Sensor de temperatura ambiental

Introducción:

La plataforma Android proporciona cuatro sensores que le permiten controlar diversas propiedades ambientales. Podemos utilizar estos sensores para controlar la humedad relativa del ambiente, iluminación, la presión ambiental y la temperatura ambiente.

En este caso me centraré en desarrollar el sensor de temperatura ambiental, pues es el objetivo de este pequeño proyecto.

Se trata de un proceso muy sencillo y sin demasiadas complicaciones, dado que no requiere de permisos ni de a penas comprobaciones para poder comenzar a usar este sensor.

Cabe mencionar que el valor que obtendremos al recoger los datos estará determinado en grados Celsius, lo cual es importante a la hora de realizar cálculos con diferentes unidades termométricas.

¿Cómo usarlo?

Para comenzar implementaremos la clase *“SensorEventListener”*, lo que nos obligará a hacer uso de los métodos *“onSensorChanged”* y *“onAccuracyChanged”,* los cuales explicaré más adelante.

A continuación necesitaremos crear dos objetos, los cuales es probable que sean utilizados en varias partes de nuestra aplicación, por lo que recomiendo crearlos como atributos:

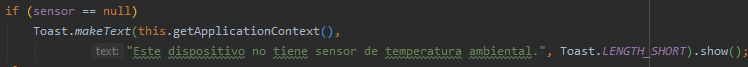


Tras esto deberemos inicializar ambos objetos:



Como se puede ver en el código anterior, primero deberemos inicializar el *“SensorManager”,* de esta manera podremos inicializar el *“Sensor”* a partir de este.

Dado que es posible que el dispositivo que se esté usando no cuente con el sensor de temperatura ambiental y con el objetivo de prevenir errores futuros, realizaremos la siguiente comprobación:

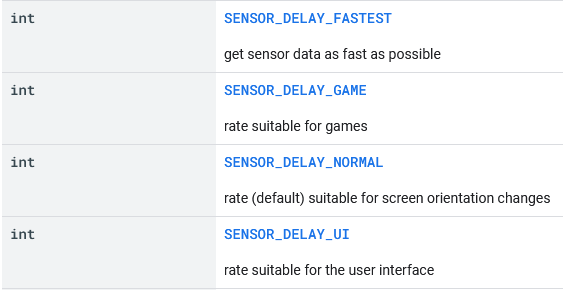


En caso contrario, es decir, en caso de que el dispositivo cuente con dicho sensor, deberemos “suscribirnos” a sus cambios de la siguiente forma:



El tercer parámetro que debemos introducir hace referencia a la resolución, o por decirlo de otra manera la velocidad con la que queremos hacer uso del sensor.

En este caso utilizaremos *“SENSOR\_DELAY\_NORMAL”,* pero existen otras opciones que pueden ajustarse más a nuestras necesidades en función de la tasa de refresco que busquemos obtener, en cuyo caso puedes consultar la siguiente tabla:



Con esto tendremos hecho todo lo necesario para comenzar a implementar el método *“onAccuracyChanged”,* el cual nos devolverá el grado de precisión que tiene nuestro sensor en un determinado momento.

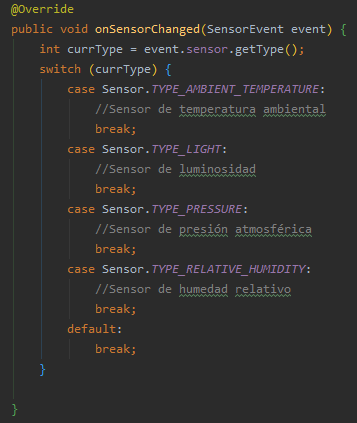
En nuestro caso esto carece de demasiada utilidad, dado que este sensor en concreto no posee variaciones de precisión como puede ser en el caso del GPS, por ejemplo, pero si aún así queremos hacer uso de este método, aquí dejo un ejemplo de cómo debería utilizarse:



Como podemos observar es un proceso muy sencillo y haciendo uso de los atributos estáticos de la clase *“SensorManager”* prácticamente no tenemos memorizar nada.

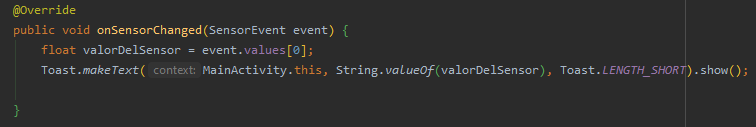
Ahora implementaremos el método *“onSensorChanged”,* el cual posee información mucho más interesante.

En caso de que hayamos registrado más de un sensor será necesario, antes de nada, crear un switch que diferencie entre los eventos producidos por cada sensor de la siguiente forma:



Como se puede observar en el ejemplo se han incluido otro tipo de sensores, en este caso del grupo de sensores ambientales, pero como he mencionado anteriormente, esto no será necesario si sólo manejamos un sensor.

Una vez hecho esto, si queremos recuperar el valor actual de dicho sensor deberemos hacer lo siguiente:



De esta forma tendremos el valor devuelto por el sensor almacenado en una variable de tipo *“float”* y podremos tratar dicha información de la manera que nos plazca, aunque en este caso simplemente se muestra el valor por pantalla.

Por último cabe mencionar que el uso de sensores de manera continua supone un gasto de recursos elevado y se debe evitar en la medida de lo posible, por lo que una vez hayamos recogido los datos necesarios es recomendable cambiar el valor de *“sensor”* a nulo y “desuscribirse” del *“sensorManager”* de la siguiente forma:



También podemos reducir el consumo de recursos cuando la aplicación esté en pausa mediante el uso del método *“onPause”.*

