Εργασία για το μάθημα Ανάπτυξη Λογισμικού για Δίκτυα και Τηλεπικοινωνίες (Δικτυακός Προγραμματισμός) (Κ23-Β)

Μέλη Ομάδας:

ΓΕΩΡΓΑΡΑ ΒΑΣΙΛΙΚΗ 1115201400026
DOMA ANDIA 1115201400230

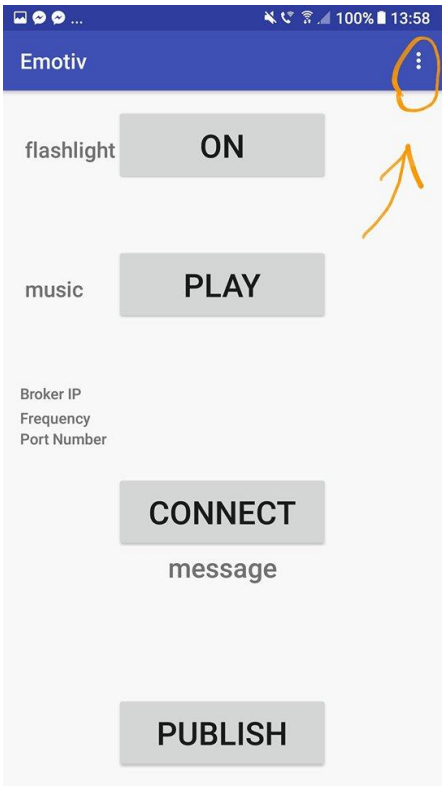
Γενικότερα για την εκτέλεση της εργασίας

Implementation environment: Android Studio (Windows OS).

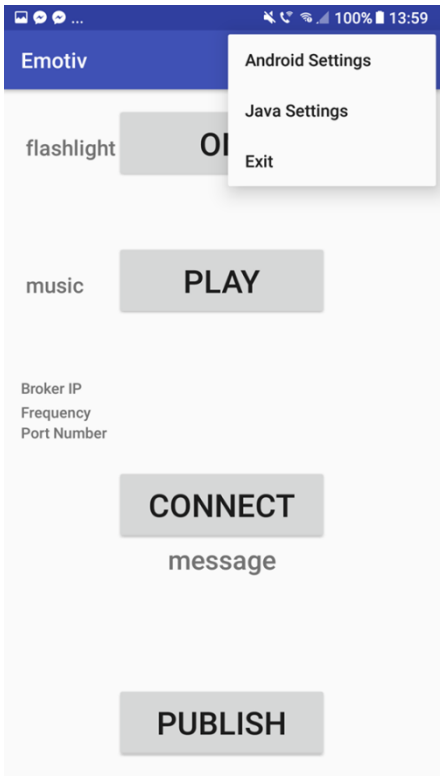
Βήματα για να «τρέξει» η εφαρμογή όπως προβλέπεται:

- Συνδέουμε το κινητό με τον υπολογιστή.
- Ανοίγουμε στο κινητό δεδομένα και mobile hotspot.
- Συνδέουμε το laptop με το wifi του κινητού.
- Στο κινητό, ανοίγουμε την εφαρμογή "Emotiv".
- Στο Menu bar της εφαρμογής εισάγουμε τις παραμετροποιημένες μεταβλητές, δηλαδή εισάγουμε στο Android Settings την broker IP και την port, ενώ στο Java Settings εισάγουμε την επιθυμητή συχνότητα που θα στέλνονται τα δεδομένα.
- Πατάμε το κουμπί CONNECT, ώστε να πραγματοποιηθεί η σύνδεση.
- Έπειτα, τρέχουμε το πρόγραμμα Java.
- Όταν στην κονσόλα της Java εφαρμογής εμφανιστεί το μήνυμα ότι γίνεται subscribe, τότε στην Android εφαρμογή πατάμε το κουμπί PUBLISH.
- Τέλος, σύμφωνα με τη συχνότητα που δώσαμε τα μηνύματα στέλνονται και εμφανίζονται στην οθόνη του κινητού.
- Στην περίπτωση που έρθουν δύο διαδοχικά μηνύματα ίδιας κατηγορίας (2 διαδοχικά Execute Eyes Closed ή 2 διαδοχικά Execute Eyes Opened), εμφανίζεται toast που ενημερώνει τον χρήστη πως η μουσική και ο φακός είναι ήδη κλειστά ή ανοιχτά.

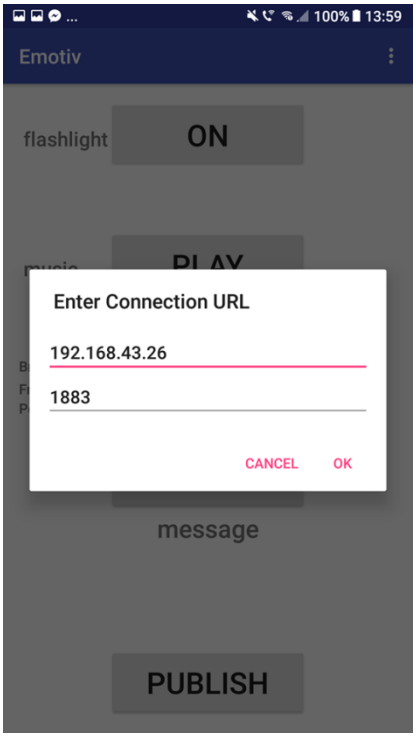
Η εμφάνιση εφαρμογής είναι η εξής, με το βέλος δείχνει το menu bar:



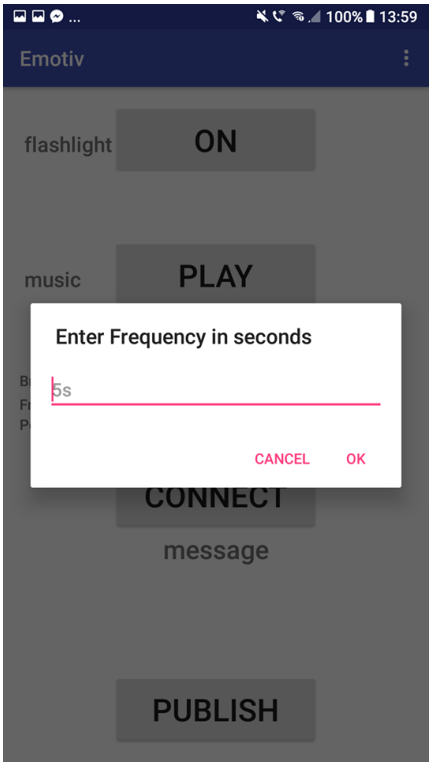
όπου όταν πατηθεί:



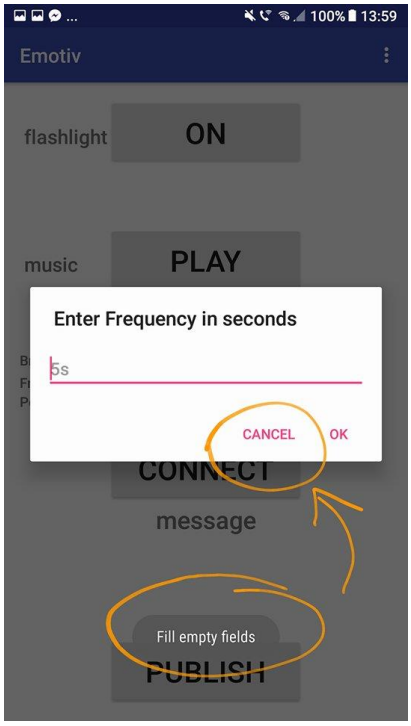
Το πάτημα του Android Settings έχει ως Αποτέλεσμα την εμφάνιση του εξής dialog:



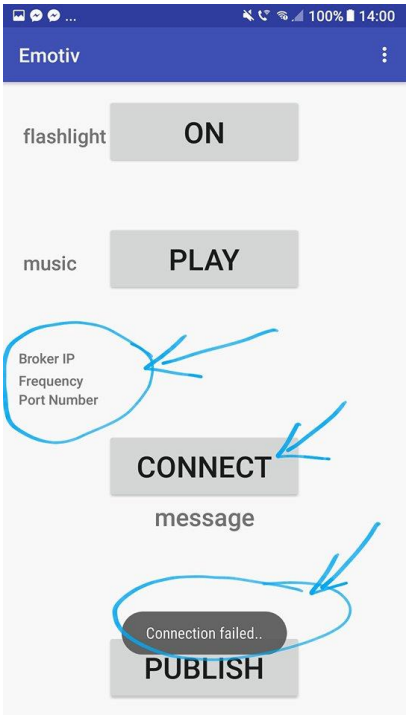
Ενώ το Java Settings:



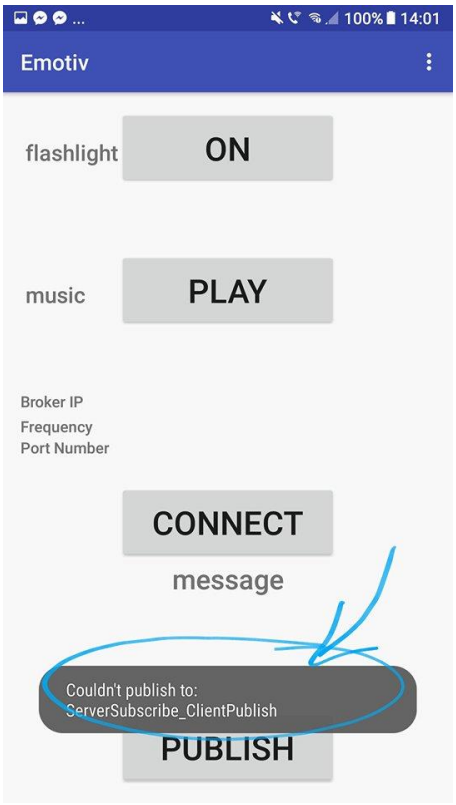
Επιπλέον, έχουμε υλοποιήσει την δυνατότητα να μην γίνεται ο χρήστης προχωρήσει χωρίς να εισάγει τα δεδομένα:



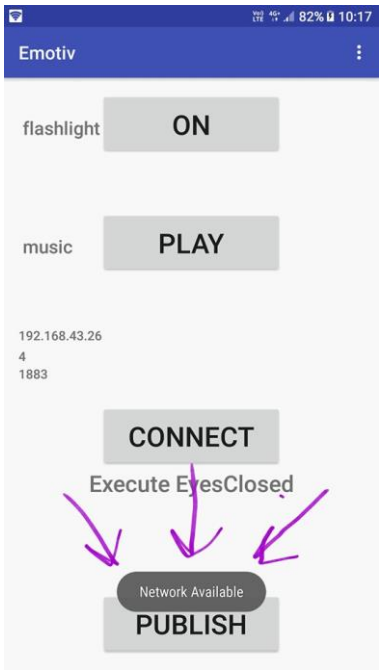
Αν ο χρήστης δεν έχει βάλει σωστά τις παραμέτρους τότε η σύνδεση είναι ανεπιτυχής:



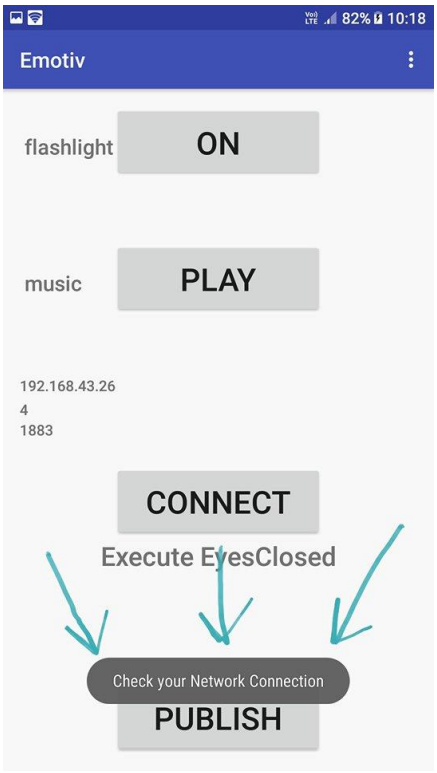
Το ίδιο συμβαίνει και με το κουμπί PUBLISH:



Κάθε 10 δευτερόλεπτα
εμφανίζονται Toast
για σύνδεση στο δίκτυο
Network Available(όταν υπάρχει σύνδεση)



Check your Network Connection
(όταν δεν υπάρχει σύνδεση)



Όσον αφορά το Android πρόγραμμα:

Υλοποιούμε τρία πακέτα μέσα στο κύριο πακέτο «com.example.project.myapplication» καθώς και τη MainActivity:

- πακέτο dialog –
  δύο αρχεία όπου γίνονται οι λειτουργίες που σχετίζονται με το dialog box που ανοίγει για να εισχωρήσουμε τις παραμέτρους της σύνδεσης, ένα για τις παραμέτρους του broker και ένα για τη συχνότητα.

- πακέτο flash –
  κλάση όπου γίνεται υλοποίηση της λειτουργίας του φακού με τις απαραίτητες συναρτήσεις για ενεργοποίηση, απενεργοποίηση κλπ.

- InternetConnectionClass –
  Κλάση όπου υλοποιείται η isConnected(), συνάρτηση που ελέγχει την σύνδεση με το δίκτυο, η οποία καλείται μέσα σε thread στην MainActivity

Έπειτα, προχωράμε στην υλοποίηση της MainActivity:
Περιέχει τις εξής συναρτήσεις:
•onClick() : όπου συνδέει όλα τα κουμπιά με συγκεκριμένες λειτουργίες. Τα κουμπιά του φακού και της μουσικής είναι toggle buttons και με το ON εκτελούν την ενεργοποίηση ενώ με το OFF την απενεργοποίηση. Το κουμπί CONNECT πραγματοποιεί τη σύνδεση και ταυτόχρονα την εγγραφή στο topic του server. Επίσης, υλοποιούνται οι μέθοδοι callback, όπου στην messageArrived() γίνεται η ανάθεση των λειτουργιών, ανάλογα με τον τύπο της εντολής που θα φτάσει από τον server.
•setSubscription() : καλείται μέσα στην onSuccess του CONNECT και κάνει subscribe στο topic που έχει πάρει από τον broker, βγάζοντας το αντίστοιχο toast για την επιτυχία ή την αποτυχία.
•setPublication() : εκτελείται μόλις πατηθεί το PUBLISH και δημοσιεύει στο topic που έχει πάρει από τον σέρβερ τη συχνότητα, βγάζοντας το αντίστοιχο toast για την επιτυχία ή την αποτυχία.
•turnOnMusic()/turnOffMusic() : λειτουργίες μουσικής
•onBackPressed(): double click στο back button για έξοδο από την εφαρμογή. Αναμονή έως 3 δευτερόλεπτα για να πατηθεί το 2ο back ώστε να κάνει exit (exit ο χρήστης μπορεί επίσης να κάνει και με το menu bar).

●onCreateOptionsMenu(): δημιουργία του menu bar.
●onOptionsItemSelected(): αν επιλέξουμε Android Settings ή Java Settingstότε ανοίγει το πακέτο dialog.
●displayData(): ανάλογα με την τιμή που επιστρέφει η isConnected() (true/false) εμφανίζει το κατάλληλο notification μέσω Toast. Η συνάρτηση καλείται σε ένα Thread κάθε 10 δευτερόλεπτα.

πηγές κώδικα για android:
https://www.journaldev.com/9357/android-actionbar-example-tutorial
https://www.youtube.com/watch?v=6X_1WhPcmYU
https://eclipse.org/paho/clients/android
https://www.youtube.com/watch?v=6AE4D8INs_U
https://www.youtube.com/watch?v=qcbpUPAZ2mc&t=334s
https://www.youtube.com/watch?v=BV_KLo-5X10
http://www.viralandroid.com/2016/01/turn-on-and-off-camera-led-flashlight-programmatically.html
https://www.youtube.com/watch?v=Ech5WkaZUVw
https://www.youtube.com/watch?v=ARezg1D9Zd0
https://www.youtube.com/watch?v=ARezg1D9Zd0
https://stackoverflow.com/questions/2620444/how-to-prevent-a-com.example.project.myapplication.dialog-from-closing-when-a-button-is-clicked

Όσον αφορά το Java πρόγραμμα

Implementation environment: IntelliJ IDEA (WINDOWS OS)
Στη Java, η δυνατότητα παραμετροποίησης γίνεται με default ορίσματα, τα οποία αλλάζουν με βάση τα arguments που δίνουμε. Αυτά είναι:

```
JavaApp [-h] [-a publish|subscribe] [-t <topic>] [-m <message text>]\n" +
[-s 0|1|2] -b <hostname|IP address>] [-p <brokerport>] [-i <clientID>]\n\n" +

        -h  Print this help text and quit\n" +

        -q  Quiet mode (default is false)\n" +

        -a  Perform the relevant action (default is publish)\n" +

        -t  Publish/subscribe to <topic> instead of the default\n" +
  (publish: \"JavaApp/Java/v3\", subscribe: \"JavaApp/#\")\n" +

        -m  Use <message text> instead of the default\n" +
  (\"Message from MQTTv3 Java client\")\n" +

        -s  Use this QoS instead of the default (2)\n" +

        -i  Use this client ID instead of SampleJavaV3_<action>\n" +

        -c  Connect to the server with a clean session (default is false)\n" +
```

► Στο πακέτο EntropyPckg:

Αρχικά, περάσαμε όλα τα πειράματα σε έναν φάκελο μέσα στο project, που λέγεται experiments ώστε να διαβάζεται ένα ένα.
Περνάμε τα ορίσματα σε μια λίστα και ύστερα σε 2Δ πίνακα. Στο πίνακα Channels αποθηκεύουμε τις 14 πρώτες στήλες, αφού εκεί βρίσκονται τα κανάλια, χωρίς όμως την πρώτη γραμμή καθώς αυτή αποτελεί τους τίτλους.
Έπειτα, υπολογίζουμε την εντροπία για καθένα από τα 14 κανάλια και για κάθε csv αποθηκεύεται σε 1Δ πίνακα, σύμφωνα με το όρισμα της calculateEntropy.
ύστερα, βάζουμε τις εντροπίες σε έναν vector και μαζί με το όνομα του πειράματος (από το όνομα του αρχείου), αποθηκεύονται όλα σε μια arraylist που έχει πεδία: index, name και vector.
Η λίστα επιστρέφεται ολόκληρη αφού διαβαστούν και τα 264 csv.

► Στο πακέτο kNN:

Περνάμε το TrainingSet σε 2Δ πίνακα και για κάθε κόμβο της λίστας, για κάθε στοιχείο του TRainingSet απομονώνω στον vectorA τις εντροπίες για το κάθε csv(το πεδίο vector της λίστας), και στον vectorB κάθε γραμμή εντροπιών του Trainingset, σώζωντας και τα Labels των αρχείων για το classification αργότερα. Έπειτα, αποθηκεύω τις ευκλείδιες αποστάσεις των vector σε πίνακα, καθώς και τους δείκτες που βρίσκονται οι k μικρότερες αποστάσεις. Οι μετρητές για κάθε Label καθορίζουν την classified Label κάθε πειράματος, μέσα απο τη συνάρτηση classify που λαμβάνει υπόψη και τα βάρη. Το αρχείο αυτό επιστρέφει μια arraylist που έχει κόμβους με 2 συμβολοσειρές: την classifiedLabel και την actualLabel. Τέλος, αποθηκεύει και τα αποτελέσματα και σε ένα txt αρχείο για προσωπικό debug αργότερα.

► Στο πακέτο ProducerConsumerPckg:

Η συνάρτηση ProducerConsumerSolution δέχεται σαν όρισμα την λίστα που έχει επιστραφεί με τα classified και actual Labels. Δημιουργεί έναν buffer χωρητικότητας 10, και χρησιμοποιώντας 2 νήματα, διαχειρίζεται το πρόβλημα παραγωγού και καταναλωτή με συγχρονισμό των 2. Όταν ένα αντικείμενο μπορεί να καταναλωθεί, τότε καλείται και η συνάρτηση publish η οποία στέλνει τα strings "Execute Eyes ******" στο android τερματικό.

► Όλα τα παραπάνω πακέτα βρίσκονται μέσα στο κύριο πακέτο JavaApp μαζί με τη main, Subscriber και Publisher κλάσεις. Η main διευθετεί τα ορίσματα, για αρχή, και έπειτα κάνει πρώτα Subscribe και μετά Publish. Μέσα στην κλάση της εγγραφής γίνονται και όλες οι λειτουργίες των προηγούμενων πακέτων, δηλαδή υπολογίζονται οι εντροπίες των 264 csv, περνάνε από τον Knn Algorithm και μετά επιστρέφονται οι γείτονες, όπου δημοσιεύονται με τη διαδικασία του παραγωγού και καταναλωτή.

Συγκεκριμένα, τα παρακάτω screenshots δείχνουν τι εμφανίζεται στο terminal της Java, εξηγώντας κάθε βήμα αναλυτικά.
Μόλις τρέξει το πρόγραμμα εμφανίζεται η επιβεβαίωση της σύνδεσης αλλά και της εγγραφής από την πλευρά της Java.

```
Connected to tcp://localhost:1883 with client ID SampleJavaV3_publish
Subscribing to topic "ServerSubscribe_ClientPublish" qos 0
```

Τότε, μπορούμε να κάνουμε PUBLISH τη συχνότητα από τη μεριά του Android και παίρνω ως παράδειγμα εισόδου συχνότητας 7 δευτερόλεπτα. Εκτυπώνονται όλα τα στατιστικά στοιχεία στην οθόνη και στο τέλος αποσυνδέεται ο client.

```
Connected to tcp://localhost:1883 with client ID SampleJavaV3_publish
Subscribing to topic "ServerSubscribe_ClientPublish" qos 0
Time:   2018-02-12 12:59:54.978  Topic: ServerSubscribe_ClientPublish  Message: 7  QoS: 0
Frequency given from the Android terminal is 7
Disconnected
```

Στη συνέχεια, υπολογίζονται οι εντροπίες και περνάνε μέσα από τον αλγόριθμο. Τα αποτελέσματα τα αποθηκεύουμε σε μια arraylist, καθώς και σε ένα αρχείο txt για debug. Το txt ονομάζεται out.txt.

```
Frequency given from the Android terminal is 7
Disconnected
Entropies were calculated successfully!
kNN Algorithm done! Results extracted to an arraylist and a txt file!
```

Έπειτα, εμφανίζουμε και στην οθόνη το ποσοστό ακρίβειας του αλγορίθμου μας για επιλογή του κ = 11, το οποίο είναι:

```
Accuracy Percentage = 74.24%
```

Και για τέλος, παρουσιάζουμε το πακέτο με τον παραγωγό και τον καταναλωτή. Έχουμε ορίσει τον buffer μας να έχει χωρητικότητα 10. Οι εικόνες που ακολουθούν δείχνουν ορισμένα παραδείγματα.

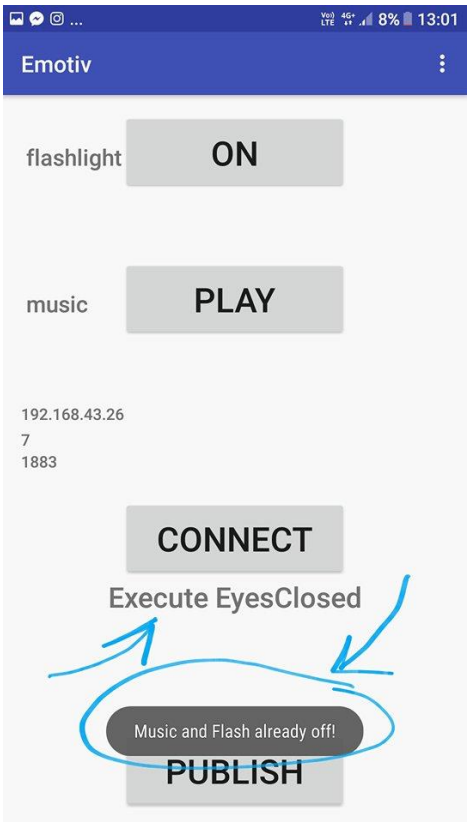
```
Produced: Execute EyesClosed
Queue is empty Consumer is waiting , size: 0
Produced: Execute EyesOpened
Produced: Execute EyesClosed
//////////////////////////
Connecting to tcp://localhost:1883 with client ID SampleJavaV3_publish
Connected
Publishing at: 2018-02-12 13:00:28.706 to topic "ServerPublish_ClientSubscribe" qos 0
 with the message: Execute EyesClosed

Disconnected
//////////////////////////
```

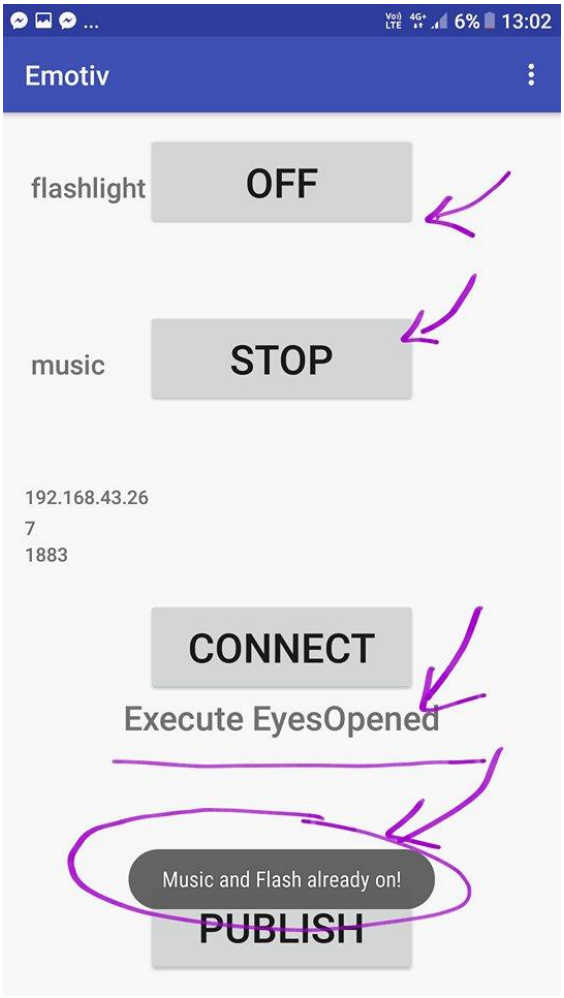
```
Consumed: Execute EyesClosed
Produced: Execute EyesOpened
Produced: Execute EyesClosed
Produced: Execute EyesOpened
Produced: Execute EyesOpened
Produced: Execute EyesOpened
Produced: Execute EyesClosed
Produced: Execute EyesClosed
Produced: Execute EyesClosed
Produced: Execute EyesOpened
Queue is full Producer is waiting , size: 10
//////////////////////////
Connecting to tcp://localhost:1883 with client ID SampleJavaV3_publish
Connected
Publishing at: 2018-02-12 13:00:35.784 to topic "ServerPublish_ClientSubscribe" qos 0
 with the message: Execute EyesOpened

Disconnected
//////////////////////////
```

Με την παραδοχή πως το πρώτο μήνυμα που θα έρθει είναι Eyes Closed και δεν έχουμε ανοίξει προηγουμένως τον φακό και τη μουσική, τότε στο Android εμφανίζεται το συγκεκριμένο toast.



Και η αντίστοιχη περίπτωση όπου έρχονται διαδοχικά Eyes Opened:



πηγές που χρησιμοποιήσαμε για την υλοποίηση:

♦ για τη main
https://github.com/eclipse/paho.mqtt.java/blob/master/org.eclipse.paho.sample.mqttv3app/src/main/java/org/eclipse/paho/sample/mqttv3app/Sample.java
για το CSVReader
https://stackoverflow.com/questions/5932604/return-number-of-rows-in-2d-array
https://beginnersbook.com/2013/12/vector-in-java/
https://stackoverflow.com/questions/17729228/java-read-many-txt-files-on-a-folder-and-process-them
https://beginnersbook.com/2013/12/hashmap-in-java-with-example/
♦ για τον αλγόριθμο kNN
http://afewguyscoding.com/2010/05/nearest-neighbors-java/
https://www.programcreek.com/2013/01/use-k-nearest-neighbors-knn-classifier-in-java/
http://www.d.umn.edu/~deoka001/InstanceWKNN.html
https://github.com/alexksikes/ML/blob/master/knn/kNN.java
https://www.mkyong.com/java/how-to-read-and-parse-csv-file-in-java/
https://github.com/deepmehtait/KNN-algo/blob/master/KnnAlgo.java
https://github.com/deepmehtait/KNN-algo
https://stackoverflow.com/questions/19602601/create-an-arraylist-with-multiple-object-types
https://www.programcreek.com/2011/03/java-write-to-a-file-code-example/
https://beginnersbook.com/2013/12/how-to-loop-arraylist-in-java/
https://www.mkyong.com/java/java-display-double-in-2-decimal-points/
♦ για τον buffer και το producer/consumer problem
https://stackoverflow.com/questions/13666091/threads-and-buffers-in-java
https://github.com/eclipse/paho.mqtt.java/issues/9
http://www.java67.com/2012/12/producer-consumer-problem-with-wait-and-notify-example.html
http://www.umeshmorsu.com/2017/05/17/producer-consumer-problem/
https://stackoverflow.com/questions/19602601/create-an-arraylist-with-multiple-object-types