



ANDROID PARA INGENIEROS

2ª EDICIÓN

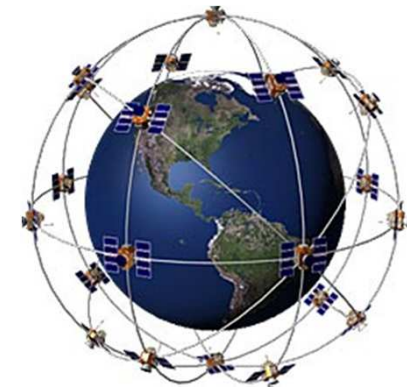
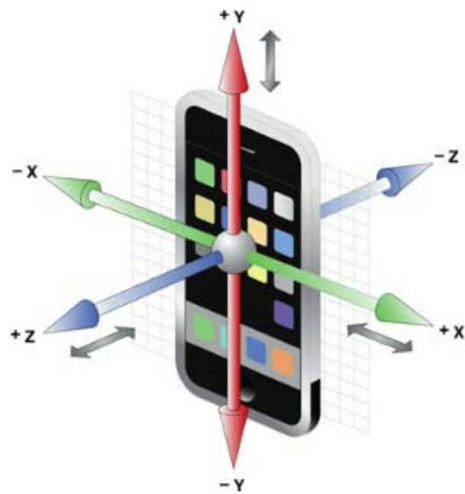
3ª SESIÓN

Curso 2015-2016

ÍNDICE

1. Sensores del Dispositivo
2. El localizador (GPS)
3. Códigos QR
4. La Cámara
5. La Galería de Imágenes
6. Mandar Emails

1-SENSORES DEL DISPOSITIVO



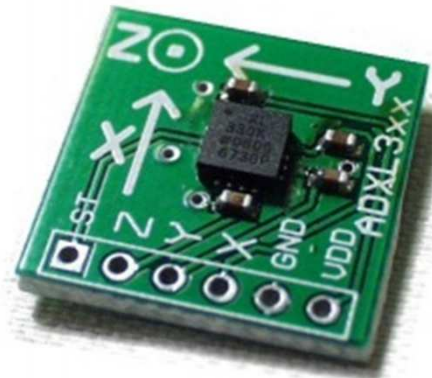
1-SENSORES DEL DISPOSITIVO

- Un sensor es un dispositivo capaz de detectar magnitudes físicas o químicas y transformarlas en variables eléctricas.
- Algunas de las magnitudes que puede medir un sensor son: temperatura, humedad, distancia, inclinación, desplazamiento, fuerza, presión, etc.
- Las dos características más importantes de un sensor son:
 - La resolución
 - La precisión

1-SENSORES DEL DISPOSITIVO

- Vamos a ver dos de los sensores más importantes que tienen la mayoría de nuestros dispositivos Android:
 - Acelerómetro
 - Sensor de luz
 - Otros

1-SENSORES. ACELERÓMETRO



1-SENSORES. ACELERÓMETRO

- El acelerómetro es un instrumento destinado a medir aceleraciones.
- Sólo mide los cambios de aceleraciones. Por ejemplo, para un objeto está en caída libre el acelerómetro dará una aceleración igual a cero.

1-SENSORES. ACELERÓMETRO

- El acelerómetro más común es el piezoeléctrico por compresión. Cuando se comprime, este sensor produce una carga proporcional a la fuerza aplicada.
- Actualmente es posible construir acelerómetros de tres ejes (X, Y, Z) en un solo chip de silicio, incluyendo toda la parte electrónica que se encarga de procesar las señales.

1-SENSORES. ACELERÓMETRO

- La medida de la fuerza del acelerómetro está en m/s^2 y se aplica a los tres ejes físicos del dispositivo (x, y, z) incluyendo la fuerza de la gravedad.
- Vamos a usar una API 14 o mayor.

1-SENSORES. ACELERÓMETRO

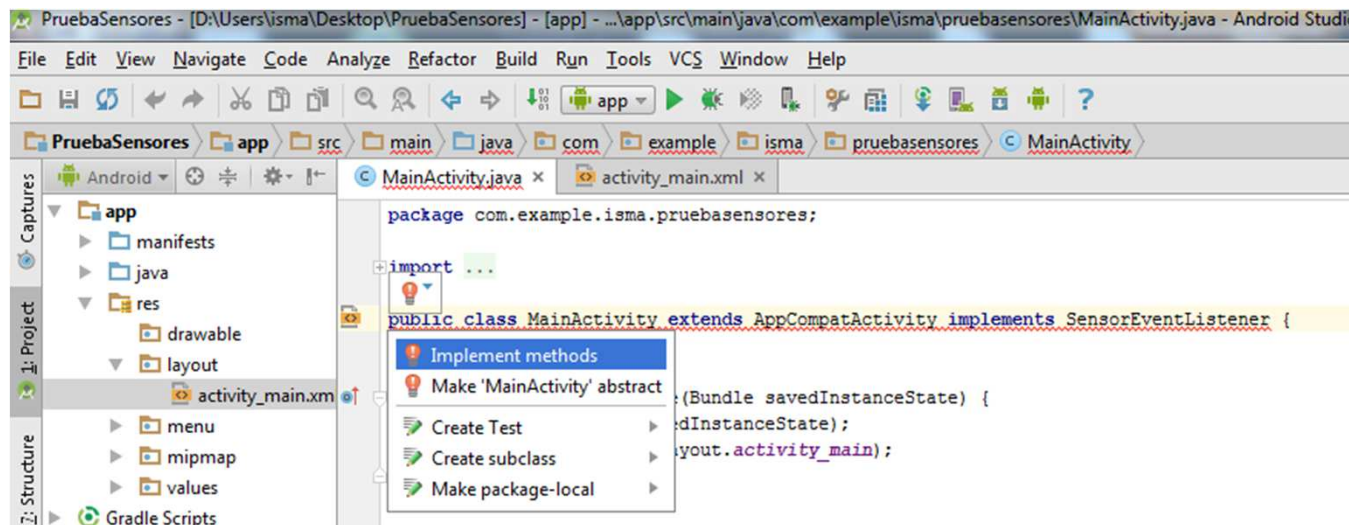
- Vamos a ver un ejemplo de como se maneje el sensor de aceleración de nuestro dispositivo. Necesitamos realizar cuatro pasos:
 - Implementar la clase «SensorEventListener»
 - Crear un «Manager» para manejar sensores.
 - Escoger el sensor que queremos manejar.
 - Registrar el sensor con el «Manager»

1-SENSORES. ACELERÓMETRO

- Para implementar la clase «SensorEventListener» solo tenemos que añadir a nuestro «MainActivity»:

public class MainActivity extends AppCompatActivity implements SensorEventListener {

- Una vez añadido nos saldrá un error:



1-SENSORES. ACELERÓMETRO

- Pulsando en implementar métodos nos creará dos métodos automáticamente que necesitamos: «onSensorChaged()» y «onAccuracyChanged()».
- Creamos dos variable globales para el «Manager» y el Sensor:

```
private SensorManager sm;  
private Sensor sensorAcelerometro;
```

- Completamos el método «onSensorChanged» para mostrar la salida del sensor por tres «TextEdit» que debemos haber creado antes:

```
@Override  
public void onSensorChanged(SensorEvent event) {  
    if(event.sensor == sensorAcelerometro) {  
        synchronized (this) {  
            txt_x.setText("X=" + event.values[0]);  
            txt_y.setText("Y=" + event.values[1]);  
            txt_z.setText("Z=" + event.values[2]);  
        }  
    }  
}
```

1-SENSORES. ACELERÓMETRO

- A continuación y usando un botón vamos a crear la funcionalidad de reanudar. Dentro del escuchador de un botón vamos a llamar a la función «onResume()» , que tendrá este contenido (dentro de la clase «MainActivity»):

```
@Override
protected void onResume() {
    sm = (SensorManager) getSystemService(SENSOR_SERVICE);
    sensorAcelerometro = sm.getSensorList(
        Sensor.TYPE_ACCELEROMETER ).get(0);

    sm.registerListener(this, sensorAcelerometro,
        SensorManager.SENSOR_DELAY_GAME);

    super.onResume();
}
```

Creamos un «Manager» de sensores

Seleccionamos el sensor que queremos manejar.

Establecemos un «delay» para obtener información del sensor (60.000ms)

1-SENSORES. ACELERÓMETRO

- Después y usando un botón vamos a crear la funcionalidad de parar. Dentro del escuchador de un botón vamos a llamar a la función «onStop()», que tendrá este contenido (también dentro de la clase «MainActivity»):

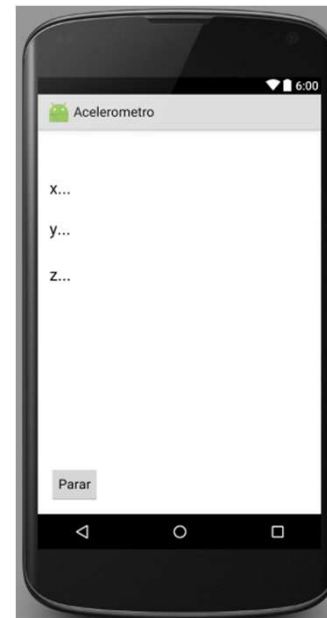
```
@Override  
protected void onStop() {
```

```
    sm.unregisterListener(this);  
    super.onStop();  
}
```

Desvinculamos el
«Manager» para detener
el sensor.

EJERCICIO PROPUESTO

- Crear una aplicación para tu dispositivo Android para obtener las aceleraciones en X, Y, Z de nuestro dispositivo.
 - Utilizar el acelerómetro.
 - Permitir parar y reanudar la recogida de datos.
 - Ver un ejemplo de interfaz:



1-SENSORES. ILUMINACIÓN

- El sensor de iluminación puede dar información de la luminosidad que hay donde está nuestro dispositivo.
- Este sensor mide si hay mucha luz o poca.
- Proporciona un valor de 1000 cuando hay máxima luminosidad y 0 cuando no hay luz.

1-SENSORES. ILUMINACIÓN

- Este sensor puede ser útil para controlar el brillo de la pantalla.
- Vamos a usar una API 14 o mayor.

1-SENSORES. ILUMINACIÓN

- El uso es muy parecido al componente anterior. Ahora vamos a crear un «Sensor» para controlar el nivel de luz (variable global):

private Sensor sensorLuz;

- Crearemos el sensor igual que antes, dentro de la función «onResume()»

...

sensorLuz = sm.getSensorList(Sensor.TYPE_LIGHT).get(0);

...

1-SENSORES. ILUMINACIÓN

- Finalmente dentro de la función «onSensorChanged()» implementamos:

```
@Override  
public void onSensorChanged(SensorEvent event) {  
    if(event.sensor == sensorLuz) {  
        txt_luz.setText("Nivel de luz = " + event.values[0]);  
    }  
}
```

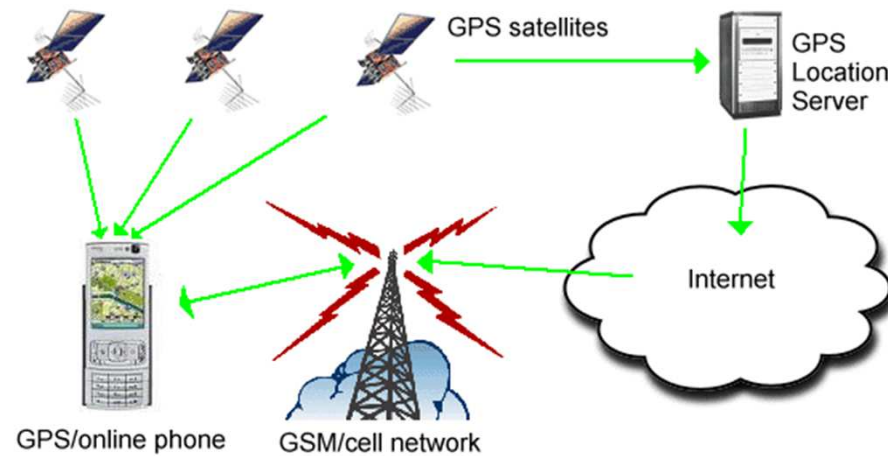
EJERCICIO PROPUESTO

- Crear una aplicación para tu dispositivo Android para obtener la luminosidad ambiental.
 - Poner un botón de reanudar y parar.
 - Mostrar la luminosidad actual con un «TextView»
 - Si la luminosidad es menor que 200 cambiar el fondo del «TextView» a rojo, si no ponerlo a verde.
 - (Ayuda) Utiliza la función siguiente para modificar el color de un componente:
`txt_luz.setBackgroundColor(Color.argb(alpha, rojo, verde, azul));`

1-SENSORES.

- A parte de los dos sensores que hemos visto, Android permite trabajar con los siguientes sensores (una vez que los tenga y soporte nuestro dispositivo):
 - *Sensor de temperatura* → *TYPE_AMBIENT_TEMPERATURE*
 - *Sensor de gravedad* → *TYPE_GRAVITY*
 - *Giroscopio* → *TYPE_GYROSCOPE*
 - *Sensor de orientación* → *TYPE_ORIENTATION*
 - *Sensor de presión* → *TYPE_PRESSURE*
 - *Sensor de proximidad* → *TYPE_PROXIMITY*
 - *Sensor de humedad* → *TYPE_RELATIVE_HUMIDITY*
 - *Sensor de temperatura* → *TYPE_TEMPERATURE*
- Se puede ver más información aquí:
http://developer.android.com/guide/topics/sensors/sensors_overview.html

2- EL LOCALIZADOR (GPS)



2- EL LOCALIZADOR (GPS)

- El sensor de localización es un dispositivo que puede obtener la posición global.
- El sensor de localización más extendido es el GPS que permite detectar en todo el mundo la posición de un objeto.
- La precisión del GPS oscila entre unos centímetros (GPS diferencial) a unos metros que es lo más habitual.

2- EL LOCALIZADOR (GPS)

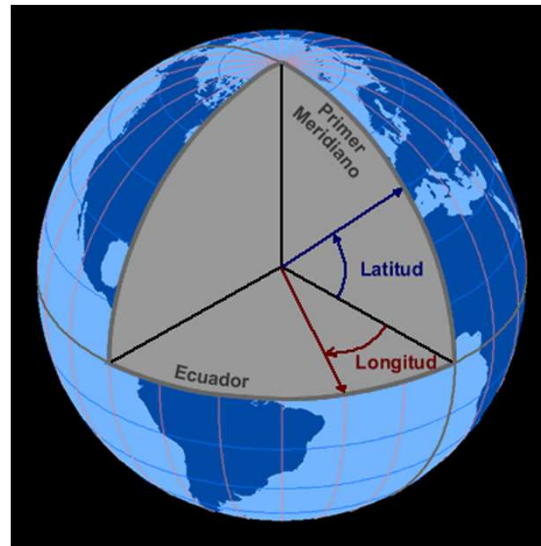
- El GPS funciona mediante una red de 24 satélites en orbita a 20.200 Km de altura, con trayectorias sincronizadas para cubrir toda la superficie de la Tierra.
- Cuando se quiere obtener la posición de un objeto, el receptor localiza automáticamente como mínimo 3 satélites de la red.
- El receptor recibe la identificación de cada satélite y su hora de reloj. Con estos datos se calcula el tiempo que tardan en llegar las señales y se calcula la distancia al satélite mediante triangulación.

2- EL LOCALIZADOR (GPS)

- Debido al carácter militar del GPS, el Departamento de Defensa de los EE. UU. se reserva la posibilidad de incluir un error aleatorio al GPS que oscila entre 15-100 metros.
- El GPS diferencial es un GPS colocado en un punto donde conocemos las coordenadas exactas. Comparando las coordenadas exactas y la posición obtenida del GPS se puede calcular el error incluido.
 - De esta forma se puede mejorar la recepción de otros GPS cercanos ya que tendrán el mismo error.

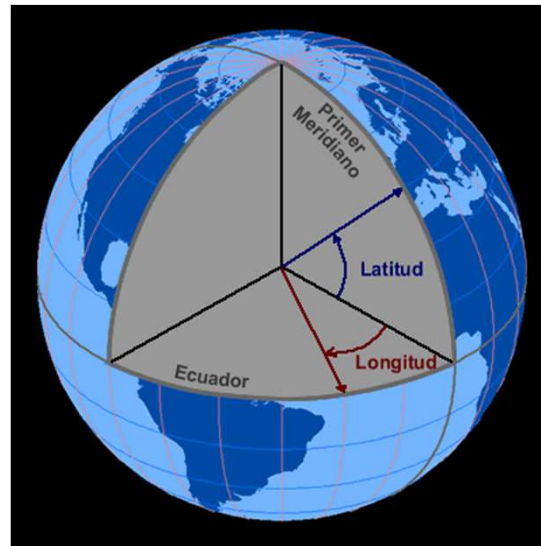
2- EL LOCALIZADOR (GPS)

- Latitud, es la distancia angular que existe desde cualquier punto de la Tierra respecto al ecuador. Todos los puntos sobre un mismo paralelo tienen la misma latitud.



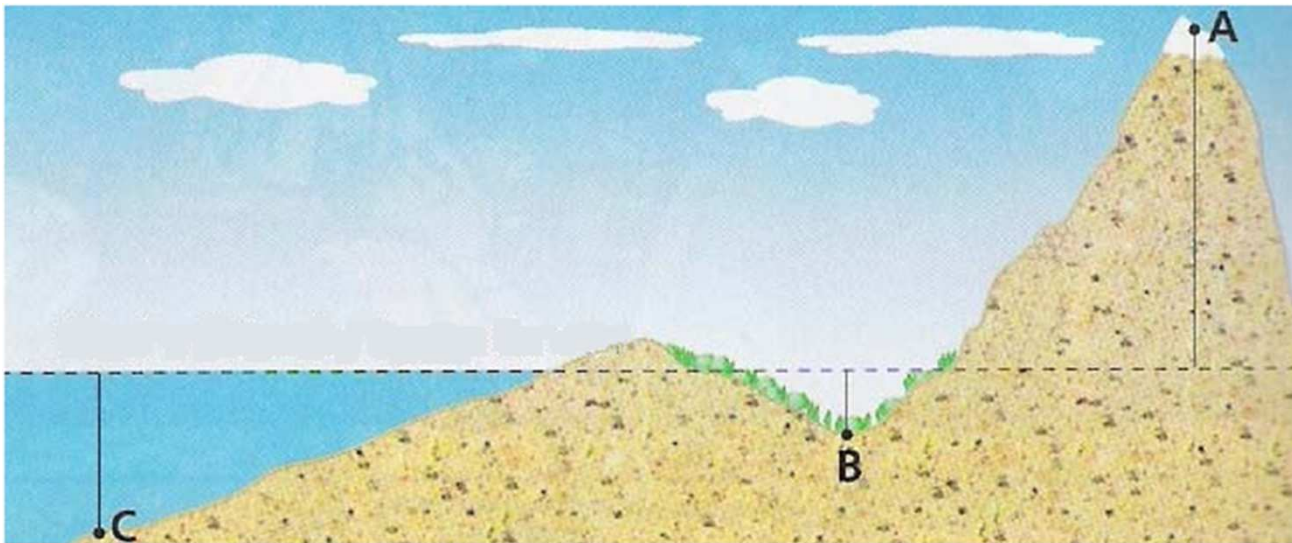
2- EL LOCALIZADOR (GPS)

- Longitud, es la distancia angular que existe desde cualquier punto de la Tierra respecto a meridiano de Greenwich. Todos los puntos ubicados en el mismo meridiano tienen la longitud.
 - Los polos Norte y Sur no tienen longitud.



2- EL LOCALIZADOR (GPS)

- Altura, es la distancia vertical de un punto dado, considerando como cero el nivel del mar.



2- EL LOCALIZADOR

- Otra forma de obtener la localización es usando las antenas de los móviles.
- Se obtienen las señales de radio entre varias torres de internet y telefonía. Con estas señales se triangula y se obtiene nuestra posición.
- No necesitamos realizar una llamada para obtener esta posición.

2- EL LOCALIZADOR

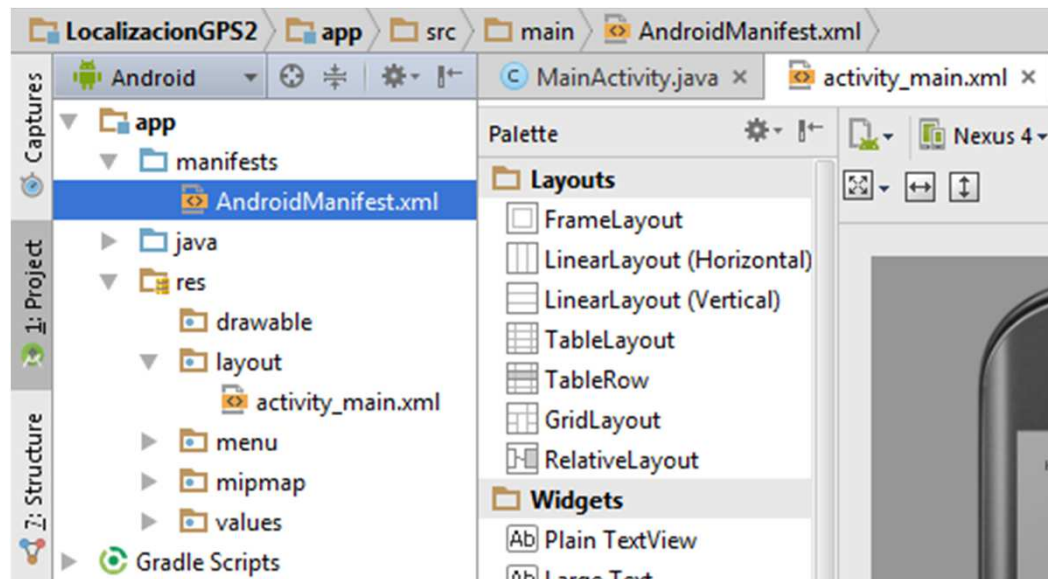
- Puede ser útil en espacios interiores donde el GPS no funciona correctamente o complementando.
- Esta forma de obtener la localización es más rápida que el GPS.
- En Android podemos seleccionar entre estos dos tipos de localización: GPS y network. A esto se le llama seleccionar el proveedor «provider».

2- EL LOCALIZADOR

- Vamos a ver como se utiliza el localizador en Android: el GPS y network.
- Necesitamos realizar los siguientes cinco pasos:
 - Añadir permisos para usar el localizador.
 - Crear un «Manager» para utilizar el localizador.
 - Comprobar que el GPS está activado.
 - Implementar el escuchador («LocationListener»).
 - Enlazar el «Manager» con el escuchador («LocationListener»).

2- EL LOCALIZADOR

- Añadir permisos para usar el localizador. Abrimos el archivo «AndroidManifest.xml» de nuestro proyecto, que está aquí:



2- EL LOCALIZADOR

- Añadimos permisos para poder usar el localizador de nuestro dispositivo. Dentro del archivo «AndroidManifest.xml».

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.isma.localizaciongps2" >

    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".MainActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

2- EL LOCALIZADOR

- Inicializamos el «Manager» para manejar el localizador así:

```
private LocationManager locationManager;
```

- Después dentro de la función «onCreate» creamos el «Manager» así:

```
locationManager = (LocationManager)  
getSystemService(Context.LOCATION_SERVICE);
```

2- EL LOCALIZADOR

- El siguiente paso es implementar el «LocationListener» en nuestra clase «MainActivity». Volvemos a escribir junto a nuestra clase: `implement LocationListener`, así:



- Encima de la clase saldrá un error, pulsamos en implementar los métodos.

2- EL LOCALIZADOR

- Completamos los métodos que han aparecido así:

```
@Override
public void onLocationChanged(Location location) {
    txt_salida.setText("Latitude:" + location.getLatitude() + ", Longitude:" + location.getLongitude()
        + "Altitude:" + location.getAltitude() + "Proveedor:" + location.getProvider());
}
```

```
@Override
public void onProviderDisabled(String provider) {
    Toast.makeText(MainActivity.this, «Localizador disable", Toast.LENGTH_SHORT).show();
}
```

```
@Override
public void onProviderEnabled(String provider) {
    Toast.makeText(MainActivity.this, " Localizador enable", Toast.LENGTH_SHORT).show();
    Log.d("Latitude","enable");
}
```

```
@Override
public void onStatusChanged(String provider, int status, Bundle extras) {
    Toast.makeText(MainActivity.this, " Localizador status", Toast.LENGTH_SHORT).show();
}
```

2- EL LOCALIZADOR

- Finalmente enlazamos el «Manager» con las clases que acabamos de implementar, el escuchador, de esta forma (dentro de la función onCreate()):

- Usando el GPS:

locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 0, 0, this);



- Usando la network:

locationManager.requestLocationUpdates(LocationManager.NETWORK_PROVIDER, 0, 0, this);



2- EL LOCALIZADOR

- Aunque este último paso aparezca subrayado en rojo, el localizador funciona correctamente. Nos dice que deberíamos comprobar los permisos.
- En mi caso el localizador GPS no me daba coordenadas al estar en un interior y he utilizado el localizador network.

EJERCICIO PROPUESTO

- Crear una aplicación para tu dispositivo Android que calcule la distancia en grados (longitud y latitud) entre nuestra posición actual y la puerta del Sol de Madrid.
 - La puerta del Sol tiene una localización de latitud 40.416939 y longitud -3.703507.
 - Utilizar el sensor de orientación (GPS o network)
 - Calcular la distancia lineal en Kilómetros.
 - (Ayuda) Puede usar la función `mi_location.distanceTo(new_location)`, se puede crear un localización así:

```
Location new_location = new Location("");  
new_location.setLatitude(40.416939);  
new_location.setLongitude(-3.703507);
```



3- CÓDIGOS QR

- Uno de los códigos más utilizados en la industria son los **códigos de barras**. El código de barras es un código basado en la representación mediante un conjunto de líneas verticales de distinto grosor y espaciado. Estos códigos de barras contienen información útil.



3- CÓDIGOS QR



3- CÓDIGOS QR

- Otro tipo de códigos son los QR. Un Código QR (Quick Response Code, código de respuesta rápida) es una matriz de puntos bidimensional que sirve para almacenar información útil.
- Fue creado en 1994 por la empresa Japonesa Denso Wave, subsidiaria de Toyota.
- Sigue el estándar internacional ISO/IEC18004 que fue aprobado en junio del 2000.

3- CÓDIGOS QR

- Se caracterizan por tener tres cuadrados que se encuentran en las esquinas y que permite detectar la posición del código lector.
- Permite almacenar bastantes caracteres en su interior.



3- CÓDIGOS QR

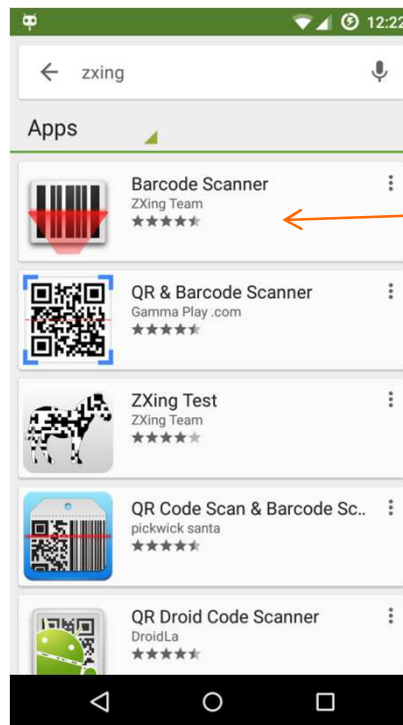
- Capacidad de los códigos QR:
 - Solo numérico → máximo 7089 caracteres
 - Alfanumérico → máximo 4296 caracteres
 - Binario → máximo 2953 bytes
- Existen diferentes niveles de seguridad que permiten restaurar menos o más errores, niveles L, M, Q y H, respectivamente.

3- CÓDIGOS QR

- Ahora vamos a ver como podemos obtener el contenido de los códigos QR desde Android. Para ellos tenemos que seguir estos tres pasos:
 - Instalar Zxing en nuestro dispositivo desde el Google Play.
 - Sobrescribir la clase «onActivityResult» para obtener el contenido de los códigos QR.
 - Crear una llamada al lector de códigos y a la clase que hemos sobrescrito.

3- CÓDIGOS QR

- En primer lugar vamos al Google Play y buscamos «ZXing» y lo instalamos:



3- CÓDIGOS QR

- El segundo paso es sobrescribir la clase «onActivityResult» para que reciba el contenido del código QR, de esta forma (dentro de nuestro «MainActivity»):

@Override

```
protected void onActivityResult(int requestCode, int resultCode, Intent data) {  
    super.onActivityResult(requestCode, resultCode, data);  
    if (requestCode == 0) {  
  
        if (resultCode == RESULT_OK) {  
            String contents = data.getStringExtra("SCAN_RESULT");  
            txt_salida.setText("Salida="+contents);  
        }  
        if(resultCode == RESULT_CANCELED){  
            //handle cancel  
        }  
    }  
}
```

3- CÓDIGOS QR

- Finalmente vamos a crear la llamada (Intent) al lector y a la clase que hemos sobrescrito para recibir el resultado. Por ejemplo dentro del escuchador de un botón vamos a hacer esta acción:

```
try {  
    Intent intent = new Intent("com.google.zxing.client.android.SCAN");  
    intent.putExtra("SCAN_MODE", "QR_CODE_MODE");  
  
    startActivityForResult(intent, 0);  
} catch (Exception e) {  
    Uri marketUri = Uri.parse("market://details?id=com.google.zxing.client.android");  
    Intent marketIntent = new Intent(Intent.ACTION_VIEW,marketUri);  
    startActivity(marketIntent);  
}
```

Si no tenemos
instalado el Zxing
llama a la Play
Store para que lo
descarguemos

3- CÓDIGOS QR

- ¿Cómo podemos crear nuestro propios códigos QR?
 - Existen un gran número de aplicaciones de escritorio y online para este propósito.
 - Ejemplo de aplicación de escritorio: Barcode Studio
 - Ejemplo de sitio online para crear códigos QR:

www.qrplanet.com/es/generador-qr-code/

Donde podemos introducir nuestra cadena de caracteres.
Configurar el tamaño del código y el tipo de seguridad o redundancia.

3- CÓDIGOS QR

- Ejemplo de sitio online para crear códigos QR:

Generador QR-Code
Juega y experimenta creando tus códigos

URL	Texto	SMS	E-Mail	MeCard	Teléfono	WIFI (Android)
-----	-------	-----	--------	--------	----------	----------------

Información

Enlace

Algunos ejemplos:

- <http://www.google.com> - Links
- <http://twitter.com/@user> - Twitter account
- <http://fb.me/username> - Facebook profile
- <http://youtu.be/g2NpcYuCvAo> - Video youtube

Parámetros del código

Color

Tamaño

Redundancia

Generar QR-Code

Código

Hoy hemos generado:
¡2307 QR-Codes!



↓ Descargar .png

Create and Manage
your QR Codes
with Unitag's Platform

Tracking, HD,
Web Mobile...

Este generador es **100% gratuito**.
Colabora compartiéndolo o RT please :)

3- CÓDIGOS QR

- Usando el sitio online vamos a crear códigos QR para identificar diferentes circuitos integrados. Para ello debemos crear un patrón donde introduciremos los datos y así después podemos interpretarlo.

fabricante=[fabricante_del_circuito],modelo=[modelo_del_circuito],consumo=[consumo_en_voltios],velocidad=[velocidad_del_circuito],descripcion=[otras_características]

- Ejemplo:

fabricante=Silicon Labs, modelo=EM358x, consumo=2.1-3.6V, velocidad=2.5GHz,descripcion=32-bits ARM Cortex M3 processor...

EJERCICIO PROPUESTO

- Crear una aplicación para tu dispositivo Android para leer códigos QR que reconozca circuitos integrados. Una vez leído el código muestre cada característica del circuito en un «TextView» diferente. Utilizar un botón para leer el códigos.
 - Crear 3 o 4 códigos QR en el sitio web siguiendo el patrón que se ha planteado.
 - Podemos dividir String así:

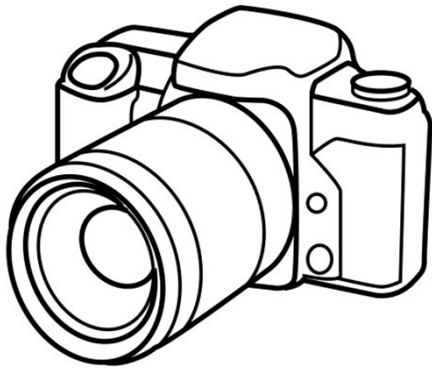
```
String [] partes = entrada.split(",");  
partes[0];  
partes[1];  
...
```

EJERCICIO PROPUESTO

- Probar con estos códigos QR:



4- LA CÁMARA



4- LA CÁMARA

- La cámara es un sensor que tienen la mayoría de los dispositivos móviles y tabletas.
- La cámara de nuestros dispositivos permite capturar fotos y grabar vídeos.
- Vamos a ver que como se realizan estas acciones en Android.

4- LA CÁMARA

- Para manejar la cámara tenemos que seguir estos tres pasos:
 - Añadir permisos de usuario.
 - Crear un directorio y un archivo donde se guardarán los archivos.
 - Lanzar la cámara y guardar el resultado.

4- LA CÁMARA

- El primer paso es añadir permisos de usuario para manejar cámara y para tener acceso de escritura a la memoria de nuestro dispositivo.
- Abrimos de nuevo el archivo «AndroidManifest.xml» de nuestro proyecto.
Añadimos:

```
<uses-permission android:name="android.permission.CAMERA"/>  
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
```

4- LA CÁMARA

- Ahora necesitamos una ruta donde guardar nuestro archivos. Vamos a utilizar la carpeta «Pictures» o «Movies» de nuestro dispositivo. Creando una carpeta dentro «misImagenes» o «misVideos». De esta forma (se guardará en sdcard/pictures/misFotos/):

```
private final String ruta_fotos =  
Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_PICTURES) +  
"/misfotos/";
```

- También vamos a crear un archivo dentro de este directorio:

```
private File file = new File(ruta_fotos);
```

4- LA CÁMARA

- Finalmente dentro del escuchador de un botón vamos a lanzar la cámara y guardar el resultado, así:

```
String file = ruta_fotos + counter + ".jpg";  
// String file = ruta_fotos + counter + ".avi";  
counter++;
```

Elegimos el formato de guardar Imagen → jpg.
Videos → avi

```
File mi_foto = new File( file );  
try {  
    mi_foto.createNewFile();  
} catch (IOException ex) {  
    Log.e("ERROR ", "Error:" + ex);  
}  
//  
Uri uri = Uri.fromFile(mi_foto);
```

Elegimos tomar fotos

```
//Abre la camara para tomar la foto  
Intent cameraIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);  
//Intent cameraIntent = new Intent(MediaStore.ACTION_VIDEO_CAPTURE);
```

```
//Guarda imagen  
cameraIntent.putExtra(MediaStore.EXTRA_OUTPUT, uri);
```

```
//Retorna a la actividad  
startActivityForResult(cameraIntent, 0);
```

También se pueden grabar vídeos

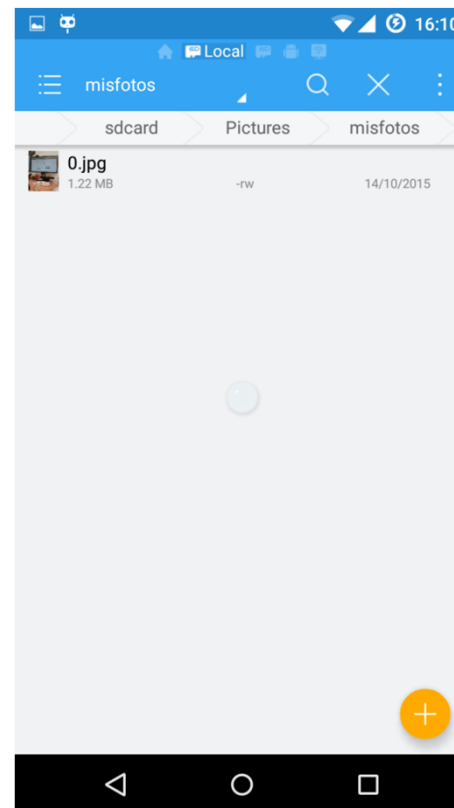
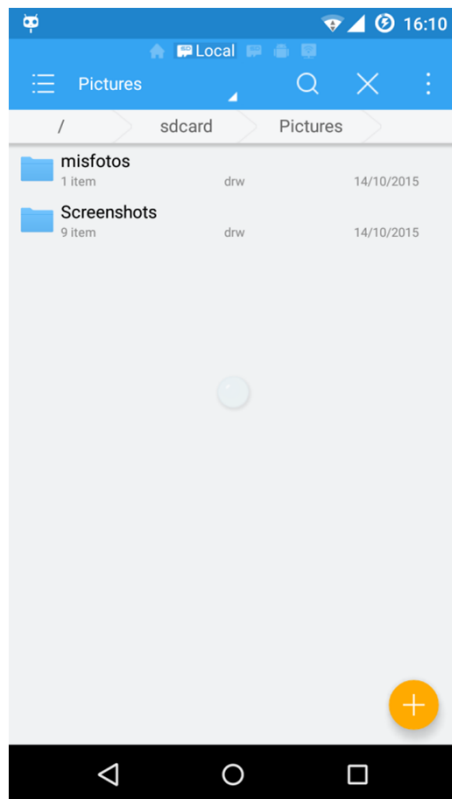
4- LA CÁMARA

- Si no existen los directorios donde vamos a guardar las imágenes o vídeos el programa fallará al no encontrar la ruta para guardar.
- para solucionar este problema, simplemente crearemos estos directorios de esta forma (dentro de la función onCreate()):

file.mkdir();

4- LA CÁMARA

- Una vez tomada la imagen si vamos a nuestro dispositivo /sdcard/ veremos:



EJERCICIO

- Crear un aplicación para nuestro dispositivo Android que nos permita tomar fotos y grabar vídeos. Usar un botón para cada acción.
 - Guardar las imágenes en /sdcard/pictures/misImagenes/
 - Guardar los vídeos en /sdcard/movies/misVideos/

5- LA GALERÍA DE IMÁGENES



5- LA GALERÍA DE IMÁGENES

- Una vez hemos tomado una imagen estas son almacenadas en la galería de imágenes de nuestro dispositivo.
- En la galería se almacenan todas las imágenes que tenemos en nuestro dispositivo.
- Android permite manejar esta galería y leer las imágenes.
- Android también permite visualizar imágenes usando el componente «ImageView».

5- LA GALERÍA DE IMÁGENES

- Para manejar la galería y mostrar la imagen seleccionada debemos seguir estos tres pasos:
 - Añadir permisos de usuario.
 - Lanzar la galería.
 - Recoger la selección de la galería y mostrar el resultado.

5- LA GALERÍA DE IMÁGENES

- El primer paso es añadir permisos de usuario para manejar cámara y para tener acceso de escritura a la memoria de nuestro dispositivo.
- Abrimos de nuevo el archivo «AndroidManifest.xml» de nuestro proyecto.
Añadimos:

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
```

5- LA GALERÍA DE IMÁGENES

- El siguiente paso es abrir la galería, por ejemplo usando el escuchador de un botón, lanzamos un «Intent» para abrir la galería, de esta forma:

```
Intent i = new Intent(Intent.ACTION_PICK, android.provider.MediaStore.Images.Media.EXTERNAL_CONTENT_URI);  
startActivityForResult(i, RESULT_LOAD_IMAGE);
```

- La variable RESULT_LOAD_IMAGE simplemente es un número que nos permite diferenciar este «Intent» de otro y así recoger el resultado después. Por defecto vamos a dejarlo a 1.

5- LA GALERÍA DE IMÁGENES

- El último paso es sobrescribir el método «onActivityResult» para que responda al «Intent» que hemos mandado antes de forma correcta.
- Finalmente coger la imagen que ha seleccionado el usuario y la mostrarla con un componente «ImageView».
- Quedaría así:

5- LA GALERÍA DE IMÁGENES


```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);

    if (requestCode == RESULT_LOAD_IMAGE && resultCode == RESULT_OK && null != data) {
        Uri selectedImage = data.getData();
        String[] filePathColumn = { MediaStore.Images.Media.DATA };

        Cursor cursor = getContentResolver().query(selectedImage, filePathColumn, null, null, null);
        cursor.moveToFirst();

        int columnIndex = cursor.getColumnIndex(filePathColumn[0]);
        String picturePath = cursor.getString(columnIndex);
        cursor.close();

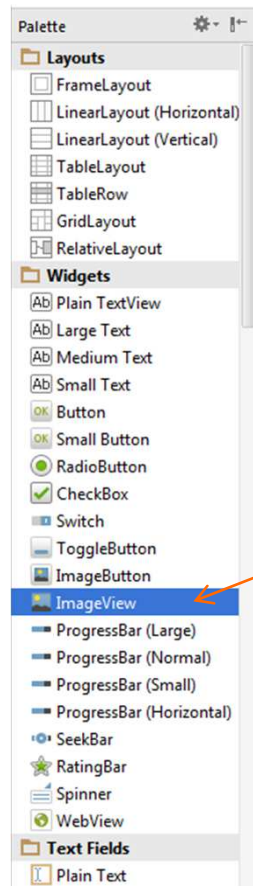
        img_imagen.setImageBitmap(BitmapFactory.decodeFile(picturePath));
    }
}
```



Este es el componente «ImageView»
que vamos a usar para mostrar el
resultado

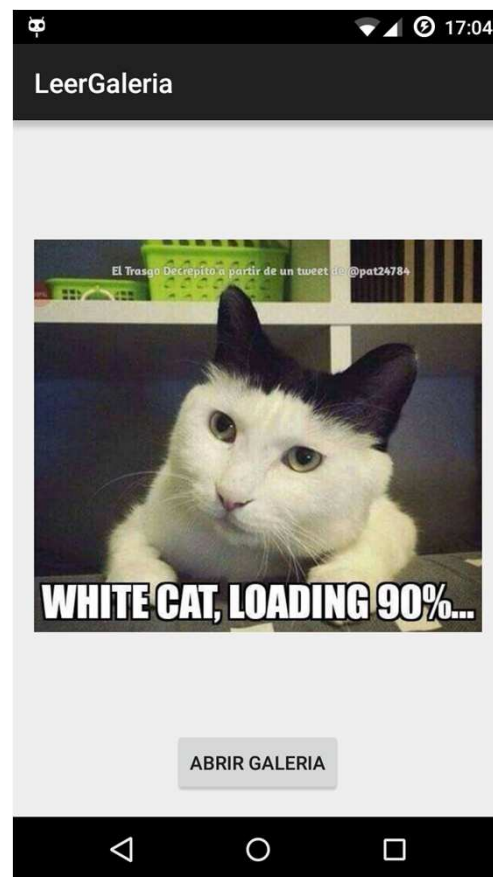
5- LA GALERÍA DE IMÁGENES

- El componente «ImageView» lo encontramos aquí:



5- LA GALERÍA DE IMÁGENES

- Vemos aquí un ejemplo:



EJERCICIO

- Implementar un aplicación para dispositivos Android que nos permita abrir la galería para seleccionar una imagen y después se muestre por pantalla.
 - Utilizar un botón para abrir la galería.
 - Utilizar un «ImageView» para mostrar la imagen seleccionada.

6- ENVIAR EMAILS



6- ENVIAR EMAILS

- Los correo electrónicos (o e-mails es inglés), es un servicio de red que permite enviar y recibir mensajes mediante sistemas de comunicación electrónica.
- Es un servicio de Internet que usa el protocolo SMTP.
- Desde Android también podemos enviar Emails a un correo determinado con un asunto y un contenido.

6- ENVIAR EMAILS

- Para enviar Emails desde Android tenemos que seguir estos dos pasos:
 - Creamos tres «EditText» para la entrada del correo, asunto y el mensaje a enviar. Además de un botón para enviar.
 - Creamos un «Intent» para enviar el Email.

6- ENVIAR EMAILS

- Creamos tres «EditText» y para la entrada del correo, asunto y el mensaje.
- Después creamos el botón «Enviar» y dentro del escuchador vamos a enviar el Email.

6- ENVIAR EMAILS

- Finalmente vamos a crear un «Intent» para enviar el correo añadiendo el correo, asunto y contenido del correo como extras del «Intent», de esta forma (dentro del escuchador del botón):

```
//es necesario un intent para mandar emails
Intent itSend = new Intent(android.content.Intent.ACTION_SEND);

//vamos a enviar texto plano
itSend.setType("plain/text");

//colocamos los datos para el envío
itSend.putExtra(android.content.Intent.EXTRA_EMAIL, new String[]{
edt_correo.getText().toString()});
itSend.putExtra(android.content.Intent.EXTRA_SUBJECT,
edt_asunto.getText().toString());
itSend.putExtra(android.content.Intent.EXTRA_TEXT, edt_mensaje.getText());

//iniciamos la actividad
startActivity(itSend);
```

6- ENVIAR EMAILS

- Cuando enviemos el Email nos pedirá que el usuario seleccione un gestor de correos.
- Vamos a ver un ejemplo de la interfaz:



EJERCICIO

- Crear un aplicación Android (parecida a la imagen anterior) que nos permita mandarnos recordatorios a nuestro propio correo. El usuario deberá introducir solo el asunto. El correo que llegue debe tener una estructura así:
 - Asunto: "Recordatorio: " + asunto
 - No es necesario que tenga texto.
- Usar un botón para enviar.