Principios básicos de GSM / GPRS / EDGE Evolution / VAMOS



**Las redes GSM** (sistema global para comunicaciones móviles) se han desplegado en todo el mundo como sistemas de comunicaciones móviles de segunda generación. El GSM se desarrolló originalmente para transmisión de voz, pero también es capaz de transmitir datos con una carga máx. de 9,6 kbit/s en el enlace descendente y en el enlace ascendente. Con la solución técnica provisional HSCSD se consiguió aumentar la velocidad de transmisión máx. hasta 43,2 kbit/s en el enlace descendente y hasta 14,4 kbit/s en el enlace ascendente.

Se añadió capacidad para datos de paquete en la versión '97 del estándar GSM mediante GPRS (servicio general de radio por paquetes). **GPRS** es un servicio de datos que está disponible para los usuarios de GSM y dispositivos portátiles IS-136. GPRS se puede utilizar para todos los servicios de datos, como navegación por Internet, acceso WAP, SMS y MMS. A diferencia del GSM, el GPRS utiliza conmutación de paquetes; en caso de multiplicidad de usuarios, solo comparte el canal de transmisión cuando es necesario enviar datos. La velocidad de transmisión que se puede alcanzar en GPRS depende de la clase de multintervalo que admita. La máx. velocidad de transmisión por cada intervalo de tiempo es de 21,4 kbit/s con un número máx. posible de 8 intervalos tanto en enlace descendente como ascendente. Una clase multintervalo 10 del dispositivo celular, por ejemplo, admite un máx. de 4 intervalos de recepción en el enlace descendente y un máx. de 2 intervalos de transmisión en el enlace ascendente, pero la suma de intervalos de recepción y transmisión tiene un límite máx. de 5.

Para aumentar la velocidad de transmisión de datos y mejorar la capacidad de la red, se introdujo **EDGE** (mejora de datos para evolución de GSM) en la versión '98. Con EDGE se aplica un nuevo esquema de modulación: 8PSK (modulación por desplazamiento de fase), es decir, tres bits por símbolo RF modulado en contraste con el único bit original por símbolo en GPRS. Este cambio de tecnología triplica la velocidad de transmisión eficaz en EDGE y ofrece una velocidad máxima de datos de 59,2 kbit/s por intervalo de tiempo. De forma similar a GPRS, la velocidad de transmisión máxima que se puede conseguir depende de la clase de multintervalo que admita el teléfono celular. La velocidad de transmisión media del enlace descendente en redes comerciales está en la gama de 300 kbit/s. **EGPRS** es una expresión jerárquicamente superior que comprende EDGE y GPRS.

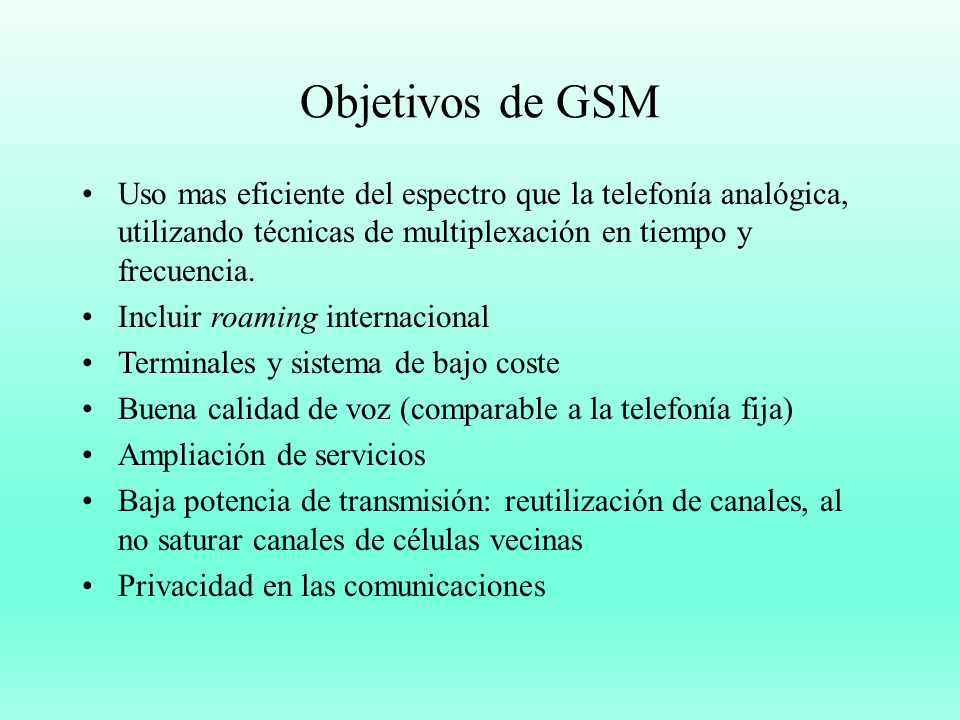
**EDGE evolution** contiene un conjunto de características especificadas en 3GPP versión 7. El principal motivo para mejorar EDGE es garantizar la competitividad futura de la tecnología dominante de segunda generación y en concreto proporcionar continuidad de servicio entre las diversas tecnologías de radio (GERAN, UTRAN, LTE) admitidas por la evolución del núcleo de la red, p. ej., IMS. Entre las mejoras se incluye el funcionamiento de portadora dual del enlace descendente (DL DC), la modulación de orden superior (EGPRS2 con 16 QAM y 32 QAM) en combinación con una velocidad de símbolos opcional superior (nivel B) y la codificación turbo en el enlace descendente, mejoras de latencia y el uso de diversidad de recepción en la estación móvil. Al combinar DL DC y EGPRS2 nivel B, las velocidades máximas de transmisión se elevan hasta 1,9 Mbit/s en el enlace descendente, en tanto que es realista alcanzar una velocidad de transmisión de 1 Mbit/s.

**VAMOS** es una de las características más importantes entre las añadidas en 3GPP versión 9 porque tiene potencial para duplicar la capacidad de voz en la red GSM. El nuevo esquema de modulación AQPSK en DL permite la asignación de distintos niveles de potencia en las fases I y Q. En combinación con nuevas secuencias de entrenamiento ortogonales, dos usuarios de voz podrán compartir el mismo recurso físico.

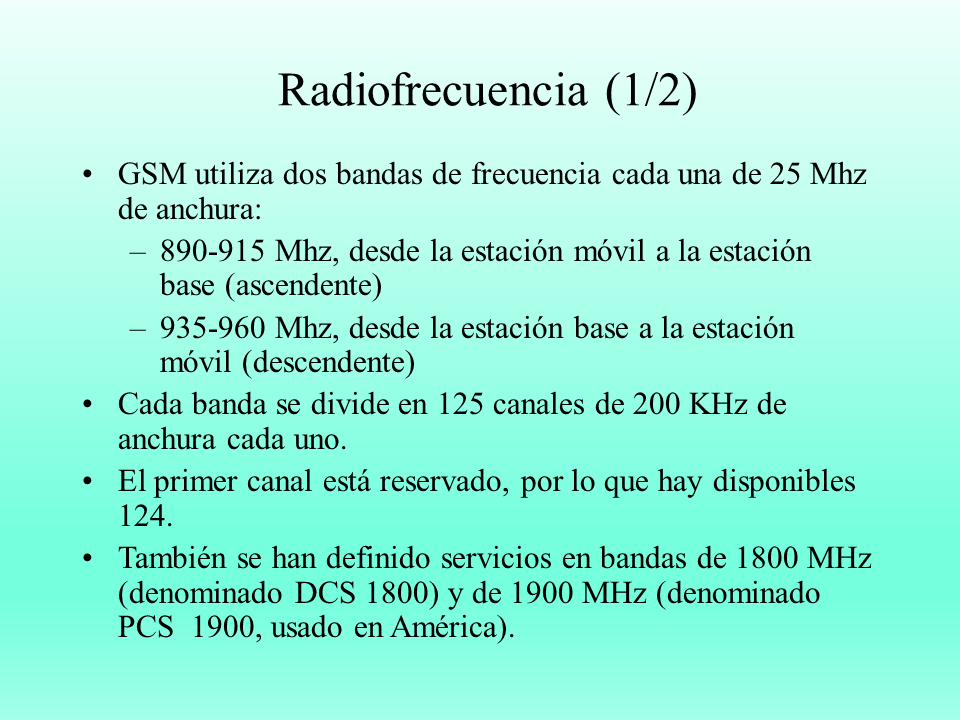
En 1989, el Groupe Speciale Mobile definió el estándar GSM que se convirtió en un estándar de telefonía celular digital aceptado internacionalmente. GSM fue transferido después al comité técnico de eTSI que continuó definiendo los estándares GSM. Hoy en día, GSM y EDGE están especificados en 3GPP (3rd Generation Partnership Project) y coordinados en TSG GERAN

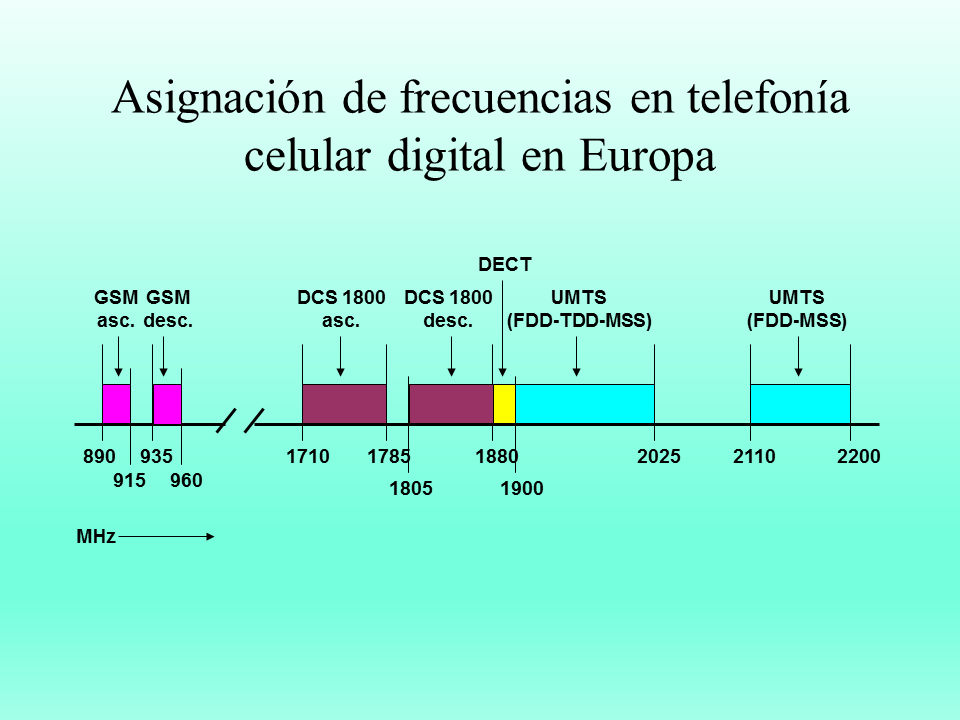
……………….

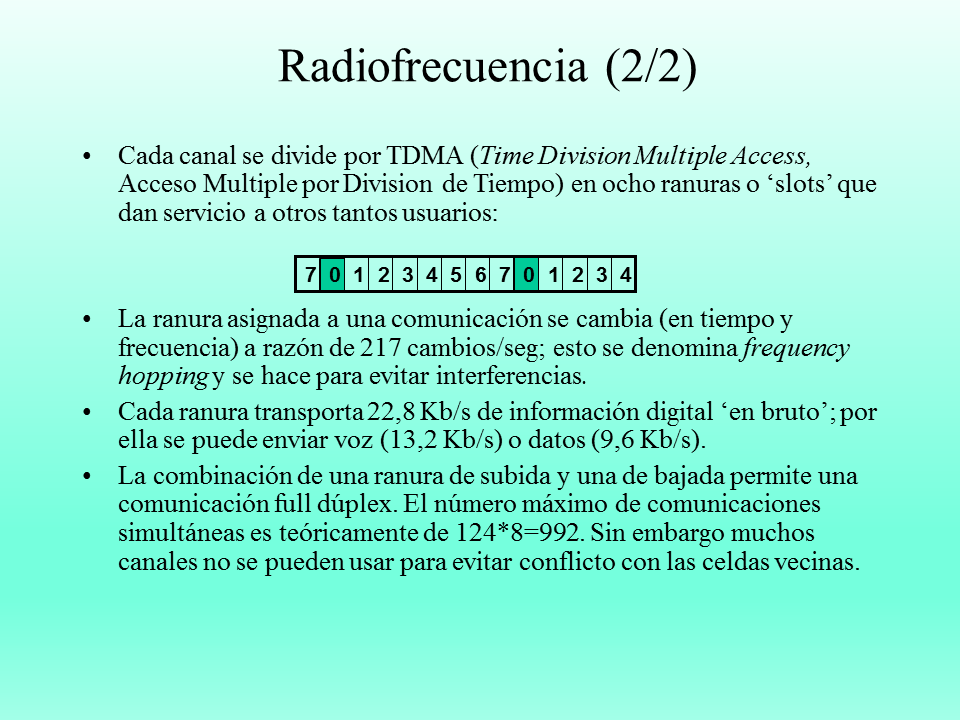


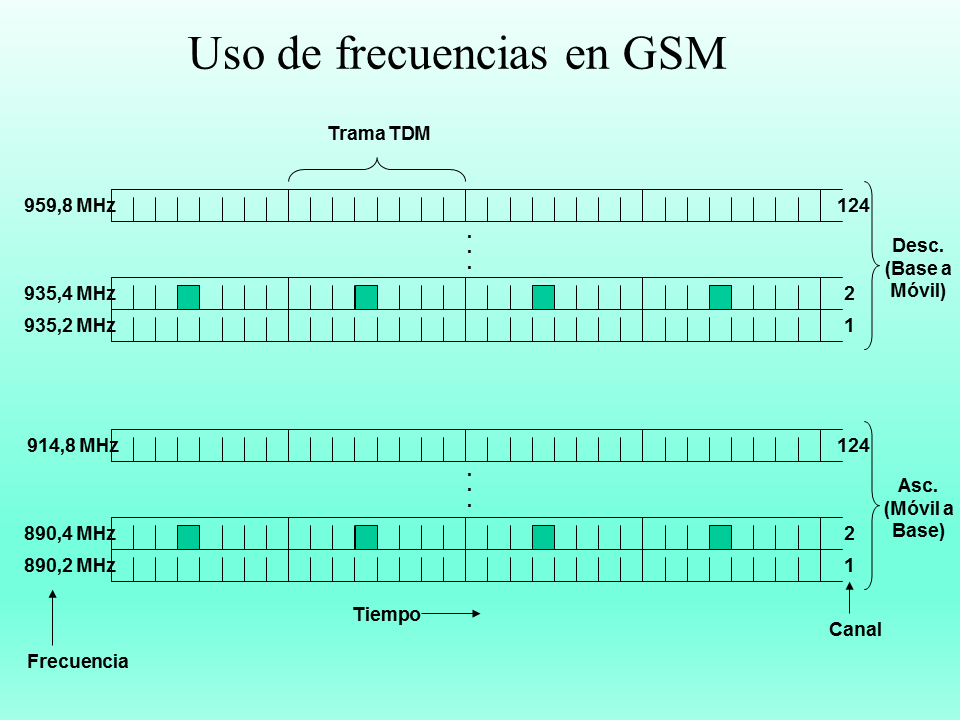


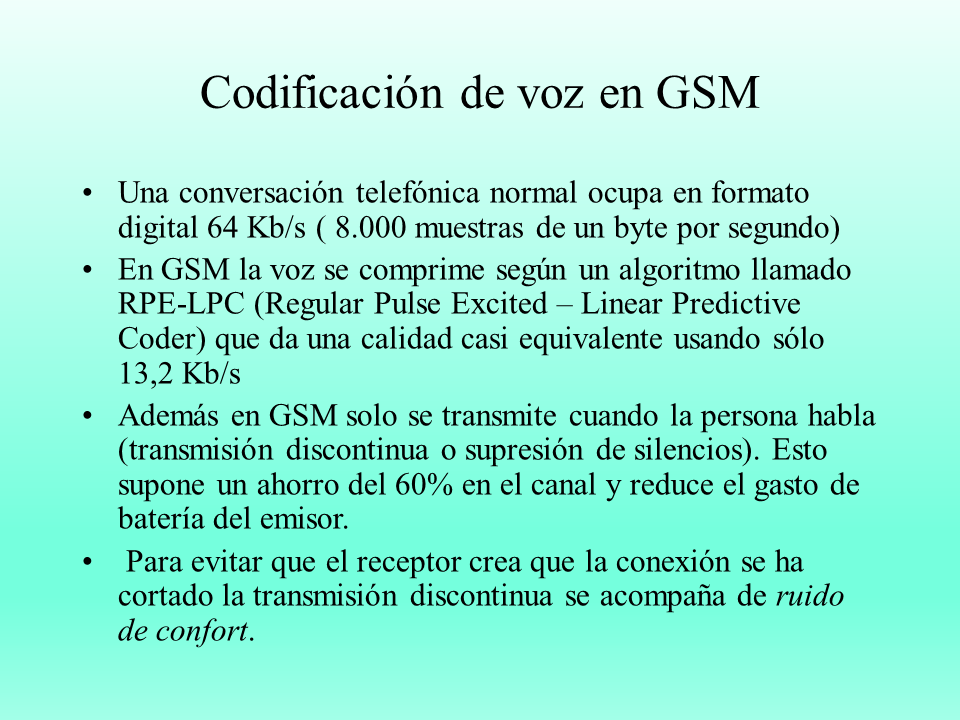


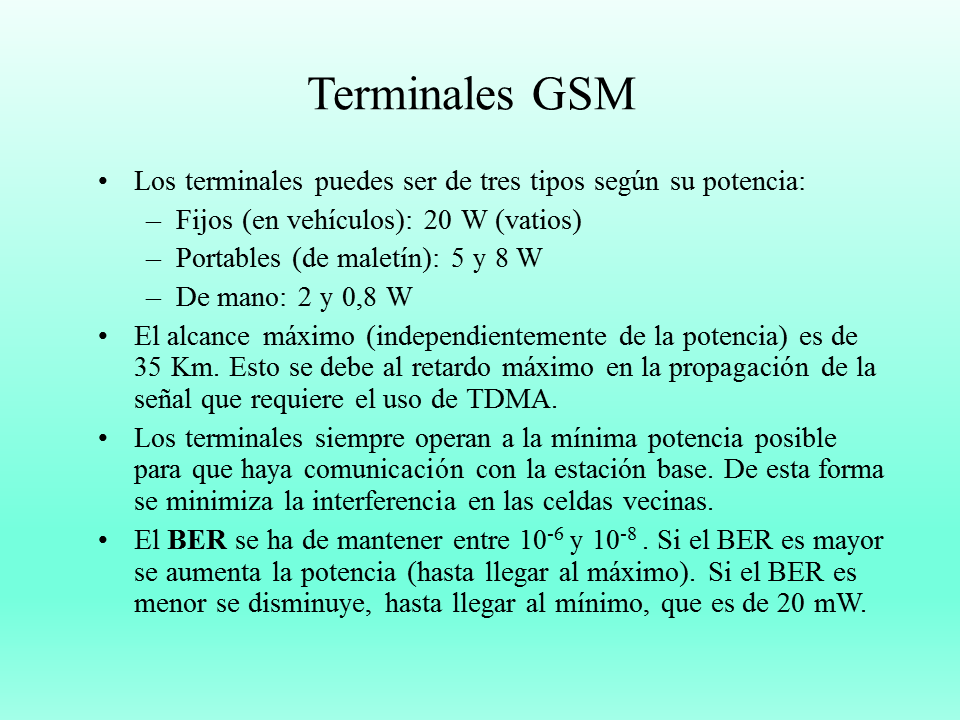


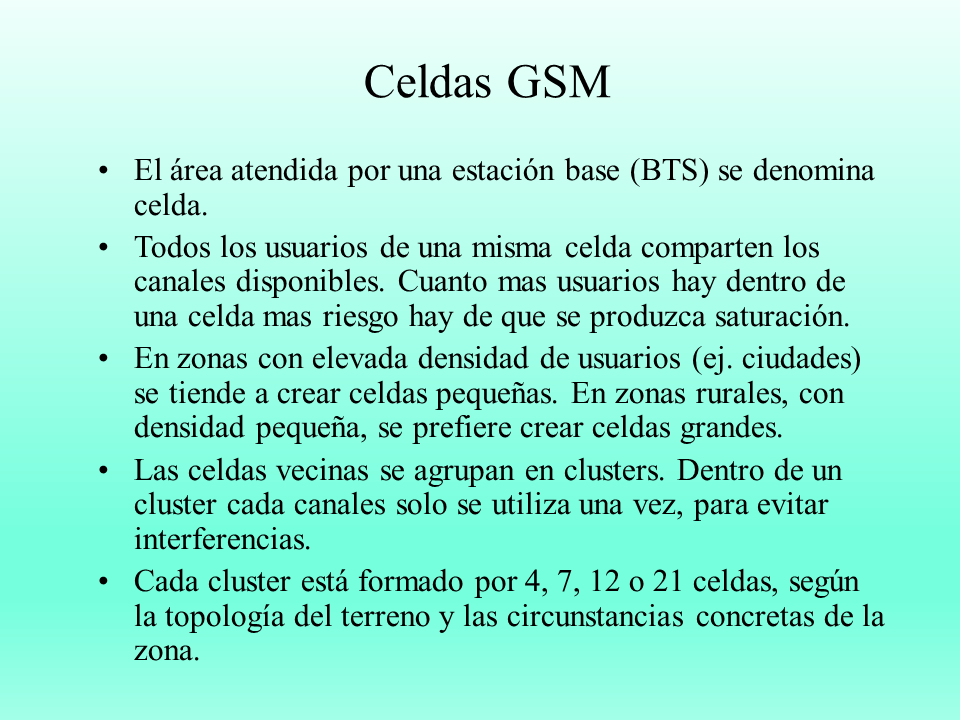


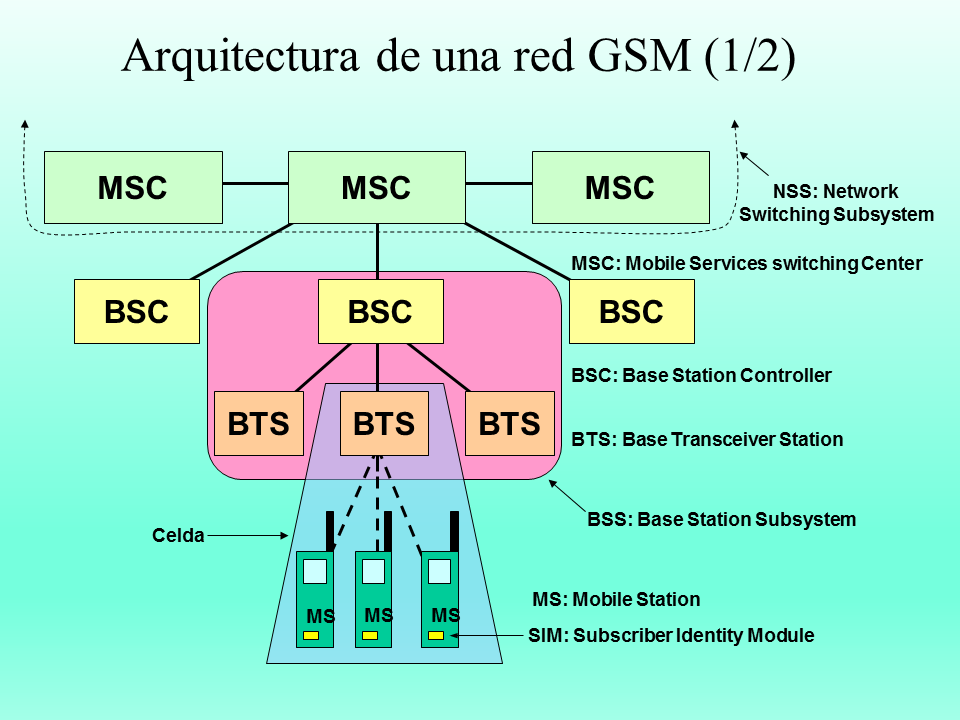


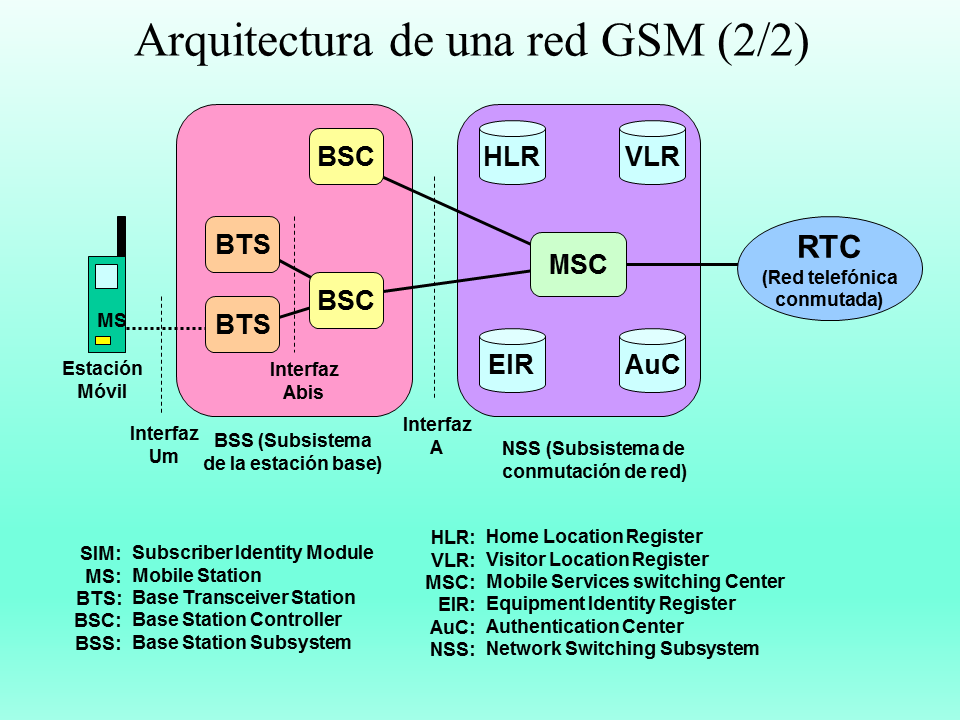




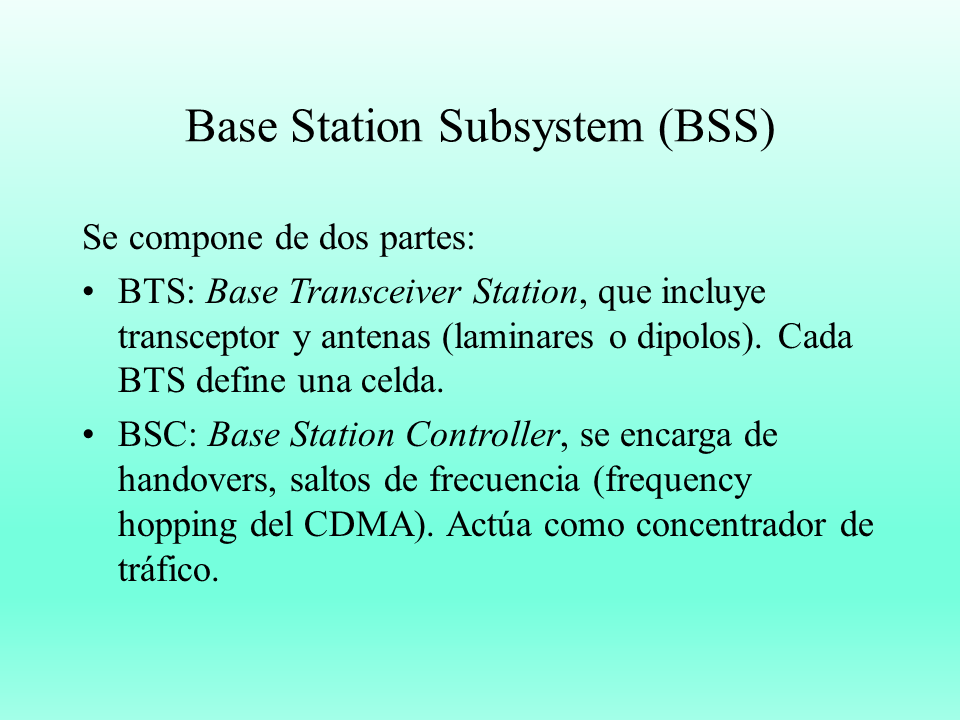


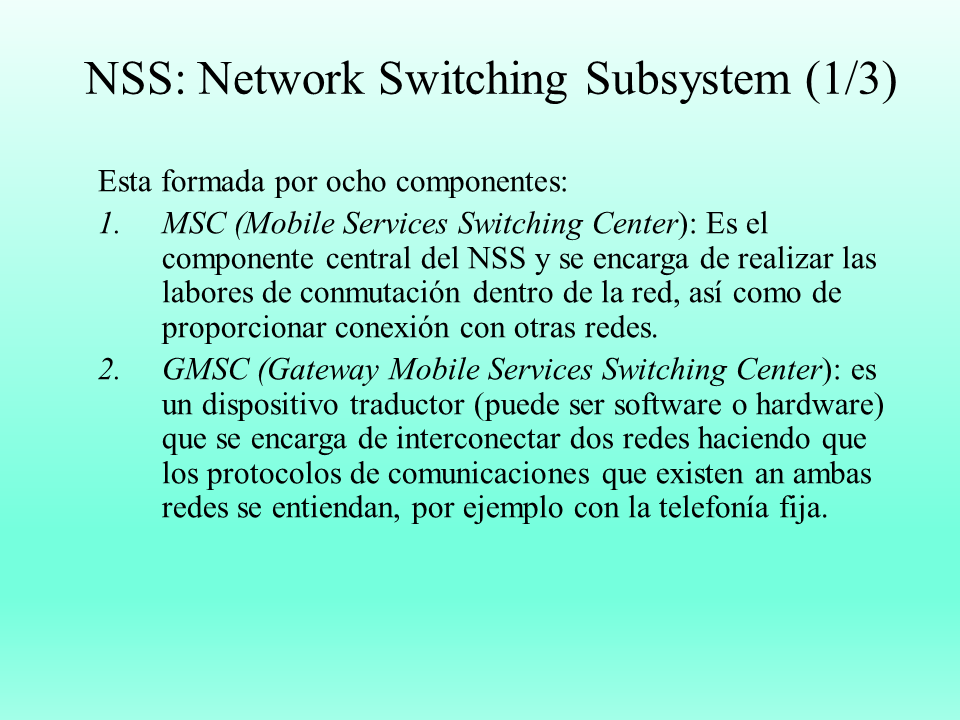


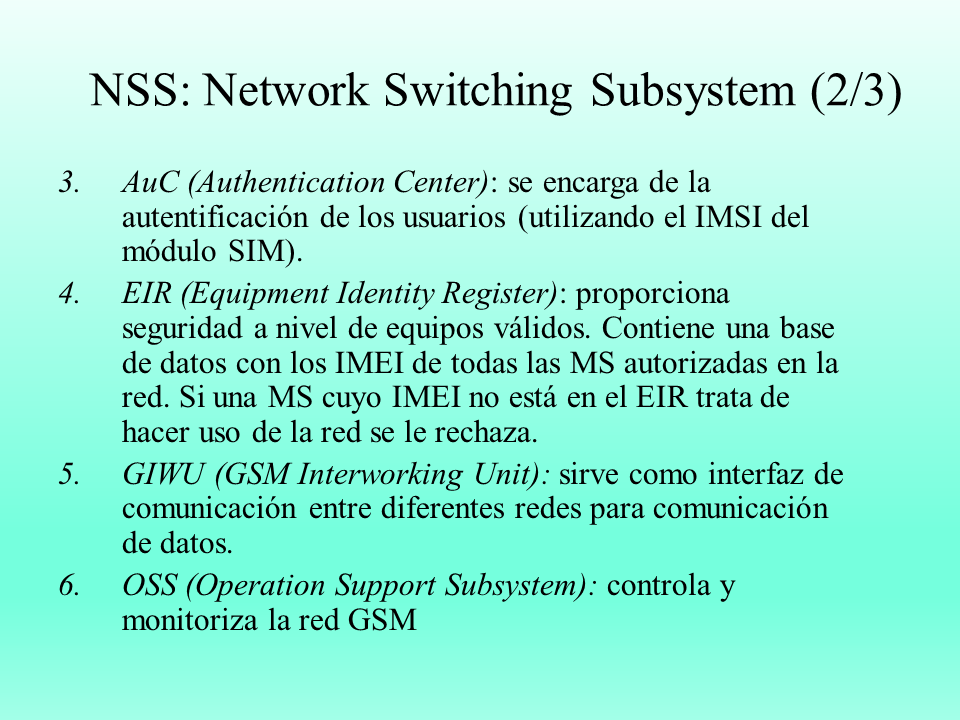


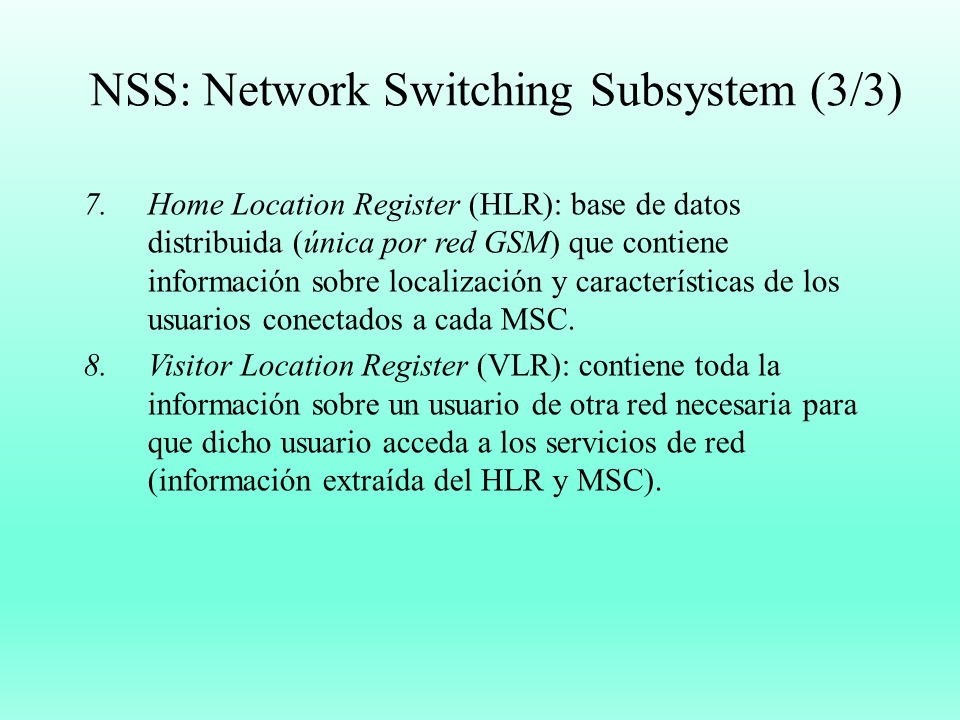


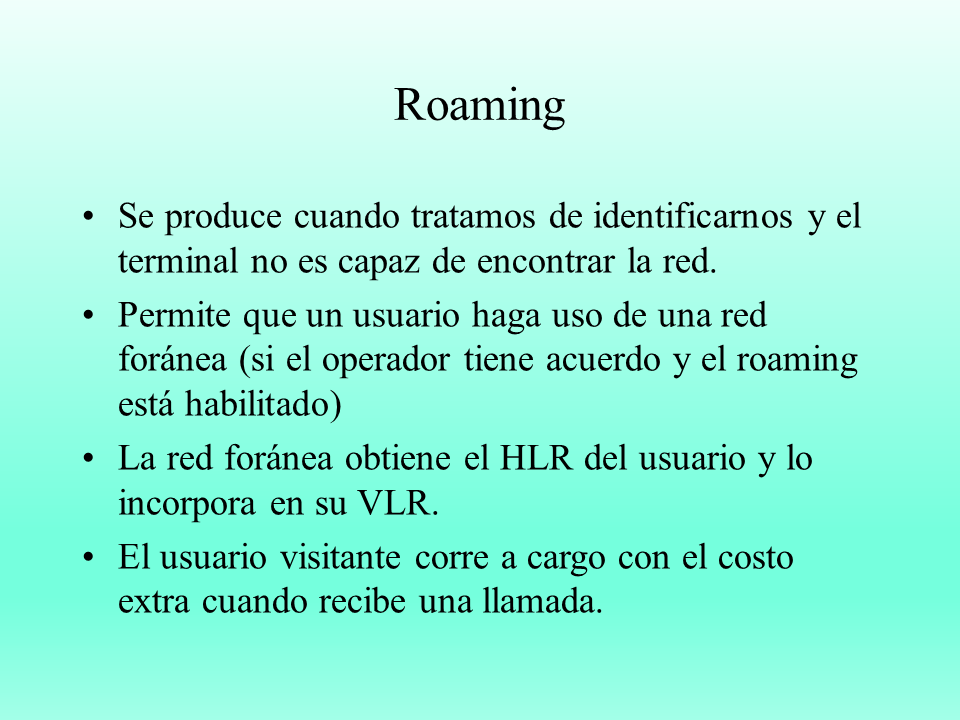


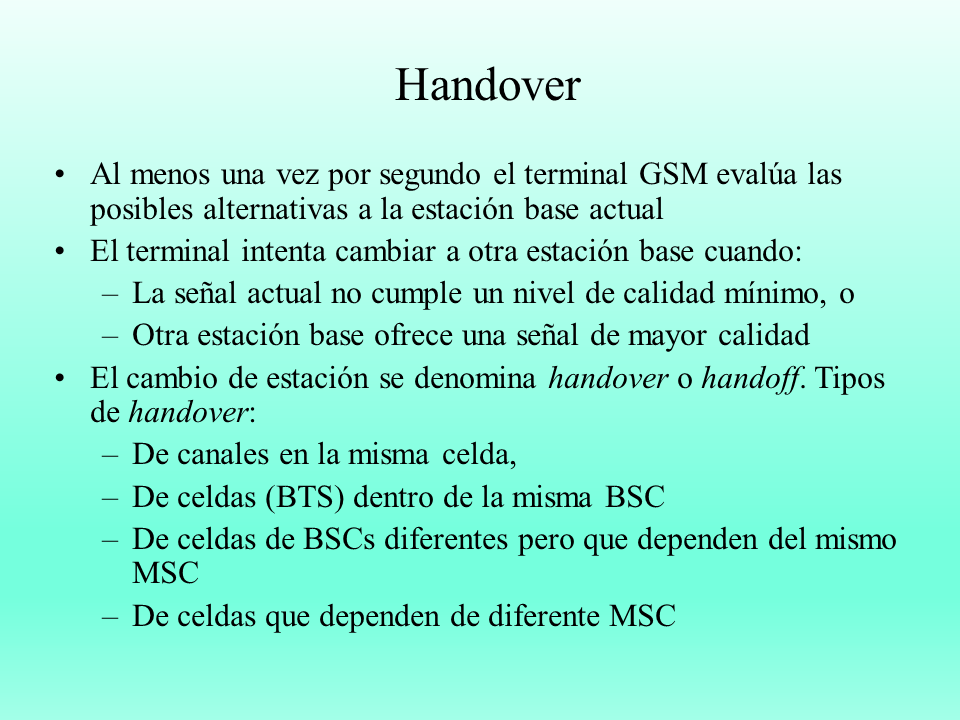


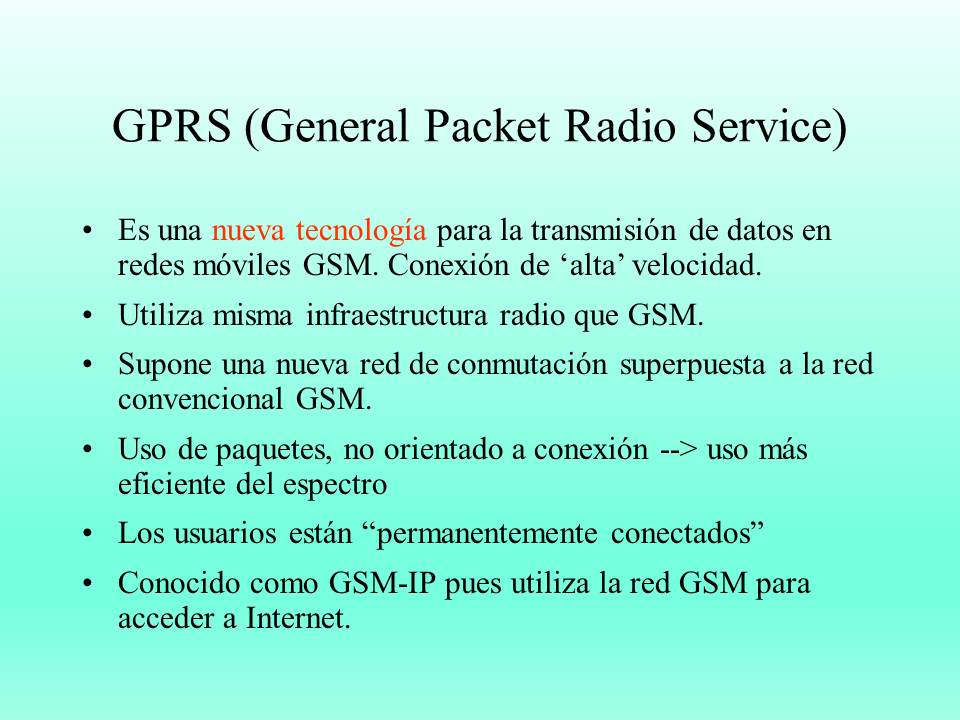


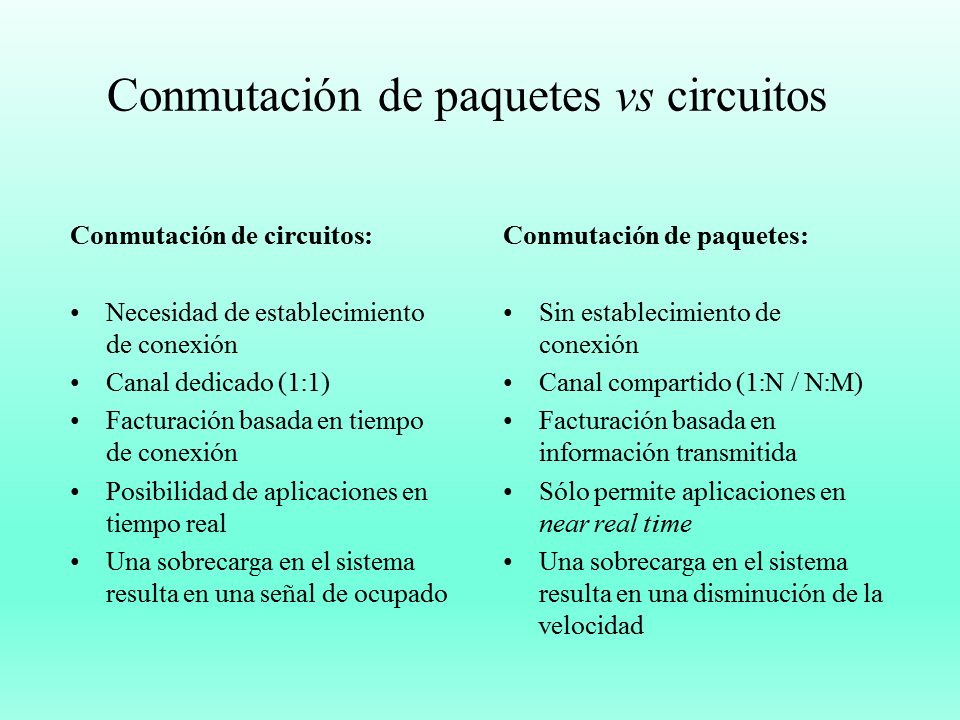


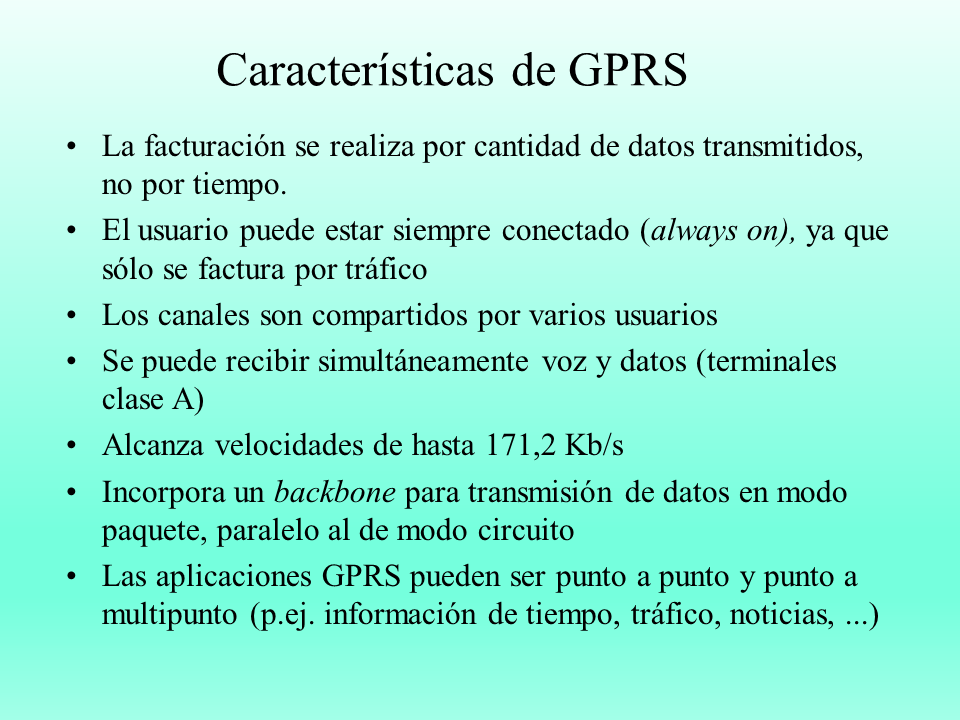


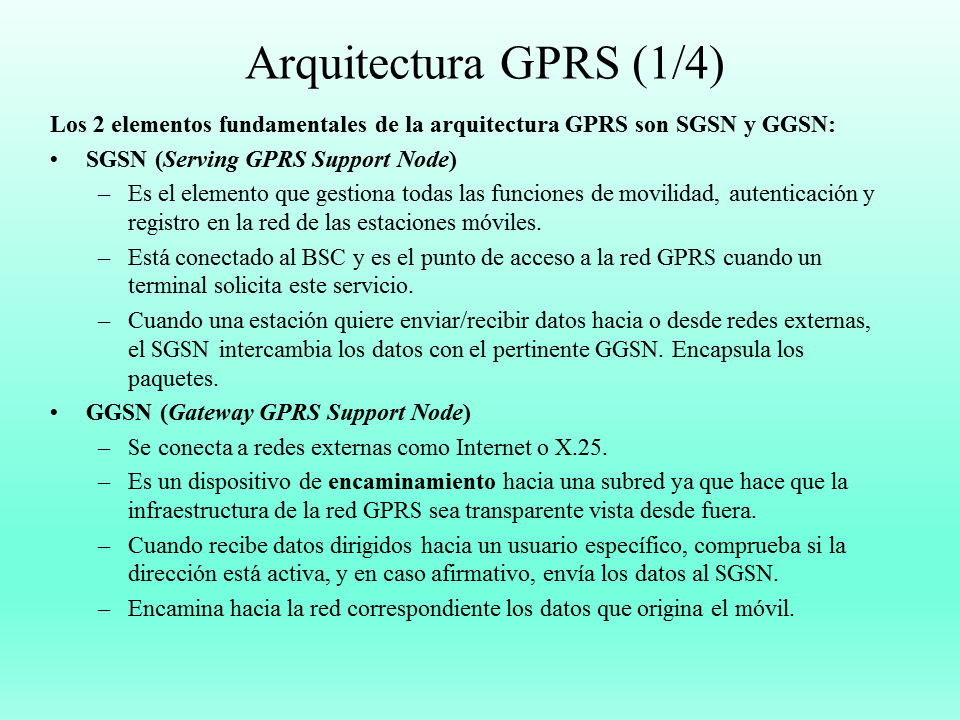


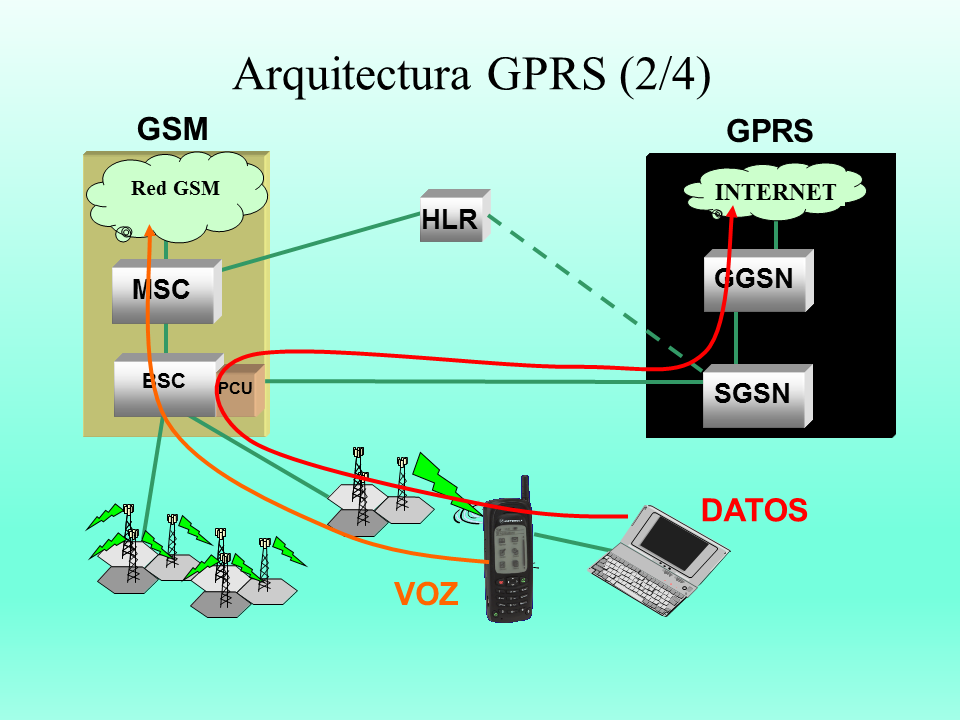


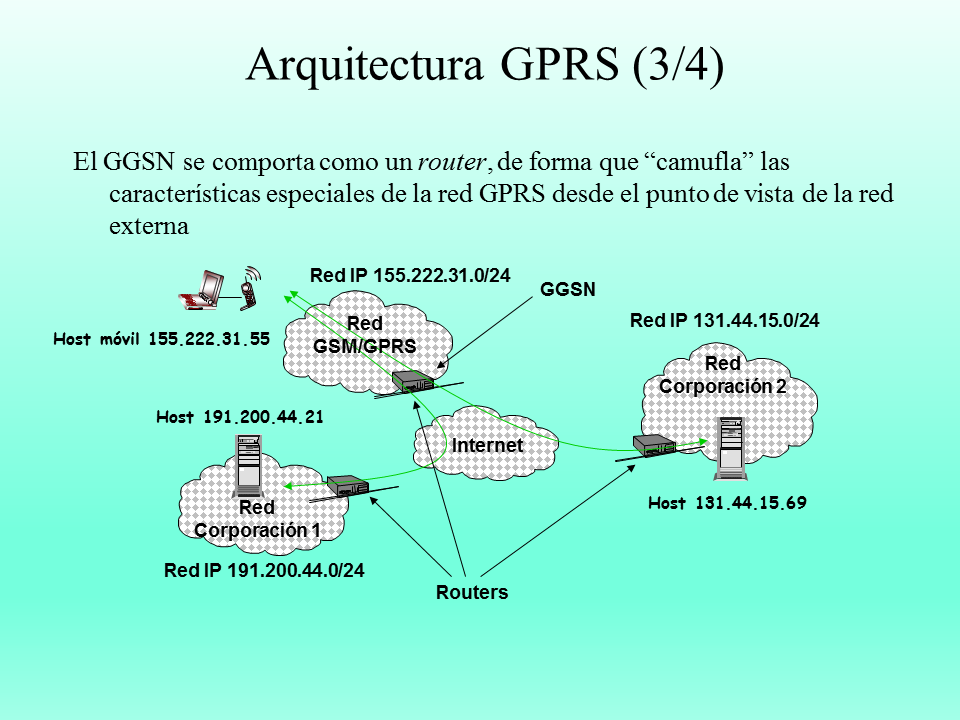


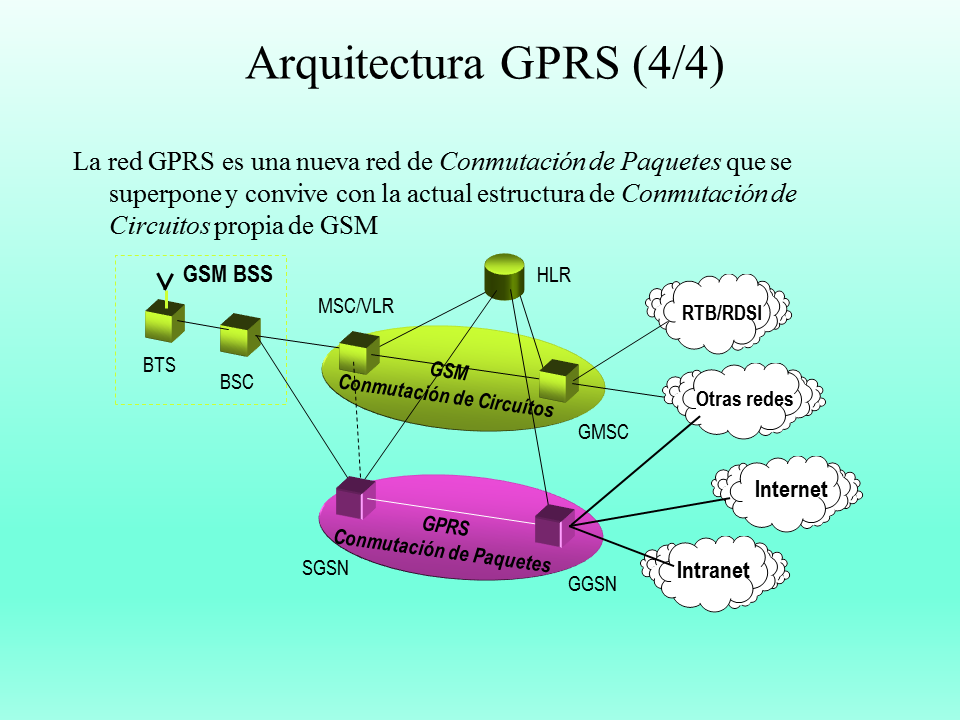


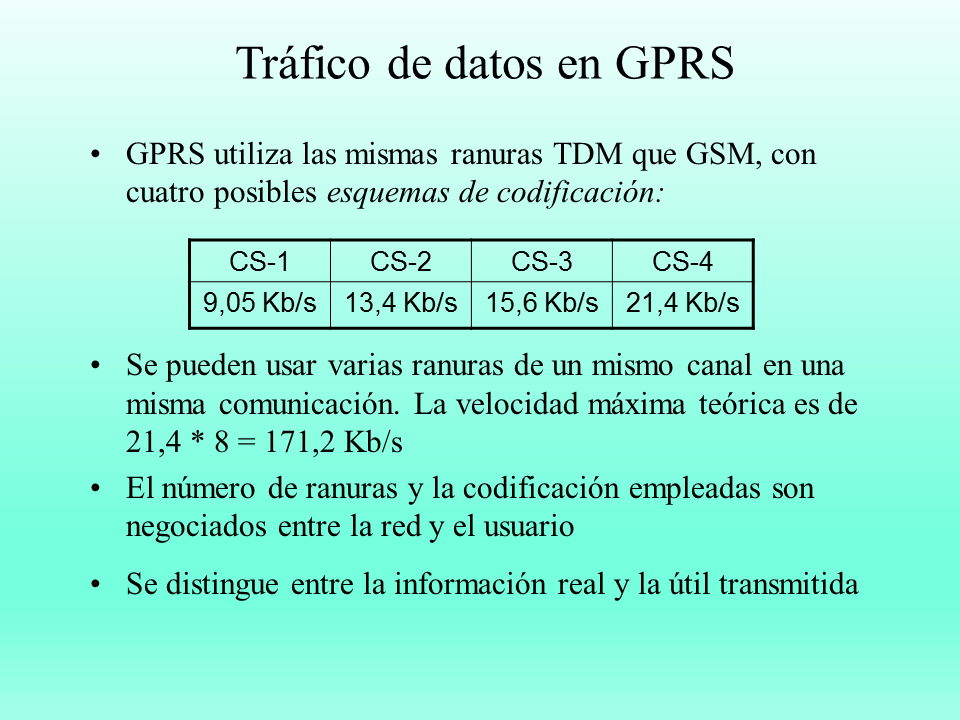


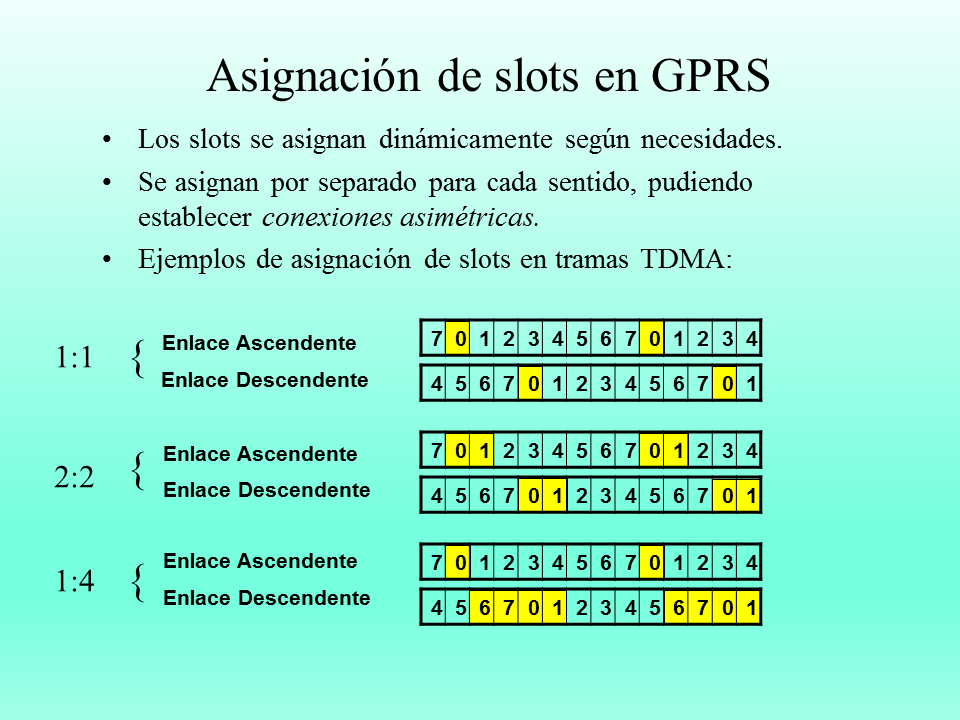


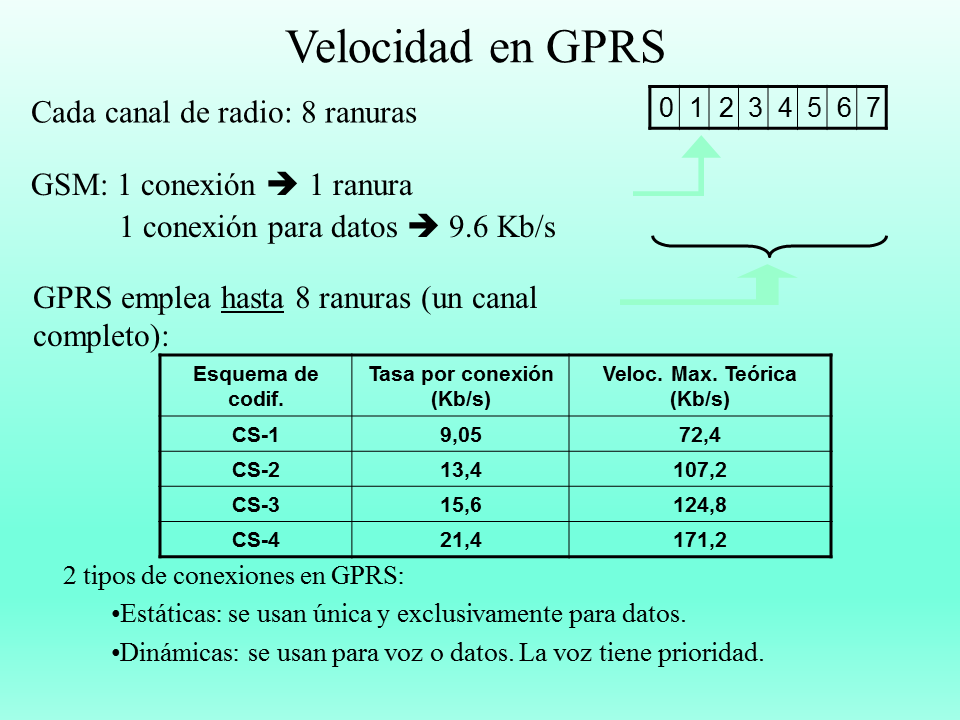


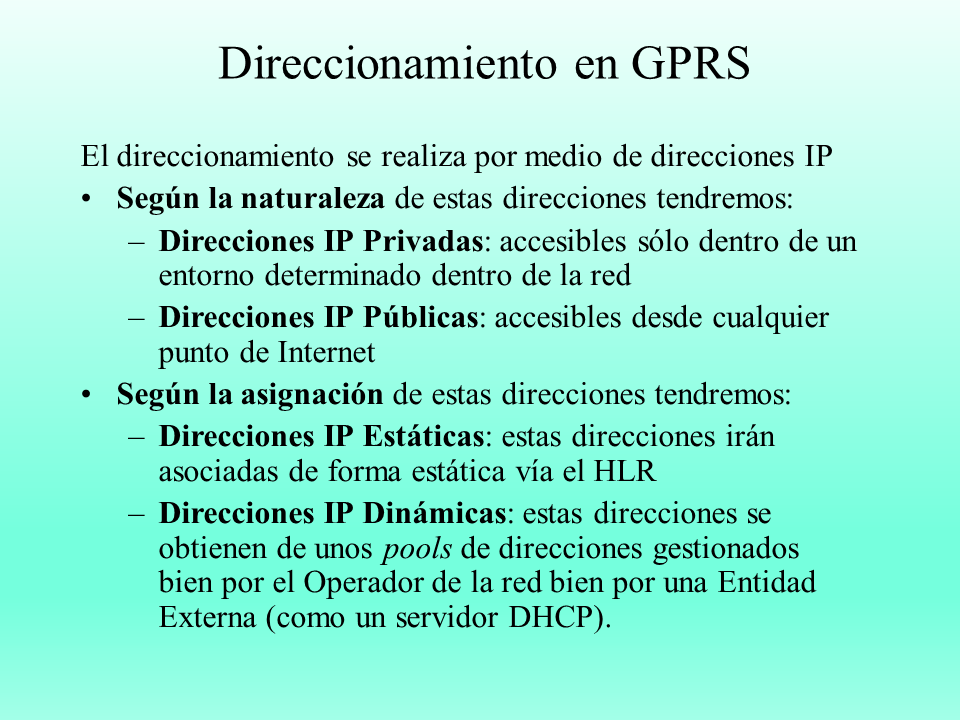


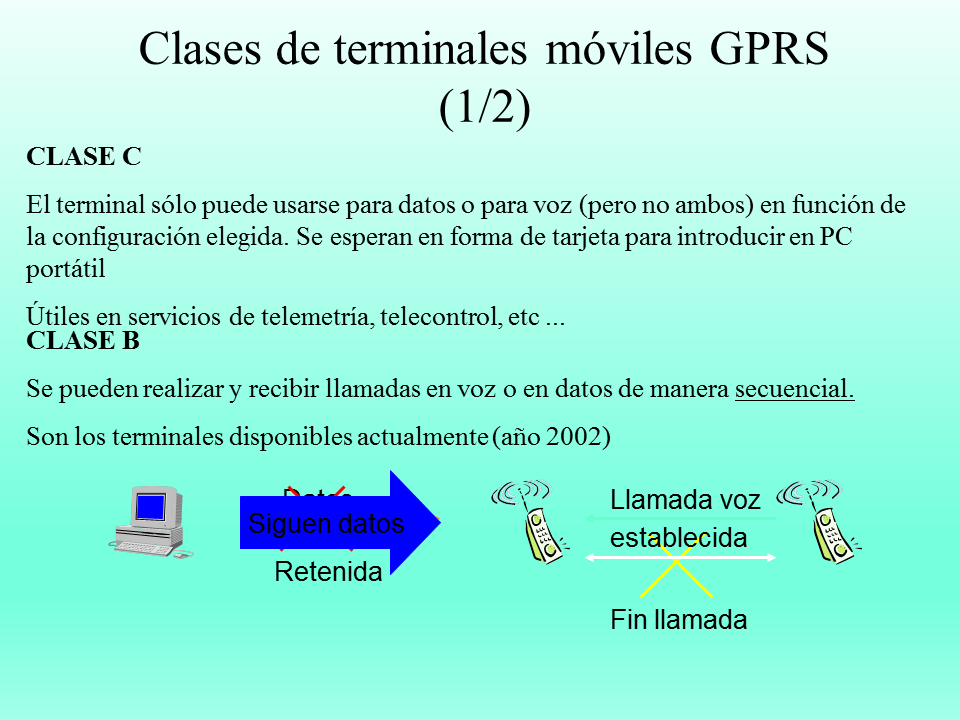


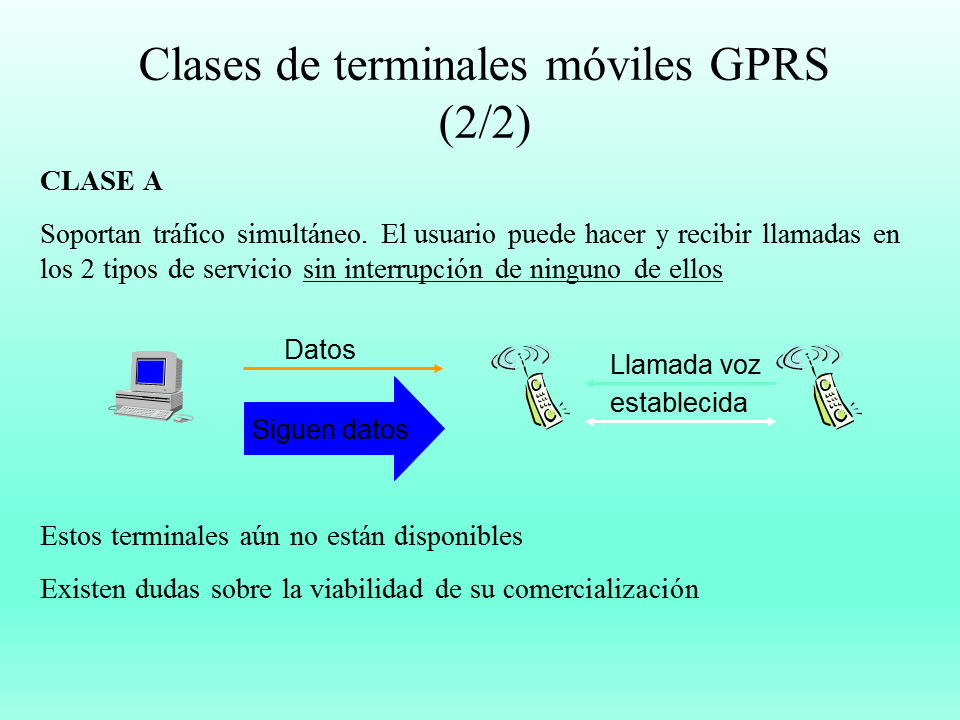




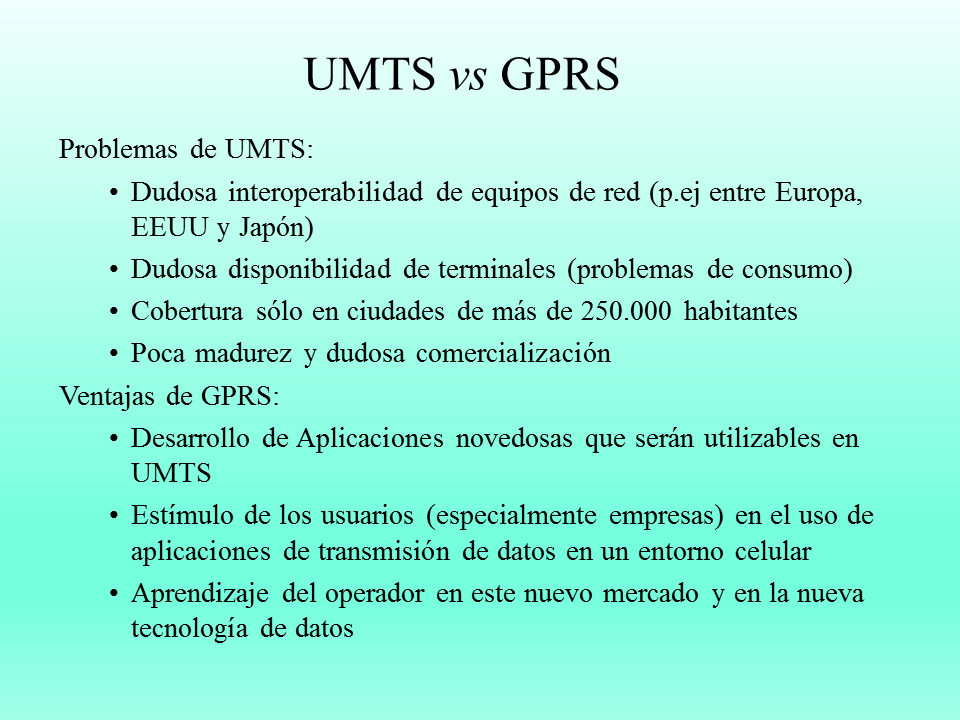










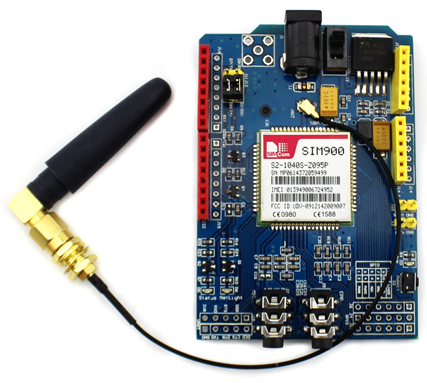


<https://www.monografias.com/trabajos107/redes-telefonia-celular-gsm-gprs-umts/redes-telefonia-celular-gsm-gprs-umts.shtml>

GSM Y GPRS

Hemos visto varias formas de conectar nuestro **Arduino** al exterior, como Ethernet y wifi, pero puede que alguna vez queramos comunicarnos con él y no tengamos acceso a ninguna de estas redes, o no queramos depender de ellas. Puede ser muy útil, por ejemplo, para colocar en una estación meteorológica.

Para este tipo de propósitos podemos utilizar un módulo **GSM/GPRS** con una tarjeta SIM, de forma que podamos comunicarnos con él como si se tratase de un teléfono móvil. Y es que esta tarjeta basada en el módulo **SIM900** nos permite enviar y recibir llamadas y SMS y conectarnos a Internet, transformando nuestro Arduino en un teléfono móvil.



EL **GSM** (Global System for Global Communications) es el sistema de comunicaciones que más se utiliza en teléfonos móviles y es un estándar en Europa.  La primera funcionalidad de esta tecnología es la transmisión de voz, pero también permite la transmisión de datos (SMS, Internet), eso sí,  a una velocidad muy baja de 9kb/s.

El **GPRS** (General Packet Radio Service) es una extensión del GSM basada en la transmisión por paquetes que ofrece un servicio más eficiente para las comunicaciones de datos, especialmente en el caso del acceso a Internet. La velocidad máxima (en teoría) del GPRS es de 171kb/s, aunque en la práctica es bastante más pequeña.

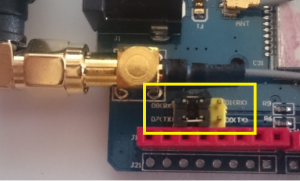
CONEXIONES Y PUESTA EN MARCHA

El módulo que vamos a usar para el tutorial no tiene pines para montarla directamente sobre el Arduino, pero hay modelos que sí lo permiten. De todas formas la conexión es muy sencilla.

Esta tarjeta está basada en el módulo SIM900, y la configuraremos y controlaremos vía UART utilizando los **comandos AT**.

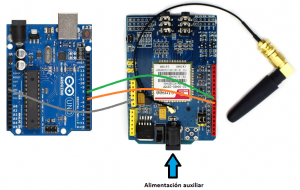
Lo primero tendremos que colocar la tarjeta **SIM** que vayamos a usar. El adaptador para las tarjetas es del tamaño de las SIM normales, que en su mayoría han sido sustituidas por MicroSIM  y NanoSim, así que tendremos que hacernos con una o enganchar una de estas últimas en un adaptador *(el módulo que utilizamos en la sesión y que podéis conseguir en la tienda de esta página incluye adaptadores para todos los tamaños de tarjeta)*. Con un poco de paciencia y de pericia también podríamos colocar una nano o una micro directamente, pero en correremos peligro de que se mueva y deje de hacer contacto.

Antes de conectar nada al Arduino, vamos a colocar los **jumpers** que tiene la tarjeta de forma que utilicemos para comunicarnos los pines 7 y 8.

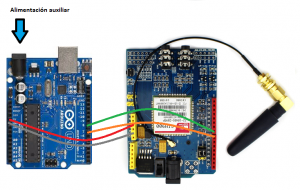
[](https://www.prometec.net/wp-content/uploads/2016/06/jumpers-tarjeta-gprs-pin-7-y-8.png)

Para conectarla a nuestro Arduino tenemos dos opciones, y para las dos necesitaremos una fuente de alimentación externa, ya que con conectar el Arduino por USB será suficiente para encender el módulo, pero no para alimentar también la tarjeta. Si conectamos la alimentación externa al shield GPRS y alimentamos el Arduino con el USB o una fuente aparte, sólo tendremos que unir los pines 7 y 8 para la comunicación serie y el GND entre ambas tarjetas.

* *Si utilizáis un Arduino Mega, la conexión sería de los pines 7 y 8 del GPRS a los pines 10 y 11 del Mega.*
* *En la programación también habría que cambiar la instrucción en la que definimos la instancia Software Serial que creamos y ponerle los pines 10 y 11 (lo recordaremos más adelante).*

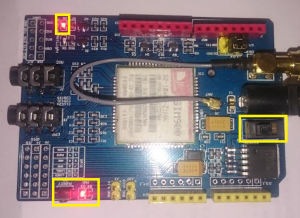


Si conectamos la alimentación externa al Arduino a parte de las tres conexiones anteriores tendremos que unir los pines 5V del Arduino y del GPRS.

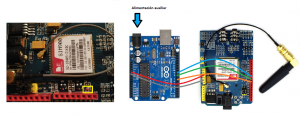
[](https://www.prometec.net/wp-content/uploads/2016/06/conexi%C3%B3n-gprs-arduino-1.png)

Para encender el shield tenemos que colocar el interruptor en la posición correcta. Las dos posiciones corresponden a cada uno de los tipos de conexión que hemos explicado arriba. Una vez hagamos esto se encenderán dos LEDs

Para activar la alimentación de la tarjeta SIM también nos encontramos con dos opciones. Podemos hacerlo manualmente presionando el pulsador que hay en uno de los laterales durante uno o dos segundos y veremos que se enciende otro LED, y que uno de los que se había encendido antes empieza a parpadear una vez por segundo. Este parpadeo nos indica que está buscando la **red.**Cuando la encuentre y se conecte cambiará la frecuencia de parpadeo y lo hará cada 3 o 4 segundos. Eso sí, no se conectará hasta que no metamos el **PIN** de la SIM, que ya veremos ahora cómo hacerlo.

[](https://www.prometec.net/wp-content/uploads/2016/06/leds-gprs.png)

También podemos activarla por programa, pero antes tendremos que realizar una soldadura en el **pad *“R13”*** que está justo al lado de la regleta de pines roja ***“J16”****.* Y una vez hecho esto la activaremos mandando un pulso de 1 segundo al pin 9 de la tarjeta. Así que si queréis usar este tipo de activación tendréis que añadir un cable entre un pin digital *(en nuestro caso el 9)* y el pin 9 de la tarjeta que hemos conectado. Eso sí, recordad que si está activada la tarjeta y volvemos a meter el pulso, lo que haremos será apagarla.

[](https://www.prometec.net/wp-content/uploads/2016/06/Soldadura-R13-1.png)

PROBANDO LOS COMANDOS AT

Para poder comunicarnos vía **comandos AT** tendremos que cargar un programa para la **comunicación serie** como hemos hecho tantas veces anteriormente. Crearemos una instancia llamada SIM808 y seleccionaremos los pines del Arduino que queramos usar para comunicarnos *(Rx y Tx)*. Nosotros hemos elegido el 7 y el 8, pero podéis usar cualquiera que sea compatible con la librería. También podéis cambiar la **velocidad de comunicación**, pero debe ser la misma para el puerto serie y para la instancia que hemos creado. Nosotros hemos elegido 19200 porque es la que usa el SIM900, de forma que podamos usar los programas que ya tenemos de esas sesiones.

#include <SoftwareSerial.h>

SoftwareSerial SIM900(7, 8); //Seleccionamos los pines 7 como Rx y 8 como Tx

void setup()

{

SIM900.begin(19200);

Serial.begin(19200);

delay(1000);

}

void loop()

{

//Envíamos y recibimos datos

if (Serial.available() > 0)

SIM808.write(Serial.read());

if (SIM808.available() > 0)

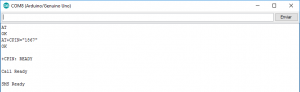
Serial.write(SIM808.read());

}

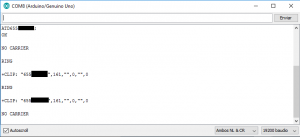
Una vez hayamos cargado el programa abrimos el monitor serie y seleccionamos la velocidad correcta. El primer **comando AT** nos servirá simplemente para saber si el módulo responde y que por lo tanto la comunicación funciona. Y este comando es simplemente ***AT***, lo escribimos y pulsamos *INTRO*. Debería respondernos con un ***OK***; si no deberíamos repasar que esté todo en orden: conexiones, encendido y velocidad correcta.

[monitor serie AT](https://www.prometec.net/wp-content/uploads/2017/01/SIM808-AT.png)

Comprobado esto, ya podemos meter el PIN de la tarjeta SIM, que se hace mediante el comando ***AT+CPIN=”XXXX”****; donde tendréis que sustituir XXXX por el pin que corresponda, en mi caso 1867, o sea,****AT+CPIN=”1867″.***Obtendremos un mensaje de respuesta indicando si el PIN es correcto, y en caso de que lo sea, el LED que parpadeaba una vez por segundo pasará a hacerlo cada 3 segundos (más o menos), indicando que ha encontrado la red móvil.

[](https://www.prometec.net/wp-content/uploads/2017/01/SIM808-ATCPIN.png)

Y ahora ya estamos en disposición de enviar y recibir llamadas y conectarnos a Internet. Para comprobar que efectivamente podéis llamaros al numero de teléfono de la SIM que hayáis puesto en el módulo, o usar el comando ***ATDXXXXXXXXX;*** *(sustituyendo las X por el número de teléfono y respetando el “;”)* para llamar a quien queráis. Y si le conectáis unos cascos y un micrófono al módulo podréis hablar como si fuera un teléfono normal.

[](https://www.prometec.net/wp-content/uploads/2017/01/SIM808-LLAMADAS.png)

O si llamáis al número de teléfono de la SIM que hayáis colocado, oiréis tono y en el monitor serie veréis como aparece “RING”.

REALIZANDO LLAMADAS DESDE ARDUINO

Para comunicarnos con el módulo utilizaremos **comandos AT**, pero lo primero que haremos será incluir la librería *SoftwareSerial.h* y configurar la comunicación por los pines 7 y 8 *(10 y 11 para el Arduino Mega)*.

#include <SoftwareSerial.h>

SoftwareSerial SIM900(7, 8); // 10 y 11 para el Arduino Mega. Configura el puerto serie para el SIM900.

El módulo utiliza una velocidad de comunicación de 19200KHz, así que en el setup configuraremos el puerto serie para el SIM900 y para Arduino a esa velocidad.

Además introduciremos el **PIN** de la tarjeta para desbloquearla utilizando comandos AT y le daremos un poco de tiempo para que se conecte a la red. El comando que utilizaremos es ***AT+CPIN=****“XXXX****“***, donde sustituiremos XXXX por el PIN de nuestra tarjeta.

* *Recordad que como vamos a meter el comando dentro de un println, tendremos que utilizar las \ antes de cada “ para que no se comporten como un carácter especial.*

Así que el **setup** nos quedaría así:

void setup()

{

//digitalWrite(9, HIGH); // Descomentar para activar la alimentación de la tarjeta por Software

//delay(1000);

//digitalWrite(9, LOW);

delay (5000); //Nos damos un tiempo para encender el GPRS y la alimentación de la tarjeta

SIM900.begin(19200);  //Configura velocidad del puerto serie para el SIM900

Serial.begin(19200);  //Configura velocidad del puerto serie del Arduino

Serial.println("OK");

delay (1000);

SIM900.println("AT+CPIN=\"XXXX\"");  //Comando AT para introducir el PIN de la tarjeta

delay(25000);  //Tiempo para que encuentre una RED

}

Crearemos una función en la que meteremos las instrucciones necesarias para llamar y colgar la llamada, también mediante comandos AT. Para llamar usaremos el comando ***ATD****XXXXXXXXX*, *(sustituyendo las X por el número al que queramos llamar)*y para colgar ***ATH***.

No hay que ser tacaño con el tiempo que mantenemos la llamada, porque a veces puede tardar en empezar a dar tono y sino podríamos colgar antes de recibir nada en nuestro móvil.

void llamar()

{

Serial.println("Realizando llamada...");

SIM900.println("ATDXXXXXXXXX;");  //Comando AT para realizar una llamada

delay(20000);  // Espera 20 segundos mientras realiza la llamada

SIM900.println("ATH");  // Cuelga la llamada

delay(1000);

Serial.println("Llamada finalizada");

}

Y ya sólo tenemos que llamar a la función cuando nosotros queramos. En este ejemplo simplemente hacemos que llame una vez y se quede en espera indefinidamente.

void loop()

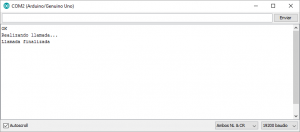
{

llamar();  //Realiza la llamada

while (1);

}

Aquí os dejamos el programa completo: [**llamar**](https://www.prometec.net/wp-content/uploads/2016/06/llamar-2.rar)**.**

[](https://www.prometec.net/wp-content/uploads/2016/06/monitor-serie-realizar-llamada-modulo-gprs-sim900.png)

ENVIAR SMS

La programación para **enviar** un **SMS** va a ser idéntica, pero crearemos otra función que se encargará de mandar los comandos AT para enviar el SMS.

Primero usaremos el comando ***AT+CMGF=1\r*** para indicarle al módulo GPRS que vamos a mandar un mensaje, y a continuación introducimos el número al que va dirigido con el comando ***AT+CMGS =****“XXXXXXXXX”*.

Una vez hecho esto simplemente le enviamos el contenido del mensaje y lo finalizamos con el caracter ***^Z***. La función quedaría así:

void mensaje\_sms()

{

Serial.println("Enviando SMS...");

SIM900.print("AT+CMGF=1\r");  //Configura el modo texto para enviar o recibir mensajes

delay(1000);

SIM900.println("AT+CMGS=\"XXXXXXXXX\"");  //Numero al que vamos a enviar el mensaje

delay(1000);

SIM900.println("SMS enviado desde un Arduino. Saludos de Prometec.");  // Texto del SMS

delay(100);

SIM900.println((char)26); //Comando de finalización ^Z

delay(100);

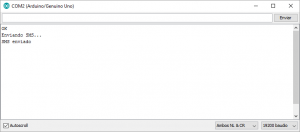
SIM900.println();

delay(5000);  // Esperamos un tiempo para que envíe el SMS

Serial.println("SMS enviado");

}

Podéis descargar el programa completo aquí: [**mandar\_sms**](https://www.prometec.net/wp-content/uploads/2016/06/mandar_sms-1.rar)**.**

[](https://www.prometec.net/wp-content/uploads/2016/06/monitor-serie-enviar-sms-tarjeta-gprs-sim900.png)

Aquí les dejamos un vídeo para que veáis cómo nos ha ido a nosotros:

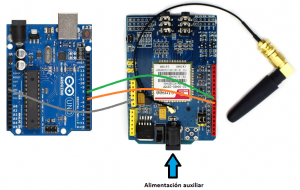
<https://www.youtube.com/watch?time_continue=2&v=AOeOzeRBpkA>

En la siguiente sesión aprenderemos a recibir llamadas y SMS. Y, si os habéis quedáis con ganas de más, os recomendamos que echéis un ojo a [**esta sesión**](https://www.prometec.net/gprs-enviar-recibir-llamadas-sms/). En ella encontraréis un programa más trabajado en el que, entre otras cosas, vamos a usar una función para enviar los comandos AT y asegurarnos de que la respuesta del módulo sea la que esperamos.

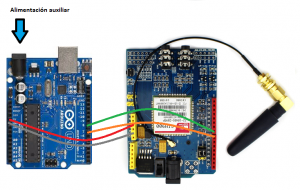
RESUMEN DE LA SESIÓN

En esta sesión hemos aprendido varias cosas importantes:

* + Podemos hacer que nuestro **Arduino** se comporte como **un teléfono móvil.**
  + Cómo conectar y configurar la **tarjeta GPRS.**
  + A **enviar llamadas y SMS** a cualquier teléfono.

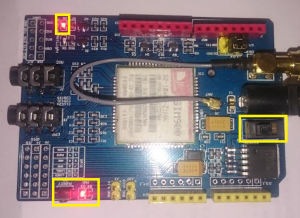


Si conectamos la alimentación externa al Arduino a parte de las tres conexiones anteriores tendremos que unir los pines 5V del Arduino y del GPRS.

[](https://www.prometec.net/wp-content/uploads/2016/06/conexi%C3%B3n-gprs-arduino-1.png)

Para encender el shield tenemos que colocar el interruptor en la posición correcta. Las dos posiciones corresponden a cada uno de los tipos de conexión que hemos explicado arriba. Una vez hagamos esto se encenderán dos LEDs

Para activar la alimentación de la tarjeta SIM también nos encontramos con dos opciones. Podemos hacerlo manualmente presionando el pulsador que hay en uno de los laterales durante uno o dos segundos y veremos que se enciende otro LED, y que uno de los que se había encendido antes empieza a parpadear una vez por segundo. Este parpadeo nos indica que está buscando la **red.**Cuando la encuentre y se conecte cambiará la frecuencia de parpadeo y lo hará cada 3 o 4 segundos. Eso sí, no se conectará hasta que no metamos el **PIN** de la SIM, que ya veremos ahora cómo hacerlo.

[](https://www.prometec.net/wp-content/uploads/2016/06/leds-gprs.png)

También podemos activarla por programa, pero antes tendremos que realizar una soldadura en el **pad *“R13”*** que está justo al lado de la regleta de pines roja ***“J16”****.* Y una vez hecho esto la activaremos mandando un pulso de 1 segundo al pin 9 de la tarjeta. Así que si queréis usar este tipo de activación tendréis que añadir un cable entre un pin digital *(en nuestro caso el 9)* y el pin 9 de la tarjeta que hemos conectado. Eso sí, recuerda que si está activada la tarjeta y volvemos a meter el pulso, lo que haremos será apagarla.

## Getting Started with the Arduino GSM Shield

The [Arduino GSM shield](https://www.arduino.cc/en/Main/ArduinoGSMShield) allows an Arduino board to connect to the internet, send and receive SMS, and make voice calls using the [GSM library](https://www.arduino.cc/en/Reference/GSM).

The shield will work with the Arduino Uno out of the box. The shield will work with the Mega, Mega ADK, Yun, and Leonardo boards with [a minor modification](https://www.arduino.cc/en/Guide/GSMShieldLeonardoMega). The Due is not supported at this time.

The [GSM library](https://www.arduino.cc/en/Reference/GSM) is included with [Arduino IDE 1.0.4 and later](https://www.arduino.cc/en/Main/Software).

* [What is GSM](https://www.arduino.cc/en/Guide/ArduinoGSMShield#toc1)
* [What is GPRS](https://www.arduino.cc/en/Guide/ArduinoGSMShield#toc2)
* [Network operator requirements](https://www.arduino.cc/en/Guide/ArduinoGSMShield#toc3)
* [SIM cards](https://www.arduino.cc/en/Guide/ArduinoGSMShield#toc4)
* [Notes on the Telefonica/Movilforum SIM included with the shield](https://www.arduino.cc/en/Guide/ArduinoGSMShield#toc5)
* [Connecting the Shield](https://www.arduino.cc/en/Guide/ArduinoGSMShield#toc6)
* [GSM Library](https://www.arduino.cc/en/Guide/ArduinoGSMShield#toc7)
* [Testing the modem and network connection](https://www.arduino.cc/en/Guide/ArduinoGSMShield#toc8)
* [Sending a SMS message](https://www.arduino.cc/en/Guide/ArduinoGSMShield#toc9)
* [Connecting to the internet](https://www.arduino.cc/en/Guide/ArduinoGSMShield#toc10)
* [Making voice calls](https://www.arduino.cc/en/Guide/ArduinoGSMShield#toc11)
* [Next steps](https://www.arduino.cc/en/Guide/ArduinoGSMShield#toc12)

### What is GSM

GSM is an international standard for mobile telephones. It is an acronym that stands for Global System for Mobile Communications. It is also sometimes referred to as 2G, as it is a second-generation cellular network.

To use GPRS for internet access, and for the Arduino to request or serve webpages, you need to obtain the Access Point Name (APN) and a username/password from the network operator. See the information in Connecting to the Internet for more information about using the data capabilities of the shield.

Among other things, GSM supports outgoing and incoming voice calls, Simple Message System (SMS or text messaging), and data communication (via GPRS).

The Arduino GSM shield is a a GSM modem. From the mobile operator perspective, the Arduino GSM shield looks just like a mobile phone. From the Arduino perspective, the Arduino GSM shield looks just like a modem.

### What is GPRS

GPRS is a packet switching technology that stands for General Packet Radio Service. It can provide idealized data rates between 56-114 kbit per second.

A number of technologies such as SMS rely on GPRS to function. With the GSM shield, it is also possible to leverage the data communication to access the internet. Similar to the Ethernet and WiFi libraries, the GSM library allows the Arduino to act as a client or server, using http calls to send and receive web pages.

### Network operator requirements

To access a network, you must have a subscription with a mobile phone operator (either prepaid or contract), a GSM compliant device like the GSM shield or mobile phone, and a Subscriber Identity Module (SIM) card. The network operator provides the SIM card, which contains information like the mobile number, and can store limited amounts of contacts and SMS messages.

To use GPRS for internet access, and for the Arduino to request or serve webpages, you need to obtain the Access Point Name (APN) and a username/password from the network operator. See the information in [Connecting to the Internet](https://www.arduino.cc/en/Guide/ArduinoGSMShield#toc10)for more information about using the data capabilities of the shield.

### SIM cards

In addition to the GSM shield and an Arduino, you need a SIM card. The SIM represents a contract with a communications provider. The communications provider selling you the SIM has to either provide GSM coverage where you are, or have a roaming agreement with a company providing GSM coverage in your location.

It's common for SIM cards to have a four-digit PIN number associated with them for security purposes. Keep note of this number, as it's necessary for connecting to a network. If you lose the PIN associated with your SIM card, you may need to contact your network operator to retrieve it. Some SIM cards become locked if an incorrect PIN is entered too many times. If you're unsure of what the PIN is, look at the documentation that came with your SIM.

Using a PUK (PIN Unlock Code), it is possible to reset a lost PIN with the GSM shield and an Arduino. The PUK number will come with your SIM card documentation.

Look at the [PIN Management](https://www.arduino.cc/en/Tutorial/GSMToolsPinManagement) example in the "tools" folder, bundled with the GSM library for an example of how to manage your PIN number with the PUK.

There are a few different sizes of SIM cards; the GSM shield accepts cards in the mini-SIM format (25mm long and 15mm wide).

### Notes on the Telefonica/Movilforum SIM included with the shield

The GSM shield comes bundled with a SIM from Telefonica/Movilforum that will work well for developing machine to machine (M2M) applications. It is not necessary to use this specific card with the shield. You may use any SIM that works on a network in your area.

The Movilforum SIM card includes a roaming plan. It can be used on any supported GSM network. There is coverage throughout the Americas and Europe for this SIM, check the [Movilforum service availability page](http://arduinosim.movilforum.com/service.php) for specific countries that have supported networks.

Activation of the SIM is handled by Movilforum. Detailed instructions on how to register and activate your SIM online and add credit are included on a [small pamphlet](https://www.arduino.cc/en/uploads/Guide/GSMShield_Flyer.pdf) that comes with your shield. The SIM must be inserted into a powered GSM shield that is mounted on an Arduino for activation.

These SIM card come without a PIN, but it is possible to set one using the GSM library's [GSMPIN class](https://www.arduino.cc/en/Reference/GSMPINConstructor).

You cannot use the included SIM to place or receive voice calls.

You can only place and receive SMS with other SIMs on the Movilforum network.

It's not possible to create a server that accepts incoming requests from the public internet. However, the Movilforum SIM will accept incoming requests from other SIM cards on the Movilforum network.

For using the voice, and other functions of the shield, you'll need to find a different network provider and SIM. Operators will have different policies for their SIM cards, check with them directly to determine what types of connections are supported.

### Connecting the Shield

If you are using an Arduino Uno, follow the instructions below. If you are using an Arduino Mega, Mega ADK, Yun, or Leonardo, you must [follow these instructions](https://www.arduino.cc/en/Guide/GSMShieldLeonardoMega). The GSM shield is not currently supported on the Due.

To use the shield, you'll need to insert a SIM card into the holder. Slide the metal bracket away from the edge of the shield and lift the cradle up.



Insert the SIM in the plastic holder so the metal contacts are facing the shield, with the notch of the card at the top of the bracket.



Slide the SIM all the way into the bracket



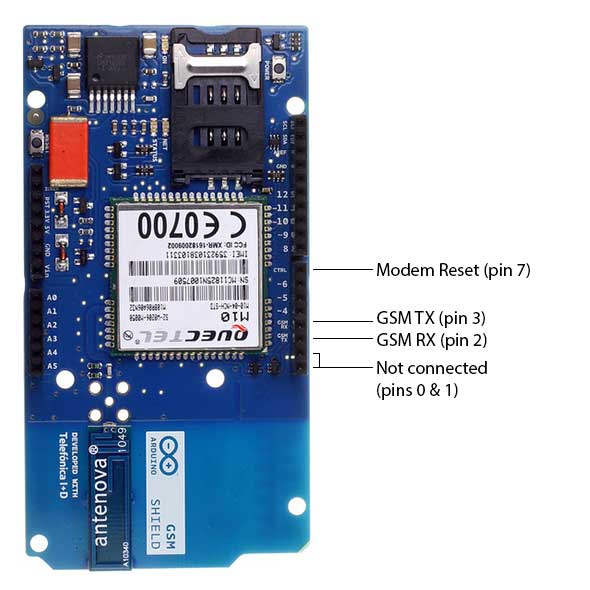
Push the SIM to the board and slide the metal bracket towards the edge of the shield to lock it in place.



Once the SIM is inserted, mount it on top of an Arduino board.



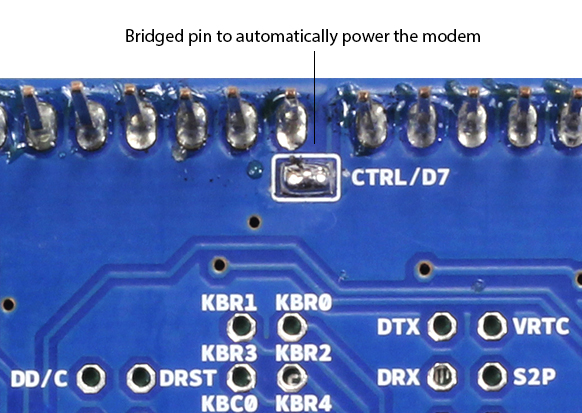
To upload sketches to the board, connect it to your computer with a USB cable and upload your sketch with the Arduino IDE. Once the sketch has been uploaded, you can disconnect the board from your computer and power it with an external power supply.



Digital pins 2, 3 and 7 are reserved for communication between the Arduino and modem and cannot be used by your sketches. Communication between the moden and Arduino is handled by the [Software Serial library](https://www.arduino.cc/en/Reference/SoftwareSerial) on pins 2 and 3. Pin 7 is used for the modem reset.

When the yellow status LED turns on, it means the modem is powered, and you can try connecting to the network.

Developer versions of the GSM shield required you to press press the Power button on the shield for a few moments to turn the modem on. If you have an early version of the shield, and it does not turn on automatically, you can solder a jumper to the CTRL/D7 pad on the reverse side of the board, and it will turn on when an attached Arduino receives power.



The shield should work in any area with GSM coverage. Before buying the shield please verify that there is this kind of coverage where you plan to use it.

### GSM Library

The GSM library handles communication between Arduino and the GSM shield. The majority of functions are for managing data, voice, and SMS communication. There are also a number of utilities for managing information about the modem and the SIM card's PIN. See the [library reference](https://www.arduino.cc/en/Reference/GSM) pages for more information and a complete set of examples.

### Testing the modem and network connection

This sketch will check the modem's IMEI number. This number is unique to each modem, and is used to identify valid devices that can connect to a GSM network. Once the number has been read from the modem, the Arduino will print out the network carrier it is connected to, and the signal strength of the network over the serial port.

*// import the GSM library*  
#include <GSM.h>  
  
*// PIN Number*  
#define PINNUMBER ""  
  
*// initialize the library instance*  
GSM gsmAccess(true);     *// include a 'true' parameter for debug enabled*  
GSMScanner scannerNetworks;  
GSMModem modemTest;  
  
*// Save data variables*  
String IMEI = "";  
  
*// serial monitor result messages*  
String errortext = "ERROR";  
  
void **setup**()  
{  
  *// initialize serial communications*  
  Serial.begin(9600);  
  Serial.println("GSM networks scanner");  
  scannerNetworks.begin();  
  
  *// connection state*  
  boolean notConnected = true;  
  
  *// Start GSM shield*  
  *// If your SIM has PIN, pass it as a parameter of begin() in quotes*  
  while(notConnected)  
  {  
    if(gsmAccess.begin(PINNUMBER)==GSM\_READY)  
      notConnected = false;  
    else  
    {  
      Serial.println("Not connected");  
      delay(1000);  
    }  
  }  
  
  *// get modem parameters*  
  *// IMEI, modem unique identifier*  
  Serial.print("Modem IMEI: ");  
  IMEI = modemTest.getIMEI();  
  IMEI.replace("**\n**","");  
  if(IMEI != NULL)  
    Serial.println(IMEI);  
  
  *// currently connected carrier*  
  Serial.print("Current carrier: ");  
  Serial.println(scannerNetworks.getCurrentCarrier());  
  
  *// returns strength and ber*  
  *// signal strength in 0-31 scale. 31 means power > 51dBm*  
  *// BER is the Bit Error Rate. 0-7 scale. 99=not detectable*  
  Serial.print("Signal Strength: ");  
  Serial.print(scannerNetworks.getSignalStrength());  
  Serial.println(" [0-31]");  
}  
  
void **loop**()  
{  
  *// scan for existing networks, displays a list of networks*  
  Serial.println("Scanning available networks. May take some seconds.");  
  
  Serial.println(scannerNetworks.readNetworks());  
  
    *// currently connected carrier*  
  Serial.print("Current carrier: ");  
  Serial.println(scannerNetworks.getCurrentCarrier());  
  
  *// returns strength and ber*  
  *// signal strength in 0-31 scale. 31 means power > 51dBm*  
  *// BER is the Bit Error Rate. 0-7 scale. 99=not detectable*  
  Serial.print("Signal Strength: ");  
  Serial.print(scannerNetworks.getSignalStrength());  
  Serial.println(" [0-31]");  
  
}

[[Get Code]](https://www.arduino.cc/en/Guide/ArduinoGSMShield?action=sourceblock&num=1)

### Sending a SMS message

Once you have connected to your network with the sketch above, you can test some of the other functionality of the board. This sketch will connect to a GSM network and send a SMS message to a phone number of your choosing.

#include <GSM.h>  
  
#define PINNUMBER ""  
  
*// initialize the library instance*  
GSM gsmAccess; *// include a 'true' parameter for debug enabled*  
GSM\_SMS sms;  
  
*// char array of the telephone number to send SMS*  
*// change the number 1-212-555-1212 to a number*  
*// you have access to*  
char remoteNumber[20]= "12125551212";    
  
*// char array of the message*  
char txtMsg[200]="Test";  
  
void **setup**()  
{  
  *// initialize serial communications*  
  Serial.begin(9600);  
  
  Serial.println("SMS Messages Sender");  
  
  *// connection state*  
  boolean notConnected = true;  
  
  *// Start GSM shield*  
  *// If your SIM has PIN, pass it as a parameter of begin() in quotes*  
  while(notConnected)  
  {  
    if(gsmAccess.begin(PINNUMBER)==GSM\_READY)  
      notConnected = false;  
    else  
    {  
      Serial.println("Not connected");  
      delay(1000);  
    }  
  }  
  Serial.println("GSM initialized");  
  sendSMS();  
}  
  
void **loop**()  
{  
*// nothing to see here*  
}  
  
void sendSMS(){  
  
  Serial.print("Message to mobile number: ");  
  Serial.println(remoteNumber);  
  
  *// sms text*  
  Serial.println("SENDING");  
  Serial.println();  
  Serial.println("Message:");  
  Serial.println(txtMsg);  
  
  *// send the message*  
  sms.beginSMS(remoteNumber);  
  sms.print(txtMsg);  
  sms.endSMS();   
  Serial.println("**\n**COMPLETE!**\n**");    
}

[[Get Code]](https://www.arduino.cc/en/Guide/ArduinoGSMShield?action=sourceblock&num=2)

### Connecting to the internet

In addition to the SIM card and a data plan, you will need some additional information from your cellular provider to connect to the internet. Every cellular provider has an Access Point Name (APN) that serves as a bridge between the cellular network and the internet. Sometimes, there is a username and password associated with the connection point. For example, the Movilforum APN is sm2ms.movilforum.es, but it has no password or login name.

[This page](http://arduinosim.movilforum.com/service.php) lists a number of carrier's information, but it may not be up to date. You may need to get this information from your service provider.

The sketch below will connect to arduino.cc/latest.txt and print out its contents.

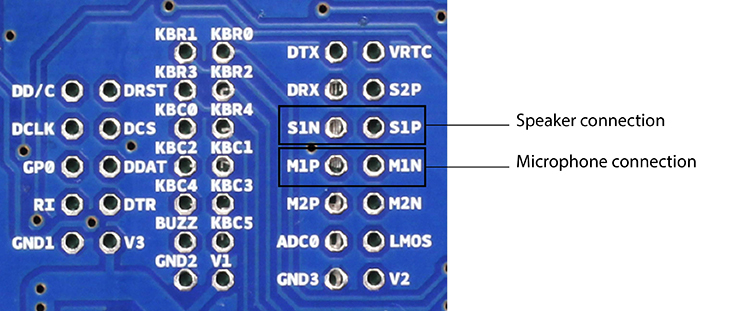
NB: Some network operators block incoming IP traffic. You should be able to run client functions, such as the sketch below, with no issues.

*// include the GSM library*  
#include <GSM.h>  
  
*// PIN number if necessary*  
#define PINNUMBER ""  
  
*// APN information obrained from your network provider*  
#define GPRS\_APN       "GPRS\_APN" // replace with your GPRS APN  
#define GPRS\_LOGIN     "login"    // replace with your GPRS login  
#define GPRS\_PASSWORD  "password" // replace with your GPRS password  
  
*// initialize the library instances*  
GSMClient client;  
GPRS gprs;  
GSM gsmAccess;  
  
*// This example downloads the URL "http://arduino.cc/latest.txt"*  
  
char server[] = "arduino.cc"; *// the base URL*  
char path[] = "/latest.txt"; *// the path*  
int port = 80; *// the port, 80 for HTTP*  
  
void **setup**()  
{  
  *// initialize serial communications*  
  Serial.begin(9600);  
  Serial.println("Starting Arduino web client.");  
  *// connection state*  
  boolean notConnected = true;  
  
  *// Start GSM shield*  
  *// pass the PIN of your SIM as a parameter of gsmAccess.begin()*  
  while(notConnected)  
  {  
    if((gsmAccess.begin(PINNUMBER)==GSM\_READY) &  
        (gprs.attachGPRS(GPRS\_APN, GPRS\_LOGIN, GPRS\_PASSWORD)==GPRS\_READY))  
      notConnected = false;  
    else  
    {  
      Serial.println("Not connected");  
      delay(1000);  
    }  
  }  
  
  Serial.println("connecting...");  
  
  *// if you get a connection, report back via serial:*  
  if (client.connect(server, port))  
  {  
    Serial.println("connected");  
    *// Make a HTTP request:*  
    client.print("GET ");  
    client.print(path);  
    client.println(" HTTP/1.0");  
    client.println();  
  }   
  else  
  {  
    *// if you didn't get a connection to the server:*  
    Serial.println("connection failed");  
  }  
}  
  
void **loop**()  
{  
  *// if there are incoming bytes available*  
  *// from the server, read them and print them:*  
  if (client.available())  
  {  
    char c = client.read();  
    Serial.print(c);  
  }  
  
  *// if the server's disconnected, stop the client:*  
  if (!client.available() && !client.connected())  
  {  
    Serial.println();  
    Serial.println("disconnecting.");  
    client.stop();  
  
    *// do nothing forevermore:*  
    for(;;)  
      ;  
  }  
}

[[Get Code]](https://www.arduino.cc/en/Guide/ArduinoGSMShield?action=sourceblock&num=3)

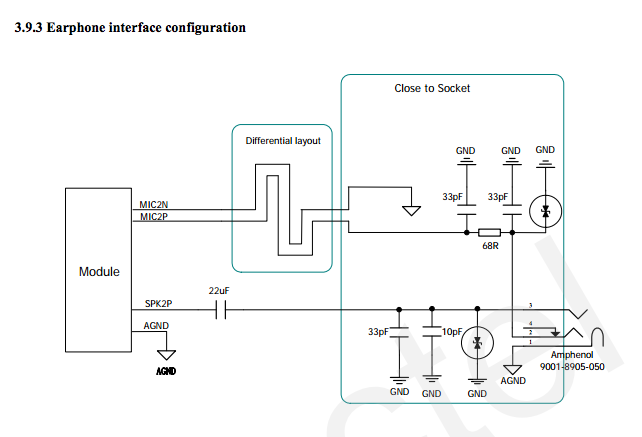
### Making voice calls

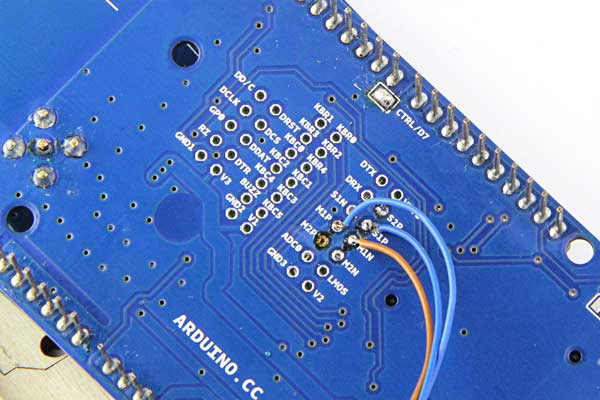
Through the modem, it is possible to make voice calls. In order to speak to and hear the other party, you will need to add a speaker and microphone.

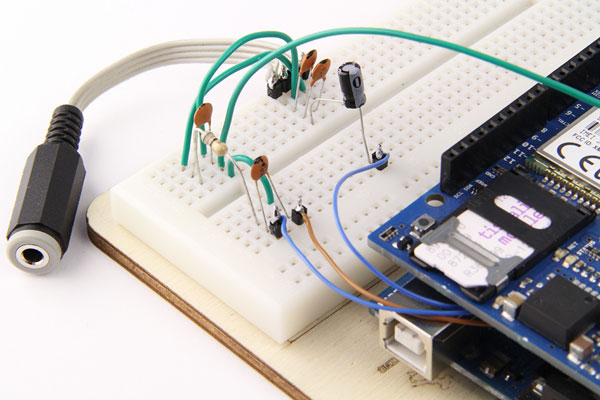


On the underside of the shield, there are through-holes labeled M1P and M1N. These are the positive and negative voice input pins for a microphone. The through-holes labeled S1P and S1N are the positive and negative voice output pins, to which you need to connect a speaker.

On page 43 of the [modem documentation](http://www.thaieasyelec.net/archives/Manual/M10_HD_V1.00.pdf), there is an example voice and sound circuit that will connect to an earphone:







The following sketch allows you to place a voice call. Using the serial monitor, you can enter the remote phone number and terminate the call. When you see the READY message, type a phone number. Make sure the serial monitor is set to send a just newline when you press return.

#include <GSM.h>  
  
*// PIN Number*  
#define PINNUMBER ""  
  
*// initialize the library instance*  
GSM gsmAccess; *// include a 'true' parameter for debug enabled*  
GSMVoiceCall vcs;  
  
String remoteNumber = "";  *// the number you will call*  
char charbuffer[20];  
  
void **setup**()  
{  
  
  *// initialize serial communications*  
  Serial.begin(9600);   
  
  Serial.println("Make Voice Call");  
  
  *// connection state*  
  boolean notConnected = true;  
  
  *// Start GSM shield*  
  *// If your SIM has PIN, pass it as a parameter of begin() in quotes*  
  while(notConnected)  
  {  
    if(gsmAccess.begin(PINNUMBER)==GSM\_READY)  
      notConnected = false;  
    else  
    {  
      Serial.println("Not connected");  
      delay(1000);  
    }  
  }  
  
  Serial.println("GSM initialized.");  
  Serial.println("Enter phone number to call.");  
  
}  
  
void **loop**()  
{  
  
  *// add any incoming characters to the String:*  
  while (Serial.available() > 0)  
  {  
    char inChar = Serial.read();  
    *// if it's a newline, that means you should make the call:*  
    if (inChar == '**\n**')  
    {  
      *// make sure the phone number is not too long:*  
      if (remoteNumber.length() < 20)  
      {  
        *// show the number you're calling:*  
        Serial.print("Calling to : ");  
        Serial.println(remoteNumber);  
        Serial.println();  
  
        *// Call the remote number*  
        remoteNumber.toCharArray(charbuffer, 20);  
  
  
        *// Check if the receiving end has picked up the call*  
        if(vcs.voiceCall(charbuffer))  
        {  
          Serial.println("Call Established. Enter line to end");  
          *// Wait for some input from the line*  
          while(Serial.read()!='**\n**' && (vcs.getvoiceCallStatus()==TALKING));            
          *// And hang up*  
          vcs.hangCall();  
        }  
        Serial.println("Call Finished");  
        remoteNumber="";  
        Serial.println("Enter phone number to call.");  
      }   
      else  
      {  
        Serial.println("That's too long for a phone number. I'm forgetting it");   
        remoteNumber = "";  
      }  
    }   
    else  
    {  
      *// add the latest character to the message to send:*  
      if(inChar!='**\r**')  
        remoteNumber += inChar;  
    }  
  }   
}

[[Get Code]](https://www.arduino.cc/en/Guide/ArduinoGSMShield?action=sourceblock&num=4)

### Next steps

Now that you have tested the basic functionality of the board, see the [GSM library](https://www.arduino.cc/en/Reference/GSM) pages for information about the library's API and additional examples.   
  
The text of the Arduino getting started guide is licensed under a [Creative Commons Attribution-ShareAlike 3.0 License](http://creativecommons.org/licenses/by-sa/3.0/). Code samples in the guide are released into the public domain.