Android Project Requirements

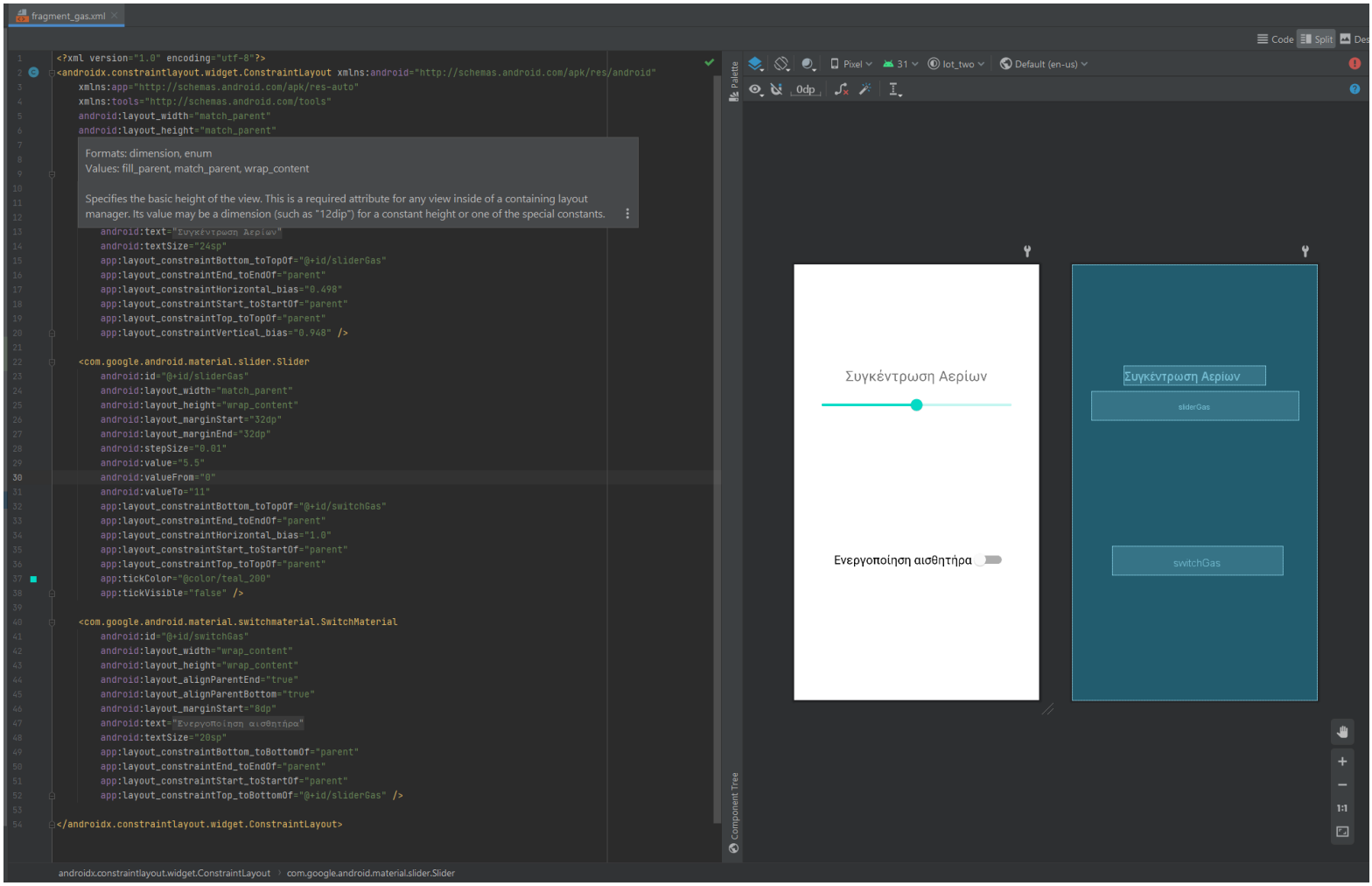
Σκοπός των δύο (2) συσκευών Android είναι να προσομοιώσουν τέσσερις (4) αισθητήρες:

- Αερίου
- Καπνού
- Θερμοκρασίας
- UV ακτινοβολίας

και να αποστείλουν τις τιμές τους, συμπεριλαμβάνοντας στοιχεία των συντεταγμένων τους και το ποσοστό της μπαταρίας τους, στον Mosquitto MQTT Broker που χρησιμοποιείται σε αυτήν την προσέγγιση.

Αισθητήρες – Fragments

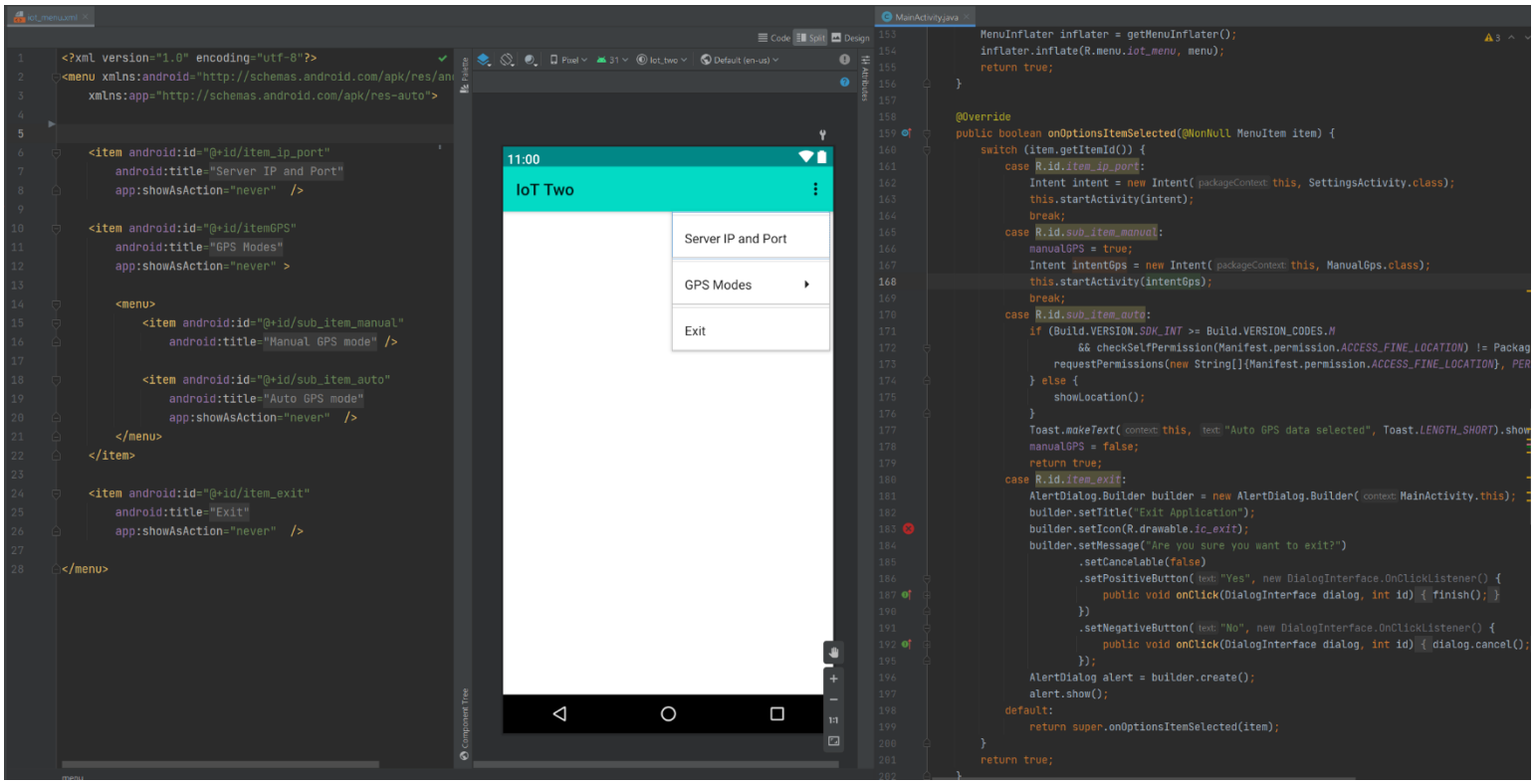
Αρχικά δημιουργούνται τα τέσσερα (4) διαφορετικά Fragments στο καθένα από τα οποία θα απεικονισθούν οι αισθητήρες και το slider που θα προσομοιώνει τις τιμές τους. Η ενεργοποίηση/απενεργοποίηση των αισθητήρων επιτυγχάνεται με ένα Switch Button.



Και στη συνέχεια θα ομαδοποιηθούν σε ένα TabLayout.

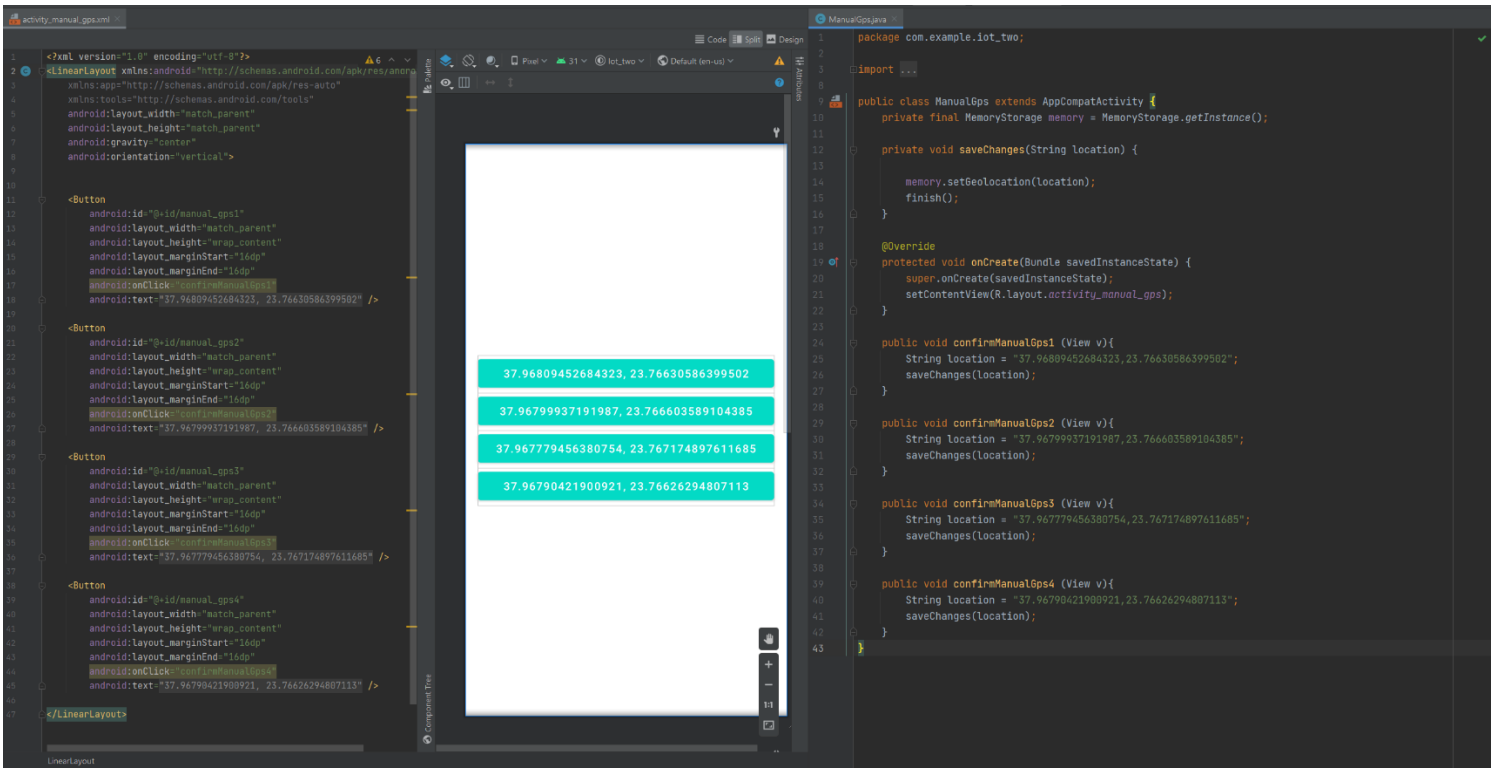
Menu – Ρυθμίσεις

Απαραίτητη υλοποίηση για τη λειτουργικότητα της εφαρμογής αποτελεί το Menu, το οποίο δίνει την δυνατότητα επιλογής Διεύθυνσης και Θύρας του MQTT Broker (IP Address and Port), επιλογή τρόπου διαχείρισης της τοποθεσίας (επιλογή προεπιλεγμένων θέσεων τοποθέτησης των αισθητήρων ή Αυτόματα, λαμβάνοντας στοιχεία από των αισθητήρα GPS της συσκευής) και τέλος την δυνατότητα εξόδου από την εφαρμογή, μετά από επιβεβαίωση.



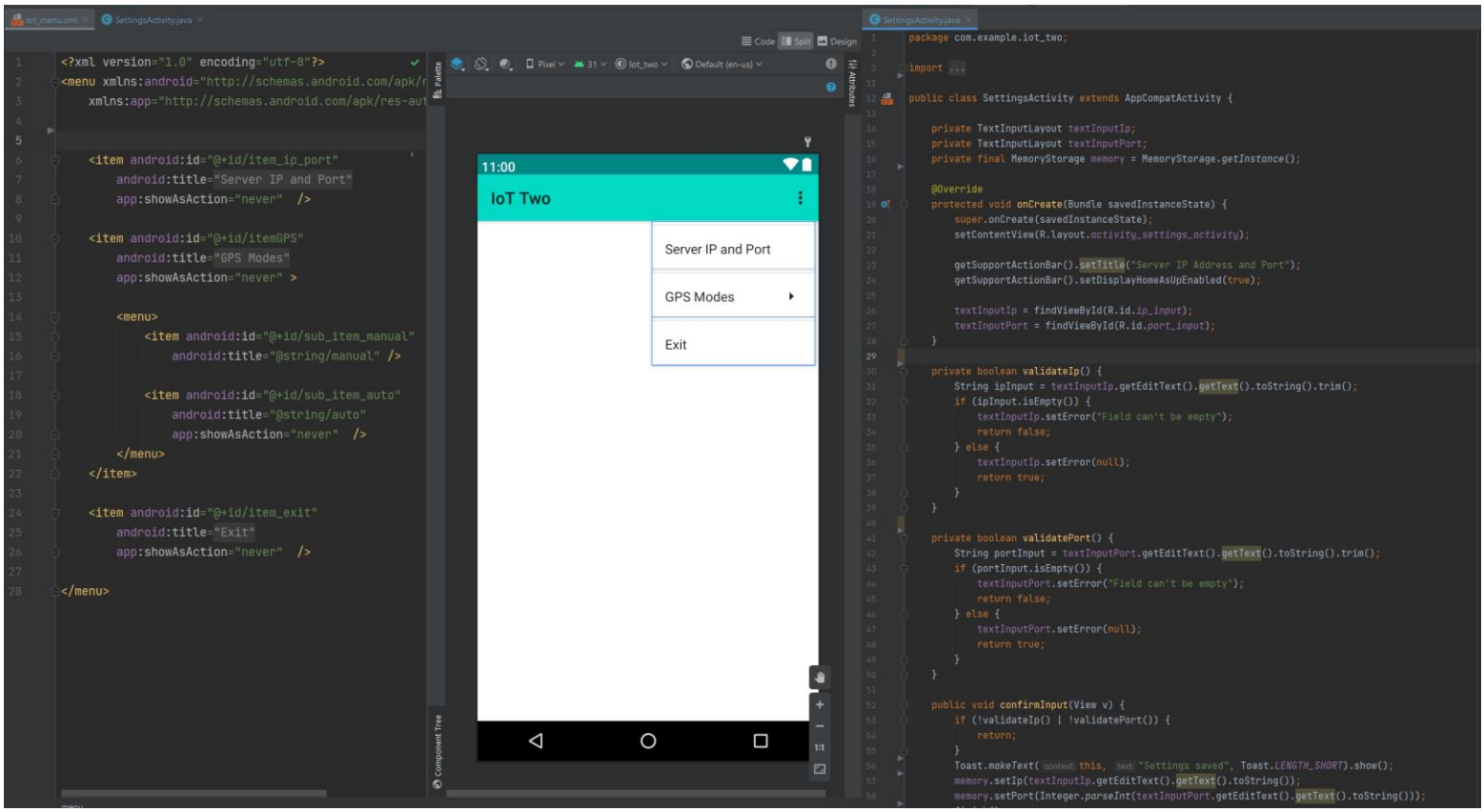
Manual GPS

Η επιλογή των προεπιλεγμένων θέσεων επιτυγχάνεται επιλέγοντας το αντίστοιχο Button.



IP Address - Port

Η εφαρμογή δίνει τη δυνατότητα στον χειριστή να επιλέξει νέα διεύθυνση για τον MQTT Broker στον οποίο θα γίνονται Publish τα μηνύματα MQTT (payload).



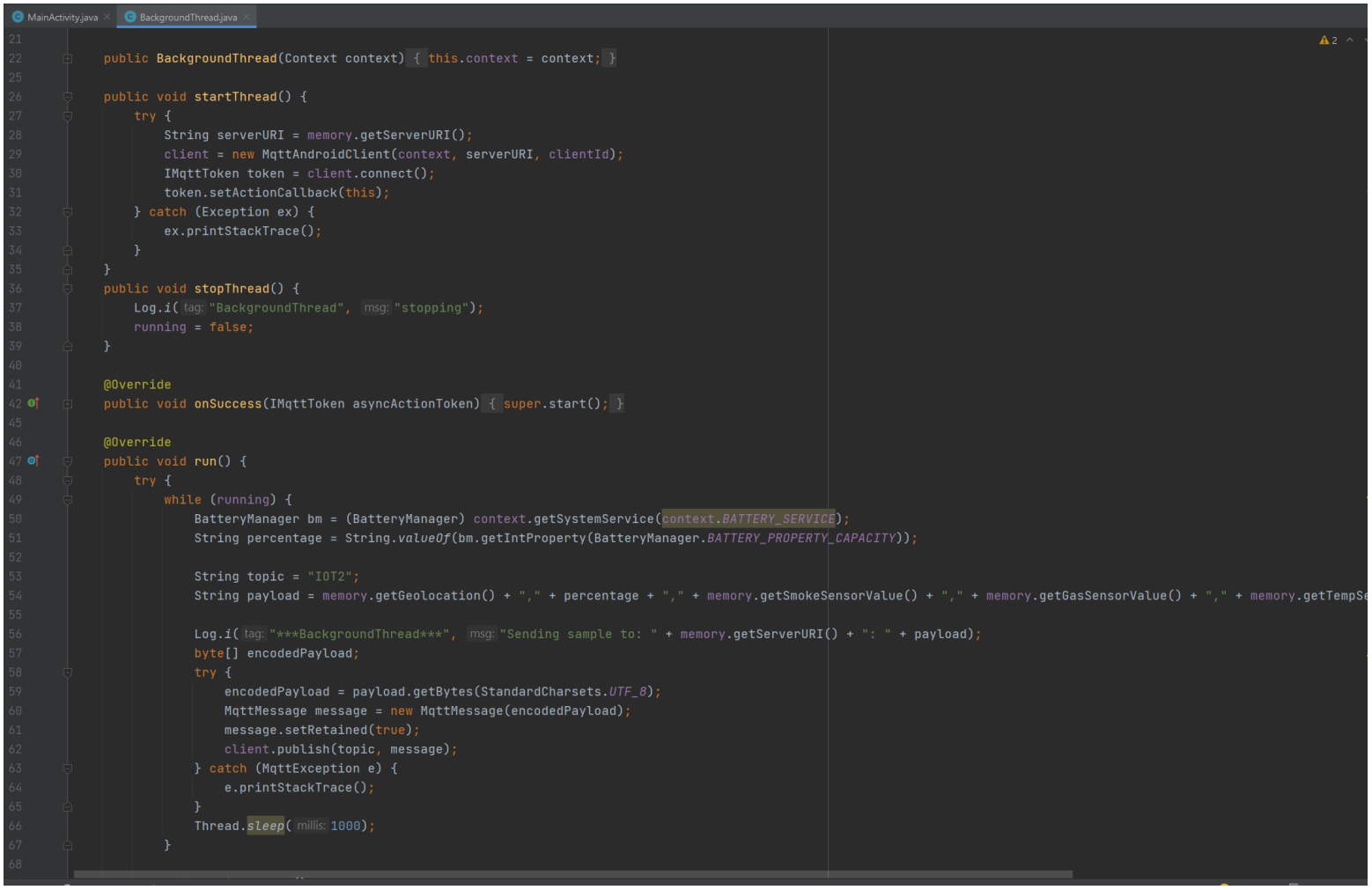
Memory

Για να λειτουργήσει η εφαρμογή ομαλά, δίνοντας τη δυνατότητα στο χρήστη να τροποποιεί όλα αυτά τα στοιχεία σύνδεσης του MQTT καθώς και οι διαφορετικές τιμές του Payload, απαιτείται η δημιουργία μιας μνήμης, ανεξάρτητης από το Lifecycle του Android. Εκεί αποθηκεύονται οι διαφορετικές τιμές των αισθητήρων, οι τιμές των συντεταγμένων (μέσα από και τις δύο διαδικασίες) καθώς και η Διεύθυνση και θύρα του MQTT Broker.

```
public synchronized String getServerURI() { return "tcp://" + ip + ":" + port; }

public synchronized String getGeolocation() { return geolocation; }

public synchronized void setGeolocation(String geolocation) { this.geolocation = geolocation; }

public synchronized String getSmokeSensorValue() { return smokeSensorValue; }

public synchronized void setSmokeSensorValue(String smokeSensorValue) {
    this.smokeSensorValue = smokeSensorValue;
}

public synchronized String getGasSensorValue() { return gasSensorValue; }

public synchronized void setGasSensorValue(String gasSensorValue) {
    this.gasSensorValue = gasSensorValue;
```

Android MQTT Client

Η λειτουργία του MQTT Client επιτυγχάνεται με τη δημιουργία ενός Thread, το οποίο διαβάζει όλα τα στοιχεία από τη μνήμη που δημιουργήσαμε, τα ενσωματώνει στα options σύνδεσης και στο payload του MQTT και ενεργοποιεί τον Client με αποστολή του payload ανά ένα δευτερόλεπτο.



```java
public BackgroundThread(Context context) { this.context = context; }

public void startThread() {
    try {
        String serverURI = memory.getServerURI();
        client = new MqttAndroidClient(context, serverURI, clientId);
        IMqttToken token = client.connect();
        token.setActionCallback(this);
    } catch (Exception ex) {
        ex.printStackTrace();
    }
}
public void stopThread() {
    Log.i( tag: "BackgroundThread",  msg: "stopping");
    running = false;
}

@Override
public void onSuccess(IMqttToken asyncActionToken) { super.start(); }

@Override
public void run() {
    try {
        while (running) {
            BatteryManager bm = (BatteryManager) context.getSystemService(context.BATTERY_SERVICE);
            String percentage = String.valueOf(bm.getIntProperty(BatteryManager.BATTERY_PROPERTY_CAPACITY));

            String topic = "IOT2";
            String payload = memory.getGeolocation() + "," + percentage + "," + memory.getSmokeSensorValue() + "," + memory.getGasSensorValue() + "," + memory.getTempSe

            Log.i( tag: "***BackgroundThread***",  msg: "Sending sample to: " + memory.getServerURI() + ": " + payload);
            byte[] encodedPayload;
            try {
                encodedPayload = payload.getBytes(StandardCharsets.UTF_8);
                MqttMessage message = new MqttMessage(encodedPayload);
                message.setRetained(true);
                client.publish(topic, message);
            } catch (MqttException e) {
                e.printStackTrace();
            }
            Thread.sleep( millis: 1000);
        }
```

Κλείσιμο Εφαρμογής

Και τέλος ελέγχουμε με ενδεικτικό μήνυμα ότι ο χρήστης επιθυμεί να κλείσει την εφαρμογή.

```java
@Override
public void onBackPressed() {
    if (backPressedTime + 2000 > System.currentTimeMillis()) {
        backToast.cancel();
        super.onBackPressed();
        return;
    } else {
        backToast = Toast.makeText(getBaseContext(),  text: "Press back again to exit", Toast.LENGTH_SHORT);
        backToast.show();
    }
    backPressedTime = System.currentTimeMillis();
}
```