**Date Acquisition and** **Virtualization**

Any sensors, both physical and virtual, can be supported. To acquire sensor data, the Android platform is used in the prototype and a mobile client application was developed.

Most Android platforms have various sensors built in and the Android sensor framework allows the user to access sensors available on the device and acquire raw sensor data, however sensor availability differs between device types, device manufacturers and Android platform versions.

**Sensors**

The prototype Android mobile client application is currently reading four sensors.

**Light sensor:**

The light sensor measures the ambient light level in lux. The light sensor is supported by most Android platform versions. Sensor values are detected automatically when values have changed.

[reference No <http://developer.android.com/guide/topics/sensors/sensors_overview.html> ]

**Proximity sensor:**

The proximity sensor measures how close something is to the phone in centimetres. The proximity sensor is supported by most Android platform versions. This can be used to determine the phone position during a call. However, some proximity sensors only support a binary *near* or *far* measurement. Its maximum range value indicates the far state and its minimum range values indicates the near state. Sensor values are detected automatically when values have changed.

[reference No <http://developer.android.com/guide/topics/sensors/sensors_overview.html>]

**Location sensor:**

A location sensor detects the user's current location in geographic latitude and longitude.

Locations are periodically updated. The Google Location APIs are used to track the current location. The Google Location APIs combines multiple location technologies into an easy to use API which allows the developer to specify accuracy and power requirements, it also transparently changes location provider. Currently the location sensor uses all location providers to get the best, most recent location and updates every 5 seconds.

**Noise sensor:**

A noise sensor measures noise level in decibel. The noise sensor is not supported by the Android platform by default. To detect noise level, sounds are captured from the device's microphone and are recorded continuously. Then maximum aptitude is checked every interval and converted into decibels.

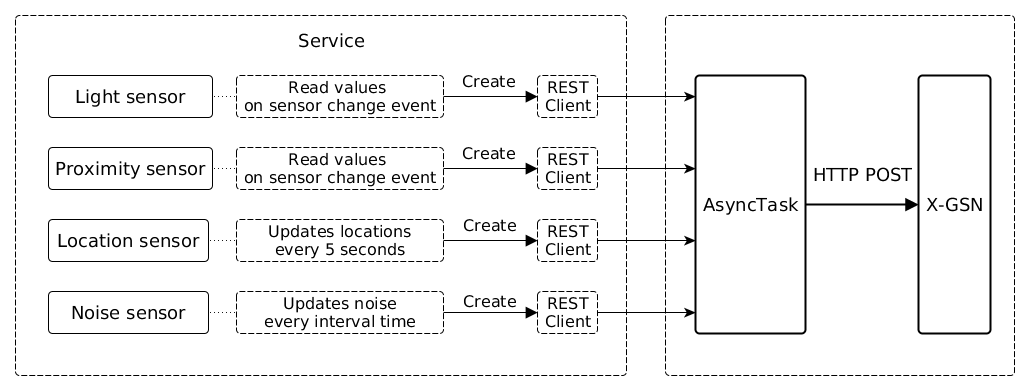
**Android mobile client application**

A prototype Android mobile client application as shown in Figure is being developed for reading sensor data and sending it to the OpenIoT platform. Four sensors are registered to the OpenIoT so each sensor has a unique sensorID provided by the OpenIoT . When clicking any of sensors, it starts reading sensor data and sending it to the HTTP-REST Listener wrapper running on X-GSN and it stops sending when unchecked.

|  |  |
| --- | --- |
|  |  |

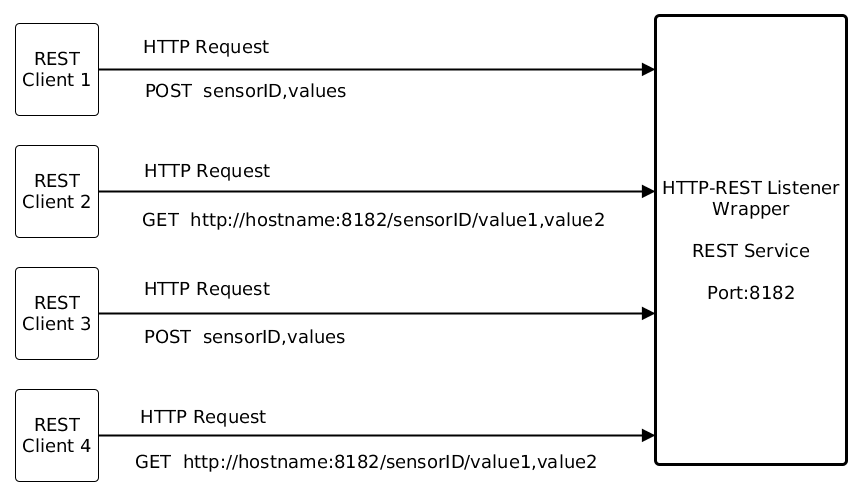
[Figure No] Mobile client application

Sensor values are read in the background using a *service* that is an application component that can perform long-running operations in the background. After reading a value, a rest client is created and the value is sent to X-GSN in the background using *AsyncTask*.

*[Figure No] *

**Communication method:**

A REST service is running in the HTTP-REST Listener wrapper in X-GSN. Any clients can send raw sensor data with HTTP POST or GET request.

*[Figure No]*

In the prototype Android mobile client application, sensorIDs and values are sent with HTTP POST request.

