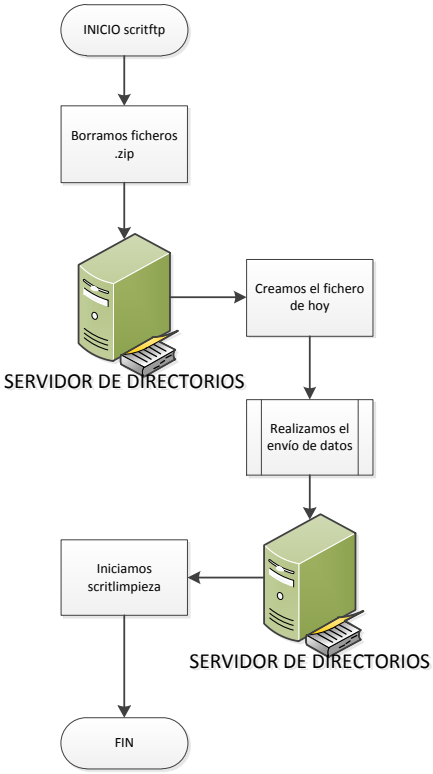


FIGURA 33 Scriptftp

Primeramente entramos al directorio donde vamos guardando la información y borramos todos los archivos con extensión zip que haya en la carpeta (envíos realizados anteriormente).

Seguidamente procedemos a crear el archivo comprimido con la información de hoy.

A continuación conectamos con el Servidor vía Ftp y realizamos el envío de la información.

```
…
lftp -u $FTPU,$FTPP -e "mput …/$DIA.zip;
quit" $FTPS
…
```

donde las variables FTPU y FTPP son las de usuario y password.

Por último llamamos al scriptlimpieza.

```
…
sleep 2
…/scriptlimpieza  > /dev/null &
…
```

4.1.3. scriptlimpieza

Es un script que de forma sencilla realiza una copia de seguridad de los archivos que son anteriores al día de hoy, guardándolos en la carpeta log.

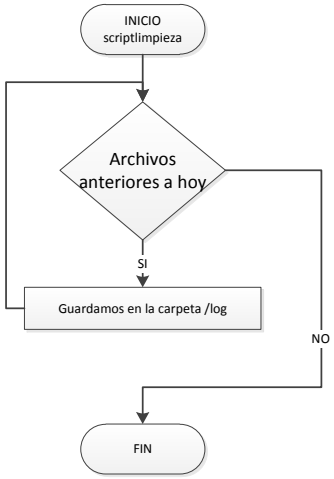


FIGURA 34 Scritplimpieza

```
…
#mover los .txt antiguos a log sin tocar los actuales
for i in $(ls --ignore=*$DIA | grep .txt);
do
mv $i …/log/$i
done
…
```

4.1.4. informes

Realiza el envío de los informes guardados en el directorio log de nuestro servidor al Servidor principal por medio de conexión ftp.

FIGURA 35 Informes

Utilizamos el mismo comando que para el scripftp.

```
…
lftp –u $FTPU,$FTPP -e "mput …/parada_actual.log;
quit" $FTPS
…
```

Con este script nos aseguramos que la información queda guardada en el Servidor, para posteriores consultas, como copia de seguridad en la nube, u otras diversas opciones.

4.1.5. watchdog

Se creó pensando en hacer la aplicación más robusta y a prueba de fallos.

Su función es solucionar los posibles problemas de conectividad a internet que puedan surgir.

Cuando se produce una pérdida de conexión el programa llama a este script, que comprueba si hay conexión, si el empotrado sigue sin conexión esperamos durante un tiempo prudente y sino recupera la conexión reiniciamos el sistema.

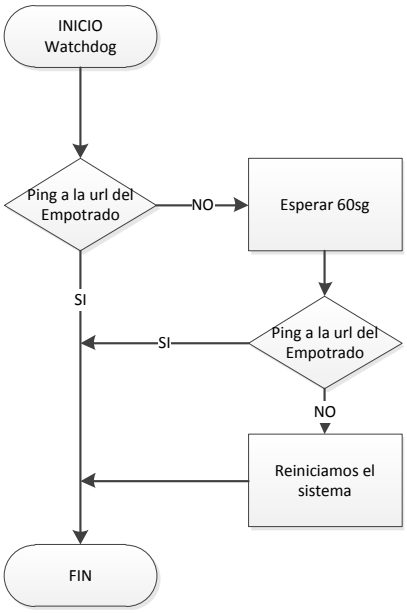


FIGURA 36 Watchdog

```
…
if ping -c 2  8.8.8.8 > /dev/null
then
      echo "Ping a 8.8.8.8 da OK"
else
      echo "Ping a 8.8.8.8 perdió un paquete"
      ## Esperar 1 segundos y probar de nuevo
      sleep 1
      if ! ping -c 2  8.8.4.4 >/dev/null
      then
            echo "8.8.4.4 conectando el modem"
            sudo /home/proeyectosidi1/Escritorio/watchdog5
      fi
fi
…
```

4.2. Aplicación Cliente

La aplicación Cliente, es la aplicación principal y es la que se encarga de gestionar la captación de los dispositivos bluetooth encontrados. Es una aplicación creada en el lenguaje C, por lo que ya citamos anteriormente, para procesos de bajo nivel se comporta mucho más rápido que otros lenguajes de alto nivel como Java.

Lo primero que hacemos es crear un canal de comunicaciones con nuestro dispositivo Bluetooth, por el cual vamos a ir recibiendo la información.

```
…
//Obtenemos el identificador del adaptador local Bluetooth
      dev_id = hci_get_route(NULL);
//Abrimos un socket local HCI
```

```
        socket = hci_open_dev(dev_id);
…
```

Una vez creado el canal lo dejamos en modo escucha para que capte todos los dispositivos posibles, posteriormente iremos discriminando los que no son usuarios de la línea de trenes en estudio.

```
…
while(1){
…
num_rsp    =    hci_inquiry(dev_id,    len,    max_rsp,    NULL,    &ii,
IREQ_CACHE_FLUSH);
            printf("\nDetectadaos %d dispositivos BT \n",num_rsp);
```

Lo primero que hacemos es recoger la velocidad del tren en cada instante, ya que solo vamos a guardar los dispositivos captados cuando el tren esté en marcha. Esto lo hemos hecho así para englobar a los usuarios por tramos de linea en distintos ficheros para su posterior estudio.

```
…
// Leemos la velocidad mínima a la que puede ir el tren
fichero=fopen("velocidad_minima.log","r");
fscanf(fichero,"%s/n", velocidad_min_char);
fclose(fichero);
velocidad_min = atof(velocidad_min_char);
// Leemos la velocidad a la que circula el tren
fichero=fopen("velocidad.log","r");
fscanf(fichero,"%s", velocidad_char);
fclose(fichero);
velocidad = 3.6*atof(velocidad_char);//Pasamos la velocidad a km/h
…
```

Hay que decir que antes de tomar esta decisión de separar los ficheros según la velocidad del autobús se estuvo viendo la posibilidad de hacerlo con el sensor de la puerta del autobús, que según se abriese o cerrase lo tomase como una nueva parada, pero esto podía llevar a error en aquellos casos en los que en una misma parada la puerta se abriese/ cerrase más de una vez, además se optó por la velocidad a parte de no dar lugar a error, por su fácil extracción del sistema GPS, ya que es este el que nos indica si está en marcha y no el autobús, con lo que no necesitamos interactuar con el autobús, haciendo el sistema más independiente para otros usos.

Para realizar la escritura en el fichero debemos estar en el caso antes mencionado (el autobús esté en marcha), si se cumple esto y que la parada actual coincide con la de nuestro fichero de paradas, además hay que comprobar si es la primera vez que se escribe en el fichero, habría que crear uno nuevo, o ya está creado, en cuyo caso procederíamos a la modificación del fichero.

```
if(velocidad>velocidad_min){
//si el bus está andando, se tienen en cuenta las detecciones, si no,
no.
```

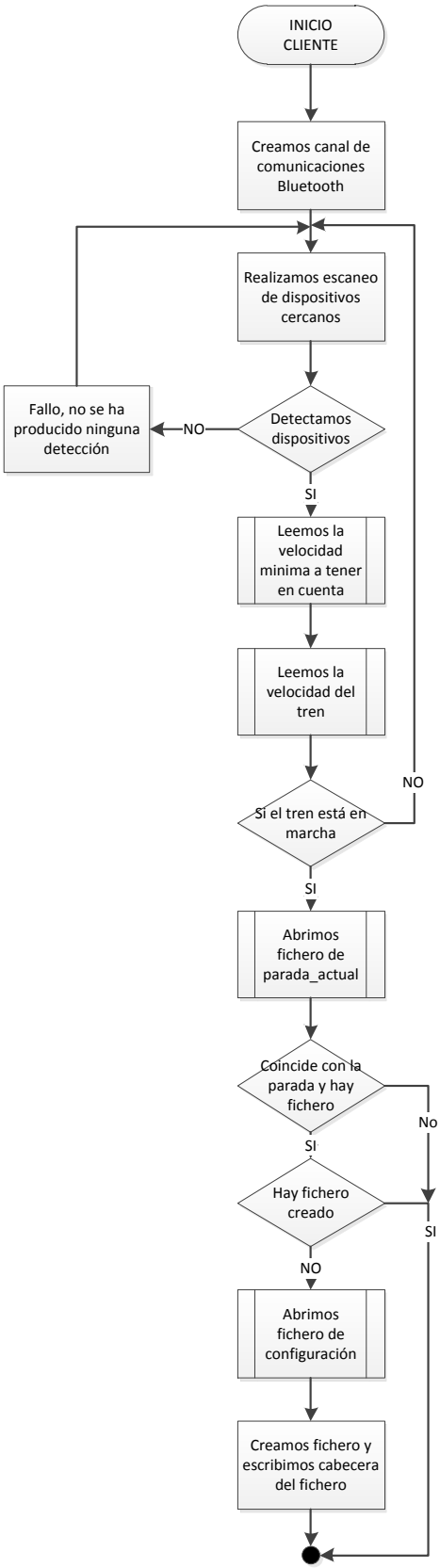
```
…
// Si es la 1°vez que entramos aqui no hay que cerrar fichero anterior
porque no lo hay
if (strcmp(id_parada, parada) != 0 || flag_inicio == 1) {
…
```

La escritura del fichero se hace por medio de un algoritmo que va comprobando si es la primera vez que se detecta ese dispositivo o ya estaba detectado. Si es la primera vez se inserta, sino se incrementa.

```
…
if(strcmp(MAC_dev,              MAC_Rep)              ==              0){
fsetpos(fichero,&posicionanterior);
auxdetecciones = atoi (detecciones);
auxdetecciones = auxdetecciones + 1;
sprintf(detecciones, "%04d", auxdetecciones);
…

if  (flag_MAC_repetida == 0 ){
      printf("Primera vez que es detectado\n");
      fseek (fichero, 0, SEEK_END);
      strcpy (detecciones, "0001");

//Como esta MAC ha sido detectada en este escaneo y no estaba, al
añadirla, hay que rellenar de ceros
//hasta la posición anterior a este escaneo y, en la posición de este
escaneo, poner un 1.
```

Al principio de cada fichero hemos creado como encacbezado un CRC, cabecera y nº de dispositivos distintos, que irán actualizándose y/o cambiando a lo largo del procesado de dispositivos encontrados. Pudiendo así tener una idea global del contenido del fichero con solo mirar el encabezado del mismo.

```
…
fscanf(fichero, "%s\n", CRC);
fscanf(fichero, "%s\n", cabecera);
fscanf(fichero,"%s\n", equipos_distintos_char);
…
```


INICIO
CLIENTE
Creamos canal de comunicaciones Bluetooth
Realizamos escaneo de dispositivos cercanos
Detectamos dispositivos
NO
Fallo, no se ha producido ninguna detección
SI
Leemos la velocidad minima a tener en cuenta
Leemos la velocidad del tren
Si el tren está en marcha
NO
SI
Abrimos fichero de parada_actual
Coincide con la parada y hay fichero
No
SI
Hay fichero creado
SI
NO
Abrimos fichero de configuración
Creamos fichero y escribimos cabecera del fichero

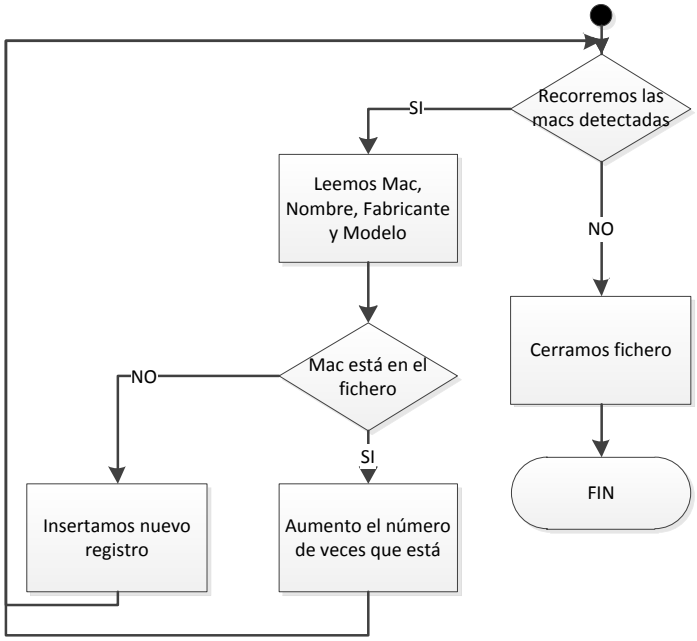


FIGURA 37 Aplicación Cliente

Lo hemos realizado así ya que para su posterior estudio era necesario, pues mediante un algoritmo matemático (BUSCAR PARA EXPLICAR EN CODIGO) de exclusión nos quedaremos sólo con aquellos que están dentro del autobús y descartaremos todos aquellos que hayamos detectado alrededor del autobús considerándolos como 'ruido'.

4.3. Aplicación Paradas

Esta aplicación se encarga de llevar el registro de paradas e ir indicando a la aplicación principal en qué parada nos encontramos exactamente.

Lo primero que haremos será crear un retardo de 1 segundo para que el sistema no consuma muchos recursos.

Una vez hecho esto pasamos a comprobar el fichero de 'recorrido.log'.

```
fichero_recorrido = fopen("recorrido.log","r");
while(fscanf(fichero_recorrido, "%s\n", penultima_parada) != EOF){
        if((strcmp(penultima_parada,"FIN")==0)){
        break;
        }
        strcpy(ultima_parada,penultima_parada);
        printf("ultima: %s\n",ultima_parada);

…
```

Un ejemplo del fichero de recorrido.log:

```
Ori
```

p01

p02

p03

p04

p05

p06

…

Fin

Seguidamente recogeremos la posición GPS actual del tren, para ello leeremos dicha posición de los ficheros 'latitud.log' y 'longitud.log', en los cuales se ha guardado la posición recogida por el GPS que nos proporciona la aplicación GPS.

```
fichero= fopen("latitud.log","r");
               fscanf(fichero, "%s\n", latitud_char);
               //printf("Latitud char: %s\n", latitud_char);
               latitud_double   = atof(latitud_char);// aquí pasar a double para poder
comparar
               //printf("Latitud double: %lf\n", latitud_double);
               fclose(fichero);
…

//De igual manera con longitud.log
```

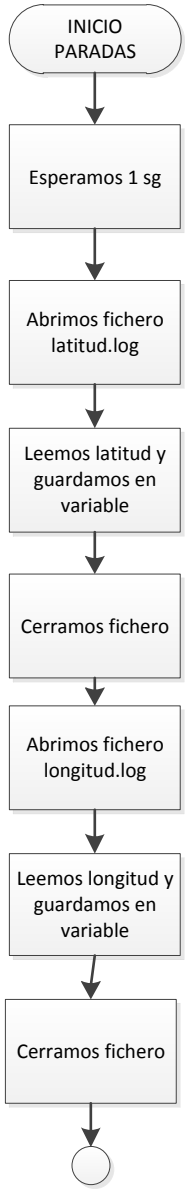
El siguiente paso es crear el fichero donde se van a ir guardando posteriormente los registros, con un formato concreto.

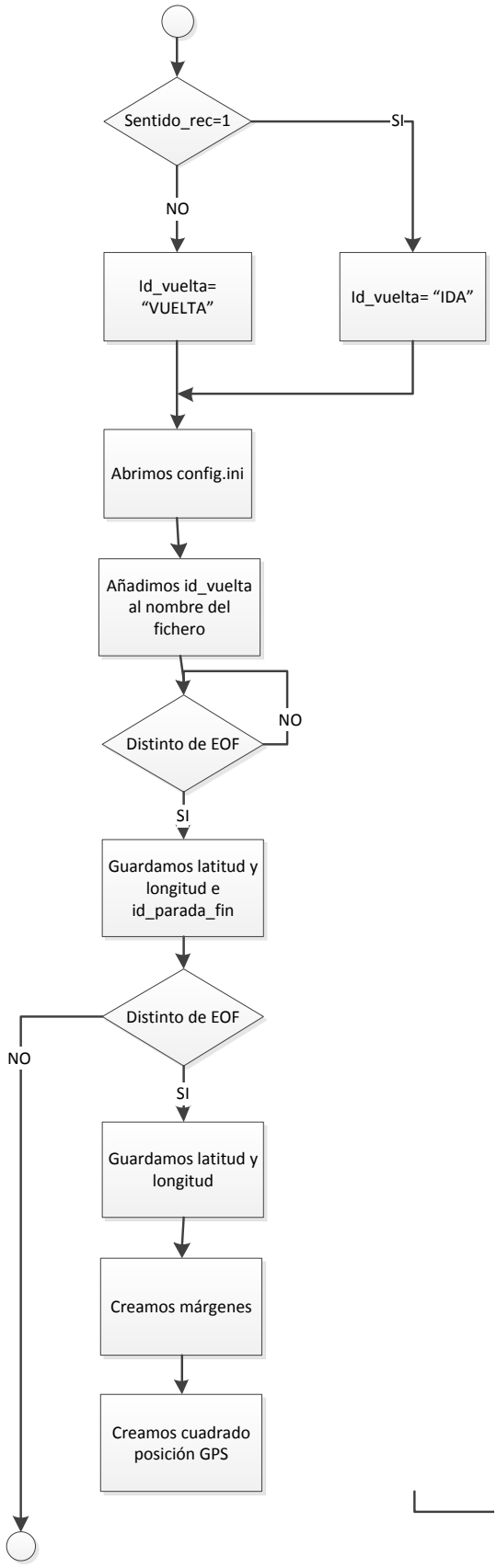
Primero leemos del archivo 'config.ini' para saber el operador, tren y linea concretos (este es uno de los ficheros que podría ser configurado para otros usos). Comprobamos si estamos en el sentido de ida o de vuelta.

```
if(sentido_rec == 1)
        strcpy(ida_vuelta, "IDA");
else
        strcpy(ida_vuelta, "VTA");


…

fichero = fopen("config.ini", "r");
               fscanf(fichero, "%s\t%s\t%s\n", operador, bus, linea);
               fclose(fichero);
…
```

En este fichero se recogen todas las posiciones GPS de cada una de las paradas de la línea actual. Procedemos entonces a crear nuestro algoritmo de detección de parada, basado en crear un cuadrado, en cuyo centro se encontrará la parada y cuyos bordes nos dirán si estamos dentro de lo que consideramos la zona de la parada o estamos fuera de ella, comparando con la posición GPS recogida, de esta forma se irán comprobando cada una de las paradas que hay en el fichero y en el momento que nos encontremos en la última parada de la línea cambiamos el sentido para hacer el recorrido de vuelta de la misma manera que el de ida.

De esta forma podremos saber en cada momento en que parada nos encontramos y englobar los datos obtenidos en cada una de esas paradas.




Sentido_rec=1
SI
NO
Id_vuelta= "VUELTA"
Id_vuelta= "IDA"
Abrimos config.ini
Añadimos id_vuelta al nombre del fichero
Distinto de EOF
NO
SI
Guardamos latitud y longitud e id_parada_fin
Distinto de EOF
NO
SI
Guardamos latitud y longitud
Creamos márgenes
Creamos cuadrado posición GPS

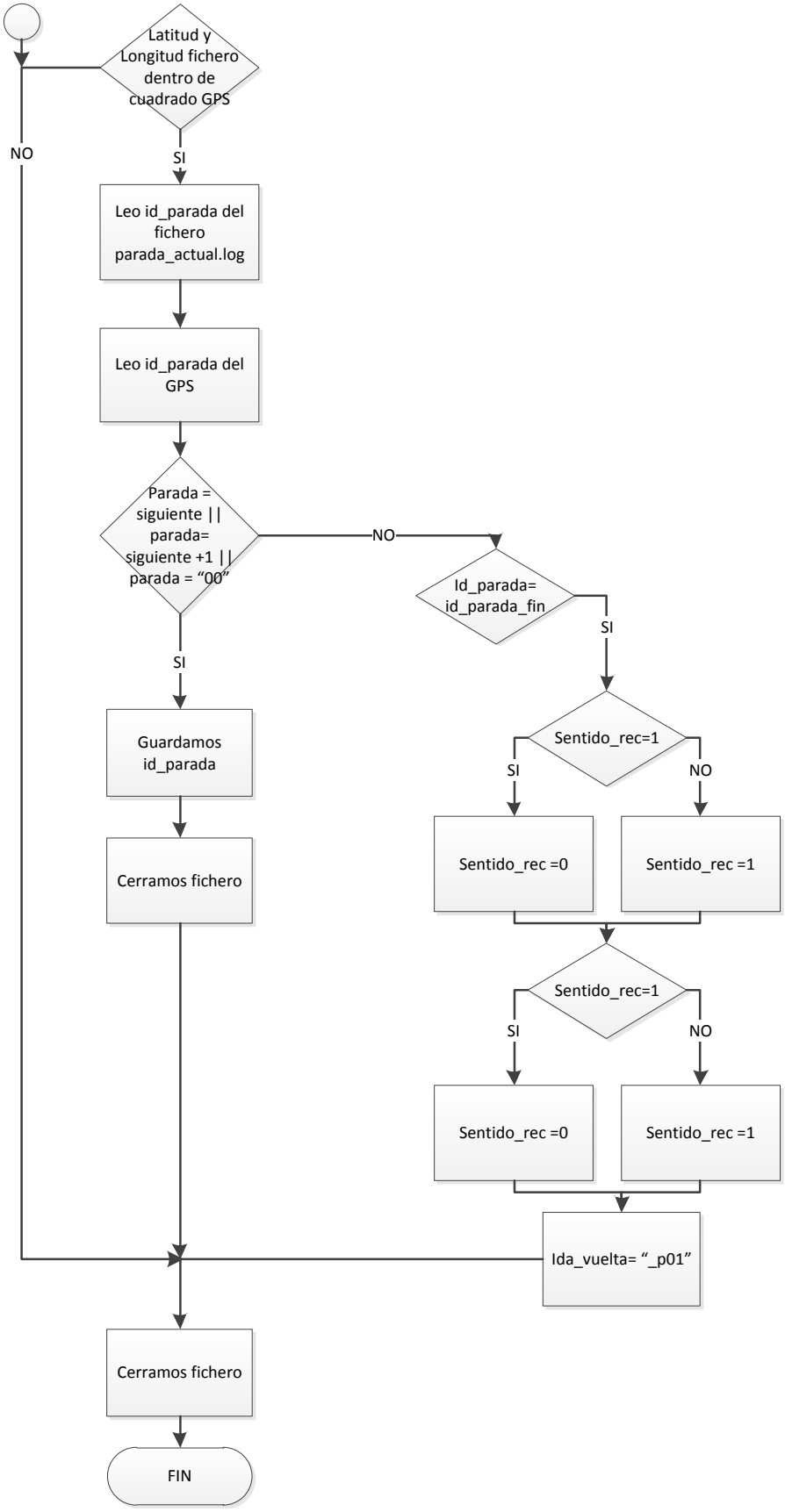


FIGURA 38 Aplicación Paradas

4.4. Aplicación GPS

Esta fue una tarea que nos llevó nuestro tiempo, ya que como hemos comentado encontrar un GPS que se adaptase a nuestras requerimientos para poder acceder a los datos no fue fácil, y tampoco fue fácil crear dicha aplicación, ya que en C, que es como pensamos en un principio la conexión al GPS es bastante tediosa. Esto nos hizo darnos unos cuantos cabezazos sin obtener grandes resultados, por lo que decidimos enfocar el problema desde otro punto de vista: ¿porque no cambiar el lenguaje?

Esto fue la solución a nuestros problemas ya que encontramos en el lenguaje python una serie de librerías que incluían funciones que realizaban las conversiones a las coordenadas UMTS utilizadas por google que es lo que nos interesaba para la posterior representación de mapas de las diferentes líneas de estudio.

Esta aplicación se encarga únicamente de recoger las coordenadas (latitud, longitud) y la velocidad del vehículo y guardarlas en un fichero que será accedido por la aplicación principal para englobar las detecciones según la parada en la que nos encontremos, que se determina por estas coordenadas.

La aplicación está pensada para recoger coordenadas cada vez que el vehículo se mueva, es decir descartaremos la repetición de coordenadas de una misma posición por medio de un algoritmo.

En sí la aplicación es un servicio que se llama desde el script Arrancar sistema.

La estructura que el programa sigue para devolver tanto la posición GPS, como la velocidad a la que circula el vehículo es la siguiente:

Creamos la conexión con el GPS.

```
…
session = gps.gps()
session.poll()
session.stream()
…
```

Guardamos latitud, longitud y velocidad.

```
…
session.poll()
latitud = str(session.fix.latitude)
longitud = str(session.fix.longitude)
velocidad = str(session.fix.speed)
…
```

Creamos un control de errores por si no se obtuviesen datos del GPS.

```
…
if latitud == latold:
            if longitud == longold:
                  contador = contador +1
…
if contador > 60:
                  sys.exit()
…
```

Lo siguientes es crear los archivos (con dicha información por separado), que después leerán la aplicación principal y la aplicación paradas.

```
…
else:
            contador = 0
            t =
open("/home/proyectosidi1/Escritorio/codigocsaumi/velocidad.log","w")
                t.write(velocidad)
                t.close
…
(mismo proceso para latitud y longitud)
```

Este proceso se realiza cada vez que el vehículo se mueva, en caso contrario no es necesario mandar de nuevo la información.

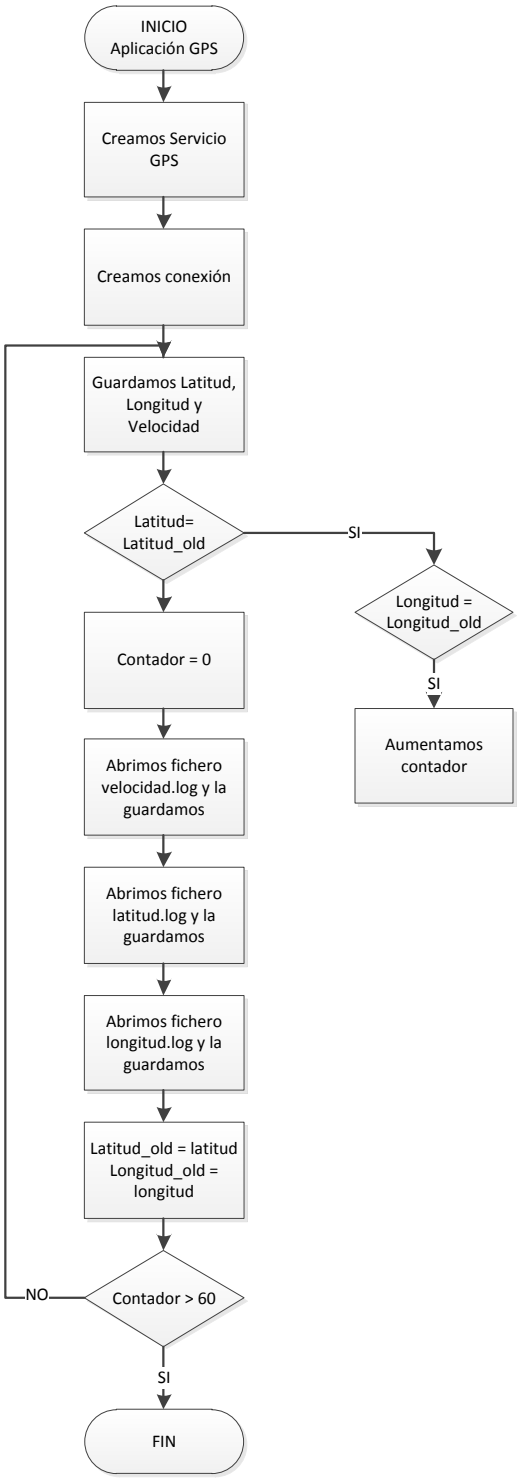


FIGURA 39 Aplicación GPS

5. Conclusión

En este capítulo se ha pretendido mostrar el funcionamiento interno de la aplicación SAUMI, a través de la explicación de parte del código creado. Es una forma fácil de entender la aplicación y eso es lo que se ha pretendido con este capítulo.

CAPÍTULO 6. CONCLUSIONES

Conclusión

Al final se ha desarrollado una aplicación que responde a la problemática planteada, no obstante la aplicación podría sufrir modificaciones para hacerla más eficaz y genérica, pero en definitiva se ha logrado cumplir con los objetivos para los cuales fue creado este proyecto, que eran los de crear un sistema móvil capaz de recoger datos de campo en tiempo real y según un itinerario.

Tras el desarrollo del mismo, nos hemos dado cuenta de la gran cantidad de ventajas y algunas desventajas de utilizar el lenguaje de programación C, que desde nuestro punto de vista, son muchas más las ventajas y consideramos que este lenguaje nos ofrece una gran cantidad de herramientas y capacidades para el desarrollo de nuestra aplicación.

Gracias a las librerías estándar que C nos ofrece para usar la mayoría de recursos, tanto del sistema como de dispositivos, seríamos capaces de implementar, si fuera necesario, una aplicación de extracción de datos que cumpla las necesidades deseadas en función de la demanda.

En nuestro caso concreto, la posibilidad de usar las librerías de conexión Bluetooth para comunicarnos con la CPU, el establecimiento del GPS y la facilidad de configuración del sistema Linux para automatizarlo… han hecho posible cumplir los objetivos propuestos que deseábamos resolver mediante el desarrollo de nuestro proyecto, que era la captura de datos en tiempo real y el posicionamiento de los mismos.

Combinando todas estas posibilidades, se podrían resolver infinidad de problemas, siendo la capacidad del manejo de un dispositivo GPS una gran ventaja que podría ir mucho más allá de los objetivos de nuestra aplicación, por mencionar uno de los dispositivos.

Líneas de futuro

Pensamos que la aplicación aquí desarrollada es muy útil para todo tipo de seguimientos, ya sea colectivos, como el caso de estudio que se ha realizado, o individuales, por ejemplo podría usarse para realizar estudios de mercado de grandes almacenes, tiendas, o realizar el seguimiento de una ruta de un empleado de seguridad, o incluso adaptarse para tener localizadas a personas en todo momento.

También sería útil para conocer el porcentaje de bluetooth que sigue utilizando la población o concretamente en un servicio como puede ser el tren, líneas de metro, etc y así conocer de antemano si se puede dar información de servicios ciudadanos al usuario a través de este medio, por ejemplo.

Estos son ejemplos de las posibilidades de la aplicación aquí realizada, pero realmente el abanico de posibilidades es muy grande, y las posibilidades de mejora de la aplicación también.

En este proyecto nos centramos en un caso particular y la aplicación fue creada teniendo en cuenta las particularidades para las que se creaba pero dejando abierto el código para líneas de mejora y por supuesto semi-adaptado a otros posibles usos, como los anteriormente mencionados.

Sería necesario también realizar un estudio a fondo de los datos contrastándolos, ya que en este proyecto nos hemos centrado en cómo sacar esos datos y no en la finalidad para la que se usen.

Pensamos también que es una aplicación que puede tener futuro gracias a su bajo coste, ya que sólo es necesario un mini-ordenador, un adaptador bluetooth y un GPS, como piezas fundamentales y optativamente el envío de datos por medio de GPRS. Este factor del bajo coste la convierte en una aplicación ideal para hacer estudios de población.

Una posible línea de futuro sería la adaptación de la aplicación a la tecnología wifi como medio captador de dispositivos, en vez del bluetooth que hemos empleado para nuestro estudio, esto permitiría adaptarlo a nuevos y más modernos dispositivos y así no quedar en el olvido.

Otra línea de futuro podría ser su adaptación a un Smartphone ya que este integra todo lo necesario para que la aplicación pueda correr, y con la entrada de Linux como sistema operativo en breve, es una posibilidad más que real, pero que no se ha tenido en cuenta ahora para la realización del proyecto, y que dejamos como posible adaptación futura.

CAPITULO 7.BIBLIOGRAFÍA

[1] Página Web http://www.wikipedia.com

[2] Página Web http://www.bluez.org

[3] Página Web http://www.seguridadmobile.com/bluetooth/especificacion-bluetooth/bluez/bluezscanner.html

[4] Página Web http://geolocalizacion.blogspot.com/

[5] Página Web http://www.clubdelamar.org/sistemagps.htm

[6] Página Web http://www.uv.es/~montanan/redes/GPRS

[7] Página Web http://www.mobilegprs.com

[8] Página Web http://www.chuidiang.com/

[9] Página Web http://www.c.conclase.net/

[10] Página Web http://www.forosdelweb.com

[11] Página Web http://gospel.endorasoft.es

[12] Biblioteca UPCT: PFC localización móviles GPS.

[13] Biblioteca UPCT: PFC Seguridad en Bluetooth.

ANEXO I Manual de Instalación y Configuración

1. Configuración de Ubuntu para correr SAUMI.

- Configurar la Conectividad de red:

  - Activamos la configuración de los parámetros de red con el protocolo DHCP modificando el fichero “interfaces”

  - Añadimos manualmente los DNSs en el archivo “resolv.conf”, puesto que el sistema no va a estar en nuestra red.

- Con Mozilla descargamos la última versión de BlueZ.

- Antes de instalar BlueZ habrá que instalar la librería D-Bus. Para ello habrá que irse al gestor de paquetes Synaptic (en Sistema -> Administración) y buscar "dbus". Después seleccionamos todos los de dbus y aplicamos.

- Ahora ya podemos instalar BlueZ con los comandos por línea de comandos o con el gestor de descargas.
  5. Descomprimir todo el contenido del programa.

- Por último habrá que añadir al cron las líneas para que se ejecuten los scripts de forma periódica:
  - Arrancar Sistema
  - Watchdog

- También será necesario instalar el paquete “curl” para los script que reciben la IP del empotrado.

- Configuración usb 3G. Nosotros hemos utilizado el de la marca Vodafone, pero cada uno deberá instalar el de la operadora contratada.

  - sudo dpkg -i vodafone-mobile-connect_1.99.17-8_all.deb
  - sudo apt-get -f install
  - vodafone-mobile-connect-card-driver-for-linux

  Una vez ejecutados estos comandos debería reconocer el usb 3G automáticamente, lo seleccionamos y listo para usar.

ANEXO II Ejemplo de Uso de la Aplicación

En esta parte explicaremos paso a paso la ejecución del sistema completo, antes de entrar a fondo comentar que el sistema está concebido para que se ejecute de forma automática, pero también es posible lanzar cada una de las aplicaciones que lo componen por separado. Esto mismo es lo que hemos hecho para poder sacar capturas de pantalla y hacer una ejecución de ejemplo del sistema.

Antes de comenzar realizaré un resumen de cómo actúa el sistema en modo automático y luego ya pasaremos a explicarlo en modo manual.

1. Modo Automático

En este tipo de ejecución es el sistema mediante el archivo contrab, pre programado, el que se encarga de lanzar los scripts correspondientes al iniciarse:

- ArrancarSistema
- Watchdog

En este modo sólo son necesarios estos 2 debido a que el primero, ArrrancarSistema, engloba las ejecuciones necesarias como se explicó anteriormente, y Watchdog para controlar que no se pierda la señal por algún error del sistema.

2. Modo Manual

En este tipo de ejecución es el usuario el que se encarga mediante un terminal de lanzar cada una de los script/aplicaciones, en paralelo o en serie según indicaremos a continuación.

Obviando el lanzador ArrancarSistema, lo primero que habría que hacer sería poner en marcha la aplicación principal Cliente2, la aplicación Paradas y la aplicación latlong.py:

El Cliente 2 va englobando dentro del archivo generado por Paradas los dispositivos encontrados, el nombre de este archivo dependerá del archivo config.ini y la aplicación latlong.py corre en modo background devolviendo los archivos longitud.log, latitud.log y velocidad.log

```
pedro_juan@PJ-Ubuntu:~/Escritorio/codigocsumi$ ./Cliente2



Detectando dispositivos...

Detectadaos 2 dispositivos BT

Creo fichero con nombre: Operador_autobus_linea_13-02-07_18-57_p04.txt


Dispositivo 1:  MAC: E8:99:C4:11:88:D1          Nombre: [Desconocido]
MAC_dev: E8:99:C4:11:88:D1
caso fichero vacio o no esta

Dispositivo 2:  MAC: A8:92:2C:27:B8:19          Nombre: [Desconocido]
MAC_dev: A8:92:2C:27:B8:19
caso fichero vacio o no esta



Detectando dispositivos...
```

FIGURA 40 Captura Aplicación Cliente

```
Detectando dispositivos...

Detectadaos 1 dispositivos BT

Dispositivo 1:  MAC: A8:92:2C:27:B8:19          Nombre: Pedro_J



Detectando dispositivos...

Detectadaos 2 dispositivos BT

Dispositivo 1:  MAC: E8:99:C4:11:88:D1          Nombre: HTC_One_S

Dispositivo 2:  MAC: A8:92:2C:27:B8:19          Nombre: [Desconocido]



Detectando dispositivos...
```

FIGURA 41 Captura Aplicación Cliente

```
pedro_juan@PJ-Ubuntu:~/Escritorio/codigocsumi$ ./paradas
p
Parada leida del fichero: 24
Parada incrementada: 25
Parada escrita: IDA_25
```

Figura 42 Captura Aplicación Paradas

Una vez recogida la información correspondiente, se envía al servidor en formato comprimido. Normalmente será un .zip con todos los archivos de cada día:

Figura 43 Captura Aplicación scriptftp

Comprobamos conectándonos al servidor mediante terminal o clienteFTP que el proceso ha ido correctamente.

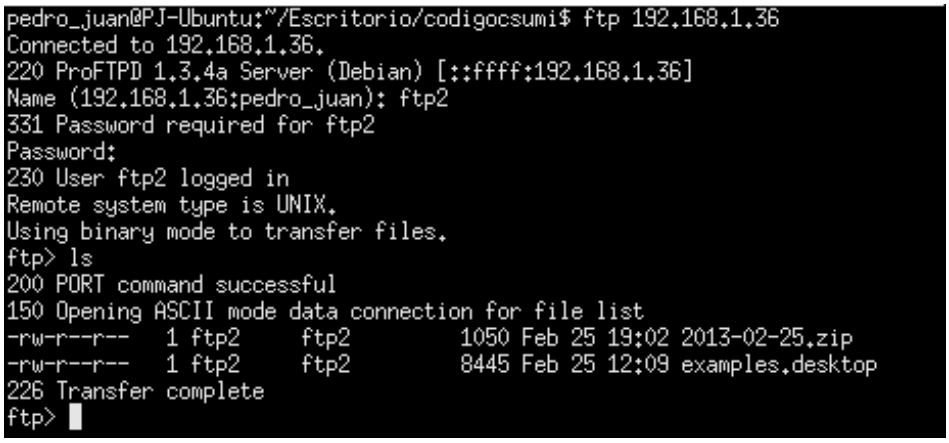


Figura 44 Captura cliente FTP

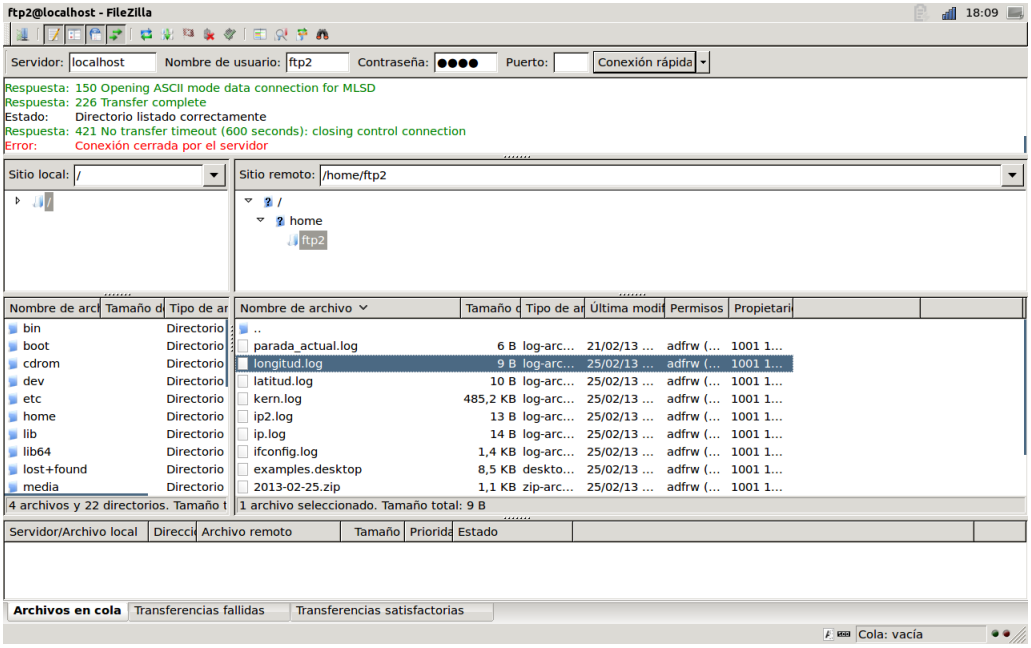


Figura 45 Captura terminal cliente FTP

Los siguientes archivos ejecutados son para el diagnóstico de problemas. Al ejecutarlos envían la información al servidor para su estudio.

Lanzamos el script infokern que nos hará una copia del kern.log del sistema.

```
pedro_juan@PJ-Ubuntu:~/Escritorio/codigocsumi$ ./infokern
485157 bytes transferidos.
```

Figura 46 Captura Aplicación Infokern

El siguiente script a lanzar es el informes, que nos hará una copia en el servidor de la última posición de envío, para poder reanudar el sistema desde esa posición.

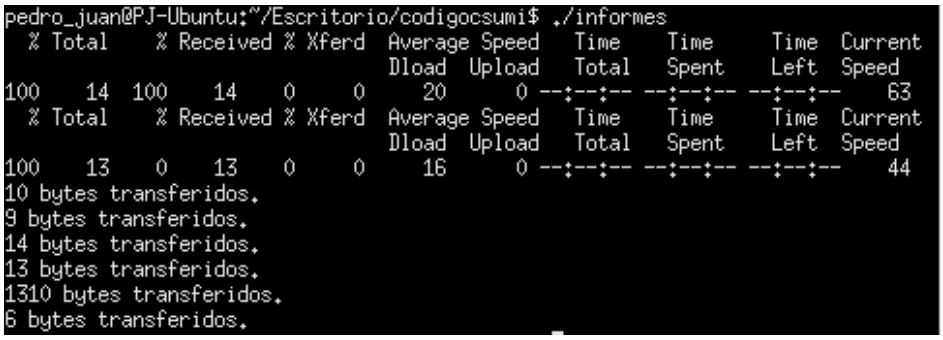
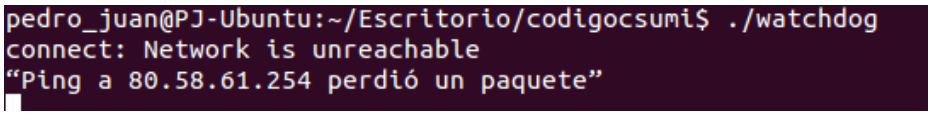
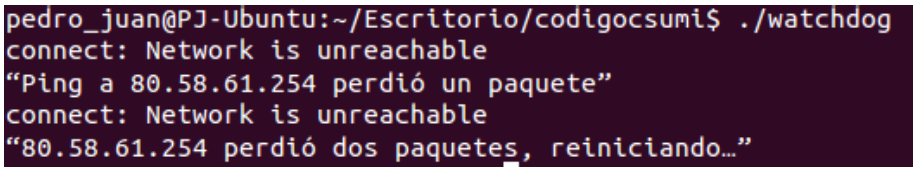


Figura 47 Captura Aplicación Informes

El Script de comprobación watchdog, nos dirá si se ha perdido la conexión a internet. No es necesario lanzarlo para el transcurso habitual de la aplicación, a no ser que notemos que algo falla en el envío de archivos al servidor.





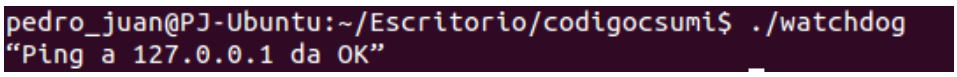


Figura 48 Capturas Aplicación Watchdog