



ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ  
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ  
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ  
ΤΟΜΕΑΣ ΗΛΕΚΤΡΟΜΑΓΝΗΤΙΚΩΝ ΕΦΑΡΜΟΓΩΝ  
ΗΛΕΚΤΡΟΟΠΤΙΚΗΣ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΥΛΙΚΩΝ

**Σχεδιασμός και κατασκευή συστήματος μετρήσεων και ελέγχου**

**ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ**

Ευάγγελος Δ. Αγγελίδης

Ιωάννης Α. Πινακούλας

**Επιβλέπων :** Πολιτόπουλος Κωνσταντίνος

Επίκουρος καθηγητής Ε.Μ.Π.

Αθήνα, Οκτώβριος 2015





ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ  
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ  
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ  
ΤΟΜΕΑΣ ΗΛΕΚΤΡΟΜΑΓΝΗΤΙΚΩΝ ΕΦΑΡΜΟΓΩΝ  
ΗΛΕΚΤΡΟΟΠΤΙΚΗΣ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΥΛΙΚΩΝ

## Σχεδιασμός και κατασκευή συστήματος μετρήσεων και ελέγχου

### ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Ευάγγελος Δ. Αγγελίδης

Ιωάννης Α. Πινακούλας

**Επιβλέπων :** Πολιτόπουλος Κωνσταντίνος

Επίκουρος καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 15<sup>η</sup> Οκτωβρίου 2015.

.....

Πολιτόπουλος Κωνσταντίνος  
Επίκουρος καθηγητής Ε.Μ.Π.

.....

Καμπουράκης Γεώργιος  
Αναπληρωτής καθηγητής Ε.Μ.Π.

.....

Αλεξανδράτου Ελένη  
Ε.ΔΙ.Π. Ε.Μ.Π.

Αθήνα, Οκτώβριος 2015

.....  
Ευάγγελος Δ. Αγγελίδης

Ιωάννης Α. Πινακούλας

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Copyright © Ευάγγελος Δ. Αγγελίδης, 2015.

Copyright © Ιωάννης Α. Πινακούλας, 2015.

Με επιφύλαξη παντός δικαιώματος. All rights reserved.

Απαγορεύεται η αντιγραφή, αποθήκευση και διανομή της παρούσας εργασίας, εξ ολοκλήρου ή τμήματος αυτής, για εμπορικό σκοπό. Επιτρέπεται η ανατύπωση, αποθήκευση και διανομή για σκοπό μη κερδοσκοπικό, εκπαιδευτικής ή ερευνητικής φύσης, υπό την προϋπόθεση να αναφέρεται η πηγή προέλευσης και να διατηρείται το παρόν μήνυμα. Ερωτήματα που αφορούν τη χρήση της εργασίας για κερδοσκοπικό σκοπό πρέπει να απευθύνονται προς τον συγγραφέα.

Οι απόψεις και τα συμπεράσματα που περιέχονται σε αυτό το έγγραφο εκφράζουν τον συγγραφέα και δεν πρέπει να ερμηνευθεί ότι αντιπροσωπεύουν τις επίσημες θέσεις του Εθνικού Μετσόβιου Πολυτεχνείου.

## Περίληψη

Σκοπός της συγκεκριμένης διπλωματικής εργασίας είναι ο σχεδιασμός και η κατασκευή ενός συστήματος μετρήσεων για χρήση στο εργαστήριο Βιοϊατρικής Οπτικής και Εφαρμοσμένης Βιοφυσικής. Η κατασκευή αποτελείται από δύο υποσυστήματα. Το πρώτο – που είναι επιφορτισμένο με την καταγραφή και την επεξεργασία των μετρήσεων– περιέχει έναν αριθμό αναλογικών και ψηφιακών αισθητήρων (θερμοκρασίας, υγρασίας και διοξειδίου του άνθρακα), οι οποίοι ελέγχονται και παρακολουθούνται από ένα μικροελεγκτή 32 bit αρχιτεκτονικής ARM. Το δεύτερο τμήμα, που έχει σκοπό το χειρισμό του μικροελεγκτή, τη γραφική απεικόνιση και αποθήκευση των μετρήσεων και την επικοινωνία με τον χρήστη μέσω της οθόνης αφής, απαρτίζεται από μια συσκευή με λειτουργικό σύστημα Android (συγκεκριμένα ένα tablet), καθώς και την αντίστοιχη εφαρμογή που φέρνει εις πέρας τη συγκεκριμένη λειτουργία. Η διασύνδεση-επικοινωνία των δύο υποσυστημάτων επιτυγχάνεται με τη χρήση USB.

**Λέξεις κλειδιά:** μικροελεγκτής ARM 32 bit, ADC, I<sup>2</sup>C, αναλογικός αισθητήρας, Android, οθόνη αφής, multithreading, AChartEngine, USB On The Go

## ΠΕΡΙΕΧΟΜΕΝΑ

Κεφάλαιο 1: Εισαγωγή.....	σελ.9
Κεφάλαιο 2: USB.....	σελ.15
2.1 Εισαγωγή.....	σελ.15
2.2 Ανώτερο στρώμα.....	σελ.16
2.3 Μεσαίο στρώμα.....	σελ.18
2.4 Κατώτερο στρώμα.....	σελ.19
Κεφάλαιο 3: Έλεγχος αισθητήρων μέσω μικροελεγκτή.....	σελ.23
3.1 Περιγραφή μικροελεγκτή.....	σελ.23
3.2 Αισθητήρες.....	σελ.25
3.2.1 Πρωτόκολλο I <sup>2</sup> C.....	σελ.25
3.2.2 Αισθητήρας CC2D.....	σελ.28
3.3 Προγραμματισμός μικροελεγκτή.....	σελ.30
3.3.1 Προγραμματισμός USB.....	σελ.30
3.3.2 Προγραμματισμός ανάγνωσης αισθητήρα I <sup>2</sup> C.....	σελ.33
3.3.3 Προγραμματισμός ανάγνωσης αναλογικών αισθητήρων.....	σελ.33
Κεφάλαιο 4: Χειρισμός και καταγραφή συστήματος μέσω οθόνης αφής.....	σελ.37
4.1 Γενική περιγραφή συστήματος.....	σελ.37
4.2 Λειτουργικό σύστημα Android.....	σελ.37
4.2.1 Αρχιτεκτονική λειτουργικού συστήματος Android.....	σελ.38
4.3 Βιβλιοθήκες για σχεδίαση γραφικών παραστάσεων σε περιβάλλον Android .....	σελ.41
4.4 USB και Android.....	σελ.42
4.4.1 Προγραμματισμός USB από την πλευρά του Android.....	σελ.43
4.5 Εφαρμογή (application) για Android.....	σελ.44
4.5.1 Multithreading (πολλαπλά νήματα και παράλληλη επεξεργασία)..	σελ.45
4.5.2 Δομικά τμήματα της εφαρμογής.....	σελ.46
4.5.2.1 MainActivity.class.....	σελ.46

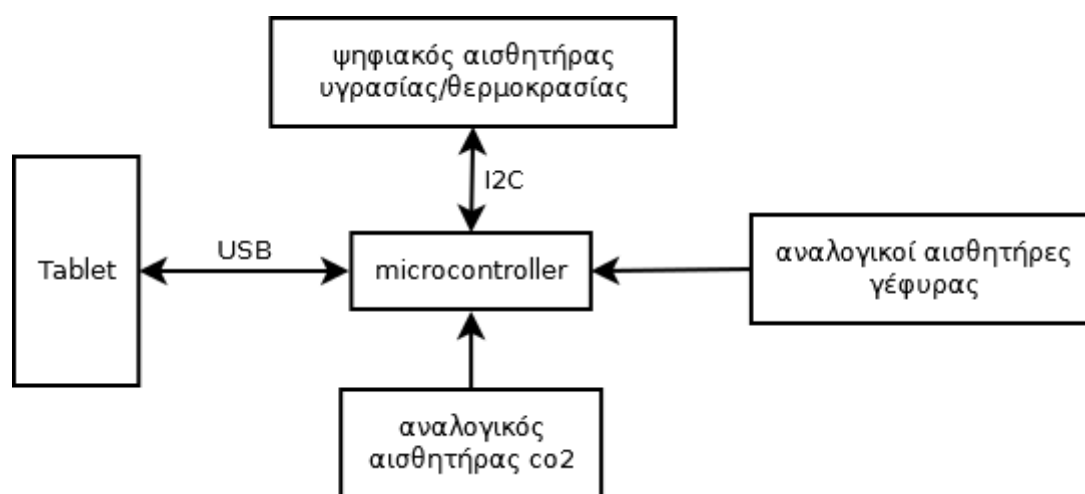
4.5.2.2 sensorsUIActivity.class.....σελ.47	σελ.47
4.5.2.3 PlotSettingsActivity.class.....σελ.50	σελ.50
4.5.2.4 SettingsActivity.class.....σελ.50	σελ.50
4.5.2.5 plotActivity.class.....σελ.51	σελ.51
4.5.3 Περιγραφή και ανάλυση των classes που δημιουργήσαμε.....σελ.56	σελ.56
4.5.4 Περιγραφή των μεθόδων του κώδικα που αφορούν τις λειτουργίες σχετικά με τη σύνδεση USB.....σελ.57	σελ.57
4.6 Μελλοντικές επεκτάσεις και αναβαθμίσεις.....σελ.59	σελ.59
Παράρτημα A: Πηγαίος κώδικας μικροελεγκτή.....σελ.61	σελ.61
A.1 Κώδικας για τη λειτουργία USB.....σελ.61	σελ.61
A.2 Κώδικας για τη λειτουργία των αισθητήρων.....σελ.70	σελ.70
Παράρτημα B: Πηγαίος κώδικας Android.....σελ.77	σελ.77
B.1 Κώδικας Android σε Java.....σελ.77	σελ.77
B.2 Κώδικας XML για σχεδιασμό των οθόνων χρήστη (User Interface).....σελ.128	σελ.128



# Κεφάλαιο 1

## Εισαγωγή

Η παρούσα διπλωματική εργασία είναι μέρος της κατασκευής ενός μετρητικού οργάνου για το εργαστήριο Βιοϊατρικής Οπτικής και Εφαρμοσμένης Βιοφυσικής. Το όργανο αυτό έχει σκοπό αφενός την παρατήρηση της μεταβολής του CO<sub>2</sub>, της υγρασίας και της θερμοκρασίας κατά την εκτέλεση πειραμάτων στο εργαστήριο και αφετέρου τον έλεγχο ενός πιεζοηλεκτρικού στοιχείου σε διάταξη γέφυρας. Προκειμένου να επιτευχθούν τα παραπάνω, χρησιμοποιούνται κάποιοι αναλογικοί αισθητήρες καθώς και ένας ψηφιακός που ελέγχονται μέσω μικροελεγκτή. Ο χρήστης χειρίζεται το συνολικό σύστημα (επιλέγει αισθητήρες προς παρακολούθηση, καταγράφει τις μετρήσεις κτλ.) μέσω της οθόνης αφής ενός tablet που συνδέεται στο μικροελεγκτή μέσω USB. Το γενικό διάγραμμα της κατασκευής φαίνεται στο σχήμα 1.1.



**Σχήμα 1.1** Γενικό διάγραμμα κατασκευής

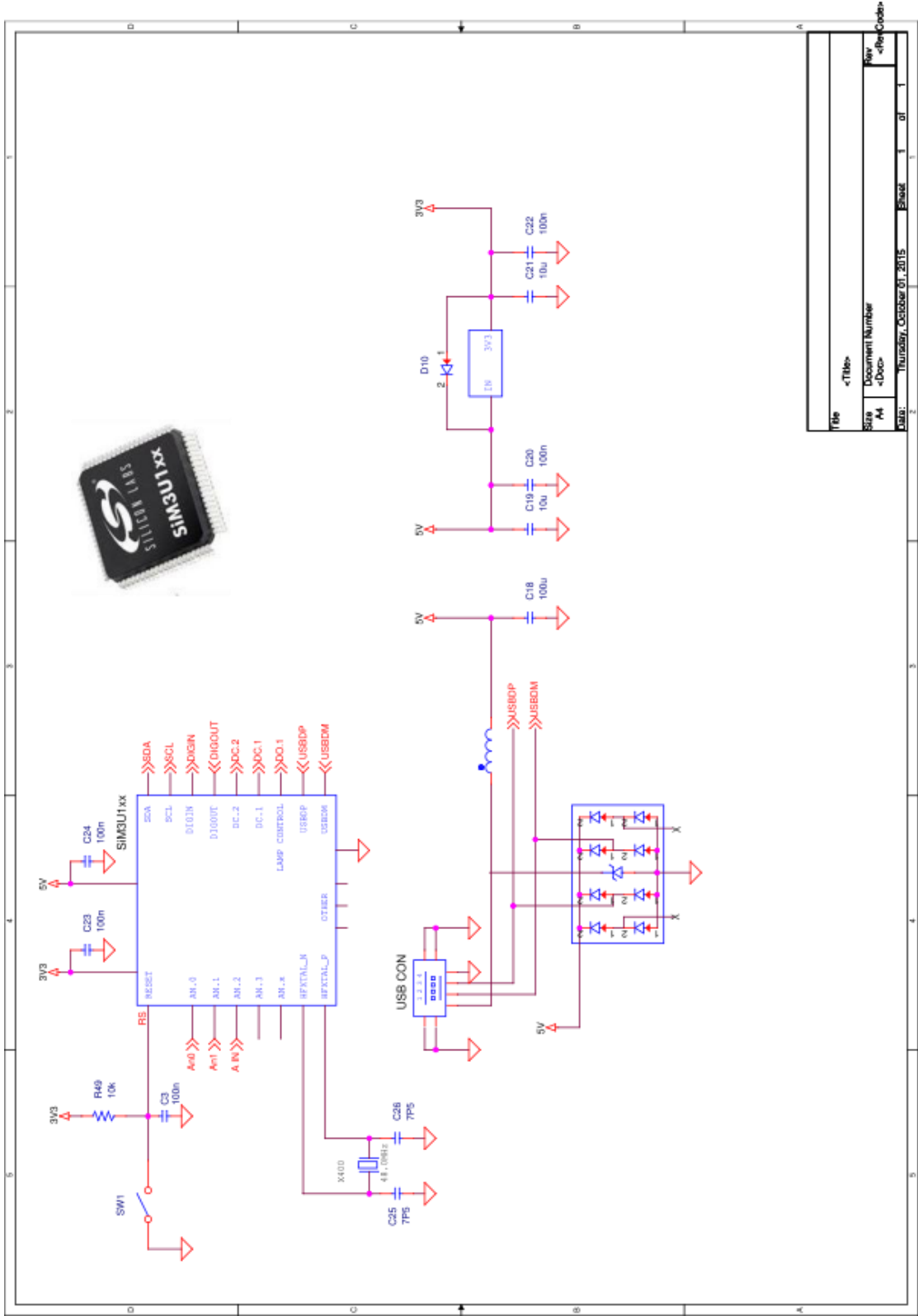
Η κατασκευή αποτελείται από δύο μέρη, το αναλογικό και το ψηφιακό. Το αναλογικό ασχολείται με την σχεδίαση και υλοποίηση των κυκλωμάτων λειτουργίας των αισθητήρων και της σύνδεσης τους με τον μικροελεγκτή. Το ψηφιακό ασχολείται με την ανάπτυξη του λογισμικού στον μικροελεγκτή και το tablet. Το αντικείμενο της παρούσας εργασίας είναι το ψηφιακό μέρος της κατασκευής.

Ο μικροελεγκτής της εργασίας είναι 32 bit τύπου ARM. Περιλαμβάνει διάφορα περιφερειακά για σύνδεση με άλλα αναλογικά ή ψηφιακά συστήματα. Ο προγραμματισμός του στα πλαίσια της εργασίας περιορίστηκε στη χρήση του USB για τη επικοινωνία με το tablet, του I<sup>2</sup>C για την επικοινωνία με τον ψηφιακό αισθητήρα υγρασίας και θερμοκρασίας και του ADC για την ανάγνωση των αναλογικών αισθητήρων. Ο μικροελεγκτής επιλέγει τους αισθητήρες που θα παρακολουθεί σύμφωνα με εντολές που δέχεται από το tablet μέσω του USB. Με τη χρήση της ίδιας σύνδεσης στέλνει τις μετρήσεις που καταγράφει στο tablet για επεξεργασία και απεικόνιση στον χρήστη.

Ο αρχικός σχεδιασμός της εργασίας ήταν να κατασκευαστεί πλακέτα πάνω στην οποία θα τοποθετούνταν ο μικροελεγκτής και οι αισθητήρες μαζί με τα απαραίτητα κυκλώματα για την λειτουργία και τη σύνδεση των επιμέρους στοιχείων (κρύσταλλος [crystal oscillator] για τον μικροελεγκτή, κύκλωμα θύρας USB, κυκλώματα τροφοδοσίας αισθητήρων κτλ.). Λόγω έλλειψης χρόνου η πλακέτα αυτή δεν κατασκευάστηκε και ο προγραμματισμός του μικροελεγκτή κατά την διάρκεια της εργασίας έγινε σε ένα evaluation board.

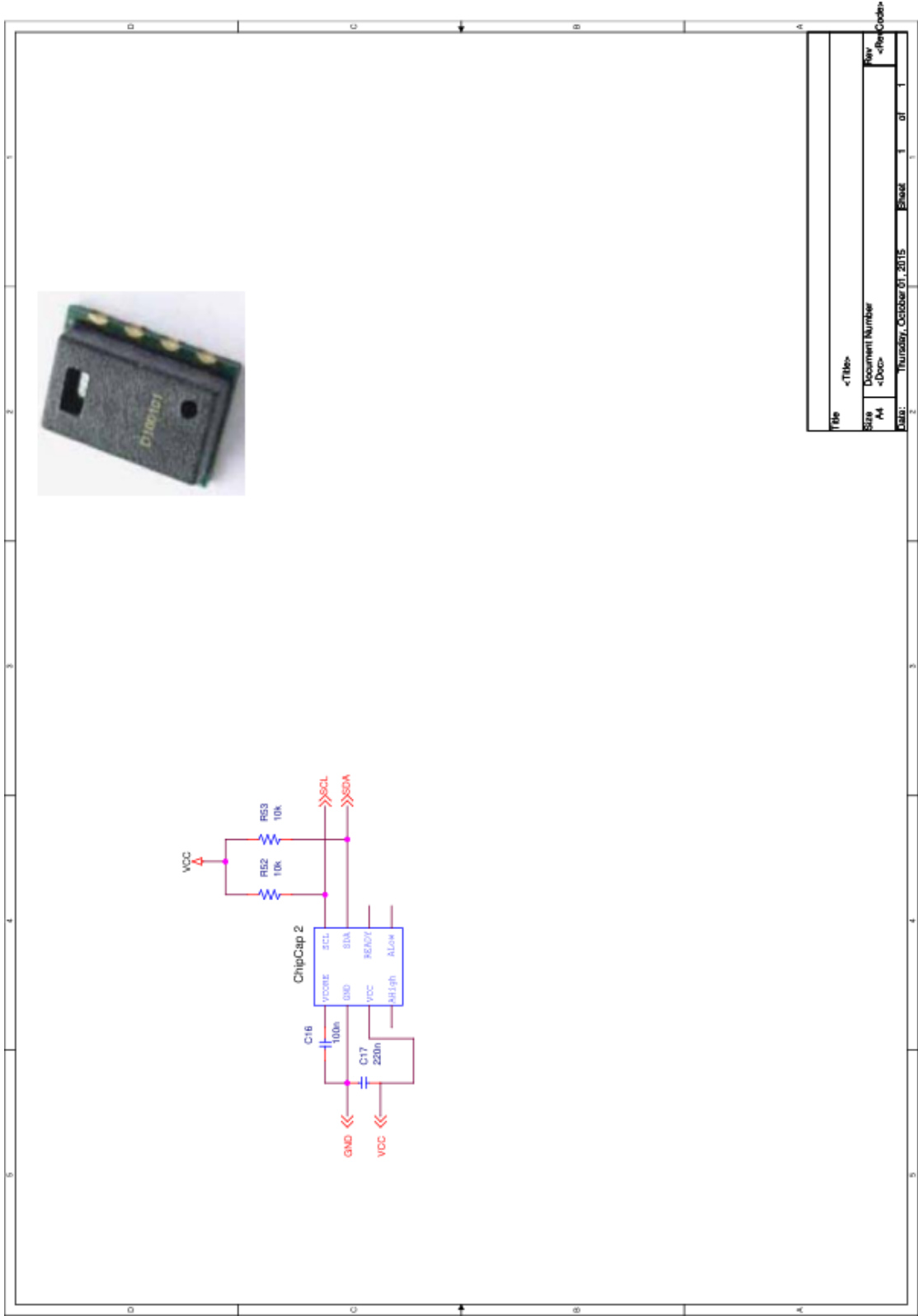
Το tablet, που περιλαμβάνει την οθόνη αφής μέσω της οποίας ελέγχεται το όργανο μέτρησης και απεικονίζονται οι μετρήσεις, έχει εγκατεστημένο το λειτουργικό σύστημα Android. Για τους σκοπούς της εργασίας δημιουργήθηκε μία εφαρμογή (Android app) με την οποία ο χρήστης εισάγει παραμέτρους που καθορίζουν, πρώτον, ποιους αισθητήρες θα παρακολουθεί και τον ρυθμό δειγματοληψίας τους (αν υπάρχει η επιλογή για τον συγκεκριμένο αισθητήρα) και, δεύτερον, τα χαρακτηριστικά απεικόνισης των μετρήσεων. Επιπλέον ο χρήστης δίνει την εντολή για την εκκίνηση και τον τερματισμό της δειγματοληψίας. Η εφαρμογή αναλαμβάνει την αποστολή των εντολών του χρήστη στο μικροελεγκτή μέσω του USB, την λήψη των μετρήσεων, την επεξεργασία, την γραφική αναπαράσταση και την αποθήκευσή τους για μελλοντική χρήση.

Στα σχήματα που ακολουθούν, παρουσιάζονται τα κυκλώματα του μικροελεγκτή και των αισθητήρων όπως θα κατασκευάζονταν πάνω στη πλακέτα του μετρητικού οργάνου. Ο μικροελεγκτής διαθέτει ένα μεγάλο αριθμό εισόδων και εξόδων για την λειτουργία των διαφόρων περιφερειακών του, αλλά στο σχήμα σημειώνονται μόνο όσες μας απασχόλησαν στην εργασία. Αυτές είναι οι γραμμές που απαιτούνται για τη σύνδεση USB, οι γραμμές SCL και SDA για τη σύνδεση I<sup>2</sup>C, τα αναλογικά κανάλια για την ανάγνωση των αντίστοιχων αισθητήρων και οι ψηφιακές έξοδοι για τον έλεγχο τους. Επιπλέον παρουσιάζονται τα κυκλώματα τροφοδοσίας του μικροελεγκτή και των αισθητήρων και η σύνδεση του πρώτου με τον κρύσταλλο (crystal oscillator).



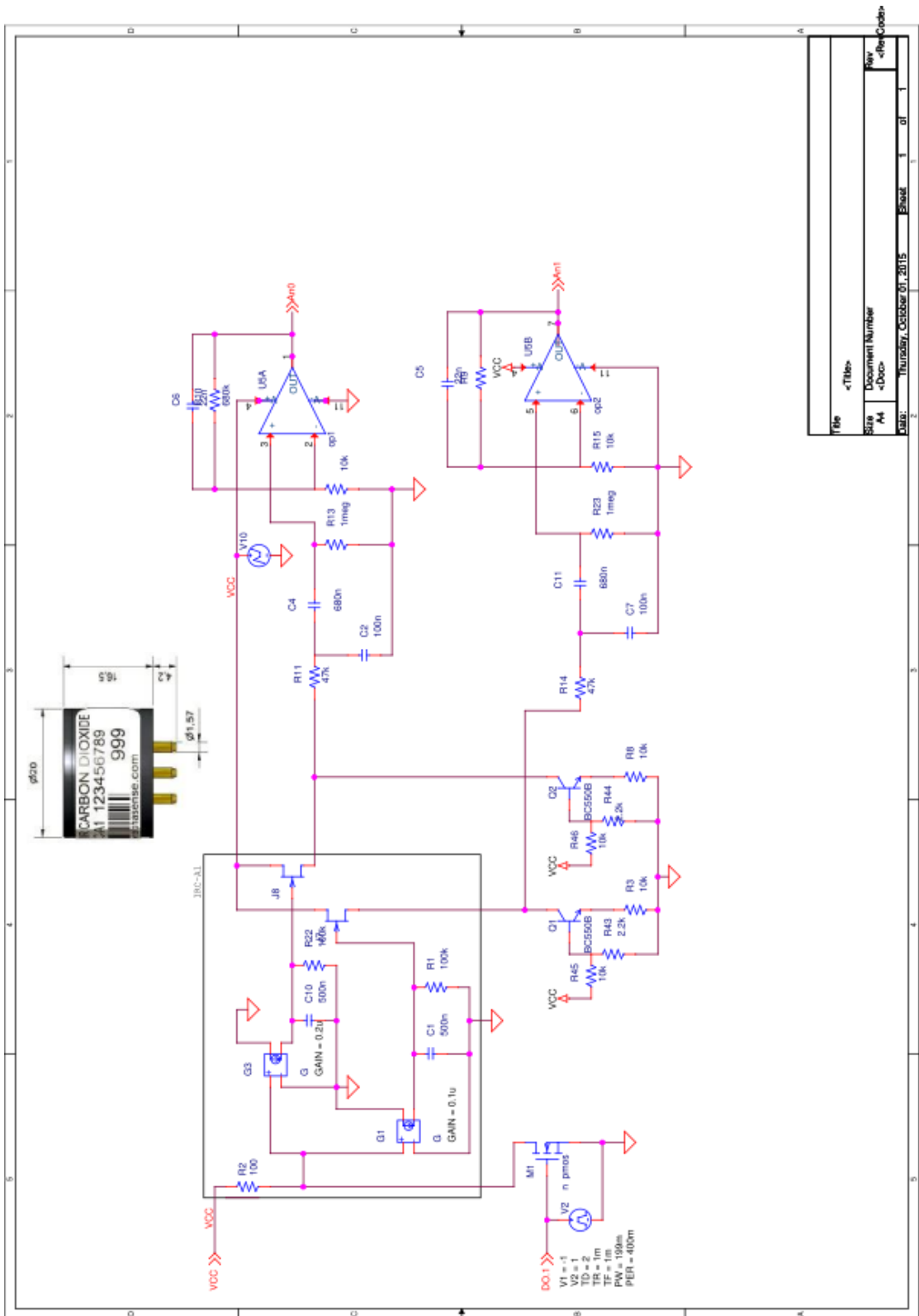
Σχήμα 1.2 Κύκλωμα μικροελεγκτή

File	<Title>
Size	Document Number
A4	<Doc>
Date:	Thursday, October 01, 2015
Sheet	1 of 1
Rev	<Rev Code>

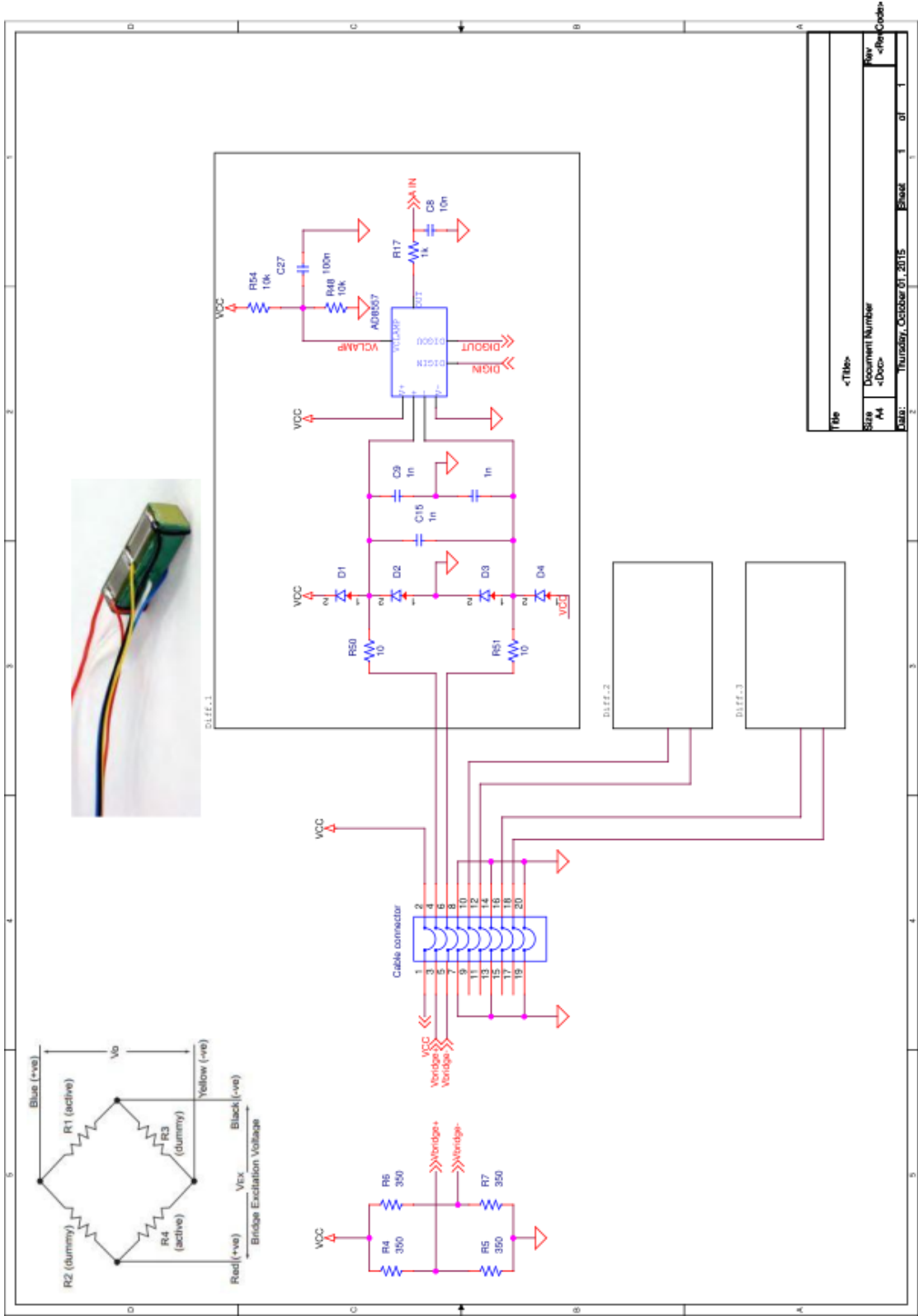


File	<Title>
Size	Document Number
A4	-Doc-
Date:	Thursday, October 01, 2015
Sheet	1 of 1
Rev	<Rev Code>

Σχήμα 1.3 Κύκλωμα ψηφιακού αισθητήρα υγρασίας και θερμοκρασίας



Σχήμα 1.4 Κύκλωμα αναλογικού αισθητήρα CO<sub>2</sub>



Σχήμα 1.5 Κύκλωμα γέφυρας

## Κεφάλαιο 2

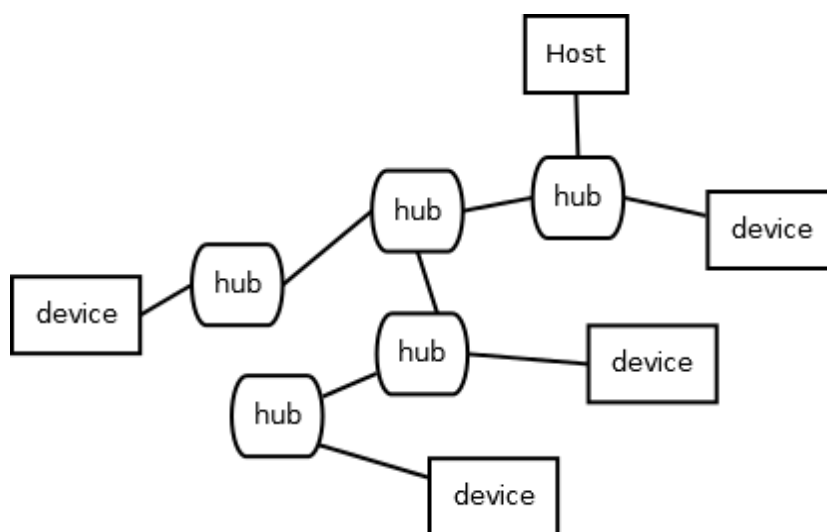
### USB

#### 2.1 Εισαγωγή

Το USB (Universal Serial Bus) ή ενιαίος σειριακός διάυλος είναι ένας τρόπος επικοινωνίας για την σύνδεση του προσωπικού υπολογιστή με τις περιφερειακές ηλεκτρονικές συσκευές. Δημιουργήθηκε με στόχο την αντικατάσταση των παλιότερων και ποικίλων μεθόδων σύνδεσης (π.χ. σειριακές/παράλληλες θύρες, θύρα για πληκτρολόγιο ή ποντίκι) με μια κοινή διεπαφή. Επιπλέον στόχευε στην απλοποίηση της χρήσης προσφέροντας δυνατότητες plug-and-play, τροφοδοσίας περιφερειακών συσκευών χαμηλής ισχύος και αποδοτικότερη χρήση των πόρων του υλικού του υπολογιστή (κανάλια DMA, interrupts). Το USB είναι ασύμμετρο πρωτόκολλο τύπου master-slave όπου ο master ονομάζεται υποδοχέας (host) και ο slave συσκευή (device). Τα βασικά χαρακτηριστικά του είναι τα εξής:

- Είναι σειριακό και τα bit μεταδίδονται σε πακέτα με πρώτο το λιγότερο σημαντικό. Προσφέρει ανταλλαγή χειραψίας (handshaking) και έλεγχο λαθών για την μεταφορά δεδομένων. Επίσης ο διάυλος χωρίζεται σε χρονικά πλαίσια (frames) μέσα στα οποία τοποθετούνται τα πακέτα.
- Είναι ημι-αμφίδρομο: Ο “host” ξεκινάει τις ανταλλαγές δεδομένων, αλλά αυτά μεταφέρονται και προς τις δύο πλευρές της σύνδεσης. Η κατεύθυνση των δεδομένων ορίζεται με αναφορά στον “host”, IN προς αυτόν και OUT προς το “device”.
- Ο “host” διαχειρίζεται τα “devices”: Ορίζει διευθύνσεις για όσα συνδέονται και σταματάει την υποστήριξη σε όσα αποσυνδέονται.
- Το USB υποστηρίζει διαχείριση της ισχύος με χρήση της κατάστασης αναστολής (Suspend mode) όταν δεν ανταλλάσσονται δεδομένα.

Η αρχιτεκτονική σύνδεσης του “host” με τα “devices” είναι μια κλιμακωτή τοπολογία αστέρα (σχήμα 1.1). Ο “host”- ριζικός κόμβος (root hub) μπορεί να επικοινωνεί με 127 κόμβους το μέγιστο που τοποθετούνται μέχρι και το τελευταίο επίπεδο της τοπολογίας δένδρου που σχηματίζεται. Το βάθος του δένδρου δεν μπορεί να υπερβαίνει τον αριθμό 5.



Σχήμα 2.1 Παράδειγμα τοπολογίας USB

Το USB έχει γνωρίσει διάφορες εκδόσεις από την εμφάνιση του (1.0,1.1,2.0,3.0,3.1). Η έκδοση που υλοποιείται στον μικροελεγκτή και το tablet είναι η 2.0 και μια επέκταση αυτής, το USB On The Go. Η έκδοση 2.0 ορίζει τρεις ταχύτητες, την Low Speed (1.5 Mbps), την Full Speed (12 Mbps) και τη High Speed (480 Mbps). Ο μικροελεγκτής δεν υποστηρίζει την τελευταία, επομένως στην εργασία χρησιμοποιείται η δεύτερη. Το USB On The Go δημιουργήθηκε για φορητούς υπολογιστές, όπως είναι τα tablets και τα smartphones, όπου υπάρχει ανάγκη να αναλαμβάνουν ρόλο “device” σε κάποιες περιπτώσεις (π.χ. σύνδεση σε προσωπικό υπολογιστή) και ρόλο “host” σε άλλες (π.χ. σύνδεση με πληκτρολόγιο ή εκτυπωτή). Στην εργασία μας ορίστηκε να χρησιμοποιηθεί ο μικροελεγκτής ως “device” και τον ρόλο του “host” αναλαμβάνει το tablet.

Η επικοινωνία μέσω μιας σύνδεσης USB μπορεί να αναλυθεί ως μια στοίβα τριών στρωμάτων. Το ανώτερο στρώμα ασχολείται με τη διαχείριση των ροών δεδομένων από και προς το “device” είτε πρόκειται για εντολές ελέγχου της σύνδεσης είτε για δεδομένα της εφαρμογής (π.χ. μετρήσεις αισθητήρων). Το μεσαίο στρώμα εμπεριέχει την πληροφορία των παραμέτρων του “device”. Βάσει αυτών ο “host” καθορίζει τα χαρακτηριστικά της σύνδεσης. Το κατώτερο στρώμα αναλαμβάνει την εφαρμογή των παραπάνω σε επίπεδο υλικού. Τα όρια μεταξύ των στρωμάτων δεν είναι πάντα ευδιάκριτα όπως θα φανεί παρακάτω.

Το λογισμικό που αναπτύχθηκε για τις ανάγκες της εργασίας στην πλευρά του tablet (host) αλληλεπιδρά μόνο με το ανώτερο στρώμα ενώ στην πλευρά του μικροελεγκτή (device) ασχολείται και με το μεσαίο. Οι λειτουργίες του κατώτερου στρώματος εκτελούνται από το υλικό. Στο υπόλοιπο του κεφαλαίου παρουσιάζεται συνοπτικά η επικοινωνία “host” -“device” ανά στρώμα, με προσέγγιση από πάνω προς τα κάτω δίνοντας έμφαση κυρίως στα σημεία που ενδιαφέρουν περισσότερο την εργασία.

## 2.2 Ανώτερο στρώμα

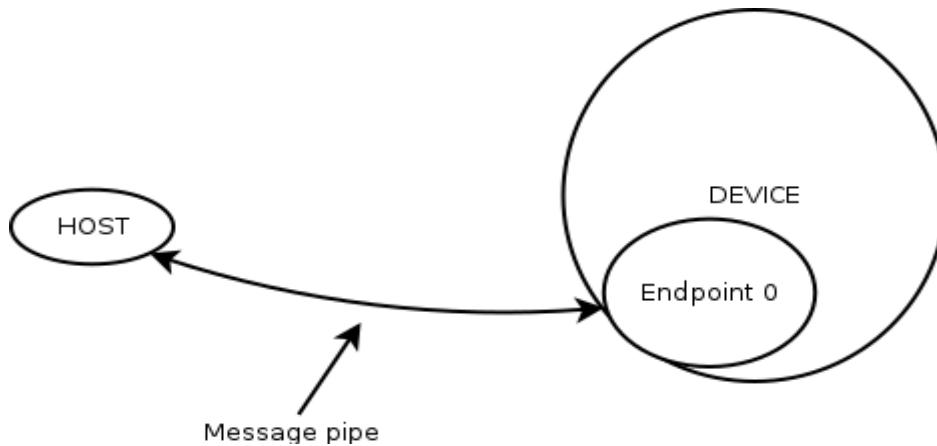
Οι ροές δεδομένων που αναφέρθηκαν ονομάζονται σωλήνες (pipes) στην ορολογία του USB. Ένα “pipe” είναι μια λογική σύνδεση μεταξύ “host” και ενός άκρου (endpoint) του “device” για την μετάδοση των δεδομένων. Τα “endpoints” είναι χώροι μνήμης στο “device” που λειτουργούν σαν buffers. Εκεί τοποθετούνται και παραλαμβάνονται τα δεδομένα από τον “host”. Η προσπέλαση τους από το λογισμικό γίνεται με τη χρήση κάποιας δομής, συνήθως έναν πίνακα byte. Οργανώνονται σε ζευγάρια για μετάδοση προς τον “host” (IN) και από τον “host” (OUT), αλλά δεν είναι υποχρεωτική η χρήση και των δύο σε ένα “pipe”. Αυτό και άλλα χαρακτηριστικά των “pipes” ορίζονται στο μεσαίο στρώμα.

Υπάρχουν δύο ειδών “pipes”:

1. **Message pipe:** Είναι ένα “pipe” που υπάρχει σε κάθε σύνδεση USB και συνδέει τον “host” με το αμφίδρομο “endpoint” 0 (το οποίο υπάρχει υποχρεωτικά σε κάθε “device”). Μέσω του “message pipe” εγκαθίσταται η σύνδεση, ελέγχεται η κατάσταση της και επιλέγεται αν προσφέρεται άλλη λειτουργία του “device”. Τα δεδομένα είναι αιτήσεις ( USB requests) από τον “host” προς το “device” και η μορφή τους καθορίζεται από το πρωτόκολλο.

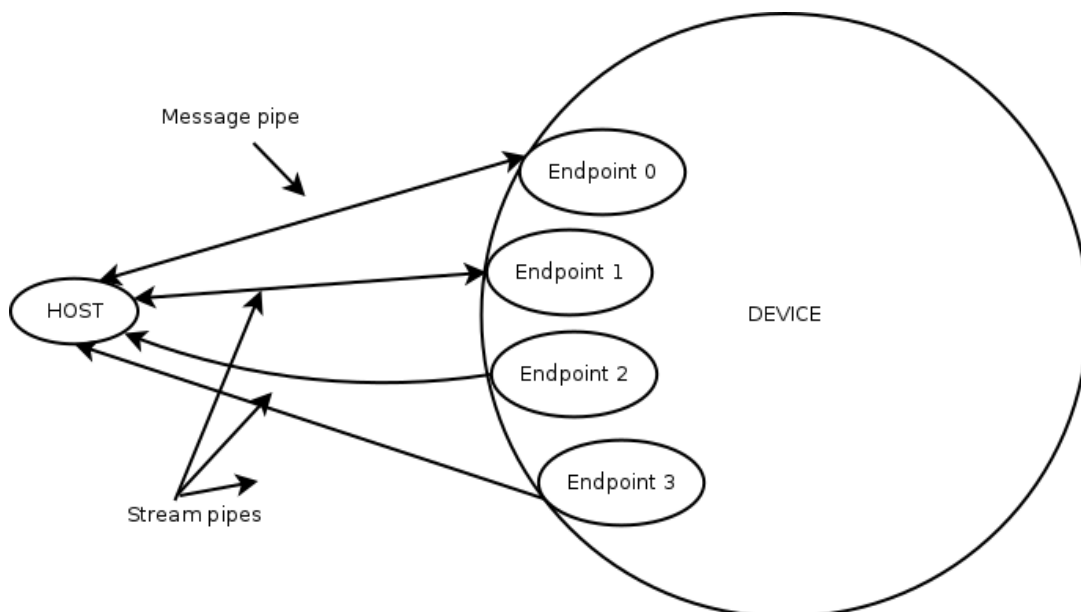
2. **Stream pipe:** Τα “pipes” αυτά μεταφέρουν την πληροφορία της εφαρμογής. Τα δεδομένα δεν έχουν καθορισμένη μορφή, αλλά είναι πακέτα από bytes που η σημασία τους ορίζεται από την εφαρμογή. Μπορεί να δημιουργηθεί ένα “stream pipe” για κάθε ζευγάρι “endpoints” της συσκευής. Η κατεύθυνση του είναι προς τον “host” αν χρησιμοποιεί το “endpoint” IN, προς το “device” αν χρησιμοποιεί το “endpoint” OUT και αμφίδρομη αν χρησιμοποιεί και τα δύο.

Στο σχήμα 2.2 παρουσιάζεται το ανώτερο στρώμα στη φάση της απαρίθμησης-καταγραφής (enumeration). Η “enumeration” είναι η διαδικασία που ξεκινάει με την ηλεκτρική σύνδεση “host” -“device”. Μέσω του “message pipe” καταγράφονται πληροφορίες για το “device” και εγκαθίσταται η σύνδεση USB, αν αυτό είναι δυνατό.



**Σχήμα 2.2 Το ανώτερο στρώμα στη διαδικασία της enumeration.**

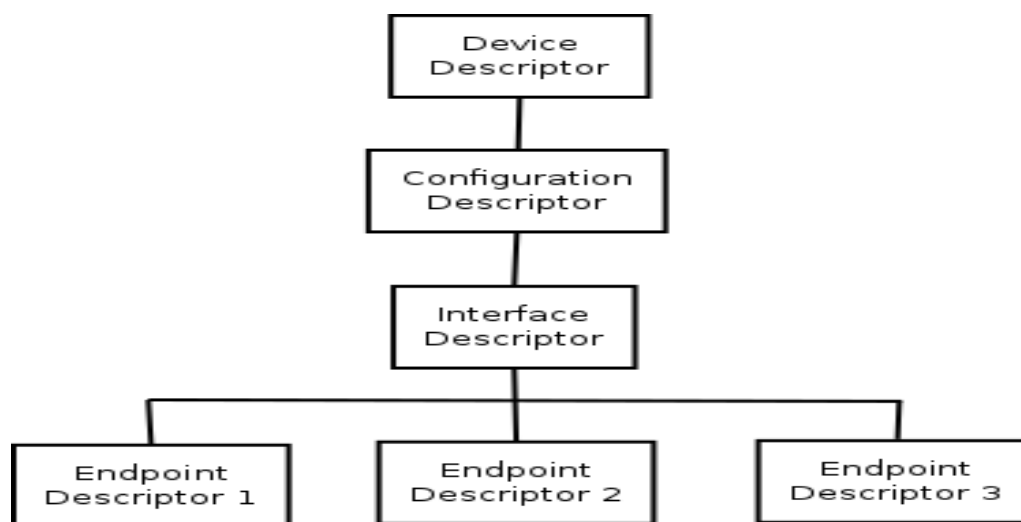
Ο “message pipe” παραμένει ενεργός όσο υπάρχει η σύνδεση παρότι δεν χρησιμοποιείται από την εφαρμογή της εργασίας μετά την “enumeration”. Για τις ανάγκες της εφαρμογής χρησιμοποιούνται τρία “stream pipes”. Το ένα είναι αμφίδρομο για αποστολή εντολών στον μικροελεγκτή και λήψη μετρήσεων από αυτόν ενώ τα άλλα δύο έχουν κατεύθυνση προς τον “host” και χρησιμοποιούνται μόνο για λήψη μετρήσεων. Στο σχήμα 2.3 παρουσιάζεται το ανώτερο στρώμα του USB κατά την λειτουργία της εφαρμογής.



**Σχήμα 2.3 Το ανώτερο στρώμα κατά τη λειτουργία της εφαρμογής.**

## 2.3 Μεσαίο στρώμα

Τα χαρακτηριστικά ενός USB device ορίζονται στους περιγραφείς (descriptors). Οι “descriptors”, όπως λέει και το όνομα τους, περιγράφουν στον “host” τα χαρακτηριστικά ενός “device” και τις λειτουργίες που αυτό προσφέρει. Είναι δομές δεδομένων που βρίσκονται στη μνήμη του “device”. Οι βασικοί τύποι “descriptors” είναι οι Device, Configuration, Interface και Endpoint. Αυτοί ακολουθούν μια ιεραρχία βάσει της οποίας γίνεται η προσπέλαση και η ανάκτηση πληροφοριών για το “device” από τον “host”. Στο σχήμα 2.4 φαίνεται αυτή η ιεραρχία όπως υλοποιήθηκε στον μικροελεγκτή.



Σχήμα 2.4 Ιεραρχία descriptors στη σύνδεση USB της εργασίας.

Όλοι οι “descriptors” έχουν κοινά τα δύο πρώτα πεδία της μορφής τους. Το πρώτο ενημερώνει τον “host” για το μέγεθος του και το δεύτερο τι τύπος είναι. Τα ιδιαίτερα χαρακτηριστικά του κάθε τύπου “descriptor” είναι τα εξής:

- Device Descriptor: Είναι μοναδικός για κάθε “device”. Περιέχει πληροφορίες για την έκδοση USB που υποστηρίζει το “device” και σε πια κατηγορία (class) αυτό ανήκει. Συνηθισμένες κατηγορίες είναι οι MSD (Mass Storage Device, π.χ. usb-sticks), HID (Human Interface Device, π.χ. πληκτρολόγιο) και CDC (Communication Device Class, προσομοίωση απλής σειριακής θύρας). Αν η λειτουργία USB δεν ανήκει σε κάποια από αυτές όπως στην περίπτωση της εργασίας μας η device class ορίζεται ως ειδική του προμηθευτή (vendor specific). Ο “Device Descriptor” ορίζει ακόμα το μέγιστο μέγεθος πακέτου για το “endpoint” 0, τον αριθμό του προμηθευτή του “device” και του προϊόντος, τον αριθμό των “Configuration Descriptors” και δείκτες για τους “String Descriptors”. Οι τελευταίοι είναι προαιρετικοί “descriptors” που περιέχουν πληροφορίες για το “device” σε μορφή αναγνώσιμη από τον άνθρωπο.
- Configuration Descriptor: Περιέχει πληροφορίες για μια λειτουργία που προσφέρει το “device”. Μπορεί να υπάρχουν περισσότεροι του ενός γενικά (όχι στον μικροελεγκτή που χρησιμοποιήθηκε), αλλά μόνο ένας επιλέγεται από τον “host”. Οι βασικές πληροφορίες που έχει είναι ο αριθμός των “Interfaces” της λειτουργίας, η τιμή που τον ξεχωρίζει από άλλους “Configuration Descriptors”, αν τροφοδοτείται από τον δίαυλο και, αν ναι, πόσο ρεύμα χρειάζεται.
- Interface Descriptor: Το “Interface” είναι μια ομαδοποίηση των “endpoints” που επιτελούν την λειτουργία του “Configuration” που ανήκουν. Κάθε “Configuration” μπορεί να έχει πολλά “Interfaces” για εναλλακτικές εκτελέσεις της ίδιας λειτουργίας ή για παροχή άλλων λειτουργιών παρακάμπτοντας έτσι τους

περιορισμούς σχετικά με τους “Configuration Descriptors”. Ο “Interface Descriptor” πληροφορεί τον “host” για τον αριθμό “endpoints” που χρησιμοποιεί και την κατηγορία που ανήκει.

- **Endpoint Descriptor:** Οι σημαντικότερες πληροφορίες ενός “Endpoint Descriptor” είναι η διεύθυνση του (αριθμός 4 bit, επομένως επιτρέπονται 16 “endpoints”), η κατεύθυνση του (IN ή OUT), τι τύπο “transfer” χρησιμοποιεί (εξηγείται παρακάτω) και το μέγιστο μέγεθος πακέτου που επιτρέπει.

Ένα ιδιαίτερο χαρακτηριστικό που ορίζεται στον “Endpoint Descriptor” είναι ο τύπος μεταφοράς (transfer) που χρησιμοποιεί το συγκεκριμένο “endpoint”.

Η “transfer” είναι μια ακολουθία από ανταλλαγές πακέτων μεταξύ “host” και “device” η οποία φαίνεται ενιαία στο ανώτερο στρώμα (π.χ. η αποστολή μιας εντολής από το tablet ή μιας σειράς μετρήσεων από τον μικροελεγκτή).

Υπάρχουν τέσσερις τύποι “transfers” και χωρίζονται ανάλογα με τη σημασία, αλλά και την επιτρεπτή χρονική καθυστέρηση:

1. **Control:** Χρησιμοποιούνται από τον “message pipe” και μόνο από αυτόν. Με αυτές ο “host” στέλνει τις τυπικές αιτήσεις (Standard requests) στη διάρκεια της “enumeration” για την εγκατάσταση της σύνδεσης και αργότερα για τον έλεγχο της.

2. **Interrupt:** Χρησιμοποιούνται για την μετάδοση μικρών μηνυμάτων. Ο “host” εγγύαται να παραλάβει το μήνυμα μέσα σε ένα χρονικό διάστημα που έχει οριστεί στη διαδικασία της “enumeration”. Τα Interrupt requests τοποθετούνται σε ουρά από το “device” μέχρις ότου ο “host” να επιλέξει το “device” και να ζητήσει δεδομένα. Παράδειγμα χρήσης αυτών είναι συνδέσεις υπολογιστή με πληκτρολόγιο ή ποντίκι.

3. **Isochronous:** Με αυτές μεταδίδεται πληροφορία περιοδικά και σε υψηλό ρυθμό. Χρησιμοποιούνται για πληροφορίες που εξαρτώνται από τον χρόνο, όπως είναι για παράδειγμα μετάδοση ροής ήχου ή βίντεο (real-time audio ή video). Απαιτούν καθορισμένο ρυθμό μεταφοράς δεδομένων (εύρος ζώνης/bandwidth), συνήθως το γρηγορότερο δυνατό, είναι πιθανό να έχουν απώλειες και δεν επιτρέπουν την επανάληψη της μετάδοσης σε περίπτωση λάθους.

4. **Bulk:** Χρησιμοποιούνται για την μετάδοση μεγάλου μεγέθους πληροφορίας που δημιουργείται σε ακανόνιστα χρονικά διαστήματα, κάνοντας χρήση όλου του διαθέσιμου bandwidth που έχει απομείνει, χωρίς όμως να έχουν εγγύηση για τυχόν καθυστέρηση. Διαθέτουν έλεγχο για σφάλματα με CRC (Κυκλικός Έλεγχος Πλεονασμού-Cyclic Redundancy Check) και έχουν εξασφαλισμένη παράδοση. Χρησιμοποιούνται για πληροφορίες που δεν εξαρτώνται από το χρόνο

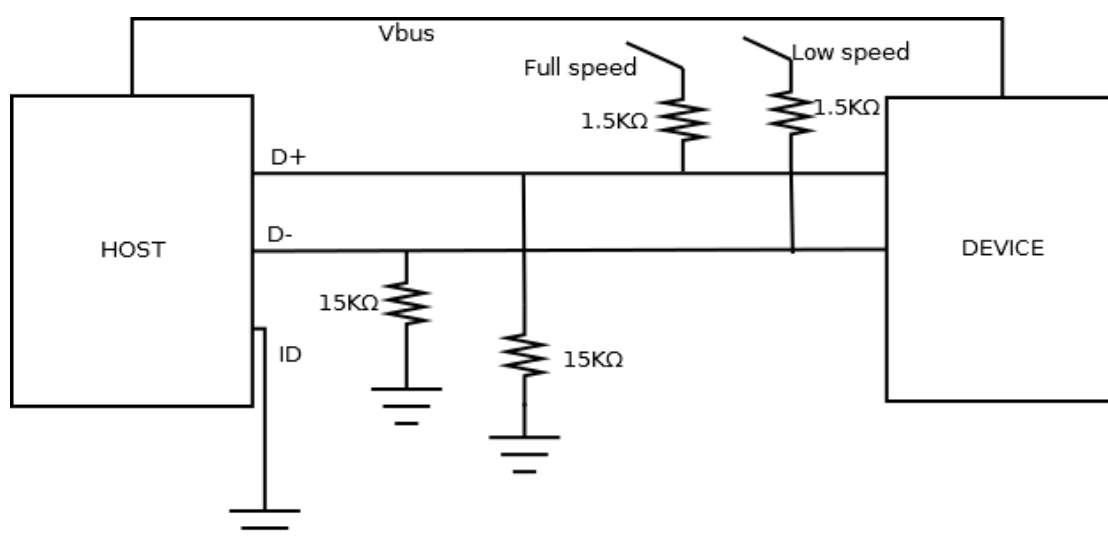
Στο δίαυλο του USB υπάρχει μια ιεραρχία κατανομής του εύρους ζώνης στους διάφορους τύπους transfers. Το 90% χρησιμοποιείται για τις Interrupt και Isochronous και το υπόλοιπο 10% για τις Control. Αν περισσεύει χώρος κατανέμεται στις Bulk. Στην εργασία τα “stream pipes” υλοποιούνται με τον τελευταίο τύπο γιατί τα δεδομένα μας παράγονται με πολύ αργούς ρυθμούς (τάξη μερικών millisecond).

## 2.4 Κατώτερο στρώμα

Οι λειτουργίες του κατώτερου στρώματος εκτελούνται από το υλικό των δύο πλευρών της σύνδεσης. Δεν απασχόλησε ιδιαίτερα την εργασία γιατί οι ρυθμίσεις που το αφορούν είτε ορίζονται έμμεσα από τα ανώτερα στρώματα (π.χ. από τους descriptors) είτε αυτόματα με τα χαρακτηριστικά της ηλεκτρικής σύνδεσης “host” και “device”. Το υλικό του USB

αποτελείται κυρίως από δύο τμήματα, τις ηλεκτρικές συνδέσεις, τα κυκλώματα προστασίας και τροφοδοσίας και ένα λογικό αυτόματο, το Serial Interface Engine (SIE).

Η ηλεκτρική σύνδεση του USB 2.0 αποτελείται από τέσσερις γραμμές, την τάση διαύλου ( $V_{BUS}$ ), την γείωση (GND), το DATA+ (D+) και το DATA- (D-). Το USB OTG πρόσθεσε μία ακόμα, το ID ή Sense. Στο σχήμα 2.5 φαίνεται μια τυπική σύνδεση "host"-“device”. Στην πλευρά του "host" οι ακροδέκτες D+ και D- συνδέονται με αντίσταση 15 KΩ στη γείωση ενώ στη πλευρά του "device" ο ένας από τους δύο συνδέεται με αντίσταση 1.5 KΩ στη  $V_{BUS}$ . Αν αυτός είναι ο ακροδέκτης D- η ταχύτητα της σύνδεσης είναι η Low Speed (1.5 Mbps), αλλιώς είναι η Full Speed (12 Mbps). Στον μικροελεγκτή χρησιμοποιείται μια ολοκληρωμένη αντίσταση και η επιλογή ποιός ακροδέκτης της συσκευής θα συνδεθεί στην  $V_{BUS}$  γίνεται με την αλλαγή ενός bit στον καταχωρητή που την ελέγχει.



**Σχήμα 2.5** Σύνδεση host-device

Οι ακροδέκτες D+ και D- χρησιμοποιούνται για την μετάδοση των bits στο κανάλι του USB. Όταν το D+ έχει υψηλή τάση και το D- χαμηλή μεταδίδεται 1 και όταν συμβαίνει το αντίθετο μεταδίδεται 0.

Ο ακροδέκτης ID χρησιμοποιείται για τον ορισμό του "host" στην αρχή της σύνδεσης δύο συσκευών USB OTG (οι ρόλοι μπορεί να αντιστραφούν αργότερα). Αν συνδέεται στη γείωση (υποδοχή τύπου A) ορίζεται ως "host" ενώ αν είναι ελεύθερη (υποδοχή τύπου B) ορίζεται ως "device". Επειδή η υποδοχή USB του tablet είναι τύπου B προεκτείνεται με ένα καλώδιο (εικόνα 2.1) που συνδέει το ID στη γείωση όπως φαίνεται στο σχήμα 2.5.



**Εικόνα 2.1 Καλώδιο host USB On-The-Go.**

Η ηλεκτρική σύνδεση του “host” και του “device” ή “attach event” ενεργοποιεί τη διαδικασία της “enumeration” που εκτελείται με Control transfers. Οι διαδικασίες αυτές καθώς και οι υπόλοιπες που συμβαίνουν κατά τη λειτουργία της εφαρμογής υλοποιούνται στο επίπεδο του υλικού από το SIE.

Το SIE αναλαμβάνει να χωρίσει μια “transfer” στις συναλλαγές (transactions) από τις οποίες αποτελείται. Η “transaction” είναι μία ακολουθία πακέτων που εντάσσεται πλήρως σε ένα frame και θεωρείται ενιαία για το κατώτερο στρώμα, δηλαδή αν αποτύχει η μετάδοση ενός πακέτου επαναλαμβάνεται ολόκληρη. Αποτελείται από τρία μέρη. Το πακέτο ένδειξης (Token) που ενημερώνει για τον τύπο της, το πακέτο δεδομένων (Data) αν χρειάζεται και το πακέτο κατάστασης (Status) που επιβεβαιώνει την επιτυχία ή όχι της “transaction”.

Η SIE εκτελεί όλες τις λειτουργίες που αφορούν την μετάδοση πακέτων από τον “host” στα “endpoints” του “device” παρέχοντας έτσι στο ανώτερο στρώμα έτοιμα τα “pipes”.



## Κεφάλαιο 3

### Έλεγχος αισθητήρων μέσω μικροελεγκτή

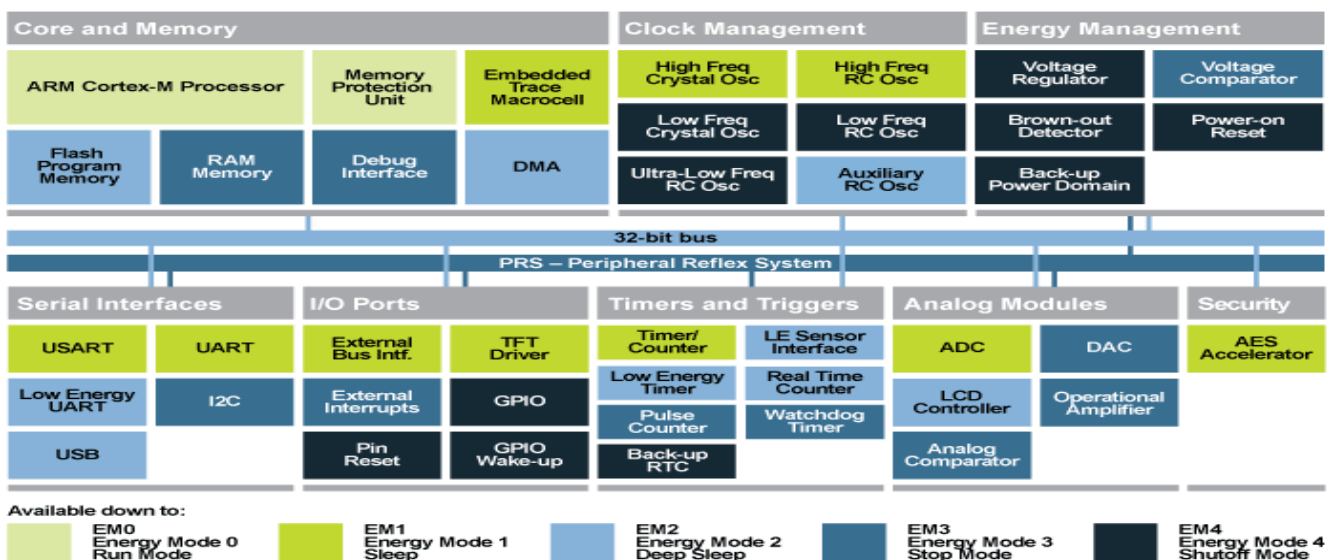
#### 3.1 Περιγραφή μικροελεγκτή

Ο μικροελεγκτής που χρησιμοποιήθηκε στην εργασία είναι ο EFM32 Wonder Gecko (EFM32WG) της Silicon Labs. Ο επεξεργαστής του είναι 32 bit τύπου ARM. Οι επεξεργαστές ARM ακολουθούν την αρχιτεκτονική RISC (Reduced Instruction Set Computer) η οποία σε σχέση με άλλους τύπους αρχιτεκτονικής υλοποιεί ένα μικρότερο σύνολο εντολών με βελτιστοποιημένη απόδοση. Ο σχεδιασμός αυτός οδηγεί σε μικρότερο κόστος και χαμηλότερη κατανάλωση ενέργειας γι' αυτό επιλέγονται να χρησιμοποιούνται σε φορητές συσκευές τροφοδοτούμενες από μπαταρία (smartphones, tablets κτλ) και γενικά σε ενσωματωμένα (embedded) συστήματα.

Ο επεξεργαστής του EFM32WG ανήκει στη σειρά επεξεργαστών ARM Cortex-M που σχεδιάστηκαν για μικροελεγκτές. Ο EFM32WG είναι Cortex-M4F δηλαδή υποστηρίζει εντολές για ψηφιακή επεξεργασία σήματος και πράξεις με αριθμούς κινητής υποδιαστολής.

Ο μικροελεγκτής EFM32WG χρησιμοποιείται για εφαρμογές αισθητήρων, συστημάτων ασφαλείας και βιομηχανικών ή οικιακών αυτοματισμών. Η σειρά μικροελεγκτών EFM σχεδιάστηκε με στόχο την χαμηλή κατανάλωση ενέργειας χωρίς την μείωση της απόδοσης. Αυτό επιτυγχάνεται με την μετάβαση κατά την λειτουργία του μικροελεγκτή σε ενεργειακές καταστάσεις (energy modes) μειωμένων αναγκών σε ισχύ στις οποίες όμως δεν είναι διαθέσιμα όλα τα περιφερειακά. Η επιστροφή στην αρχική energy mode (την 0) γίνεται με interrupt που δημιουργεί κάποιο περιφερειακό.

Ο επεξεργαστής του EFM32WG λειτουργεί με ταχύτητες 32KHz μέχρι 48 MHz , έχει 32 KB μνήμη RAM και, στην έκδοση που χρησιμοποιούμε, 256 KB μνήμη Flash. Στην εικόνα 3.1 φαίνονται τα περιφερειακά που περιέχει ο EFM32WG πέρα από τον επεξεργαστή και την μνήμη. Επίσης το χρώμα κάθε περιφερειακού δείχνει σε ποιες ενεργειακές καταστάσεις είναι προσβάσιμο.



Εικόνα 3.1 Περιφερειακά EFM32WG

Τα περιφερειακά του μικροελεγκτή με τα οποία ασχοληθήκαμε στο πρόγραμμα της εργασίας είναι το USB (για την επικοινωνία με το tablet), το I<sup>2</sup>C (για την επικοινωνία με τον ψηφιακό αισθητήρα υγρασίας και θερμοκρασίας) και το ADC (για την ανάγνωση αναλογικών αισθητήρων). Επιπλέον για την λειτουργία αυτών χρειάστηκε προγραμματισμός των ταλαντωτών (RC/crystal oscillators) που τροφοδοτούν τα ρολόγια των διάφορων περιφερειακών, των θυρών εισόδου/εξόδου γενικής χρήσης (GPIO) και των χρονομέτρων ή μετρητών (timers και counters).

Η εταιρεία ARM η οποία σχεδιάζει τους ομώνυμους επεξεργαστές έχει δημιουργήσει για τον προγραμματισμό τους το πρότυπο CMSIS (Cortex Microcontroller Software Interface Standard). Το πρότυπο αυτό ορίζει μια κοινή διεπαφή προγραμματισμού για τους μικροελεγκτές Cortex-M ανεξαρτήτως εταιρείας κατασκευής με στόχο την απλοποίηση και επαναχρησιμοποίηση του κώδικα. Ο προγραμματισμός του μικροελεγκτή στην παρούσα εργασία έγινε με χρήση της βιβλιοθήκης emlib (energy micro library) η οποία χρησιμοποιεί το CMSIS και δομείται πάνω σ' αυτό. Η βιβλιοθήκη αυτή προσφέρει ένα API για τη γλώσσα C με συναρτήσεις και δομές δεδομένων με τις οποίες ελέγχονται οι λειτουργίες και οι καταχωρητές των διάφορων περιφερειακών του μικροελεγκτή.

Η πλατφόρμα που χρησιμοποιήθηκε για τον προγραμματισμό είναι το Simplicity Studio. Έχει δημιουργηθεί από την Silicon Labs και προσφέρει πολλές λειτουργίες για την ανάπτυξη εφαρμογών στους μικροελεγκτές της. Κάποιες από αυτές είναι οι ακόλουθες: ένα ολοκληρωμένο περιβάλλον ανάπτυξης (Simplicity IDE), ένα γραφικό περιβάλλον δημιουργίας κώδικα για την προετοιμασία (initialization) των περιφερειακών (Simplicity Configurator) και δυνατότητες για εντοπισμό σφαλμάτων (debugging). Επίσης περιέχει το εργαλείο energyAware Commander με το οποίο προγραμματίζεται ο μικροελεγκτής της πλακέτας σχεδίασης (evaluation board) που χρησιμοποιείται.

Στόχος της εργασίας ήταν να προγραμματιστεί ο μικροελεγκτής πάνω στην πλακέτα που θα κατασκευαζόταν. Επειδή δεν ολοκληρώθηκε, για τις δοκιμές του προγράμματος χρησιμοποιήθηκε το evaluation board που φαίνεται στην εικόνα 3.2.



**Εικόνα 3.2 Το evaluation board STK3800**

## 3.2 Αισθητήρες

Ο μικροελεγκτής συνδέεται είτε με ψηφιακούς είτε με αναλογικούς αισθητήρες. Στην εργασία για την μέτρηση υγρασίας και θερμοκρασίας χρησιμοποιήθηκε ο ψηφιακός αισθητήρας ChipCap2 (CC2D) της εταιρείας Amphenol Advanced Sensors. Αυτός επικοινωνεί με τον μικροελεγκτή με μία σύνδεση I<sup>2</sup>C. Η σύνδεση με τους αναλογικούς αισθητήρες (π.χ. CO<sub>2</sub>) έγινε μέσω του ADC και των 8 καναλιών μέτρησης που αυτό διαθέτει.

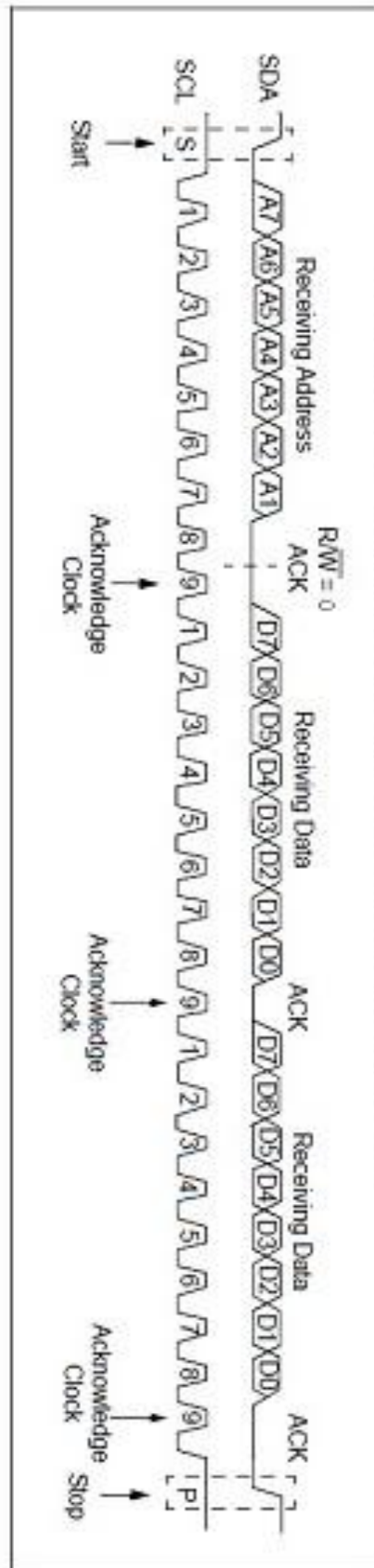
### 3.2.1 Πρωτόκολλο I<sup>2</sup>C

Το I<sup>2</sup>C (Inter-Integrated Circuit) είναι ένα αμφίδρομο σειριακό πρωτόκολλο για την επικοινωνία μικροελεγκτών και μικροεπεξεργαστών με εξωτερικά περιφερειακά που δεν απαιτούν πολύ υψηλές ταχύτητες. Οι ταχύτητες που υποστηρίζει είναι οι Standard-mode (έως 100 Kbps), Fast-mode (έως 400 Kbps), Fast-mode Plus (έως 1 Mbps), High-speed mode (έως 3.4 Mbps) και Ultra Fast-mode (έως 5 Mbps). Στην λειτουργία με την μέγιστη ταχύτητα η επικοινωνία γίνεται μονόδρομη. Ο EFM32WG υποστηρίζει μόνο τις τρεις χαμηλότερες ταχύτητες και ο CC2D μόνο τις δύο πρώτες. Η επικοινωνία τους στα πλαίσια της εργασίας έγινε με τη Standard-mode. Δεν υπήρχε ανάγκη μεγαλύτερης ταχύτητας γιατί η μεταβολή στις μετρήσεις του αισθητήρα είναι σχετικά αργή. Το κανάλι I<sup>2</sup>C αποτελείται από μόνο δύο γραμμές, τη σειριακή γραμμή δεδομένων (serial data line, SDA) για την μετάδοση των bits και τη σειριακή γραμμή ρολογιού (serial clock line, SCL) για τον χρονισμό.

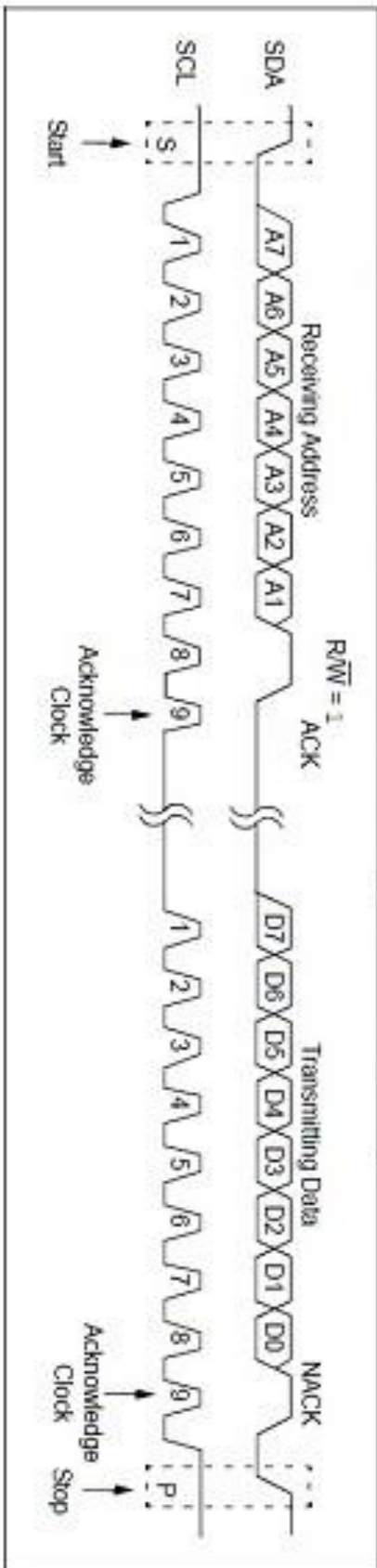
Το I<sup>2</sup>C είναι πρωτόκολλο τύπου master-slave όπου ο master ξεκινάει και τερματίζει τις μεταφορές δεδομένων και παράγει τα σήματα του ρολογιού. Ο slave μπορεί να επέμβει στον χρονισμό (στην γραμμή SCL) μόνο για να τον διακόψει (clock stretching) ώστε να κερδίσει χρόνο για εκτελέσεις άλλων εργασιών. Εκτός από πολλούς slaves στο κανάλι μπορούν να είναι συνδεδεμένοι και πολλοί masters. Λόγω αυτού και για την αποφυγή συγκρούσεων το πρωτόκολλο εφαρμόζει τεχνικές διαιτησίας (arbitration) και συγχρονισμού ρολογιών μεταξύ των masters. Κάθε συσκευή (master ή slave) που συνδέεται στο κανάλι έχει μια διεύθυνση. Οι διευθύνσεις αποδίδονται από έναν master και είναι 7 ή 10 bit. Τα δεδομένα μεταφέρονται σε μορφή byte με μετάδοση πρώτα του πιο σημαντικού ψηφίου. Στην αρχή κάθε μεταφοράς δεδομένων ο master στέλνει ένα byte με τη διεύθυνση του slave και ένα bit που ενημερώνει αν θα είναι ο πομπός (Write) ή ο δέκτης (Read) της πληροφορίας. Στην περίπτωση διευθύνσεων 10 bit χρησιμοποιεί ένα byte ακόμα για την διαδικασία αυτή.

Τα βασικά γεγονότα στην επικοινωνία I<sup>2</sup>C και οι αντίστοιχες μεταβολές στην τάση των γραμμών SDA και SCL φαίνονται στην εικόνα 3.3.

### TYPICAL I<sup>2</sup>C™ WRITE TRANSMISSION (7-BIT ADDRESS)



### TYPICAL I<sup>2</sup>C™ READ TRANSMISSION (7-BIT ADDRESS)

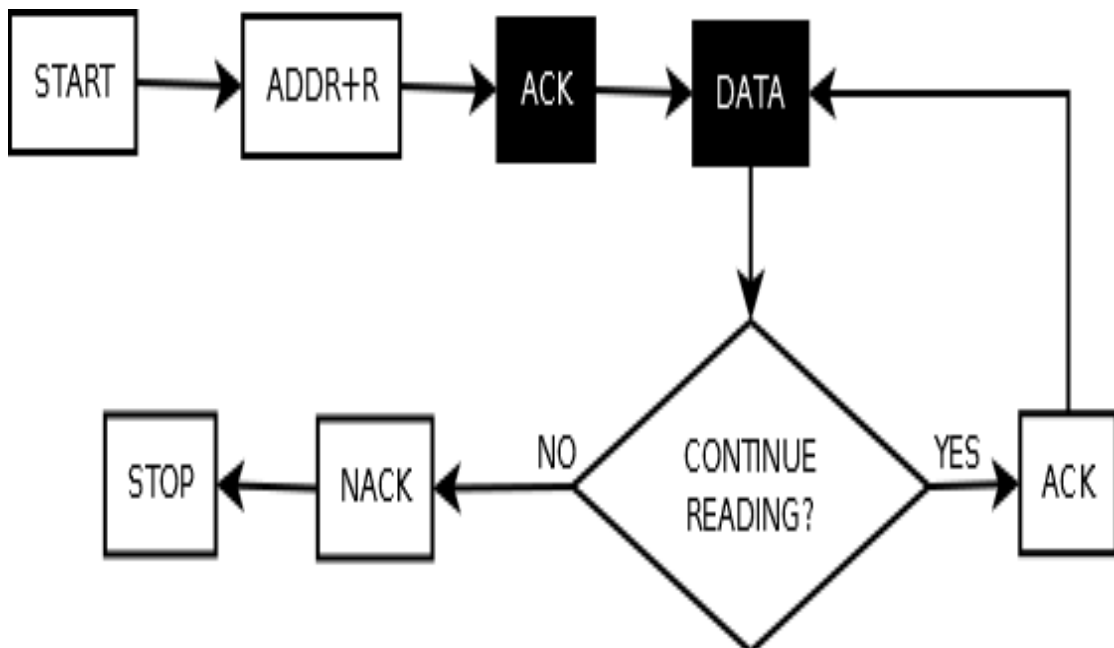


Εικόνα 3.3 Μεταβολές SCL, SDA κατά τη λειτουργία I<sup>2</sup>C

Αυτά είναι τα εξής:

- **START:** Η εκκίνηση μιας μεταφοράς δεδομένων. Ορίζεται από τον master με μετάβαση της SDA χαμηλά ενώ η SCL είναι ψηλά.
- **STOP :** Ο τερματισμός μιας μεταφοράς από τον master με μετάβαση της SDA ψηλά ενώ η SCL είναι ψηλά.
- **DATA :** Η μετάδοση ενός byte στο κανάλι. Μετά το START ξεκινάνε οι παλμοί του ρολογιού (γραμμή SCL). Κάθε bit είναι έγκυρο όταν η SCL είναι ψηλά και επιτρέπεται να αλλάξει τιμή όταν αυτή είναι χαμηλά.
- **ADDR + W/R:** Η μετάδοση από τον master της διεύθυνσης του slave με τον οποίο θα επικοινωνήσει μαζί με το Write bit (0) ή το Read bit (1).
- **ACK:** Η επιβεβαίωση της λήψης ενός byte από τον δέκτη ή η απάντηση στην αποστολή μιας διεύθυνσης από τον slave που του ανήκει. Μετά την μετάδοση ενός byte ο πομπός αφήνει την SDA έτσι ώστε ο δέκτης να επιβεβαιώσει οδηγώντας την χαμηλά.
- **NACK:** Η μη επιβεβαίωση της λήψης ενός byte γίνεται με το να κρατήσει ψηλά ο δέκτης την SDA. Χρησιμοποιείται από τον master και πριν το STOP στην περίπτωση που είναι δέκτης των δεδομένων.

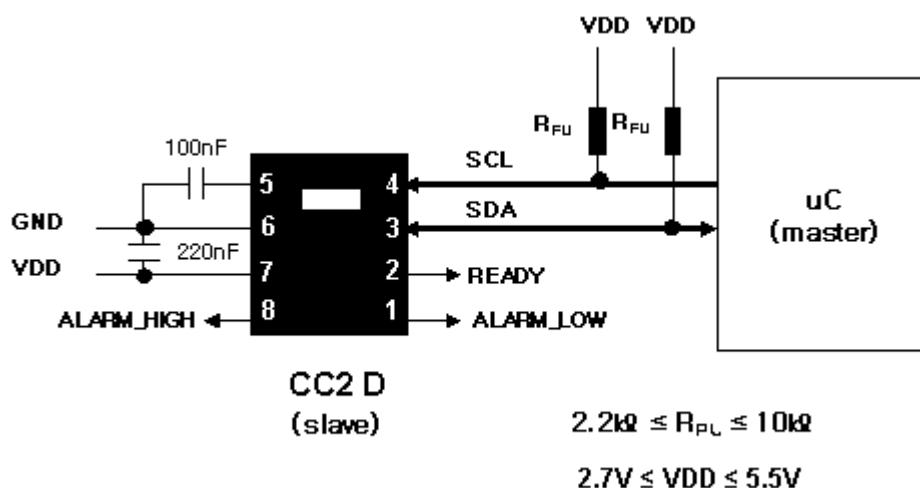
Στο σχήμα 3.1 παρουσιάζεται η λειτουργία του I<sup>2</sup>C στην περίπτωση που ο master δέχεται δεδομένα (με άσπρο πλαίσιο οι ενέργειες του master και με μαύρο του slave). Αυτή είναι και η περίπτωση της εργασίας όπου ο μικροελεγκτής (master) λαμβάνει τις μετρήσεις από τον αισθητήρα (slave).



Σχήμα 3.1 Λειτουργία I<sup>2</sup>C με τον master να δέχεται δεδομένα

### 3.2.2 Αισθητήρας CC2D

Η εικόνα 3.4 παρουσιάζει τον αισθητήρα CC2D με τις εισόδους/εξόδους του και την σύνδεση του με τον μικροελεγκτή.



Εικόνα 3.4 Αισθητήρας ChipCap2

Τα pins 3 (SDA) και 4 (SCL) χρησιμοποιούνται για την επικοινωνία I<sup>2</sup>C, το 5 είναι η τάση πυρήνα (V<sub>CORE</sub>), το 6 η γείωση (GND), το 7 η τάση τροφοδοσίας (VDD). Το 2 (READY) σηματοδοτεί την ύπαρξη νέας μέτρησης (εξηγείται παρακάτω) και τα pins 1 (ALARM\_LOW), 8 (ALARM\_HIGH) μπορούν να χρησιμοποιηθούν για έλεγχο αν η υγρασία περάσει κάποιες ακραίες τιμές.

Ο CC2D μπορεί να βρίσκεται σε δύο καταστάσεις κατά την λειτουργία του, την κατάσταση εντολών (Command Mode) και την κατάσταση μετρήσεων (Measurement Mode). Η Measurement Mode μπορεί να είναι είτε Update Mode είτε Sleep Mode. Στον CC2D που χρησιμοποιήθηκε είναι προεπιλεγμένη η Update Mode, δηλαδή μόλις ο αισθητήρας ολοκληρώσει μία μέτρηση ξεκινάει την επόμενη. Αντίθετα στη Sleep Mode ο αισθητήρας μετράει μόνο αν λάβει εντολή με στόχο την χαμηλότερη κατανάλωση ενέργειας.

Ο CC2D αφού ξεκινήσει να λειτουργεί εισέρχεται στο παράθυρο εντολών (Command Window). Αυτό είναι ένα χρονικό διάστημα 10ms (ή 3ms αν έχει επιλεγεί Fast Startup) στο οποίο ο CC2D μπορεί να λάβει την εντολή Start\_CM να μπει στη Command Mode, αλλιώς αν αυτό λήξει ή λάβει την εντολή Start\_NOM μπαίνει στη Update Mode.

Η Command Mode χρησιμοποιείται για την αναγνώριση ή αλλαγή ρυθμίσεων του CC2D (π.χ. αλλαγή διεύθυνσης I<sup>2</sup>C) μέσω της ανάγνωσης ή εγγραφής μιας μνήμης EEPROM που αυτός περιέχει. Κάθε εντολή της Command Mode αποτελείται από την εντολή Write του I<sup>2</sup>C, το byte-αναγνωριστικό της εντολής και δύο bytes με τα δεδομένα. Στον πίνακα 3.1 φαίνονται οι εντολές της Command Mode και η περιγραφή καθεμιάς.

Command byte (Hex)	Περιγραφή
0x16 έως 0x1F	Ανάγνωση θέσης μνήμης 0x16 έως 0x1F
0x56 έως 0x5F	Εγγραφή θέσης μνήμης 0x16 έως 0x1F
0x80	Start_NOM
0xA0	Start_CM

**Πίνακας 3.1**

Μετά από κάθε εντολή Command Mode μπορεί να δοθεί η εντολή Data Fetch (DF) που είναι η εντολή Read του I<sup>2</sup>C. Εκτός αν έχει δοθεί εντολή ανάγνωσης θέσης μνήμης η DF χρειάζεται να διαβάσει μόνο ένα byte, την κατάσταση απόκρισης της εντολής (πληροφορεί αν εκτελέστηκε η εντολή ή αν έγινε κάποιο λάθος), αλλιώς διαβάζει τρία bytes με τα δύο τελευταία να είναι τα δεδομένα της μνήμης.

Η DF είναι η μόνη εντολή της Update Mode και τα δύο πιο σημαντικά bits του πρώτου byte της απάντησης έχουν κοινή σημασία και για τις δύο Modes η οποία φαίνεται στον πίνακα 3.2.

Bits	Περιγραφή
00	Νέα μέτρηση
01	Μέτρηση που έχει ήδη αναγνωστεί
10	Ο CC2D βρίσκεται στην CommandMode
11	Δεν χρησιμοποιείται

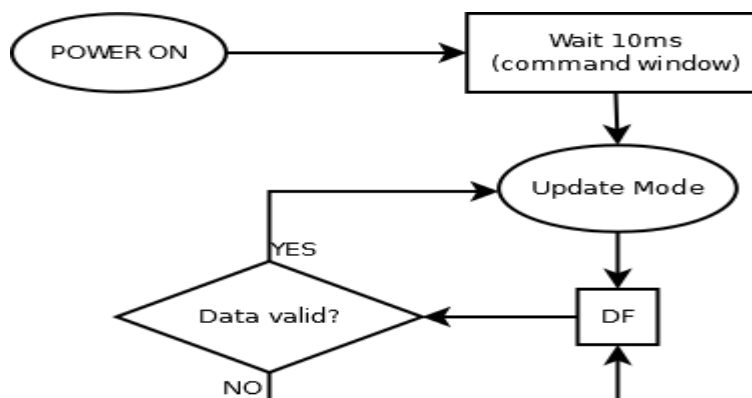
**Πίνακας 3.2**

Η Update Mode χρησιμοποιεί μόνο την εντολή DF και μπορεί να διαβάσει από δύο έως τέσσερα bytes. Τα πρώτα δύο bytes εκτός από τα δύο σημαντικότερα bits (πίνακας 3.2) είναι η μέτρηση της υγρασίας (Hum\_high και Hum\_low) και τα επόμενα δύο bytes εκτός από τα δύο λιγότερο σημαντικά bits είναι η μέτρηση της θερμοκρασίας (Temp\_high και Temp\_low). Οι εξισώσεις μετατροπής που υπολογίζουν τα πραγματικά μεγέθη είναι οι εξής:

$$\text{Σχετική υγρασία (\%)} = ((Hum_{high} \times 256 + Hum_{low}) / 2^{14}) \times 100$$

$$\text{Θερμοκρασία (}^{\circ}\text{C)} = ((Temp_{high} \times 64 + Temp_{low} / 4) / 2^{14}) \times 165 - 40$$

Το READY pin μεταβαίνει σε υψηλή τάση όταν υπάρχει νέα μέτρηση στην Update Mode ή έχει ολοκληρωθεί η εκτέλεση μιας εντολής στην Command Mode. Το λογισμικό του μικροελεγκτή μπορεί να το παρακολουθεί και να δημιουργεί ένα interrupt όταν συμβεί ένα από αυτά τα γεγονότα βελτιώνοντας έτσι την απόδοση του κώδικα. Η λειτουργία του CC2D φαίνεται στο σχήμα 3.2.



**Σχήμα 3.2 Λειτουργία CC2D**

### 3.3 Προγραμματισμός μικροελεγκτή

Το πρόγραμμα του μικροελεγκτή επιλέχθηκε να δομηθεί γύρω από την επικοινωνία με το tablet μέσω του USB. Αυτό έγινε γιατί ο χειρισμός της συνολικής κατασκευής, δηλαδή η επιλογή αισθητήρων προς παρακολούθηση και ο ορισμός παραμέτρων λειτουργίας τους, αν χρειάζονται, γίνεται από το tablet.

#### 3.3.1 Προγραμματισμός USB

Η βιβλιοθήκη emlib προσφέρει ένα API για την απλοποίηση του προγραμματισμού της στοιβάς USB στον μικροελεγκτή είτε προγραμματίζεται ως “host” είτε ως “device”. Βασικό στοιχείο της υλοποίησης του USB στον μικροελεγκτή είναι το αρχείο επικεφαλίδα usbconfig.h. Μέσα σε αυτό ορίζεται αν ο προγραμματισμός αφορά λειτουργία “host” ή “device”. Στην δεύτερη περίπτωση, που αφορά την παρούσα εργασία, ορίζονται έπειτα ο αριθμός των “endpoints” που θα χρησιμοποιηθούν, εκτός του “endpoint” 0, και ποια είναι αυτά. Στην υλοποίηση USB του μικροελεγκτή υπάρχουν έξι ζεύγη “endpoints” που μπορούν να χρησιμοποιηθούν. Επιπλέον στο usbconfig.h ορίζεται ο αριθμός των timers που θα χρειαστεί η στοιβά USB και ποιοί θα είναι αυτοί από τους τέσσερις που υπάρχουν συνολικά στον μικροελεγκτή. Το τελευταίο είναι πολύ σημαντικό γιατί με τη δήλωση ενός timer στο usbconfig.h αυτός ρυθμίζεται αυτόματα, με την εκκίνηση του USB, χωρίς ανάγκη επιπλέον κώδικα από τον προγραμματιστή. Επιπλέον με τις συναρτήσεις που προσφέρει το USB API για τους timers αυτούς μπορεί να εισαχθούν καθυστερήσεις στον κώδικα με διάκριση millisecond ή microsecond. Στο usbconfig.h μπορούν ακόμη να οριστούν λειτουργίες debugging για αποστολή απλών χαρακτήρων ή και strings στη θύρα debug του μικροελεγκτή. Επιπλέον μπορούν να οριστούν λειτουργίες εξοικονόμησης ενέργειας που προσφέρει το USB API για περιπτώσεις όπου το USB δεν συνδέεται σε κάποιο host ή βρίσκεται σε κατάσταση Suspend .

Οι “descriptors” του “device” ορίζονται με δομές ,σε γλώσσα C, που προσφέρει η emlib. Ο Device Descriptor δηλώνεται ξεχωριστά με την δομή USB\_DeviceDescriptor\_TypeDef ενώ οι υπόλοιποι “descriptors” με τις αντίστοιχες δομές τους τοποθετούνται σε έναν πίνακα bytes με την σειρά σύμφωνα με την ιεραρχία που ορίζει το πρωτόκολλο USB. Επίσης οι δηλώσεις αυτές τοποθετούνται στο αρχείο επικεφαλίδα descriptors.h. Εκεί δηλώνεται και ο πίνακας bufferingMultiplier που ορίζει το μέγεθος buffer που θα χρειαστεί κάθε “endpoint”.

Η λειτουργία του USB προκαλεί διάφορα interrupts που εξυπηρετούνται από ρουτίνες ανάκλησης (callbacks). Οι ρουτίνες αυτές μπορούν να χωριστούν σε τρεις κατηγορίες.

1. Ρουτίνες που ορίζονται ,προαιρετικά, στην στοιβά USB (DeviceInitCallbacks) και καλούνται όταν συμβεί ένα από τα παρακάτω γεγονότα:
  - USBReset: Καλείται κάθε φορά που γίνεται επανεκκίνηση του USB χωρίς παραμέτρους.
  - SOF: Καλείται στην αρχή κάθε frame του καναλιού του USB με παράμετρο τον αριθμό του παραθύρου (μέχρι τον αριθμό 16383, μετά αρχίζει ξανά από το 0).
  - USBStatechange: Καλείται όταν αλλάζει η κατάσταση του “device” με παραμέτρους τα ονόματα της προηγούμενης και της καινούργιας κατάστασης.
  - IsSelfpowered: Καλείται όταν ο host στέλνει ένα USB request για να μάθει αν το “device” τροφοδοτείται από το κανάλι του USB ή όχι (δεν έχει παραμέτρους).
  - Setupcommand: Καλείται κάθε φορά που ο “host” στέλνει εντολή Setup με

παράμετρο τα περιεχόμενα της εντολής. Αυτή η συνάρτηση callback μπορεί να χρησιμοποιηθεί για επέκταση ή αντικατάσταση από τον προγραμματιστή ενός τύπου USB request από αυτά που ορίζονται στο πρωτόκολλο USB.

2. Ρουτίνες που καλούνται ,προαιρετικά, με την ολοκλήρωση μιας “transfer”, είτε IN είτε OUT. Καλούνται με τρεις παραμέτρους, την κατάσταση της μεταφοράς (αν ήταν επιτυχημένη ή όχι και τον λόγο αποτυχίας στη δεύτερη περίπτωση), τον αριθμό bytes που μεταφέρθηκαν και τον αριθμό bytes που απομένουν στο buffer του “endpoint”.

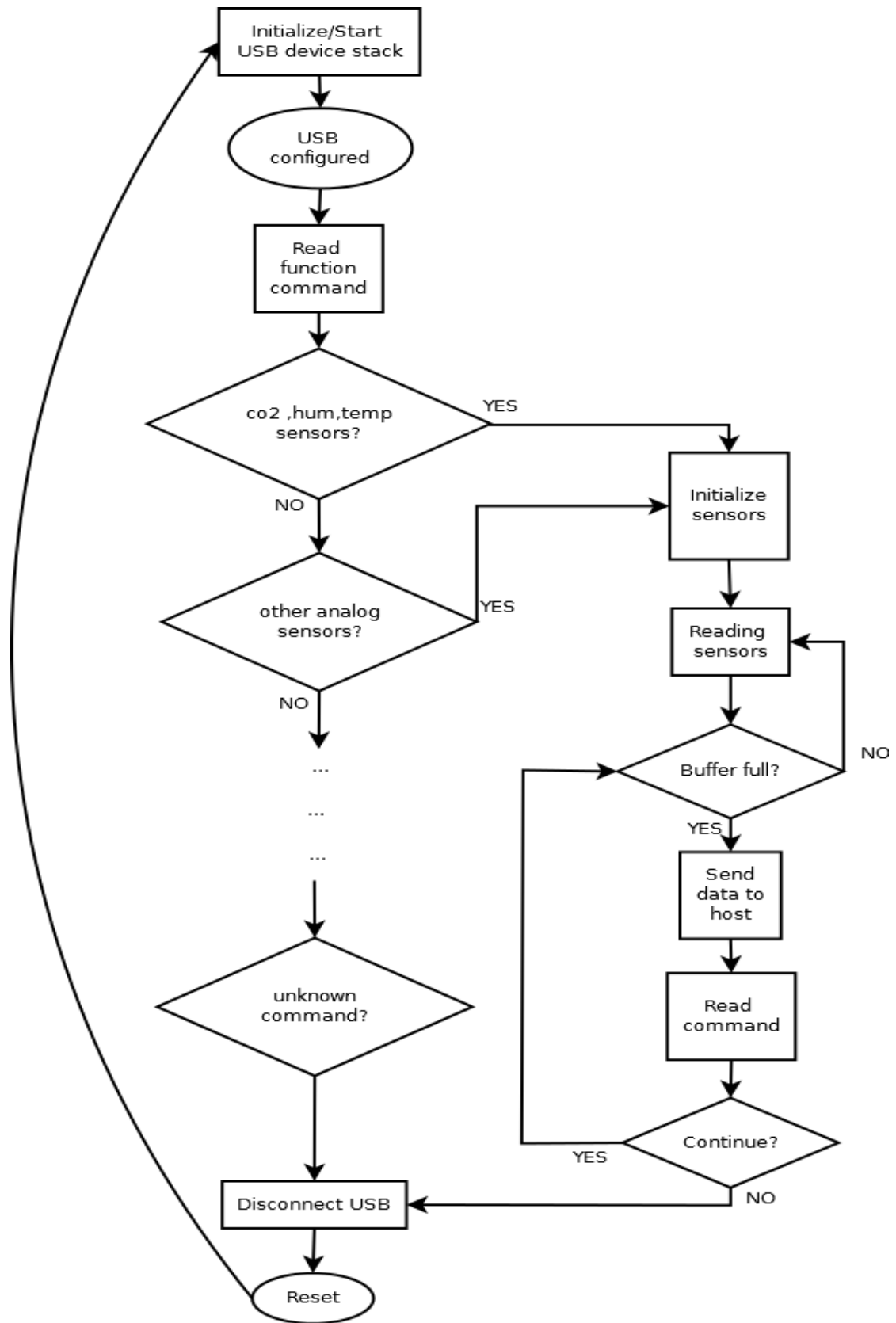
3. Ρουτίνες που ορίζονται σαν παράμετρο callback στην συνάρτηση USBTIMER\_Start(). Αυτή η συνάρτηση εκκινεί έναν timer, ορισμένο στο usbconfig.h, και μετά από κάποιο χρονικό διάστημα, που έχει δοθεί σαν παράμετρο αυτής σε milliseconds, καλεί την ρουτίνα callback.

Οι “descriptors” που έχουν οριστεί και οι ρουτίνες callback της στοίβας USB που έχουν επιλεχθεί να χρησιμοποιηθούν ορίζονται στην δομή USBD\_Init\_TypeDef που περιγράφει τα χαρακτηριστικά της συγκεκριμένης στοίβας USB. Η δομή αυτή εισάγεται σαν παράμετρος στη συνάρτηση USBD\_Init η οποία εκκινεί τη λειτουργία του USB. Άλλες συναρτήσεις που χρησιμοποιούνται είναι οι USBD\_Read (για OUT transfer), USBD\_Write ( για IN transfer), USBTIMER\_DelayMs (εισάγει καθυστέρηση σε milliseconds ) και η USBD\_Disconnect για τον τερματισμό της σύνδεσης USB.

Στο σχήμα 3.3 παρουσιάζεται η λειτουργία του USB η οποία υλοποιήθηκε. Με ορθογώνια περιγράφονται εντολές ή ομάδες εντολών που πραγματοποιούν μια ενέργεια, με ρόμβους αλλαγές στην ροή βάσει εντολών ελέγχου, και με ελλείψεις παρουσιάζονται γεγονότα που συμβαίνουν κατά την εκτέλεση του προγράμματος.

Η στοίβα USB ξεκινάει έχοντας οριστεί ως ρουτίνα callback η Statechange. Μόλις ο “host” αναγνωρίσει το “device” και εγκαθιδρύσει σύνδεση, καλείται αυτή η ρουτίνα η οποία διαβάζει εντολή από τον πρώτο (OUT transfer). Αν είναι το byte 1, επιλέγεται η λειτουργία των αισθητήρων υγρασίας, θερμοκρασίας και CO<sub>2</sub>. Αν είναι το byte 2, επιλέγεται η ανάγνωση κάποιων αναλογικών αισθητήρων. Οι αισθητήρες που επιλέγονται καθορίζονται από τα bytes που ακολουθούν την εντολή. Αυτά είναι 16 bytes (2 για κάθε κανάλι του ADC). Ανά 2 bytes αν ο αριθμός είναι 0, δεν θα ενεργοποιηθεί το συγκεκριμένο κανάλι. Αλλιώς θα λειτουργήσει με περίοδο δειγματοληψίας 1ms επί του αριθμού αυτού. Ύστερα λαμβάνονται μετρήσεις από τους αισθητήρες. Κάθε buffer των “endpoints” είναι 64 bytes (μέγιστο μέγεθος για μία bulk transfer ) και κάθε μέτρηση 4 bytes, επομένως λαμβάνονται 16 μετρήσεις. Αν γεμίσουν τα buffers των “endpoints”, οι τιμές στέλνονται στον “host” (IN transfers) και διαβάζεται πάλι εντολή από αυτόν. Η τελευταία μεταφορά ορίζει συνάρτηση callback στην οποία ελέγχεται αν ο “host” έστειλε το ίδιο byte-εντολή για συνέχιση των μετρήσεων ή έστειλε το byte 3 για τερματισμό. Στην δεύτερη περίπτωση το USB αποσυνδέεται, και όταν συμβεί επανεκκίνηση του μικροελεγκτή το πρόγραμμα ξεκινάει από την αρχή.

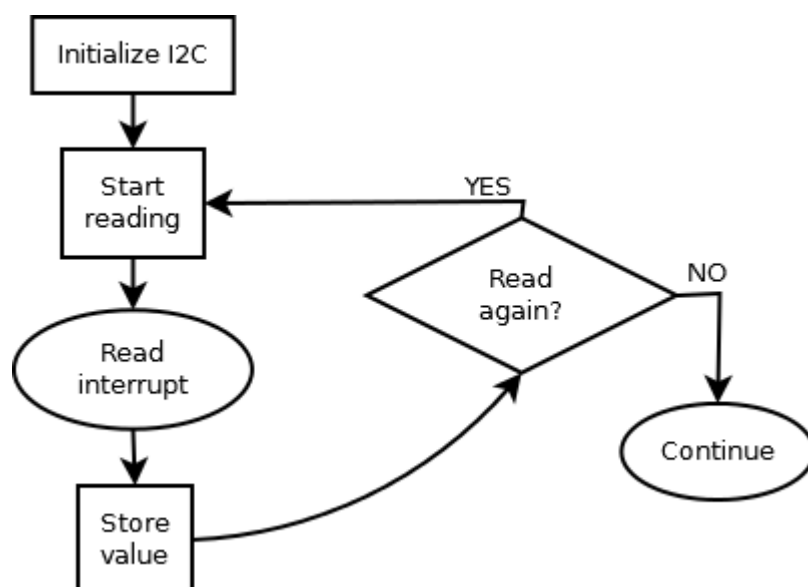
Η επανεκκίνηση γίνεται είτε εξωτερικά με χρήση του RESET pin (στο STK3800 υπάρχει κουμπί που συνδέεται με αυτό) είτε μέσω του λογισμικού. Η τελευταία περίπτωση μπορεί, για παράδειγμα, να υλοποιηθεί με χρήση του Watchdog timer στον οποίο ορίζεται ένα χρονικό διάστημα, και αφού ενεργοποιηθεί, προκαλεί την επανεκκίνηση του μικροελεγκτή μετά το πέρας αυτού.



Σχήμα 3.3 Λειτουργία USB στο πρόγραμμα του μικροελεγκτή

### 3.3.2 Προγραμματισμός ανάγνωσης αισθητήρα I<sup>2</sup>C

Η προετοιμασία της σύνδεσης I<sup>2</sup>C απαιτεί αρχικά την ενεργοποίηση των ρολογιών των περιφερειακών υψηλής συχνότητας, του I<sup>2</sup>C και του GPIO, τον ορισμό των δύο pins που θα λειτουργούν ως SCL και SDA και την ενεργοποίηση των interrupts του I<sup>2</sup>C. Γενικά στοιχεία της σύνδεσης, όπως η επιλογή της ταχύτητας Standard-Mode και η επιλογή να είναι master ο μικροελεγκτής, ορίζονται με τη χρήση της δομής I2C\_Init\_Typedef της emlib. Τα χαρακτηριστικά της μέτρησης από τον αισθητήρα ορίζονται με τη δομή I2C\_TransferSeq\_Typedef. Μέσα σε αυτή ορίζονται η διεύθυνση I<sup>2</sup>C του αισθητήρα, αν ο μικροελεγκτής θα δεχτεί ή θα στείλει δεδομένα, ο αριθμός bytes που θα μεταφερθούν και ένας χώρος μνήμης (π.χ. ένας πίνακας byte) που θα αποθηκευτούν. Στο σχήμα 3.4 φαίνεται η ροή του κώδικα που υλοποιεί την επικοινωνία με τον αισθητήρα I<sup>2</sup>C.



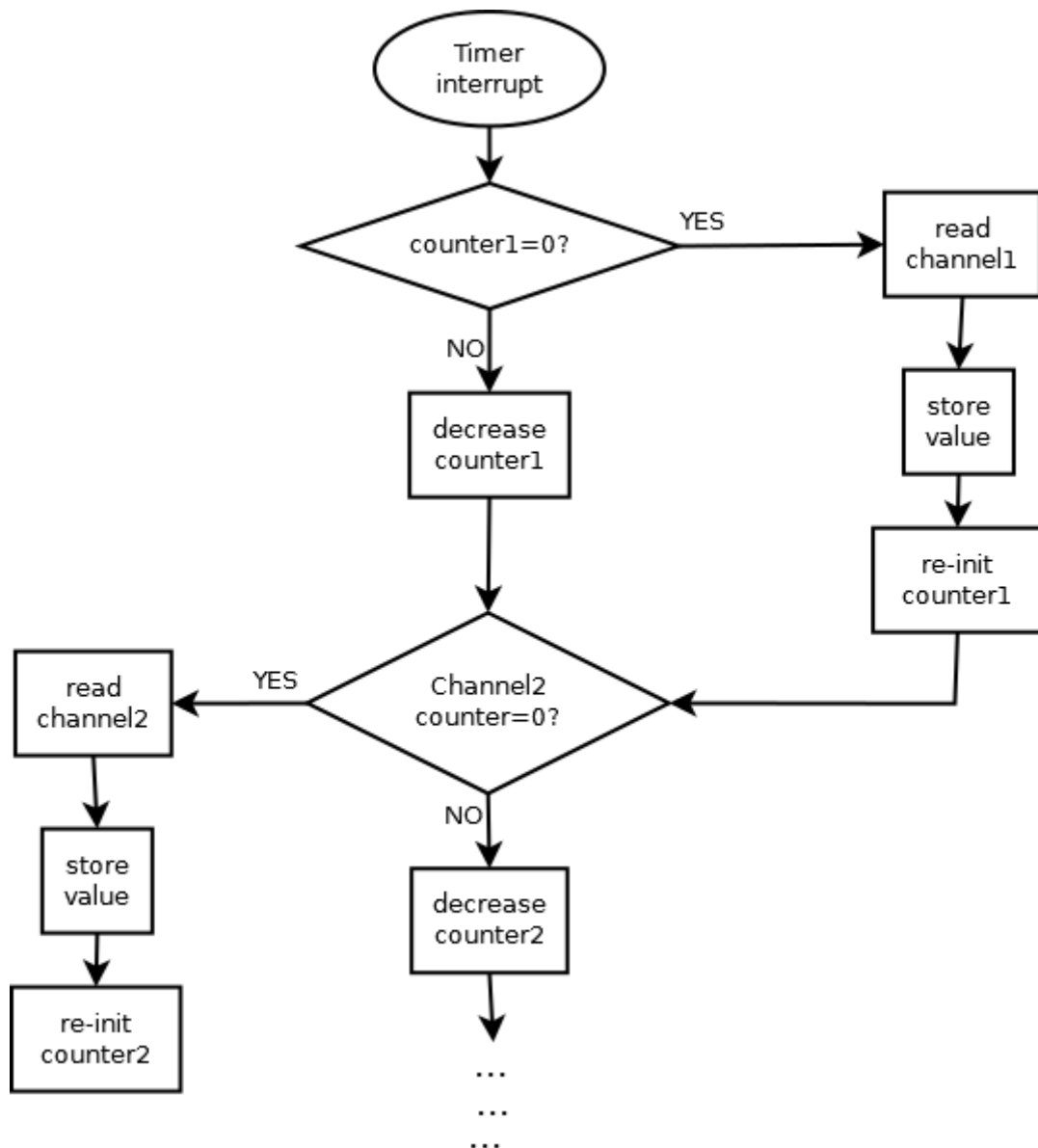
Σχήμα 3.4 Λειτουργία προγράμματος ανάγνωσης αισθητήρα I<sup>2</sup>C

### 3.3.3 Προγραμματισμός ανάγνωσης αναλογικών αισθητήρων

Η προετοιμασία του ADC αρχίζει με την ενεργοποίηση του ρολογιού του. Έπειτα με τη χρήση της δομής ADC\_Init\_Typedef ορίζονται τα γενικά χαρακτηριστικά του ADC (ταχύτητα μετατροπής κτλ). Με τη χρήση της δομής ADC\_InitSingle\_Typedef ορίζονται τα ιδιαίτερα χαρακτηριστικά κάθε καναλιού (τάση αναφοράς, μέγεθος ανάλυσης σε bit κτλ). Επιπλέον ενεργοποιούνται τα interrupts του ADC και συγκεκριμένα αυτό που δημιουργείται με την ολοκλήρωση μιας μέτρησης.

Στην εργασία, για τον συγχρονισμό της δειγματοληψίας χρησιμοποιείται ένας timer που δημιουργεί interrupt κάθε 1millisecond. Η περίοδος δειγματοληψίας κάθε καναλιού ADC, σε millisecond, ορίζεται σε μια ακέραια μεταβλητή που λειτουργεί ως μετρητής του καναλιού. Κάθε φορά που ο timer δημιουργεί interrupt ελέγχεται ο μετρητής κάθε καναλιού. Αν είναι 0 πραγματοποιείται μέτρηση που αποθηκεύεται στο buffer και ο μετρητής επιστρέφει στην αρχική του τιμή. Αν είναι μεγαλύτερος του 0, μειώνεται κατά 1 και το πρόγραμμα προχωράει

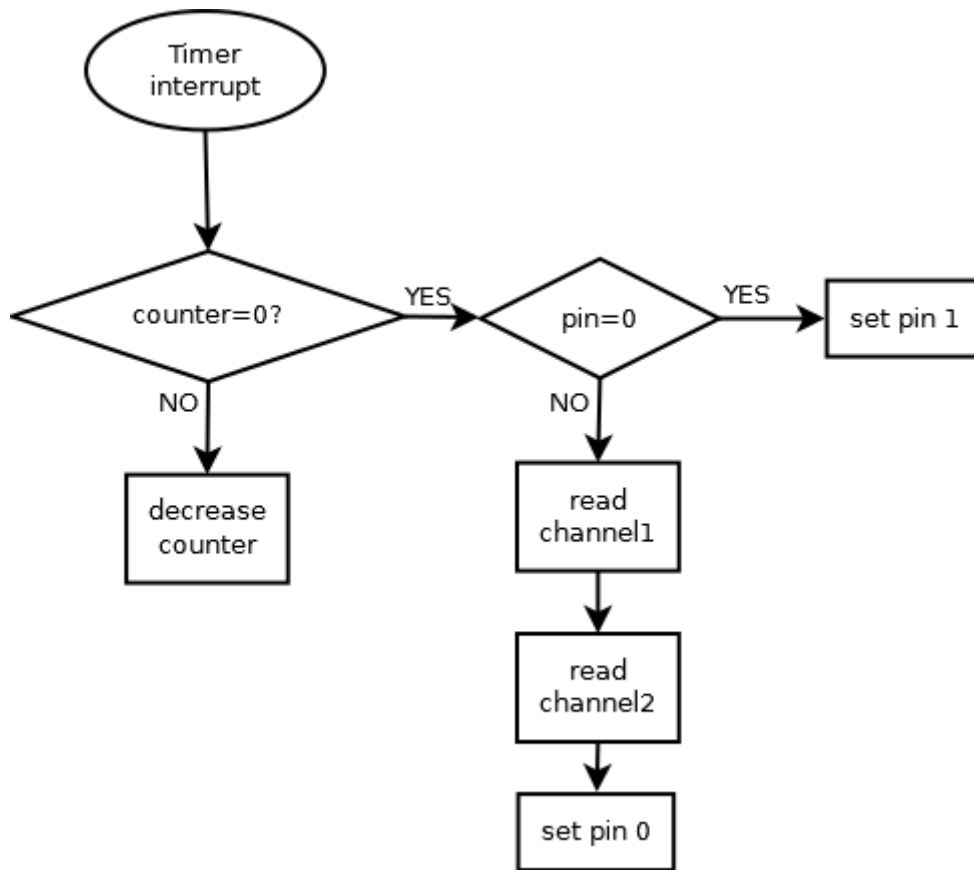
στον έλεγχο του μετρητή του επόμενου καναλιού χρησιμοποιείται. Η λειτουργία αυτή παρουσιάζεται στο σχήμα 3.5.



Σχήμα 3.5 Μέτρηση αναλογικών αισθητήρων

### Προγραμματισμός ανάγνωσης αναλογικού αισθητήρα CO<sub>2</sub>

Ο αισθητήρας CO<sub>2</sub> χρειάζεται δύο κανάλια ADC. Επίσης χρειάζεται ένα pin που δημιουργεί ένα περιοδικό σήμα τετραγωνικών παλμών συχνότητας 4 Hz (125 ms ON , 125 ms OFF). Οι μετρήσεις των δύο καναλιών συμβαίνουν πριν το pin κλείσει. Η διαδικασία της μέτρησης στην περίπτωση αυτή χρησιμοποιεί έναν μετρητή με αρχική τιμή 125. Όταν σε κάποιο interrupt του timer μηδενιστεί, οι μετρήσεις γίνονται αν το pin είναι ανοικτό, και στη συνέχεια αυτό κλείνει. Αν το pin είναι κλειστό ανοίγει. Η λειτουργία αυτή φαίνεται στο σχήμα 3.6.



Σχήμα 3.6 Λειτουργία αισθητήρα CO<sub>2</sub>



## Κεφάλαιο 4

### Χειρισμός και καταγραφή συστήματος μέσω οθόνης αφής

#### 4.1 Γενικά

Αποφασίσαμε να επιλέξουμε συσκευή με λειτουργικό σύστημα Android για πολλούς λόγους. Καταρχάς επιθυμούσαμε να χρησιμοποιήσουμε κάποια φορητή συσκευή, όπως για παράδειγμα smartphone ή tablet, λόγω των σχετικών πλεονεκτημάτων που προσφέρει (φορητότητα, ευκολία χρήσης και μεταφοράς, οθόνη αφής, φιλικότητα προς τον χρήστη, καθώς και δυνατότητα μεταφοράς της εφαρμογής σε άλλες συσκευές με το ίδιο λειτουργικό). Τελικά επιλέχθηκε tablet κυρίως λόγω του μεγέθους της οθόνης που καθιστούσε ευκολότερη και ευκρινέστερη την απεικόνιση των μετρήσεων. Το λειτουργικό σύστημα Android επιλέχθηκε γιατί είναι κατά κύριο λόγο σχεδιασμένο για συσκευές με οθόνη αφής και είναι το πιο ευρέως διαδεδομένο λογισμικό στον κόσμο καθιστώντας με αυτόν τον τρόπο τις εφαρμογές που αναπτύσσονται σε αυτό συμβατές με μια πληθώρα συσκευών.

Στην παρούσα διπλωματική χρησιμοποιήσαμε ένα tablet της εταιρίας Crypto με λειτουργικό σύστημα Android (έκδοση 4.4.2 KitKat). Το tablet διαθέτει οθόνη αφής 10.1” με ανάλυση 1024x600, επεξεργαστή (CPU) Quad Core Cortex A7 στα 1.3 GHz , RAM 1 GB DDR3 και για συνδεσιμότητα WiFi, Bluetooth και USB 2.0 (OTG). Επίσης διαθέτει εξωτερική τροφοδοσία ώστε να τροφοδοτείται ανεξάρτητα και να μπορεί να τροφοδοτεί το κύκλωμα του μικροελεγκτή με τους αισθητήρες.

#### 4.2 Λειτουργικό σύστημα Android

Το **Android** είναι λειτουργικό σύστημα (OS) για συσκευές κινητής τηλεφωνίας το οποίο τρέχει τον πυρήνα (kernel) του λειτουργικού Linux. Αναπτύχθηκε αρχικά από την Google.

Το Android είναι κατά κύριο λόγο σχεδιασμένο για συσκευές με οθόνη αφής, όπως τα έξυπνα τηλέφωνα και τα tablet, με διαφορετικό περιβάλλον χρήσης για τηλεοράσεις (Android TV), αυτοκίνητα (Android Auto) και ρολόγια χειρός (Android Wear). Παρόλο που έχει αναπτυχθεί για συσκευές με οθόνη αφής, έχει χρησιμοποιηθεί σε κονσόλες παιχνιδιών, ψηφιακές φωτογραφικές μηχανές, συνηθισμένους Η/Υ και σε άλλες ηλεκτρονικές συσκευές. Η Google δημοσίευσε το μεγαλύτερο μέρος του κώδικα του Android υπό τους όρους της Apache License, μιας ελεύθερης άδειας λογισμικού. Το λογότυπο για το λειτουργικό σύστημα Android είναι ένα ρομπότ (android).



Εικόνα 4.2. Λογότυπο του λειτουργικού συστήματος Android

Το Android είναι το πιο ευρέως διαδεδομένο λογισμικό στον κόσμο και έχει τη μεγαλύτερη εγκαταστημένη βάση από όλα τα λειτουργικά συστήματα. Οι συσκευές με Android έχουν περισσότερες πωλήσεις από όλες τις συσκευές Windows, iOS και Mac OS X μαζί.

Η ανάπτυξη λογισμικού γίνεται με τη χρήση της γλώσσας προγραμματισμού Java, και οι κατασκευαστές μπορούν να ελέγχουν την συσκευή μέσω βιβλιοθηκών λογισμικού ανεπτυγμένων από την Google.

Το Android είναι δημοφιλές σε εταιρίες τεχνολογίας που επιζητούν ένα έτοιμο, χαμηλού κόστους, τροποποιήσιμο και προσαρμόσιμο λειτουργικό σύστημα για συσκευές υψηλής τεχνολογίας. Η ανοιχτή φύση του έχει ενθαρρύνει μια μεγάλη κοινότητα δημιουργών λογισμικού να χρησιμοποιήσουν τον ανοιχτό κώδικά του για να αναπτύξουν και να διανείμουν τις δικές τους τροποποιημένες εκδόσεις του λειτουργικού συστήματος. Οι εκδόσεις αυτές συχνά φέρνουν καινούργια χαρακτηριστικά και αναβαθμίσεις στις συσκευές γρηγορότερα από τους επίσημους κατασκευαστές.

#### 4.2.1 Αρχιτεκτονική λειτουργικού συστήματος Android

Ο πυρήνας του Android είναι βασισμένος στον **πυρήνα (kernel)** του λειτουργικού συστήματος Linux. Ολόκληρο το λειτουργικό σύστημα Android είναι «χτισμένο» πάνω στον πυρήνα του Linux με κάποιες επιπλέον αρχιτεκτονικές αλλαγές. Οι συσκευές Android χρησιμοποιούν κυρίως τις εκδόσεις 3.4 ή 3.10 του πυρήνα του Linux, αν και η έκδοση του πυρήνα εξαρτάται από την εκάστοτε συσκευή και το hardware.

Ο πυρήνας αλληλεπιδρά με το υλικό (hardware) και περιέχει όλους τους απαραίτητους οδηγούς (hardware drivers). Παρέχει τη βασική λειτουργία του συστήματος όπως είναι η διαχείριση της μνήμης, η διαχείριση της ενέργειας (power management), η διαχείριση διεργασιών (process management) και η δικτύωση (networking). Επιπλέον περιλαμβάνει μια μεγάλη λίστα από οδηγούς συσκευών καθιστώντας έτσι εύκολη την αλληλεπίδραση με διάφορες περιφερειακές συσκευές.

Το επόμενο επίπεδο στην ιεραρχία είναι οι βασικές βιβλιοθήκες (core ή native libraries) που επιτρέπουν στη συσκευή να χειρίζεται διάφορους τύπους δεδομένων. Οι βιβλιοθήκες αυτές είναι γραμμένες στη γλώσσα προγραμματισμού C ή C++. Κάποιες από αυτές είναι οι βιβλιοθήκες *WebKit* (για λειτουργίες web browser), *SQLite* (βιβλιοθήκη βάσης δεδομένων που χρησιμοποιείται για λειτουργίες αποθήκευσης δεδομένων), *OpenGL* (για απεικόνιση γραφικών 2D ή 3D), καθώς και βιβλιοθήκες για αναπαραγωγή ήχου και βίντεο.

Η επόμενη κατηγορία είναι οι **βιβλιοθήκες** του Android που είναι γραμμένες σε Java. Οι βιβλιοθήκες αυτές είναι ειδικές για την ανάπτυξη εφαρμογών και προγραμμάτων σε Android. Παράδειγμα τέτοιων βιβλιοθηκών είναι οι βιβλιοθήκες για το application framework καθώς και αυτές που διευκολύνουν την ανάπτυξη της διεπαφής χρήστη (user interface) και την πρόσβαση σε βάσεις δεδομένων. Κάποιες συγκεκριμένες ονομαστικά είναι οι *android.app* (αποτελεί την βάση όλων των εφαρμογών), *android.content*, *android.database*, *android.graphics* (μια low-level API για σχεδιασμό γραφικών 2D όπως χρώματα, σημεία, κτλ), *android.hardware* (παρέχει πρόσβαση στα διάφορα τμήματα του υλικού), *android.os*, *android.media* (περιλαμβάνει classes για αναπαραγωγή ήχου και βίντεο), *android.util* (βασικά εργαλεία για λειτουργίες όπως είναι ο χειρισμός της ώρας και

της ημερομηνίας, εργαλεία XML κτλ), *android.view* (περιέχει τα θεμελιώδη δομικά στοιχεία για user interfaces) κτλ.

Στο επόμενο επίπεδο βρίσκεται το **περιβάλλον Runtime** του Android, το οποίο περιλαμβάνει ένα από τα σημαντικότερα τμήματα του Android, την **Dalvik Virtual Machine**. Η Dalvik Virtual Machine (DVS) είναι ένα είδος Java Virtual Machine (εικονική μηχανή Java) - ειδικά σχεδιασμένη για το Android και βελτιστοποιημένη για περιβάλλοντα χαμηλής επεξεργαστικής ισχύς και χαμηλής μνήμης - που χρησιμοποιεί χαρακτηριστικά του πυρήνα του Linux, όπως είναι η διαχείριση μνήμης και το multi-threading, τα οποία είναι έμφυτα στη Java. Η DVS επιτρέπει σε κάθε εφαρμογή Android να τρέχει τη δική της διεργασία με το δικό της στιγμιότυπο της DVS. Αναλυτικότερα, η DVS επιτρέπει την ταυτόχρονη δημιουργία πολλαπλών στιγμιότυπων της εικονικής μηχανής παρέχοντας ασφάλεια, απομόνωση, διαχείριση μνήμης και υποστήριξη νημάτων (threading), προσφέροντας με αυτόν τον τρόπο υψηλότερη απόδοση σε περιβάλλοντα με χαμηλούς πόρους.

Στο ίδιο επίπεδο βρίσκονται και οι βασικές βιβλιοθήκες της Java (Core Java Libraries) που δίνουν τη δυνατότητα στους κατασκευαστές εφαρμογών (developers) να γράφουν εφαρμογές Android χρησιμοποιώντας τη γλώσσα προγραμματισμού Java.

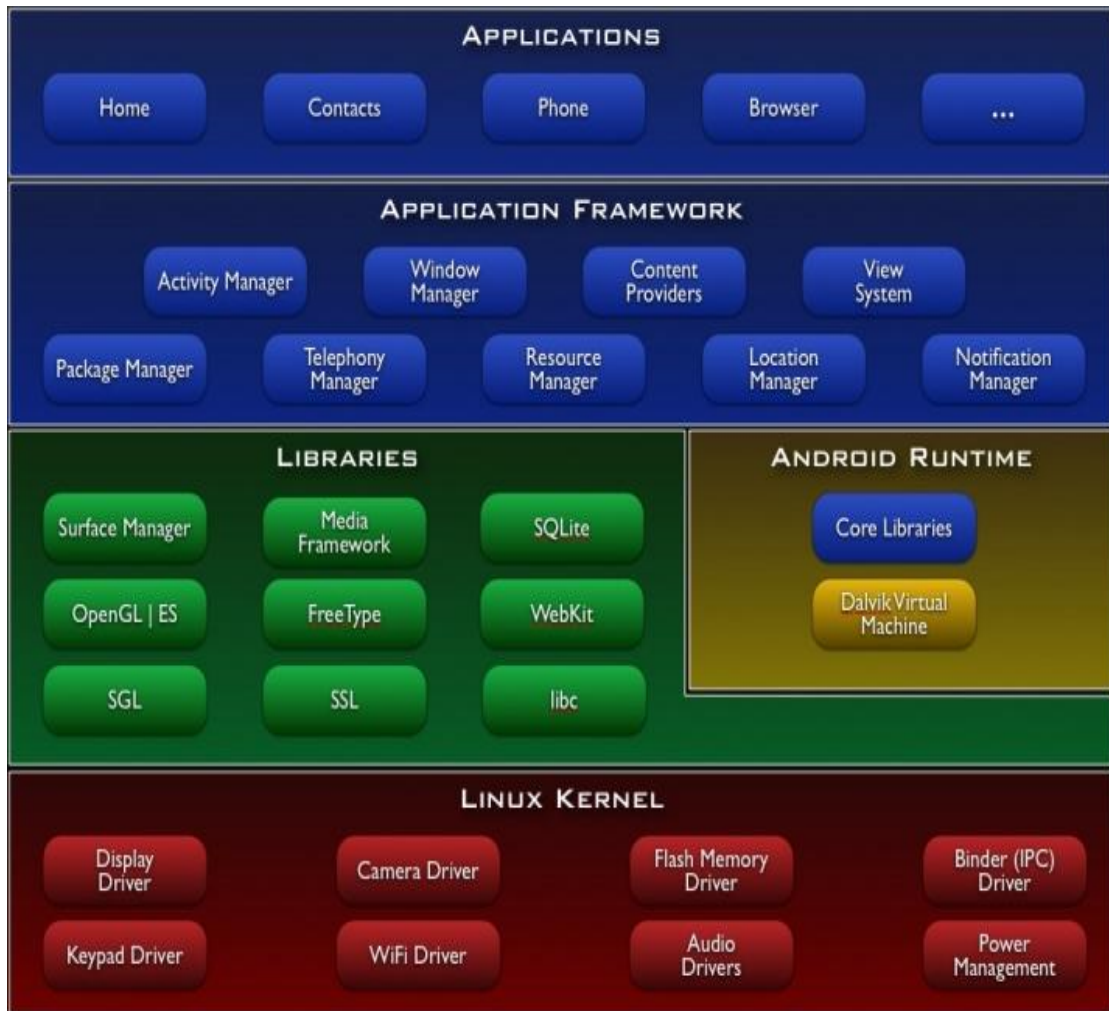
Στο προτελευταίο επίπεδο είναι το **Application Framework** το οποίο είναι ένα σύνολο υπηρεσιών που όλες μαζί σχηματίζουν το περιβάλλον πάνω στο οποίο τρέχουν όλες οι εφαρμογές android. Οι εφαρμογές android αλληλεπιδρούν απευθείας με το application framework, το οποίο θεωρεί ότι οι εφαρμογές είναι δομημένες από επαναχρησιμοποιούμενα και ανταλλάξιμα στοιχεία (components). Τέλος, το framework αυτό χειρίζεται τις βασικές λειτουργίες της συσκευής android όπως είναι για παράδειγμα η διαχείριση των πόρων. Μερικά σημαντικά τμήματα του application framework είναι τα εξής:

- **Activity Manager** : ελέγχει και διαχειρίζεται τον κύκλο ζωής όλων των εφαρμογών
- **Content Providers**: ελέγχει και επιτρέπει στις εφαρμογές να δημοσιεύουν και να διαμοιράζονται δεδομένα μεταξύ τους
- **Resource Manager**: διαχειρίζεται τους διάφορους τύπους πόρων (resources) που χρησιμοποιούνται στις εφαρμογές όπως είναι τα strings, οι ρυθμίσεις χρώματος και τα user interface layouts
- **Notifications Manager**: επιτρέπει στις εφαρμογές να εμφανίζουν ειδοποιήσεις στο χρήστη
- **Telephony Manager**: διαχειρίζεται τις τηλεφωνικές υπηρεσίες και τις φωνητικές κλήσεις
- **View System**: ένα σύνολο από views που χρησιμοποιείται για τη δημιουργία των user interfaces των εφαρμογών
- **Location Manager**: διαχειρίζεται τις υπηρεσίες τοποθεσίας και παρέχει σχετικές πληροφορίες

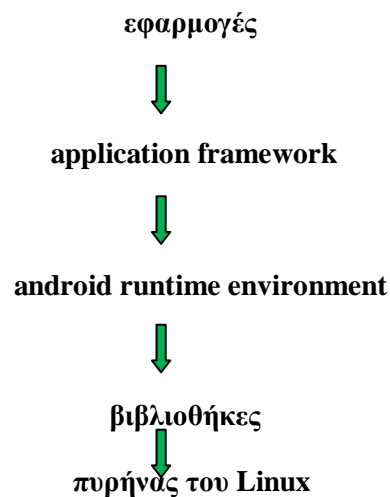
Στην κορυφή της αρχιτεκτονικής του λειτουργικού συστήματος android βρίσκονται οι **εφαρμογές**, τόσο αυτές που είναι προεγκατεστημένες στη συσκευή, (όπως για παράδειγμα η

εφαρμογή για αποστολή και λήψη SMS,η εφαρμογή για πραγματοποίηση κλήσεων κτλ), όσο και αυτές που αναπτύσσονται από τρίτους και μπορεί να εγκαταστήσει μόνος του ο χρήστης.

Συνοπτικά, όλες οι εφαρμογές χρησιμοποιούν το application framework το οποίο με τη σειρά του χρησιμοποιεί το android runtime και τις βιβλιοθήκες. Το περιβάλλον android runtime και οι βιβλιοθήκες χρησιμοποιούν τον πυρήνα του Linux.



Εικόνα 4.3.Διαγραμματική αναπαράσταση της αρχιτεκτονικής του Android



### 4.3 Βιβλιοθήκες για σχεδίαση γραφικών παραστάσεων σε περιβάλλον Android

Η σημαντικότερη και κυριότερη λειτουργία της εφαρμογής που αναπτύχθηκε στα πλαίσια της παρούσας διπλωματικής ήταν η γραφική αναπαράσταση των μετρήσεων των αισθητήρων σε πραγματικό χρόνο. Η ίδια η Google δεν παρέχει κάποια έτοιμη βιβλιοθήκη για σχεδίαση γραφικών παραστάσεων σε περιβάλλον android. Έτσι αναζητήθηκαν τέτοιες ελεύθερες βιβλιοθήκες ανοιχτού κώδικα (open source), δημιουργημένες από τρίτους, που να μπορούν να χρησιμοποιηθούν για το σκοπό αυτό.

Οι πιο διαδεδομένες και αποτελεσματικές, ελεύθερες και open source βιβλιοθήκες για δημιουργία γραφικών παραστάσεων σε android ήταν –τη στιγμή της συγγραφής της συγκεκριμένης διπλωματικής- οι **AndroidPlot** (<http://androidplot.com>) και **AChartEngine** (<http://www.achartengine.org>). Και οι δύο είναι γρήγορες, ελαφριές, εύκολα τροποποιήσιμες, αποτελεσματικές, έχουν επαρκή υποστήριξη και documentation με παραδείγματα, υποστηρίζουν πολλών ειδών γραφήματα και δυναμική σχεδίαση γραφικών παραστάσεων και είναι συμβατές με τις περισσότερες εκδόσεις του android.

Δοκιμάστηκαν και οι δύο, και παρόλο που προσφέρουν εξίσου σημαντικά πλεονεκτήματα, επιλέχθηκε τελικά η AChartEngine, κυρίως εξαιτίας του ότι κρίθηκε καταλληλότερη για σχεδίαση γραφικών παραστάσεων σε πραγματικό χρόνο(real time), που ήταν και μια από τις απαιτήσεις της εφαρμογής που δημιουργήσαμε. Η AChartEngine περιέχει classes και μεθόδους που διευκολύνουν την real time σχεδίαση καινούργιων μετρήσεων που διαβάζονται από τη συσκευή, όπως είναι για παράδειγμα η class Point που αναπαριστά ένα σημείο (με συντεταγμένες στους άξονες) και η TimeSeries (αναπαριστά μια γραφική παράσταση με οριζόντιο άξονα των χ τον χρόνο ) που περιέχει τη μέθοδο add() η οποία επιτρέπει την προσθήκη στη γραφική παράσταση νέων σημείων. Επίσης διαθέτει εργαλεία για δυνατότητα ζουμ και μετακίνησης της γραφικής παράστασης (scrolling). Με τη βιβλιοθήκη αυτή ,μπορέσαμε λοιπόν να γράψουμε κώδικα που να αναπαριστά δυναμικά τα σημεία καθώς διαβάζονταν οι αντίστοιχες μετρήσεις από τους αισθητήρες και στέλνονταν στη συσκευή μέσω USB.

Στη συνέχεια παρουσιάζονται και περιγράφονται ορισμένες από τις classes της **AChartEngine** που χρησιμοποιήθηκαν στην υλοποίησης της εφαρμογής της παρούσας διπλωματικής. Συγκεκριμένα:

- **GraphicalView** αντιπροσωπεύει το αντικείμενο απεικόνισης View που περικλείει τη γραφική παράσταση και την εμφανίζει στην οθόνη του χρήστη
- **TimeSeries** αντιπροσωπεύει μια σειρά σημείων για γραφήματα χρόνου και περιέχει μεθόδους που μπορούν να επιδρούν στα σημεία της σειράς (πχ να προσθέτει ή να αφαιρεί κάποιο σημείο, να βρίσκει το μέγιστο και το ελάχιστο σημείο της γραφικής παράστασης, να ορίζει τον τίτλο της κτλ.)
- **XYMultipleSeriesDataset** αντιπροσωπεύει μια συνολική σειρά που μπορεί να περιέχει καμία, μία ή περισσότερες σειρές σημείων (χρόνου ή άλλου τύπου). Έχει μεθόδους που επιδρούν στις σειρές που περικλείει όπως για παράδειγμα να προσθέτει ή να αφαιρεί κάποια σειρά.
- **XYSeriesRenderer** αντιπροσωπεύει ένα renderer για κάθε σειρά, που ρυθμίζει τη μορφή και τον τρόπο με τον οποίο σχεδιάζεται και παρουσιάζεται η κάθε

σειρά. Διαθέτει μεθόδους που καθορίζουν και ρυθμίζουν για παράδειγμα το χρώμα και το πάχος της γραμμής, το στυλ σχεδιασμού των σημείων, τα όρια κτλ

➤ **XYMultipleSeriesDataset** δημιουργεί ένα συνολικό renderer για όλη τη γραφική απεικόνιση στο σύνολό της (που περιλαμβάνει όλες τις σειρές) και καθορίζει τη συνολική παρουσίαση των γραφημάτων. Περιλαμβάνει μεθόδους που καθορίζουν τις λεπτομέρειες της γραφικής παρουσίασης της συνολικής σειράς και τη μορφή της απεικόνισης στο σύνολό της όπως για παράδειγμα τη σχεδίαση και τη μορφή των αξόνων, τα περιθώρια, τις διαστάσεις, το εύρος της απεικόνισης, την κλίμακα, τα όρια του ζουμ κτλ

#### 4.4 USB και Android

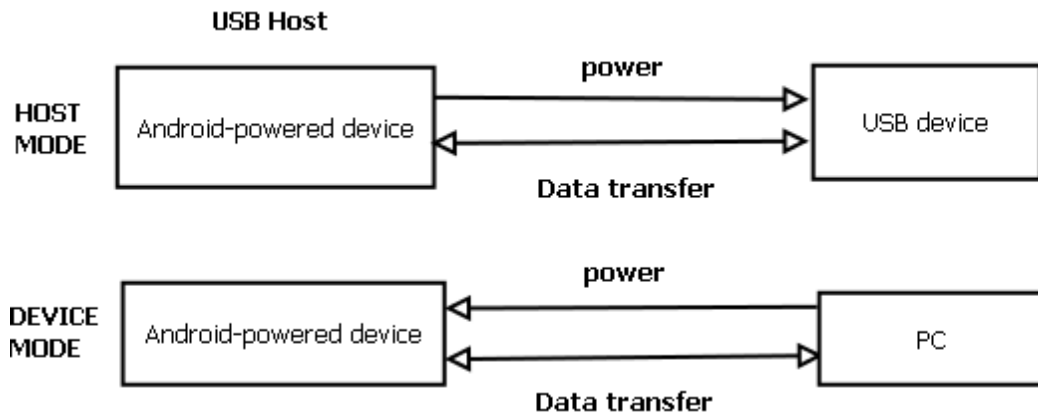
Όσον αφορά το USB μια συσκευή Android μπορεί να λειτουργήσει με δύο τρόπους, είτε ως host είτε ως device. Όταν μια συσκευή USB (device/peripheral) συνδέεται με μία συσκευή Android, τότε η συσκευή Android λειτουργεί ως host και τροφοδοτεί τον δίαυλο και απαριθμεί τις συνδεδεμένες συσκευές. Μια συσκευή Android που λειτουργεί ως host (ή On-The-Go host) πρέπει να παρέχει ισχύ 5V/500mA στη συνδεδεμένη συσκευή αν η τελευταία τροφοδοτείται μέσω USB.

Όταν μία συσκευή Android συνδέεται σε έναν υπολογιστή (που έχει το ρόλο του host) τότε λειτουργεί ως USB device. Στην περίπτωση αυτή η συσκευή Android τροφοδοτείται από τον host (σε περίπτωση που δεν έχει εξωτερική τροφοδοσία).

Το λειτουργικό, δηλαδή, αναλαμβάνει αυτόματα τη ρύθμιση των παραμέτρων και την εγκατάσταση της σύνδεσης USB, κάθε φορά που συνδέεται μια συσκευή USB.

Ξεκινώντας από την έκδοση 3.1 του Android (Honeycomb), μια συσκευή Android διαθέτει μία ξεχωριστή λειτουργία USB, αποκλειστικά για Android, που ονομάζεται **USB accessory**. Στη λειτουργία USB accessory, μία συσκευή Android που λειτουργεί ως device μπορεί να χειρίζεται εξωτερικές συσκευές. Αυτό επιτυγχάνεται συνδέοντας τη συσκευή Android σε μια εξωτερική accessory device η οποία δρα ως USB host. Τότε η συσκευή Android μπαίνει σε λειτουργία USB accessory προκειμένου να είναι σε θέση να χειρίζεται άλλες συσκευές που συνδέονται στην accessory device. Η λειτουργία USB accessory επιτρέπει στους χρήστες να συνδέουν συσκευές Android που δεν μπορούν να λειτουργήσουν ως host, με άλλες συσκευές USB που μπορούν να συμπεριφερθούν ως host, και έτσι να επικοινωνήσουν και να αλληλεπιδράσουν μαζί τους. Ένα παράδειγμα της λειτουργίας USB accessory παρουσιάζεται στην παρακάτω εικόνα.

Οι δύο λειτουργίες –host και device– καθώς και οι διαφορές τους παρουσιάζονται στο παρακάτω σχήμα.



Εικόνα 4.3.USB Host και Device Mode

#### 4.4.1 Προγραμματισμός USB από την πλευρά του Android

Το Android διαθέτει τις εξής classes για προγραμματισμό σε λειτουργία USB accessory mode:

- UsbManager —Επιτρέπει την επικοινωνία με τα συνδεδεμένα USB accessories
- UsbAccessory —Αντιπροσωπεύει μια συσκευή USB και περιέχει μεθόδους (methods) για την ανάκτηση πληροφοριών σχετικά με αυτή

Η λειτουργία host επιτρέπει στη συσκευή Android να ελέγχει οτιδήποτε συνδέεται σε αυτή. Επίσης παρέχει τροφοδοσία στη συνδεδεμένη συσκευή USB. Παραδείγματα τέτοιων USB συσκευών που συνδέονται στο Android όταν αυτό λειτουργεί ως host είναι πληκτρολόγια, ποντίκια και άλλες συσκευές εισόδου. Οι ακόλουθες κλάσεις (**classes**)-με τις αντίστοιχες μεθόδους που διαθέτουν- χρησιμοποιούνται για ανάπτυξη σε λειτουργία **USB host**:

- UsbManager —Έχει πρόσβαση στην κατάσταση των συνδέσεων USB και επικοινωνεί με τις συνδεδεμένες συσκευές
- UsbDevice —Αντιπροσωπεύει τη συνδεδεμένη συσκευή USB
- UsbInterface —Μία διεπαφή (interface) της UsbDevice
- UsbEndpoint —Ένα άκρο (endpoint) της UsbInterface
- UsbDeviceConnection —Στέλνει και λαμβάνει μηνύματα σε μια συσκευή USB
- UsbRequest —Αντιπροσωπεύει ένα πακέτο USB request
- UsbConstants —Σταθερές που χρησιμοποιούνται στο πρωτόκολλο USB

Για τις ανάγκες της εφαρμογής, συγκεκριμένα, δημιουργήσαμε έναν *BroadcastReceiver* που ανταποκρίνεται στα γεγονότα (events) *ACTION\_USB\_DEVICE\_ATTACHED* και *ACTION\_USB\_DEVICE\_DETACHED*. Αυτό θα μας ειδοποιήσει όταν συνδεθεί κάποια

συσκευή USB στο Android σύστημά μας ή αποσυνδεθεί η ήδη συνδεδεμένη συσκευή και θα πραγματοποιήσει την κατάλληλη ενέργεια ανάλογα με την περίπτωση. Ο broadcast receiver δηλαδή ανταποκρίνεται ανάλογα στο αν συνδέεται ή αποσυνδέεται μια συσκευή USB στο Android.

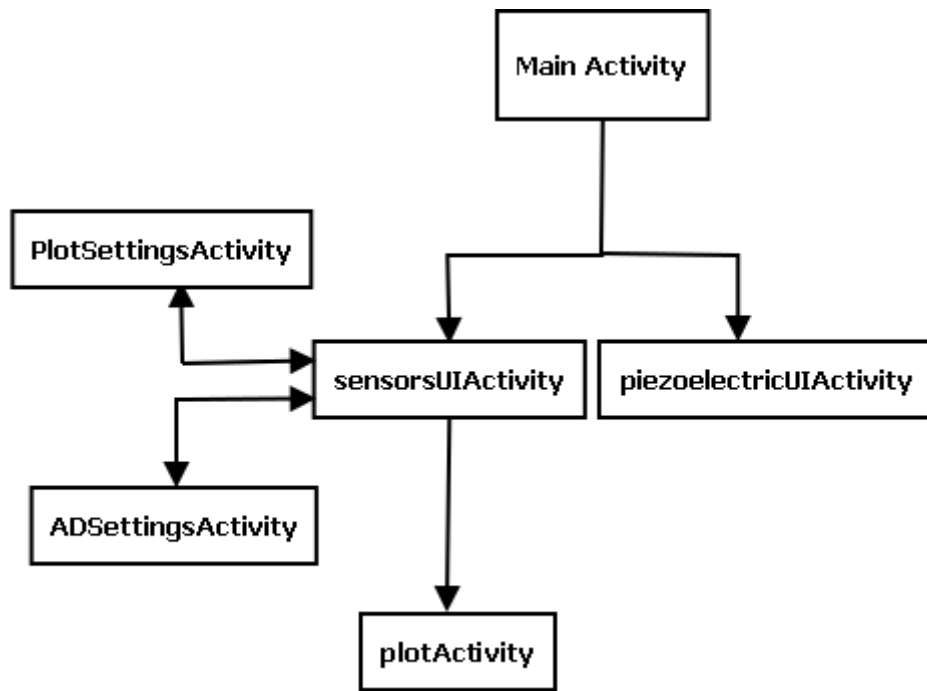
Επίσης δημιουργήσαμε έναν BroadcastReceiver που ανταποκρίνεται στην κλήση της μεθόδου requestPermission(). Η κλήση της requestPermission(), όταν συνδεθεί στο σύστημα Android μια συσκευή USB, εμφανίζει ένα παράθυρο διαλόγου στον χρήστη ζητώντας του άδεια για να επικοινωνήσει και να συνδεθεί μαζί της. Όταν ο χρήστης απαντήσει στο παράθυρο διαλόγου, ο broadcast receiver λαμβάνει το intent που περιέχει το EXTRA\_PERMISSION\_GRANTED, το οποίο είναι μια boolean τιμή που αντιπροσωπεύει την απάντηση. Αν η τιμή είναι true τότε επιτρέπεται η πρόσβαση στη συσκευή και μπορούμε να συνδεθούμε μαζί της καλώντας την connectUsb(). Αν η τιμή είναι false τότε δεν επιτρέπεται η πρόσβαση και εμφανίζεται μήνυμα λάθους.

#### **4.5 Εφαρμογή (application) για Android**

Ο κύριος σκοπός της εφαρμογής (application) που αναπτύχθηκε για Android ήταν –εκτός από το να παρέχει ένα περιβάλλον αλληλεπίδρασης του χρήστη με τη συσκευή που κατασκευάσαμε- αφενός να επικοινωνεί με το κύκλωμα του μικροελεγκτή και των αισθητήρων και να λαμβάνει δεδομένα από αυτό, και αφετέρου να απεικονίζει γραφικά σε πραγματικό χρόνο τα δεδομένα αυτά. Η εφαρμογή θα αναλάμβανε επίσης να αποθηκεύει σε αρχεία τις ληφθέντες μετρήσεις για ενδεχόμενη μελλοντική αναφορά.

Η εφαρμογή που αναπτύξαμε αποτελείται από έναν αριθμό από Java classes και θρόνες διεπαφής χρήστη (user interface-UI) προκειμένου να εξυπηρετήσουν το σκοπό της γραφικής απεικόνισης των μετρήσεων και την αποθήκευσή τους, όπως επίσης και την επικοινωνία μέσω USB με το κύκλωμα του μικροελεγκτή. Η σημαντικότερη πρόκληση ήταν η ανάγκη της σχεδίασης των γραφικών παραστάσεων σε πραγματικό χρόνο (real time), χωρίς να παρουσιάζονται καθυστερήσεις, καθώς και ο σχεδιασμός της επικοινωνίας μέσω USB. Για να επιτευχθούν τα παραπάνω χρησιμοποιήσαμε multithreading, χωρίσαμε δηλαδή κάθε λειτουργία σε ξεχωριστό νήμα (thread) ώστε να είναι ανεξάρτητη η μία από την άλλη και ταυτόχρονα να αποφεύγουμε τυχόν καθυστερήσεις και «κολλήματα» στο user interface. Τα νήματα εκτελούνται ταυτόχρονα από την εφαρμογή χωρίς να επηρεάζεται το ένα από τα υπόλοιπα.

Παρακάτω παρουσιάζονται τα σημαντικότερα τμήματα της εφαρμογής, ενώ ο πλήρης κώδικας παραθέτεται στο τέλος.



Σχήμα 4.1. Δομή εφαρμογής - ιεραρχία activities

#### 4.5.1 Multithreading (πολλαπλά νήματα και παράλληλη επεξεργασία)

Το **multithreading** είναι η δυνατότητα του επεξεργαστή (ενός ή πολλαπλών πυρήνων) να εκτελεί ταυτόχρονα πολλαπλά **νήματα (threads)** ή διαδικασίες (processes), όπου το κάθε νήμα μπορεί να αναλαμβάνει διαφορετικό έργο. Με τον τρόπο αυτό γίνεται αποδοτικότερη χρήση των διαθέσιμων πόρων του συστήματος, ιδιαίτερα όταν ο υπολογιστής διαθέτει πολλαπλούς επεξεργαστές.

Το multithreading μπορεί να εφαρμοστεί και σε μία μόνο διεργασία επιτρέποντας με αυτόν τον τρόπο την παράλληλη επεξεργασία. Τα νήματα, παρόλο που μοιράζονται τους ίδιους πόρους, μπορούν να εκτελούνται παράλληλα και ανεξάρτητα το ένα από το άλλο. Τα πλεονεκτήματα των πολλαπλών νημάτων είναι η ταχύτητα εκτέλεσης, η γρήγορη ανταπόκριση και η παραλληλοποίηση.

Η Java (και κατ' επέκταση το Android) είναι μια γλώσσα που υποστηρίζει πολλαπλά νήματα και multithreading. Μια εφαρμογή μπορεί να χωρίσει και να διαιρέσει τις λειτουργίες της (καθώς και τα δεδομένα της) σε πολλαπλά νήματα και να αφήσει στο λειτουργικό και στην αρχιτεκτονική του συστήματος να καθορίσει πως θα τρέχουν τα νήματα, είτε ταυτόχρονα σε ένα πυρήνα είτε παράλληλα σε πολλαπλούς πυρήνες.

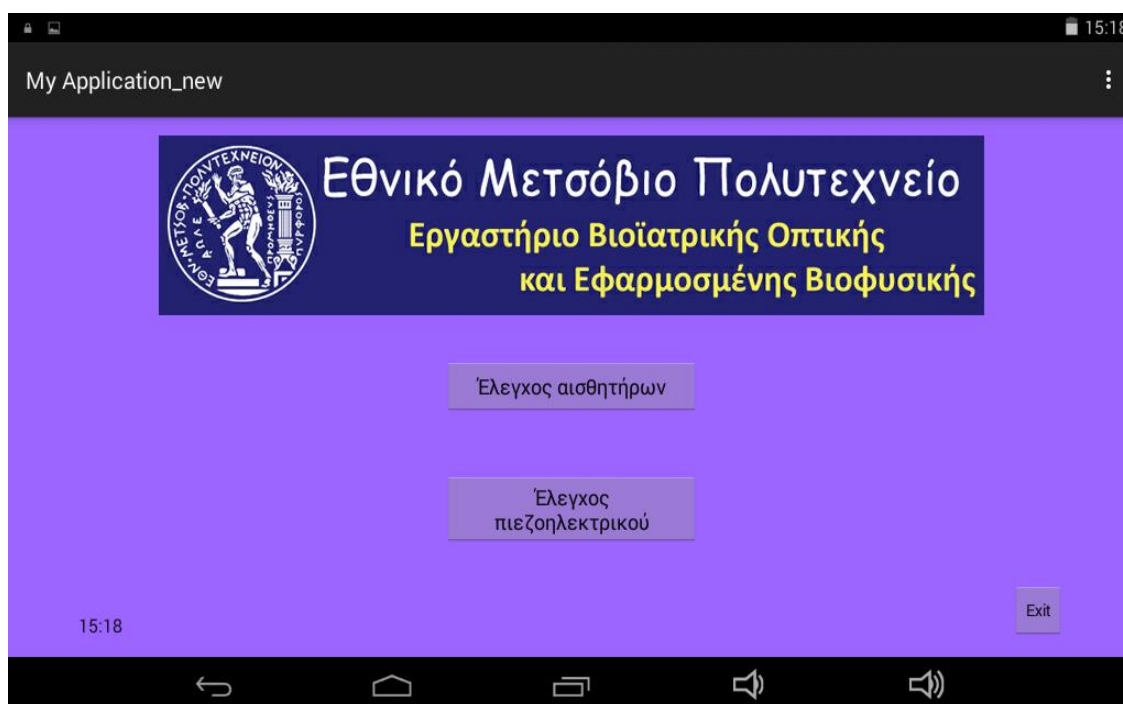
Με τον τρόπο που λειτουργεί το περιβάλλον Android επιτρέπει σε ένα μόνο νήμα (thread) να ενημερώνει και να μεταβάλλει τη διεπαφή χρήστη (user interface-UI) – το νήμα αυτό ονομάζεται UI Thread και είναι το κύριο νήμα που ξεκινάει όταν ξεκινά μια Activity. Η εφαρμογή που αναπτύξαμε έπρεπε να είναι σε θέση να απεικονίζει γραφικά σε πραγματικό χρόνο (real time) τις τιμές των αισθητήρων καθώς τις διαβάζει από το USB και να τις παρουσιάζει στο UI Thread, και ταυτόχρονα να επικοινωνεί με το κύκλωμα του

μικροελεγκτή μέσω USB, χωρίς να παρουσιάζονται καθυστερήσεις . Για να επιτευχθούν τα παραπάνω εφαρμόσαμε multithreading. Χωρίσαμε, δηλαδή, κάθε λειτουργία σε ξεχωριστό νήμα (thread) ώστε να είναι ανεξάρτητη η μία από την άλλη και ταυτόχρονα να αποφύγουμε τυχόν καθυστερήσεις και «κολλήματα» στο user interface. Τα νήματα εκτελούνται ταυτόχρονα από την εφαρμογή χωρίς να επηρεάζεται το ένα από τα υπόλοιπα.

Στην εφαρμογή μας χρησιμοποιήσαμε ξεχωριστό νήμα για την εγκατάσταση της σύνδεσης USB και ξεχωριστό νήμα για να διαβάζει συνεχώς και να απεικονίζει τα δεδομένα που λαμβάνονταν από το USB. Με τον τρόπο αυτό κατορθώσαμε να αποφύγουμε «κολλήματα» του UI (το οποίο ενημερωνόταν και άλλαζε συνεχώς για μεγάλο χρονικό διάστημα) και «κρασαρίσματα» της εφαρμογής. Το νήμα (thread) που δημιουργήσαμε έχει ως κύρια λειτουργία να διαβάζει τα δεδομένα (τις τιμές των αισθητήρων) από το USB και να τα απεικονίζει γραφικά σε πραγματικό χρόνο. Ταυτόχρονα αποθηκεύει τα δεδομένα αυτά σε αρχεία.

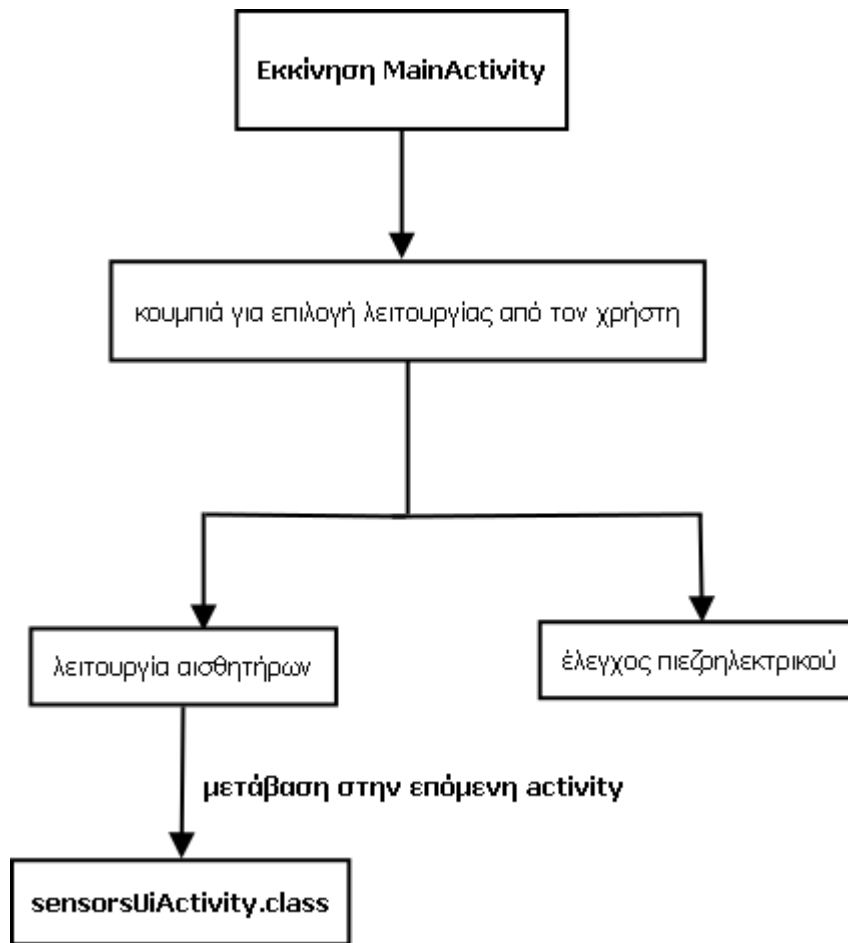
#### 4.5.2 Δομικά τμήματα της εφαρμογής

##### 1. MainActivity.class



Εικόνα 4.4.Οθόνη χρήστη (UI) της MainActivity.class

Αυτή είναι η Activity (το κύριο πρόγραμμα ή συνάρτηση) που ξεκινάει όταν εκκινεί η εφαρμογή. Προσφέρει στον χρήστη την επιλογή να διαλέξει λειτουργία για τον μικροελεγκτή ανάμεσα σε λήψη μετρήσεων από τους αισθητήρες και έλεγχο του πιεζοηλεκτρικού στοιχείου (μελλοντική επέκταση), και να μεταβεί στην αντίστοιχη οθόνη χρήστη. Η επιλογή γίνεται με πάτημα πάνω στην οθόνη αφής του αντίστοιχου κουμπιού. Τέλος υπάρχει κουμπί εξόδου από την εφαρμογή.



Σχήμα 4.2.Κύρια δομή και διάγραμμα λειτουργίας της MainActivity.class

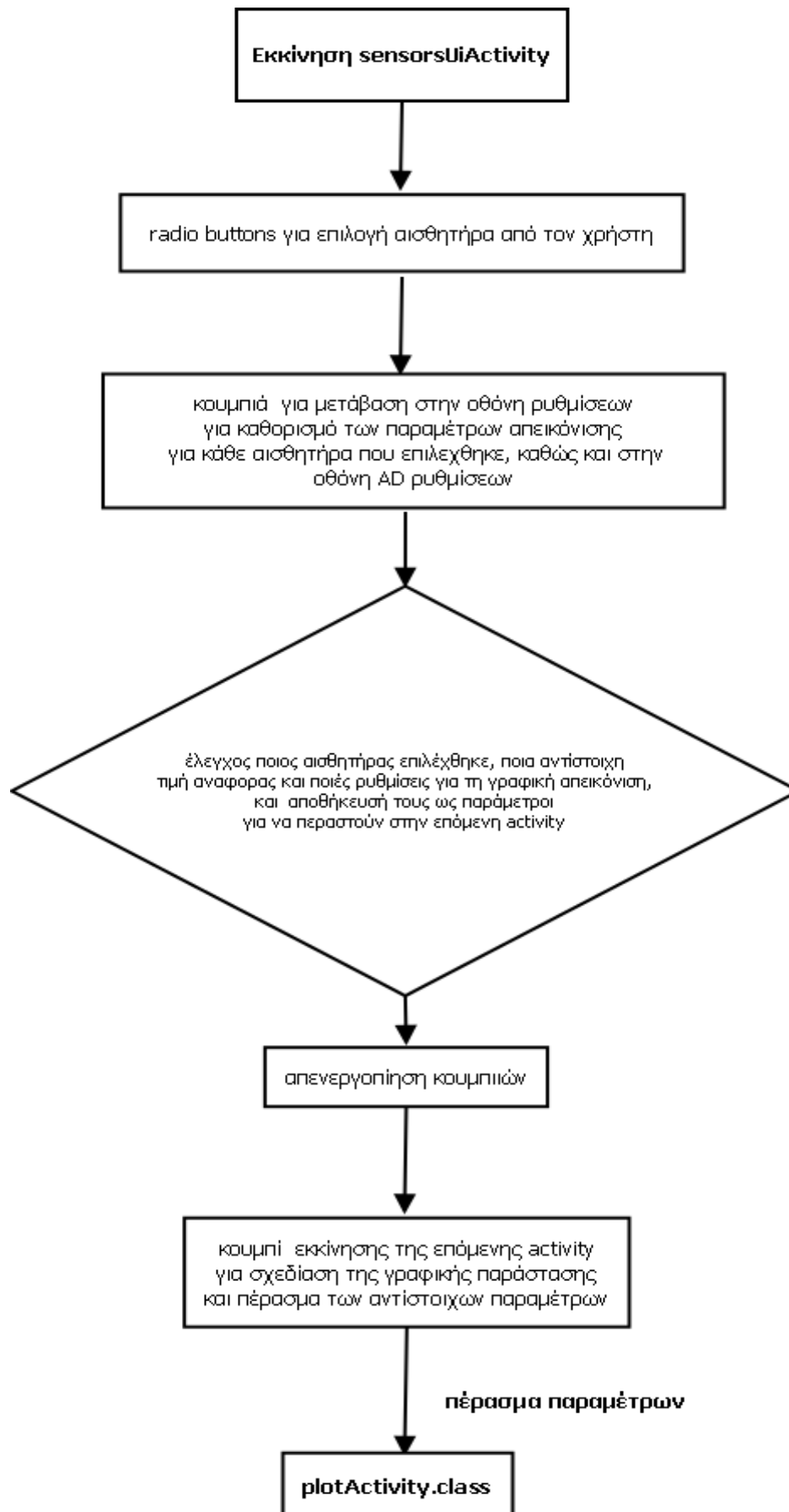
## 2. sensorsUiActivity.class

Η συγκεκριμένη Activity ξεκινάει όταν ο χρήστης επιλέξει τη λειτουργία για λήψη μετρήσεων από τους αισθητήρες, στην προηγούμενη οθόνη. Ο χρήστης έχει τη δυνατότητα να επιλέξει όποιον αισθητήρα επιθυμεί (ή όλους) διαλέγοντας το αντίστοιχο radio button. Επιπλέον –αφού επιλέξει τους αισθητήρες από τους οποίους θέλει να λάβει μετρήσεις – έχει τη δυνατότητα να καθορίσει και άλλες παραμέτρους της γραφικής απεικόνισης, όπως είναι η αντίστοιχη τιμή αναφοράς που επιθυμεί για κάθε μέγεθος, τη μέγιστη και την ελάχιστη τιμή του που καθορίζουν το διάστημα απεικόνισης, καθώς και τη χρονική υποδιαίρεση του οριζόντιου άξονα (σε δευτερόλεπτα, ώρες κτλ.), πατώντας το κουμπί *PLOT SETTINGS* και μεταβαίνοντας στην οθόνη των αντίστοιχων ρυθμίσεων. Με αυτόν τον τρόπο ο χρήστης μπορεί να παρατηρεί πιο καθαρά, κατά τη διάρκεια της γραφικής απεικόνισης, το πόσο μεταβάλλονται οι τιμές των μεγεθών σε σχέση με τη τιμή αναφοράς που έχει ορίσει. Στο τέλος, αφού έχει επιλέξει τις επιθυμητές παραμέτρους, πατάει το κουμπί για τη σχεδίαση της γραφικής παράστασης και οι παραπάνω παράμετροι περνάνε στην επόμενη Activity που ξεκινά εκείνη τη στιγμή.

Επιπλέον υπάρχει το κουμπί *ADC Settings* για επιλογή άλλων αναλογικών αισθητήρων μεταβαίνοντας στην αντίστοιχη οθόνη ρυθμίσεων.



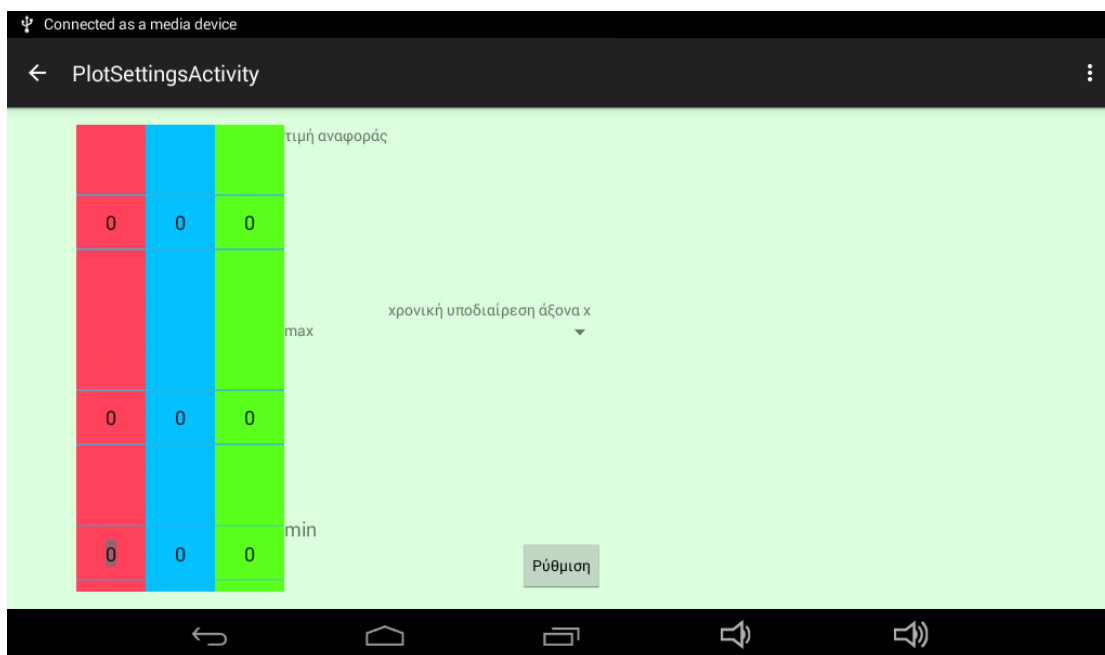
Εικόνα 4.5.Οθόνη χρήστη (UI) της sensorsUiActivity.class



Σχήμα 4.3.Κόρια δομή και διάγραμμα λειτουργίας της sensorsUiActivity.class

### 3. PlotSettingsActivity.class

Η activity αυτή παρέχει τη δυνατότητα στο χρήστη να καθορίσει διάφορες παραμέτρους της γραφικής απεικόνισης των τιμών των αισθητήρων. Συγκεκριμένα για καθένα από τα τρία προς μέτρηση μεγέθη (θερμοκρασία, υγρασία και co2) μπορεί να επιλέξει την αντίστοιχη τιμή αναφοράς που επιθυμεί, τη μέγιστη και την ελάχιστη τιμή που καθορίζουν το διάστημα απεικόνισης εντός του οποίου θα σχεδιαστούν γραφικά τα σημεία, καθώς και τη χρονική υποδιαίρεση του οριζώντιου άξονα (σε δευτερόλεπτα, ώρες κτλ.). Στη συνέχεια πατώντας το κουμπί Set αποθηκεύει τις επιλογές του ώστε να περαστούν ως παράμετροι στην plotActivity.class η οποία αναλαμβάνει το έργο της γραφικής αναπαράστασης.



Εικόνα 4.6.Οθόνη χρήστη (UI) της PlotSettingsActivity.class

### 4. SettingsActivity.class

Η συγκεκριμένη activity δίνει τη δυνατότητα στο χρήστη να επιλέξει άλλους αναλογικούς αισθητήρες (μπορεί να χρησιμοποιηθεί και για λειτουργία γέφυρας). Ο χρήστης μπορεί να διαλέξει πόσα από τα οχτώ διαθέσιμα κανάλια (8 pins) ADC επιθυμεί να χρησιμοποιήσει, ενεργοποιώντας τα με τον αντίστοιχο διακόπτη, και για κάθε κανάλι να ορίσει τη συχνότητα δειγματοληψίας σε msecs. Στη συνέχεια πατώντας το κουμπί Set αποθηκεύει τις επιλογές του ώστε να περαστούν ως παράμετροι στον μικροελεγκτή.

Αυτό γίνεται (σε άλλη activity) στέλνοντας την εντολή 2 στον μικροελεγκτή μέσω USB , όπου το πρώτο byte (ο αριθμός 2) είναι το αναγνωριστικό της εντολής για λειτουργία άλλων αναλογικών (ή γέφυρας) και τα υπόλοιπα byte είναι οι παράμετροι με τις ρυθμίσεις του χρήστη.



Εικόνα 4.7.Οθόνη χρήστη (UI) της SettingsActivity.class

## 5. plotActivity.class

Αυτή είναι η σημαντικότερη Activity που επιτελεί τις κυριότερες λειτουργίες και το βασικό σκοπό της συγκεκριμένης εφαρμογής. Καταρχάς, η Activity αυτή εγκαθιστά την επικοινωνία και συνδέεται με το κύκλωμα του μικροελεγκτή μέσω USB. Όταν ο χρήστης πατήσει το κουμπί start, που βρίσκεται στο πάνω μέρος της οθόνης, τότε η Activity, αφού εγκαταστήσει τη σύνδεση USB δίνει εντολή στο μικροελεγκτή να ξεκινήσει τη λήψη μετρήσεων από τους αισθητήρες στέλνοντας ως εντολή τον ακέραιο αριθμό 1 σε μορφή byte. Χειρίζεται την αποστολή πακέτων δεδομένων ανάμεσα στο Android και το κύκλωμα του μικροελεγκτή και όταν ο χρήστης πατήσει το κουμπί stop για να σταματήσει η εφαρμογή, τότε στέλνει τον αριθμό 3 σε byte στο μικροελεγκτή για να τερματίσει τη σύνδεση.

Αν ο χρήστης στείλει την εντολή 2, τότε εκτός από τον αριθμό 2 στέλνονται στον μικροελεγκτή και άλλα 16 bytes που ρυθμίζουν τα AD κανάλια και την περίοδο δειγματοληψίας τους σε msecs.

Η Activity αυτή είναι επίσης υπεύθυνη για τη γραφική απεικόνιση των ληφθέντων δεδομένων από το USB σε πραγματικό χρόνο, χρησιμοποιώντας τη βιβλιοθήκη ανοιχτού κώδικα achartengine για να δημιουργήσει τις γραφικές παραστάσεις. Με τον τρόπο που λειτουργεί το περιβάλλον Android επιτρέπει σε ένα μόνο νήμα (thread) να ενημερώνει και να μεταβάλλει τη διεπαφή χρήστη (user interface-UI) – το νήμα αυτό ονομάζεται UI Thread και είναι το κύριο νήμα που ξεκινάει όταν ξεκινά μια Activity. Για αυτό το λόγο εφαρμόσαμε multithreading. Στην εφαρμογή μας χρησιμοποιήσαμε ξεχωριστό νήμα για την εγκατάσταση της σύνδεσης USB και ξεχωριστό νήμα για να διαβάζει συνεχώς και να απεικονίζει τα δεδομένα που λαμβάνονταν από το USB. Με τον τρόπο αυτό κατορθώσαμε να αποφύγουμε «κολλήματα» του UI (το οποίο ενημερωνόταν και άλλαζε συνεχώς για μεγάλο χρονικό διάστημα) και «κρασαρίσματα» της εφαρμογής.

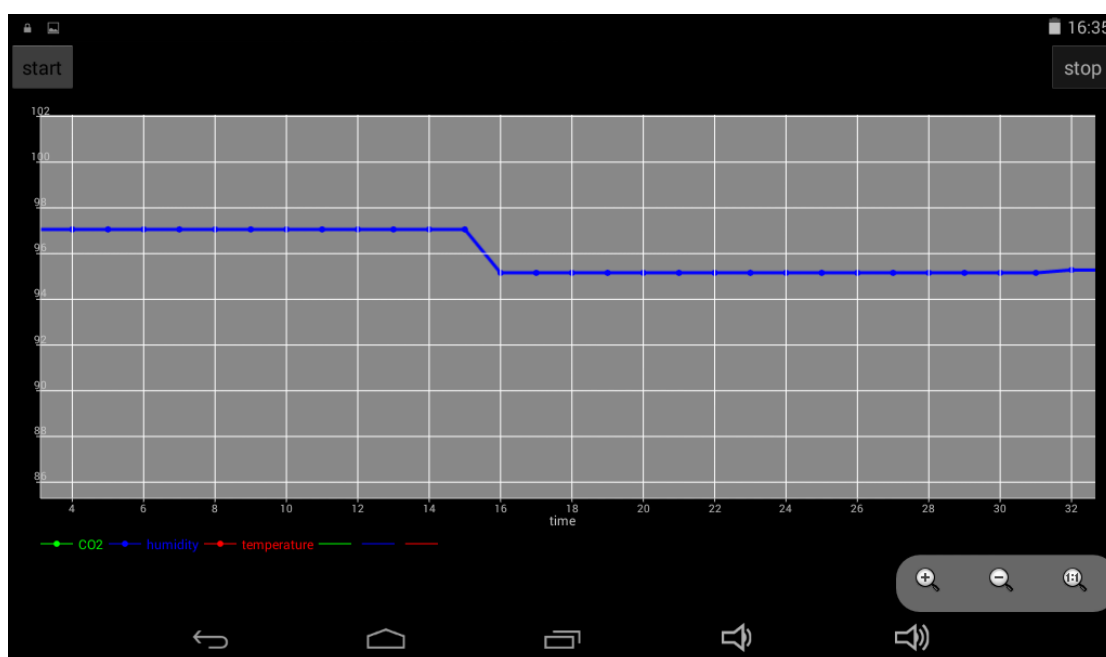
Το **νήμα (thread)** έχει ως κύρια λειτουργία να διαβάζει τα δεδομένα (τις τιμές των αισθητήρων) από το USB και να τα απεικονίζει γραφικά σε πραγματικό χρόνο. Ταυτόχρονα

αποθηκεύει τα δεδομένα αυτά σε αρχεία. Καθορίσαμε εξαρχής κάθε αισθητήρας να αντιστοιχεί σε διαφορετικό endpoint (άκρο) του USB. Έτσι από κάθε endpoint διαβάζονται οι τιμές του αντίστοιχου αισθητήρα. Κάθε endpoint έχει το δικό του ByteBuffer μεγέθους 64 bytes όπου αποθηκεύει κάθε φορά τα δεδομένα που διαβάζει. Η λήψη δεδομένων από το USB γίνεται ασύγχρονα χρησιμοποιώντας την class UsbRequest (αντιπροσωπεύει ένα πακέτο USB request) και τις μεθόδους queue(ByteBuffer, int) (που τοποθετεί σε ουρά το request για να διαβάσει τα δεδομένα και να τα αποθηκεύσει στο καθορισμένο buffer ) και requestWait() (η οποία επιστρέφει το αποτέλεσμα της queue(ByteBuffer, int) ). Στη συνέχεια από κάθε buffer διαβάζεται μία-μία μέτρηση και αναπαρίσταται ως σημείο στην αντίστοιχη γραφική παράσταση .Παράλληλα αποθηκεύεται και στο αντίστοιχο αρχείο.

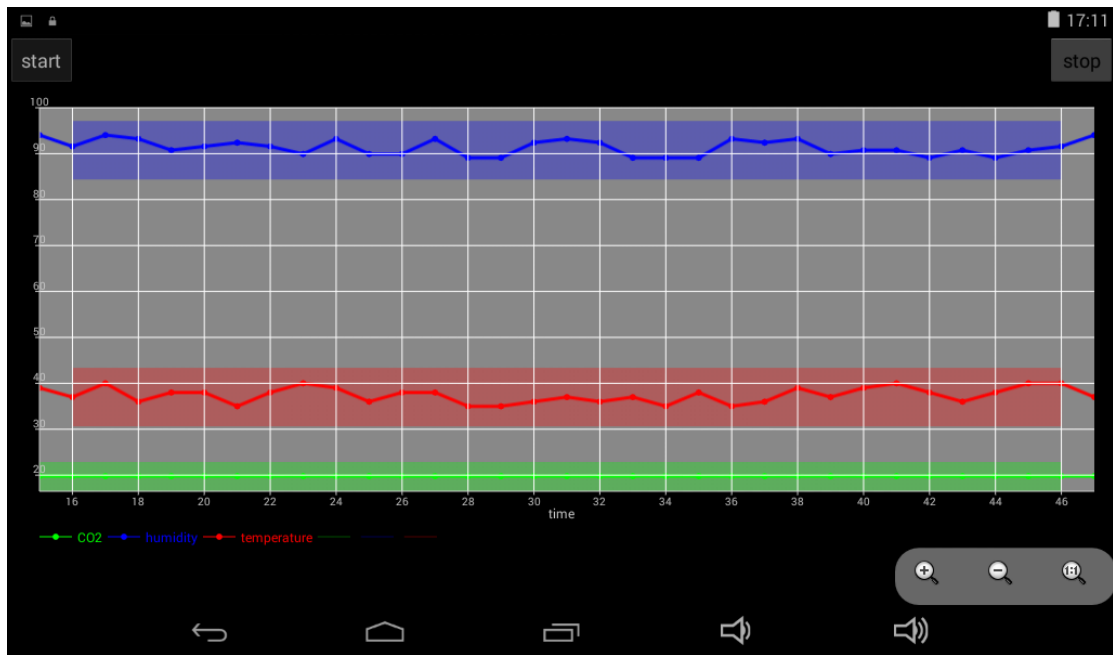
(Σημείωση: η τιμή αναφοράς, που έχει επιλέξει ο χρήστης στην προηγούμενη οθόνη, αναπαρίσταται ως μία σταθερή τιμή στη γραφική παράσταση).

Τέλος η Activity αποθηκεύει σε αρχεία τις ληφθέντες μετρήσεις σε μορφή binary για ενδεχόμενη μελλοντική αναφορά, γράφοντας επιπλέον στο όνομα του αρχείου και τη χρονική στιγμή –ημερομηνία και ώρα- που ξεκίνησε η λήψη των μετρήσεων.

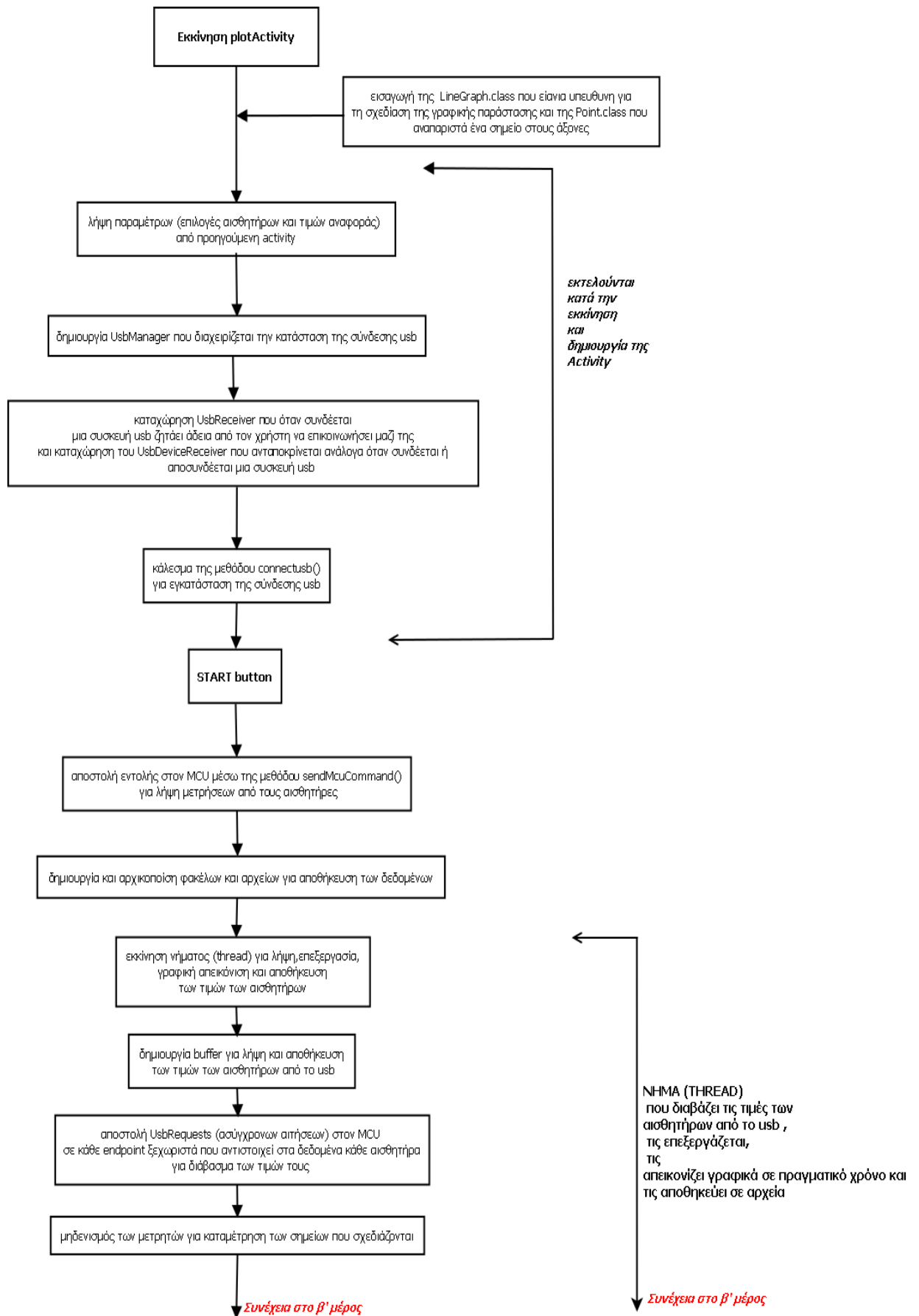
Η plotActivity.class χρησιμοποιεί τις classes LineGraph.class (είναι υπεύθυνη για τη δημιουργία και την απεικόνιση της γραφικής παράστασης καθορίζοντας διάφορες λεπτομέρειες και παραμέτρους όπως είναι η προσθήκη καινούργιων σημείων, ο σχεδιασμός και η παρουσίαση των αξόνων, τα χρώματα, το στυλ της γραφικής παράστασης κτλ.) και Point.class (αντιπροσωπεύει ένα σημείο τις γραφικής παράστασης με συντεταγμένες στους άξονες x και y ) που γράψαμε εμείς για να διευκολύνουν το έργο της.



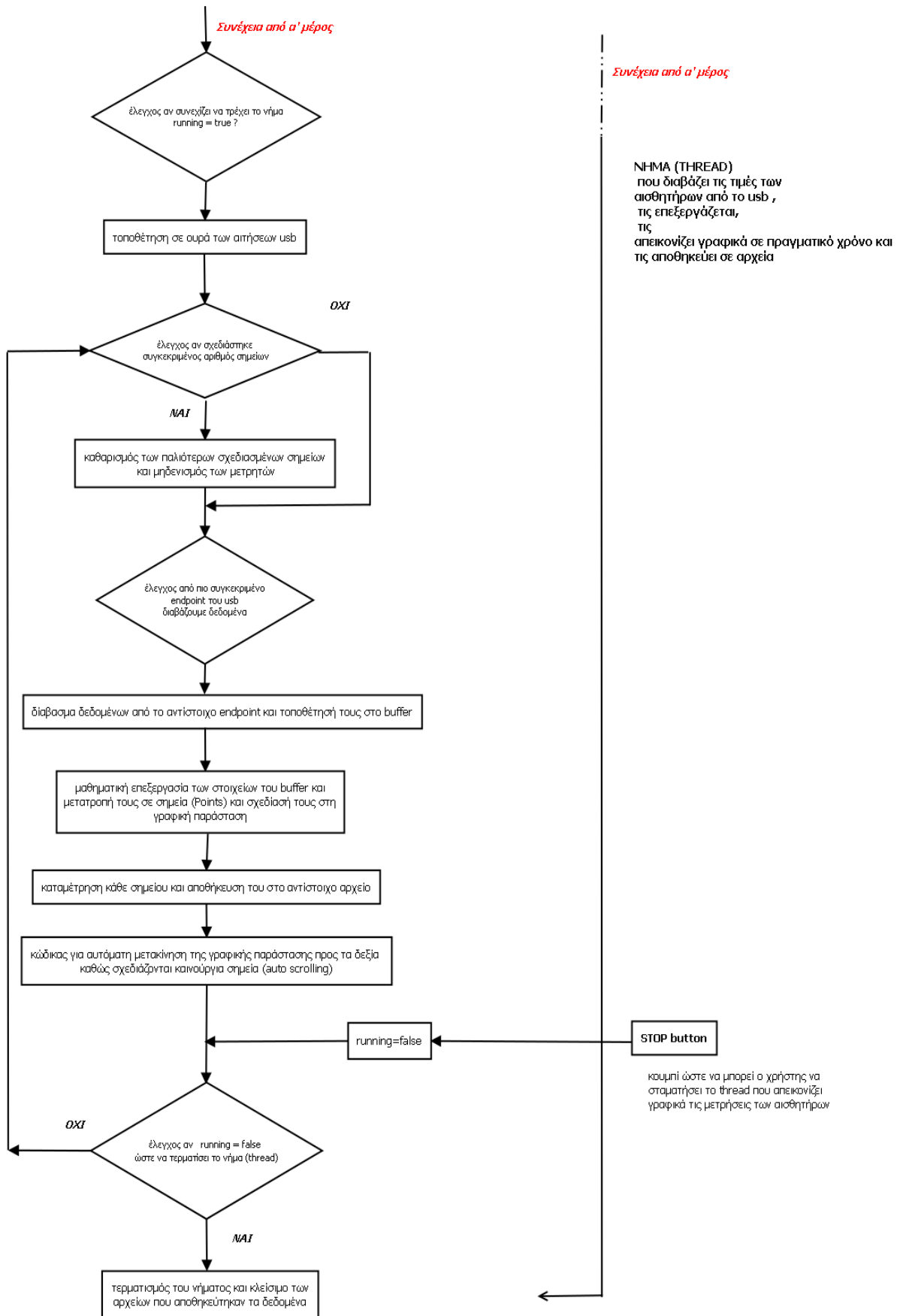
**Εικόνα 4.8.Στιγμιότυπο της plotActivity.class που παρουσιάζει τη γραφική απεικόνιση των τιμών του αισθητήρα της υγρασίας σε μεγέθυνση (zoom)**



**Εικόνα 4.9.** Στιγμιότυπο της `plotActivity.class` που παρουσιάζει τη γραφική απεικόνιση των τιμών των τριών αισθητήρων μαζί



**Σχήμα 4.4.Κύρια δομή και διάγραμμα λειτουργίας της plotActivity.class (Μέρος Α')**



**Σχήμα 4.5.Κύρια δομή και διάγραμμα λειτουργίας της plotActivity.class (Μέρος Β')**

### 4.5.3 Περιγραφή και ανάλυση των classes που δημιουργήσαμε

Για την υλοποίηση της εφαρμογής μας δημιουργήσαμε τις classes της Java, `LineGraph.class` και `Point.class`, για να μας βοηθήσουν στη γραφική απεικόνιση και στη σχεδίαση των σημείων της γραφικής παράστασης, από τα δεδομένα των αισθητήρων που διαβάζονται μέσω USB. Ο κώδικάς τους παρατίθεται στο παράρτημα.

#### LineGraph.class

Είναι υπεύθυνη για τη δημιουργία και την απεικόνιση της γραφικής παράστασης. Δημιουργεί ένα αντικείμενο (object) που αναπαριστά τη συνολική γραφική παράσταση και ρυθμίζει, με τις μεθόδους που διαθέτει, όλες τις λεπτομέρειες και παραμέτρους της γραφικής απεικόνισης, όπως είναι το είδος του γραφήματος, η προσθήκη καινούργιων σημείων, ο σχεδιασμός και η παρουσίαση των αξόνων, τα χρώματα, το στυλ της γραφικής παράστασης, τα περιθώρια, το μέγεθος των γραμμών, τη δυνατότητα ζουμ κτλ.

Οι κύριες μέθοδοί της είναι:

- I. **LineGraph()** : (constructor method). Δημιουργεί ένα αντικείμενο (object) που αναπαριστά τη συνολική γραφική παράσταση και ρυθμίζει, με τις μεθόδους που διαθέτει, όλες τις λεπτομέρειες της γραφικής απεικόνισης
- II. **getView(Context context)** επιστρέφει τη View που περικλείει τη συνολική γραφική απεικόνιση ώστε να εμφανιστεί στην οθόνη του χρήστη
- III. **addNewPoints(Point p)** προσθέτει ένα καινούργιο σημείο στη σειρά που αντιστοιχεί στο ποσοστό CO<sub>2</sub>
- IV. **addNewPoints2(Point p)** προσθέτει ένα καινούργιο σημείο στη σειρά που αντιστοιχεί στη τιμή της υγρασίας
- V. **addNewPoints3(Point p)** προσθέτει ένα καινούργιο σημείο στη σειρά που αντιστοιχεί στη τιμή της θερμοκρασίας
- VI. **addNewPointsCn1(Point p)** προσθέτει ένα καινούργιο σταθερό σημείο που αντιστοιχεί στην επιλεγμένη από το χρήστη τιμή αναφοράς για το μέγεθος τ
- VII. **addNewPointsCn2(Point p)** προσθέτει ένα καινούργιο σταθερό σημείο που αντιστοιχεί στην επιλεγμένη από το χρήστη τιμή αναφοράς για το μέγεθος τ
- VIII. **addNewPointsCn3(Point p)** προσθέτει ένα καινούργιο σταθερό σημείο που αντιστοιχεί στην επιλεγμένη από το χρήστη τιμή αναφοράς για το μέγεθος τ
- IX. **adjust1()** προσαρμόζουν τη συνολική γραφική απεικόνιση στην οθόνη ώστε να ανανεώνεται και να μετακινείται αυτόματα προς τα δεξιά (auto scrolling) καθώς σχεδιάζονται καινούργια σημεία
- X. **adjust2()** Ανανεώνει το μέγιστο και το ελάχιστο x του διαστήματος εντός του οποίου προβάλλεται ο επιθυμητός αριθμός σημείων ώστε να γίνεται scrolling
- XI. **clearOldPoints(int count)** μέθοδος που αφαιρεί από τη γραφική παράσταση τα παλιότερα σχεδιασμένα σημεία κάθε σειράς και την καθαρίζει

### **Point.class**

Αντιπροσωπεύει **ένα σημείο** τις γραφικής παράστασης με συντεταγμένες στους άξονες  $x$  και  $y$ . Οι κύριες μέθοδοί της είναι:

- I. **Point(int x,int y)** δημιουργεί ένα αντικείμενο (object) που αναπαριστά ένα σημείο με συντεταγμένες  $x$  και  $y$
- II. **getX()** επιστρέφει τη συντεταγμένη (τη διάσταση  $x$ ) από ένα δοσμένο σημείο
- III. **getY()** επιστρέφει τη συντεταγμένη (τη διάσταση  $y$ ) από ένα δοσμένο σημείο

### **4.5.4 Περιγραφή των μεθόδων του κώδικα που αφορούν τις λειτουργίες σχετικά με τη σύνδεση USB**

Στα επόμενα αποσπάσματα κώδικα της `plotActivity.class` παρουσιάζονται οι μέθοδοι (methods) που αφορούν τις λειτουργίες σχετικά με τη σύνδεση USB.

Σημείωση: Από τις δοκιμές που έγιναν διαπιστώσαμε ότι ένα επιπλέον λανθασμένο '0x00' λαμβάνεται στη πλευρά του Android, για αυτό θα πρέπει να το αγνοήσουμε.

#### **Μέθοδος `setupUsbComm()`**

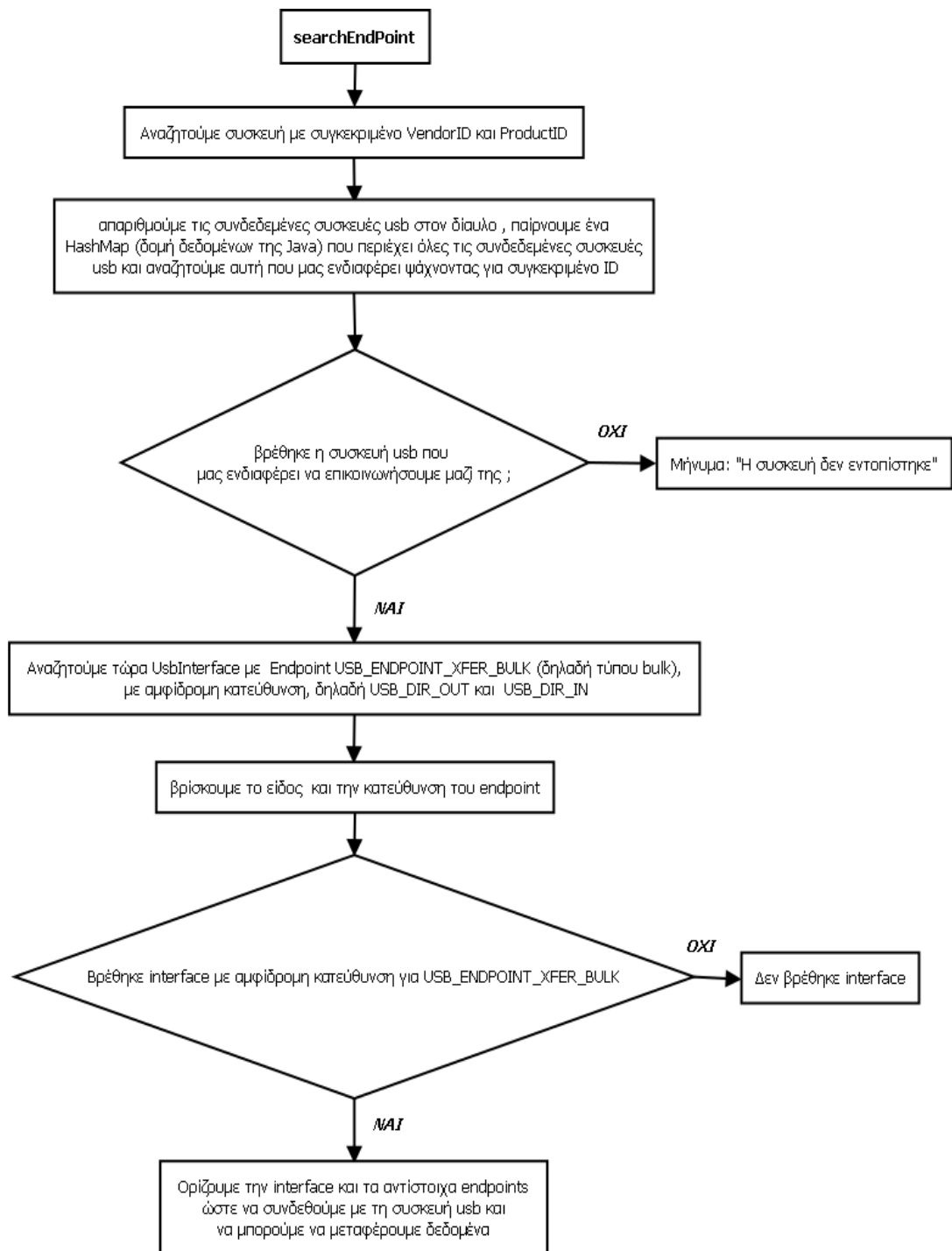
Η μέθοδος αυτή αναλαμβάνει το έργο της επικοινωνίας και της εγκατάστασης της σύνδεσης USB με το κύκλωμα του μικροελεγκτή. Επιστρέφει true ή false ανάλογα με το αν ήταν επιτυχές ή όχι το στήσιμο της σύνδεσης. Χρησιμοποιεί μεταφορά τύπου *control transfer*, που είναι αμφίδρομες μεταφορές οι οποίες χρησιμοποιούνται για αποστολή μικρών μηνυμάτων και εντολών που είναι απαραίτητες για το στήσιμο της σύνδεσης USB και την προετοιμασία της συσκευής. Αποστέλλονται από τον host, που στην περίπτωσή μας είναι το Android, στο endpoint 0 του USB (control endpoint).

#### **Μέθοδος `searchEndPoint()`**

Η συγκεκριμένη μέθοδος αναζητά τα διαθέσιμα endpoints και τη διαθέσιμη interface (που συνίσταται από τα εν λόγω endpoints) της σύνδεσης USB προκειμένου να γίνει η επικοινωνία και να εγκατασταθεί η σύνδεση USB από την προηγούμενη μέθοδο `setupUsbComm()`. Συγκεκριμένα αναζητάμε αν υπάρχουν endpoints τύπου bulk με αμφίδρομη κατεύθυνση (και προς τα μέσα και προς τα έξω - In and Out) και αν υπάρχει interface που διαθέτει τα συγκεκριμένα endpoints. Κατεύθυνση In είναι η κατεύθυνση από device στον host, ενώ κατεύθυνση Out είναι από τον host στο device.

Σημείωση: Όπως έχει δημιουργηθεί και οριστεί η interface από την πλευρά του μικροελεγκτή, γνωρίζουμε εκ των προτέρων ότι υπάρχει μία interface με τρία endpoints τύπου bulk με κατεύθυνση In (από device στον host), που χρησιμοποιούνται για να διαβάζονται τα δεδομένα από το USB, και ένα endpoint τύπου bulk με κατεύθυνση Out το οποίο χρησιμοποιείται για να στέλνονται εντολές από το Android στον μικροελεγκτή. Για

αυτό το λόγο ορίζουμε ένα πίνακα που περιέχει τρία endpoints με κατεύθυνση In χωρίς να χρειάζεται να ψάξουμε όλα τα endpoints.Επίσης ορίζουμε και ένα Out endpoint.



Σχήμα 4.6.Κύρια δομή και διάγραμμα λειτουργίας της μεθόδου searchEndPoint().

### Μέθοδος sendMcuCommand

Η μέθοδος αυτή είναι υπεύθυνη για την αποστολή εντολών στον μικροελεγκτή . Η αποστολή γίνεται χρησιμοποιώντας μεταφορά τύπου bulk (bulk transfer) στο endpoint Out που έχει κατεύθυνση από τον host (Android) στο device. Στέλνεται στον μικροελεγκτή ένα μήνυμα που αποτελείται από δύο bytes. Το πρώτο byte είναι η τιμή '0xFF' που χρησιμοποιείται για το συγχρονισμό της επικοινωνίας και για να δηλώσει την έναρξη των δεδομένων, και το δεύτερο byte είναι η κύρια εντολή ελέγχου προς τον μικροελεγκτή που μπορεί να έχει μία από τις εξής δύο τιμές

- την τιμή **0x01** σε byte (δηλαδή τον ακέραιο αριθμό 1) , που είναι η εντολή για να αρχίσει ο μικροελεγκτής να παίρνει μετρήσεις από τους αισθητήρες και να τις στέλνει στο Android
- την τιμή **0x02** σε byte (δηλαδή τον ακέραιο αριθμό 2), που εκτός από τον αριθμό 2 στέλνονται στον μικροελεγκτή και άλλα 16 bytes που ρυθμίζουν τα ADC κανάλια και την περίοδο δειγματοληψίας τους σε msecs (για λειτουργία γέφυρας).
- και την τιμή **0x03** σε byte (δηλαδή τον ακέραιο αριθμό 3), η οποία είναι η εντολή για να σταματήσει ο μικροελεγκτής να λαμβάνει και να στέλνει μετρήσεις από τους αισθητήρες και να σταματήσει τη σύνδεση με το Android

### Μέθοδος releaseUsb()

Αυτή η μέθοδος αποσυνδέει τη συνδεδεμένη συσκευή USB με την οποία επικοινωνούμε και κλείνει τη σύνδεση. Θέτει null τη σύνδεση, τη συσκευή, τα endpoints και την αντίστοιχη interface, ώστε όταν συνδεθεί κάποια καινούργια συσκευή USB (ή ξανά η ίδια), να μπορούμε να συνδεθούμε, να επικοινωνήσουμε μαζί της και να εγκαταστήσουμε νέα σύνδεση αν θέλουμε.

## 4.6 Μελλοντικές επεκτάσεις και αναβαθμίσεις

Κατά τη διάρκεια της εκπόνησης της παρούσας διπλωματικής προέκυψαν και συζητήθηκαν διάφορες ιδέες και πιθανές βελτιώσεις και προεκτάσεις της κατασκευής. Όμως δεν ήταν δυνατό να υλοποιηθούν λόγω του περιορισμού του χρόνου και εξαιτίας του ότι ξεπερνούσαν το σκοπό της παρούσας εργασίας. Αναφέρουμε ενδεικτικά ορισμένες από αυτές:

- i. Σύνδεση του κυκλώματος του μικροελεγκτή με πιεζοηλεκτρικό στοιχείο μέσω γέφυρας. Το πιεζοηλεκτρικό στοιχείο θα μετακινεί τη πλακέτα (δρομέα) μικροσκοπίου. Ο μικροελεγκτής θα ελέγχει την μετακίνηση του πιεζοηλεκτρικού και θα κάνει τις κατάλληλες διορθώσεις ώστε να παρέχει την απαραίτητη ακρίβεια. Θα υπάρχει αντίστοιχη εφαρμογή Android ή επέκταση της υπάρχουσας εφαρμογής με μια νέα Activity, που θα παρέχει ένα γραφικό περιβάλλον επικοινωνίας με τον χρήστη, ο οποίος θα ρυθμίζει από τη συσκευή Android την επιθυμητή μετακίνηση του πιεζοηλεκτρικού και θα λαμβάνει στην οθόνη μήνυμα ανάλογα με το αν ήταν επιτυχής ή όχι η μετακίνηση. Το Android θα επικοινωνεί με το κύκλωμα του

μικροελεγκτή μέσω της ήδη εγκαταστημένης σύνδεσης USB και θα του στέλνει την-ορισμένη από το χρήστη- απόσταση μέσω του πρωτοκόλλου επικοινωνίας που έχουμε αναπτύξει για τις ανάγκες της συγκεκριμένης διπλωματικής. Και ο μικροελεγκτής με τη σειρά του θα αναλαμβάνει την πραγματοποίηση της μετακίνησης του πιεζοηλεκτρικού και τον αυτόματο έλεγχο της.

ii. Βελτίωση της αποθήκευσης και της ανάλυσης των δεδομένων. Μια πρόσθετη λειτουργία και επέκταση της εφαρμογής Android που σκεφτήκαμε αλλά δεν καταφέραμε να υλοποιήσουμε λόγω του περιορισμού του χρόνου που διαθέταμε, ήταν η δυνατότητα της εφαρμογής να απεικονίζει και να αναπαριστά γραφικά τις αποθηκευμένες τιμές των αισθητήρων. Η εφαρμογή που αναπτύξαμε, αποθηκεύει όπως είδαμε σε αρχεία (που αναγράφουν και την ημερομηνία που έγινε η καταγραφή των μετρήσεων) τις τιμές που λαμβάνει από τους αισθητήρες. Ο χρήστης, λοιπόν, θα μπορεί να επιλέξει όποιο από τα αρχεία, που έχουν αποθηκευμένα τα δεδομένα, επιθυμεί και η εφαρμογή θα απεικονίζει γραφικά τις τιμές τους. Έτσι θα έχει τη δυνατότητα να ανατρέξει σε παλιότερες μετρήσεις των μεγεθών για πιθανή μελλοντική αναφορά και σύγκριση. Επιπλέον, η εφαρμογή θα μπορούσε να επεξεργάζεται τα αποθηκευμένα αρχεία και να παρουσιάζει στατιστικά αποτελέσματα στο χρήστη σχετικά με τη μεταβολή των τιμών των μεγεθών κατά τη διάρκεια του χρόνου.

iii. Άμεση, ζωντανή και απομακρυσμένη ενημέρωση του χρήστη σχετικά με την εξέλιξη των πειραματικών μετρήσεων. Όπως είδαμε, η εφαρμογή παρέχει τη δυνατότητα στο χρήστη να επιλέξει μία επιθυμητή τιμή αναφοράς για κάθε μέγεθος. Θα μπορούσε, λοιπόν, να επεκταθεί η εφαρμογή και να γραφεί κώδικας ώστε όταν οι τιμές ενός μεγέθους υπερβούν κατά πολύ την τιμή αναφοράς ή ξεπεράσουν κάποιο κατώφλι, να στέλνεται άμεσα ενημέρωση στο χρήστη, είτε με e-mail είτε με γραπτό μήνυμα στο κινητό του (sms), έτσι ώστε αυτός με τη σειρά του να προβεί στις ενέργειες που κρίνει κατάλληλες.

iv. Κατάργηση της ενσύρματης επικοινωνίας-σύνδεσης του Android με το κύκλωμα του μικροελεγκτή και προτίμηση της ασύρματης επικοινωνίας. Θα πρόσφερε σημαντικά πλεονεκτήματα αν αντί της ενσύρματης σύνδεσης μέσω USB που χρησιμοποιήσαμε στην υλοποίηση μας, χρησιμοποιούσαμε ασύρματη σύνδεση όπως για παράδειγμα Bluetooth. Με αυτό τον τρόπο θα μπορούσαμε να καταργήσουμε το καλώδιο USB και να αποκτήσει η συσκευή Android ανεξαρτησία στο χώρο από το κύκλωμα του μικροελεγκτή και των αισθητήρων. Θα ήταν δηλαδή δυνατό η συσκευή που απεικονίζει γραφικά τις τιμές των αισθητήρων - και κατ'επέκταση ο χρήστης - να βρίσκεται σε διαφορετικό δωμάτιο από το χώρο που διεξάγονται οι μετρήσεις. Μόνη προϋπόθεση βέβαια θα είναι το κύκλωμα του μικροελεγκτή να έχει δική του ανεξάρτητη τροφοδοσία (εξωτερική τροφοδοσία ή μπαταρία) και να διαθέτει συσκευή Bluetooth.

Αυτές είναι ορισμένες από τις πιθανές βελτιώσεις και επεκτάσεις της συσκευής που κατασκευάσαμε -και συγκεκριμένα της πλευράς του Android, που θα μπορούσε να αναλάβει να υλοποιήσει κάποιο άλλο άτομο ή ομάδα που θα συνέχιζε και θα βελτίωνε το έργο μας.

## ΠΑΡΑΡΤΗΜΑ Α

### Πηγαίος κώδικας μικροελεγκτή

#### A.1 Κώδικας για τη λειτουργία USB

**Αρχείο usbconfig.h** (ορισμός γενικών παραμέτρων της λειτουργίας USB του μικροελεγκτή):

```
/*Επιλογή λειτουργίας USB device για τον μικροελεγκτή */  
  
#define USB_DEVICE  
  
/*Αριθμός επιπλέον endpoints πέραν του 0 */  
  
#define NUM_EP_USED      4  
  
/*Αριθμός timers που θα χρειαστεί η λειτουργία του USB */  
  
#define NUM_APP_TIMERS   1  
  
/*Ορισμοί endpoints */  
  
#define DATA_OUT      ( 0x01 ) /* endpoint για λήψη δεδομένων (εντολών)          */  
#define DATA_IN1      ( 0x81 ) /* endpoint για αποστολή δεδομένων (μετρήσεων) */  
#define DATA_IN2      ( 0x82 ) /* endpoint για αποστολή δεδομένων (μετρήσεων) */  
#define DATA_IN3      ( 0x83 ) /* endpoint για αποστολή δεδομένων (μετρήσεων) */  
  
/*Επιλογή του timer που θα χρησιμοποιηθεί  
  
#define TIMER_ID          0
```

**Αρχείο descriptors.h** (ορισμός των descriptors) :

```
#include "em_usb.h"  
  
/*Ορισμός Device Descriptor */  
  
extern const USB_DeviceDescriptor_TypeDef USBDESC_deviceDesc;  
  
/*Ορισμός Configuration descriptor */  
  
extern const uint8_t          USBDESC_configDesc[];  
  
/*Ορισμός αριθμού buffer για κάθε endpoint */  
  
extern const uint8_t          USBDESC_bufferingMultiplier[];
```

**Αρχείο descriptors.c (στοιχεία κάθε descriptor):**

```
#include "descriptors.h"

/* Device Descriptor */

EFM32_ALIGN(4)

const USB_DeviceDescriptor_TypeDef USBDESC_deviceDesc __attribute__
((aligned(4)))=
{
    .bLength      = USB_DEVICE_DESCSIZE,
    .bDescriptorType = USB_DEVICE_DESCRIPTOR,
    .bcdUSB       = 0x0200,
    .bDeviceClass  = 0xFF,
    .bDeviceSubClass = 0,
    .bDeviceProtocol = 0,
    .bMaxPacketSize0 = USB_FS_CTRL_EP_MAXSIZE,
    .idVendor      = 0x10C4,
    .idProduct     = 0x0001,
    .bcdDevice     = 0x0000,
    .iManufacturer = 0,
    .iProduct      = 0,
    .iSerialNumber = 0,
    .bNumConfigurations = 1
};

/* Configuration descriptor**/

EFM32_ALIGN(4)

const uint8_t USBDESC_configDesc[] __attribute__ ((aligned(4)))=
{
    USB_CONFIG_DESCSIZE,
    USB_CONFIG_DESCRIPTOR,
    USB_CONFIG_DESCSIZE +
```

```

USB_INTERFACE_DESCSIZE +
(USB_ENDPOINT_DESCSIZE * NUM_EP_USED),
(USB_CONFIG_DESCSIZE +
USB_INTERFACE_DESCSIZE +
(USB_ENDPOINT_DESCSIZE * NUM_EP_USED))>>8,
1,
1,
0,
CONFIG_DESC_BM_RESERVED_D7 |
CONFIG_DESC_BM_SELFPOWERED,
CONFIG_DESC_MAXPOWER_mA( 100 ),

/*Interface descriptor */
USB_INTERFACE_DESCSIZE, /* bLength */
USB_INTERFACE_DESCRIPTOR, /* bDescriptorType */
0, /* bInterfaceNumber */
0, /* bAlternateSetting */
4, /* bNumEndpoints */
0, /* bInterfaceClass */
0, /* bInterfaceSubClass */
0, /* bInterfaceProtocol */
0, /* iInterface */

/* endpoint descriptors */
USB_ENDPOINT_DESCSIZE, /* bLength */
USB_ENDPOINT_DESCRIPTOR, /* bDescriptorType */
DATA_IN1, /* bEndpointAddress (IN) */
USB_EPTYPE_BULK, /* bmAttributes */
USB_FS_BULK_EP_MAXSIZE, /* wMaxPacketSize (LSB) */

```

```

0,          /* wMaxPacketSize (MSB) */
0,          /* bInterval      */
USB_ENDPOINT_DESCSIZE, /* bLength      */
USB_ENDPOINT_DESCRIPTOR, /* bDescriptorType */
DATA_IN2,   /* bEndpointAddress (IN) */
USB_EPTYPE_BULK, /* bmAttributes */
USB_FS_BULK_EP_MAXSIZE, /* wMaxPacketSize (LSB) */
0,          /* wMaxPacketSize (MSB) */
0,          /* bInterval      */
USB_ENDPOINT_DESCSIZE, /* bLength      */
USB_ENDPOINT_DESCRIPTOR, /* bDescriptorType */
DATA_IN3,   /* bEndpointAddress (IN) */
USB_EPTYPE_BULK, /* bmAttributes */
USB_FS_BULK_EP_MAXSIZE, /* wMaxPacketSize (LSB) */
0,          /* wMaxPacketSize (MSB) */
0,          /* bInterval      */
USB_ENDPOINT_DESCSIZE, /* bLength      */
USB_ENDPOINT_DESCRIPTOR, /* bDescriptorType */
DATA_OUT,   /* bEndpointAddress (OUT) */
USB_EPTYPE_BULK, /* bmAttributes */
USB_FS_BULK_EP_MAXSIZE, /* wMaxPacketSize (LSB) */
0,          /* wMaxPacketSize (MSB) */
0           /* bInterval      */
};

/* Διπλό μέγεθος buffer για κάθε bulk endpoint */
const uint8_t USBDESC_bufferingMultiplier[ NUM_EP_USED + 1 ] = { 1, 2, 2, 2, 2 };

```

**Αρχείο cdc.c (περιέχει τις συναρτήσεις της λειτουργίας USB):**

```
#include "em_device.h"

#include "em_common.h"

#include "em_cmu.h"

#include "em_emu.h"

#include "em_gpio.h"

#include "em_usb.h"

#include "cdc.h"

#define BULK_EP_SIZE 64

/* Μέγεθος πακέτου όταν δέχεται δεδομένα */
#define USB_RX_BUF_SIZ CDC_BULK_EP_SIZE

/* Μέγεθος πακέτου όταν στέλνει δεδομένα */
#define USB_TX_BUF_SIZ CDC_BULK_EP_SIZE

/* ορισμός buffers για αποδοχή δεδομένων */
STATIC_UBUF(usbRxBuffer0, USB_RX_BUF_SIZ); STATIC_UBUF(usbRxBuffer1,
USB_RX_BUF_SIZ);

/* ορισμός buffers για αποστολή δεδομένων */
STATIC_UBUF(usbTx1Buffer0, USB_TX_BUF_SIZ);
STATIC_UBUF(usbTx1Buffer1, USB_TX_BUF_SIZ);

STATIC_UBUF(usbTx2Buffer0, USB_TX_BUF_SIZ);
STATIC_UBUF(usbTx2Buffer1, USB_TX_BUF_SIZ);
STATIC_UBUF(usbTx3Buffer0, USB_TX_BUF_SIZ);
STATIC_UBUF(usbTx3Buffer1, USB_TX_BUF_SIZ);

/*πίνακες με δείκτες για κάθε buffer */
static const uint8_t *usbRxBuffer[2] = {usbRxBuffer0,usbRxBuffer1 };
static const uint8_t *usbTx1Buffer[2]={usbTx1Buffer0,usbTx1Buffer1 };
static const uint8_t *usbTx2Buffer[2]={usbTx2Buffer0,usbTx2Buffer1 };
static const uint8_t *usbTx3Buffer[2]={usbTx3Buffer0,usbTx3Buffer1 };
```

```

/*μεταβλητές που δείχνουν ποιο buffer χρησιμοποιούμε
   πόσα bytes δεχθήκαμε και πόσα θα στείλουμε */
static int      usbRxIndex, usbBytesReceived;
static int      usbTxIndex, usbTxCount;
/*μεταβλητή που ενημερώνει αν έχουν ξεκινήσει οι αισθητήρες*/
static int      sensors_init=0;
/*ρουτίνα ανάκλησης όταν αλλάζει η κατάσταση του USB */
void CDC_StateChangeEvent( USBD_State_TypeDef oldState,
                           USBD_State_TypeDef newState)
{
    if (newState == USBD_STATE_CONFIGURED)
    {
        /*ξεκινάει η λειτουργία*/
        /* διαβάζουμε εντολή */
        usbRxIndex = 0;
        USBD_Read(DATA_OUT, (void*) usbRxBuffer[ usbRxIndex ],
                 USB_RX_BUF_SIZ, UsbDataReceived);
    }
}
/*ρουτίνα που καλείται όταν δεχθεί δεδομένα το USB */
static int UsbDataReceived(USB_Status_TypeDef status,
                           uint32_t xferred,
                           uint32_t remaining)
{
    (void) remaining;      /* Unused parameter. */
    int a,i;
    if ((status == USB_STATUS_OK) && (xferred > 0))
    {

```

```

usbBytesReceived = xferred;

a=(usbRxBuffer[usbRxIndex]);

usbRxIndex ^= 1;

usbTxIndex = 0 ;

usbTxCount = 64 ;

switch(a){

/*εντολή για τους αισθητήρες υγρασίας,θερμοκρασίας,co2 */
    case 1 :

        if (sensors_init==0) {

/*Εδώ καλούνται οι συναρτήσεις που εκκινούν τους αισθητήρες */

            sensors_init=1;

        }

        while (1) {

/*ελέγχει αν οι αισθητήρες γέμισαν τα buffer*/

            if buffer_ready {

/*στέλνει τις μετρήσεις κάθε αισθητήρα*/

                USBD_Write(DATA_IN1,(void*) usbTx1Buffer[usbTxIndex] ,
usbTxCount, NULL);

                USBD_Write(DATA_IN2,(void*)usbTx2Buffer[usbTxIndex] ,
usbTxCount, NULL);

                USBD_Write(DATA_IN3,(void*) usbTx3Buffer[usbTxIndex] ,
usbTxCount, NULL);

/*αλλάζει buffer για την επόμενη μεταφορά*/

                usbTxIndex ^= 1;

/*διαβάζει πάλι εντολή*/

                USBD_Read(DATA_OUT, (void*) usbRxBuffer[ usbRxIndex ],
USB_RX_BUF_SIZ, UsbDataReceived);

            }

            break;

```

```

        case 2 :

/*συναρτήσεις για την εκκίνηση των αισθητήρων γέφυρας*/
        break;

/*εντολή 3 ή άγνωστη τερματίζει την σύνδεση*/
        case 3 :

                USBD_Disconnect();

        default :

                USBD_Disconnect();

        break;

    }

}

return USB_STATUS_OK;
}

```

**Αρχείο main.c (ξεκινάει απλώς τη λειτουργία USB):**

```

#include "em_device.h"

#include "em_cmu.h"

#include "em_gpio.h"

#include "em_int.h"

#include "em_usb.h"

#include "cdc.h"

#include "descriptors.h"

static const USBD_Callbacks_TypeDef callbacks =
{

```

```

.usbReset      = NULL,

.usbStateChange = CDC_StateChangeEvent,

.setupCmd      = NULL,

.isSelfPowered = NULL,

.sofInt        = NULL
};

static const USBD_Init_TypeDef usbInitStruct =
{
.deviceDescriptor = &USBDESC_deviceDesc,
.configDescriptor = USBDESC_configDesc,
.stringDescriptors = NULL,
.numberofStrings  = NULL,
.callbacks         = &callbacks,
.bufferingMultiplier = USBDESC_bufferingMultiplier,
.reserved          = 0
};

int main(void)
{
CMU_ClockSelectSet(cmuClock_HF, cmuSelect_HFXO);

USB_D_Init(&usbInitStruct);

for (;;)
{
}
}

```

## A.2 Κώδικας για τη λειτουργία των αισθητήρων

Κώδικας δειγματοληψίας για τους αισθητήρες υγρασίας ,θερμοκρασίας και CO<sub>2</sub>:

```
#include "em_device.h"

#include "em_chip.h"

#include "em_system.h"

#include "em_cmu.h"

#include "em_gpio.h"

#include "em_i2c.h"

#include "em_rtc.h"

#include "em_adc.h"

/*μετρητής για την δειγματοληψία*/

int sample_cnt=125;

/*δομές για την λειτουργία I2C*/

I2C_TransferReturn_TypeDef I2C_Status ;

I2C_TransferSeq_TypeDef I2C_Sequence ;

I2C_Init_TypeDef i2cInit = I2C_INIT_DEFAULT;

uint8_t i2cbuffer[4];

/*δομές για την λειτουργία ADC */

ADC_InitSingle_TypeDef co2_0 = ADC_INITSINGLE_DEFAULT;

ADC_InitSingle_TypeDef co2_1 = ADC_INITSINGLE_DEFAULT;

ADC_Init_TypeDef init = ADC_INIT_DEFAULT;

int adc_pin_port=2,adc_pin_num=0;

int channel_index=0 ;

/*δομή για την λειτουργία RTC που παράγει τα interrupts με ρυθμό 1KHz*/

RTC_Init_TypeDef rtcInit = RTC_INIT_DEFAULT;
```

```

/*ενεργοποίηση των απαραίτητων ρολογιών */
void setupClocks(void)
{
    /* ενεργοποίηση LFRCO και RTC */
    CMU_OscillatorEnable(cmuOsc_LFRCO, true, true);
    CMU_ClockSelectSet(cmuClock_LFA,cmuSelect_LFRCO);
    CMU_ClockEnable(cmuClock_RTC, true);
    /*βήμα RTC 1ms */
    CMU_ClockDivSet(cmuClock_RTC,cmuClkDiv_32);
    /* ενεργοποίηση περιφερειακών υψηλής συχνότητας*/
    CMU_ClockEnable(cmuClock_HFPER,true);
    /* ενεργοποίηση I2C1 */
    CMU_ClockEnable(cmuClock_I2C1, true);
    /*ενεργοποίηση GPIO */
    CMU_ClockEnable(cmuClock_GPIO, true);
    /* ενεργοποίηση ADC */
    CMU_ClockEnable(cmuClock_ADC0,true);
    /* ενεργοποίηση περιφερειακών χαμηλής ενέργειας (RTC)*/
    CMU_ClockEnable(cmuClock_CORELE, true);
}

/*προετοιμασία I2C*/
void Setup_i2c()
{
    /*επιλογή ποια pins θα είναι η θύρα I2C */
    GPIO_PinModeSet(gpioPortC, 5, gpioModeWiredAnd, 1);
    GPIO_PinModeSet(gpioPortC, 4, gpioModeWiredAnd, 1);
}

```

```

I2C1->ROUTE = I2C_ROUTE_SDAPEN |
                I2C_ROUTE_SCLPEN |
                (0 << _I2C_ROUTE_LOCATION_SHIFT);

I2C_Init(I2C1, &i2cInit);
/* ενεργοποίηση interrupts */
NVIC_ClearPendingIRQ(I2C1_IRQn);
NVIC_EnableIRQ(I2C1_IRQn);
/*χαρακτηριστικά της μέτρησης από τον αισθητήρα της εργασίας*/
I2C_Sequence.addr = 0x50 ;
I2C_Sequence.flags = I2C_FLAG_READ;
I2C_Sequence.buf[0].data = i2cbuffer ;
I2C_Sequence.buf[0].len = 4;
}

/* interrupt handler του I2C */
void I2C1_IRQHandler(void)
{
    I2C_Status = I2C_Transfer(I2C1);
    if (I2C_Status==i2cTransferDone) {
        /*αν ολοκληρώθηκε η μετρηση στέλνεται στο buffer για αποστολή*/
    }
}

/* προετοιμασία ADC */
void setupADC(void)
{
    /* pin για τη δημιουργία του τετραγωνικού παλμού*/
    GPIO_PinModeSet(adc_pin_port,adc_pin_num,gpioModePushPull,0);
}

```

```

    /*ορισμός καναλιών για τη μέτρηση του CO2*/
co2_0.reference = adcRefVDD;

    co2_0.input = adcSingleInpCh0;
    co2_0.reference = adcRefVDD;
    co2_0.input = adcSingleInpCh1;

    /*ορισμός χαρακτηριστικών του ADC και ενεργοποίηση του*/
    init.timebase = ADC_TimebaseCalc(0);
    init.prescale = ADC_PrescaleCalc(400000,0);
    ADC_Init(ADC0, &init);

    /* ενεργοποίηση ADC interrupt για ολοκλήρωση μέτρησης */
    ADC_IntEnable(ADC0, ADC_IF_SINGLE);
    NVIC_EnableIRQ(ADC0_IRQn);
}

/* interrupt handler του ADC */
void ADC0_IRQHandler(void)
{
    ADC_IntClear(ADC0, ADC_IF_SINGLE);

    /*αν μέτρησε το κανάλι 0 διαβάζει την τιμή και ξεκινάει να διαβάζει το κανάλι 1*/
    if (channel_index==0) {
        channel_index ^=1;

        /*εδώ η μετρηση στέλνεται στο buffer για αποστολή*/
        ADC_InitSingle(ADC0, &co2_1);
        ADC_Start(ADC0, adcStartSingle);
    }

    /*αν μέτρησε το κανάλι 1 άλλαξε το pin σε 0*/
    else {

```

```

        channel_index ^=1;

        //send value to buffer

        GPIO_PinOutToggle(adc_pin_port,adc_pin_num);

    }
}

```

```

/* προετοιμασία RTC */

```

```

void setupRtc(void)

```

```

{

```

```

/*χαρακτηριστικά RTC */

```

```

    rtcInit.enable = true;

```

```

    rtcInit.comp0Top = true;

```

```

    rtcInit.debugRun = false;

```

```

    RTC_CompareSet(0, 0);

```

```

    /* ενεργοποίηση RTC interrupt για όταν περνάει 1ms */

```

```

    RTC_IntEnable(RTC_IFC_COMP0);

```

```

    NVIC_EnableIRQ(RTC_IRQn);

```

```

    /* εκκίνηση RTC */

```

```

    RTC_Init(&rtcInit);

```

```

}

```

```

/* interrupt handler του RTC*/

```

```

void RTC_IRQHandler(void)

```

```

{

```

```

    RTC_IntClear(RTC_IFC_COMP0);

```

```

    /*αν δεν πέρασαν 125ms μείωσε τον μετρητή*/

```

```

    if (sample_cnt) {

```

```

        sample_cnt -= 1;

```

```

}
else {
    sample_cnt=125;
    /*αν πέρασαν 125ms άνοιξε το pin αν ήταν 0 */
    if (GPIO_PinOutGet(adc_pin_port,adc_pin_num)==0) {
        GPIO_PinOutToggle(adc_pin_port,adc_pin_num);
    }
    /*αν ήταν 1 πάρε μετρήσεις */
    else {
        I2C_Status = I2C_TransferInit(I2C1,&I2C_Sequence);
        ADC_InitSingle(ADC0, &co2_0);
        ADC_Start(ADC0, adcStartSingle);
    }
}
}

int main(void)
{
    setupClocks();
    Setup_i2c();
    setupADC();
    setupRtc();
    /* Infinite loop */
    while (1) {
    }
}

```



## Παράρτημα Β

### Πηγαίος κώδικας Android

#### Β.1 Κώδικας Android σε Java

##### MainActivity.java

```
import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.content.Intent;
import android.widget.Button;

public class MainActivity extends ActionBarActivity {

    private Button exitButton;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        //κουμπί για έξοδο και κλείσιμο εφαρμογής
        exitButton = (Button) findViewById(R.id.button4);
        exitButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                //finish();
                //System.exit(0);
            }
        });
    }
}
```

```

        moveTaskToBack(true);
    }
});
}
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu; this adds items to the action bar if it is present.
    getMenuInflater().inflate(R.menu.menu_main, menu);
    return true;
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // Handle action bar item clicks here. The action bar will
    // automatically handle clicks on the Home/Up button, so long
    // as you specify a parent activity in AndroidManifest.xml.
    int id = item.getItemId();

    //noinspection SimplifiableIfStatement
    if (id == R.id.action_settings) {
        return true;
    }

    return super.onOptionsItemSelected(item);
}

public void sensorsGui(View view) {
//εκκίνηση Activity για λειτουργία αισθητήρων
    Intent intent1 = new Intent(this, sensorsUiActivity.class);
    startActivity(intent1);
}

```

```

    public void piezoelectrGui(View view) {
// εκκίνηση Activity για λειτουργία πιεζοηλεκτρικού
        Intent intent2 = new Intent(this, piezoelectrUiActivity.class);
        startActivity(intent2);
    }
}

```

### **sensorsUiActivity.java**

```

import android.content.Context;
import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.Menu;
import android.view.MenuItem;
import android.content.Intent;
import android.view.View;
import android.widget.AdapterView;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.EditText;
import android.widget.RadioButton;
import android.widget.Spinner;
import android.widget.TextView;
import android.widget.ToggleButton;
import java.util.Arrays;

public class sensorsUiActivity extends ActionBarActivity

```

```

{
    private static final String TAG = "AndroidUsbHostMcu";

    Spinner mspin;

    Spinner mspin2;

    Spinner mspin3;

    TextView txtview1;

    TextView txtview2;

    TextView txtview3;

    Integer ref1;

    Integer ref2;

    Integer ref3;

    int indx;

    //δημιουργία λέξεων κλειδιών –αναγνωριστικών για πέρασμα ως παραμέτρων στην
    //επόμενη Activity των επιλογών του χρήστη σχετικά με ποια μεγέθη (θερμοκρασία,
    //υγρασία,co2) επιθυμεί να ληφθούν οι τιμές τους από τους αισθητήρες και να
    //παρουσιαστούν γραφικά

    public final static String My_array1 =

        "com.example.nakama.myapplication_new.myarray1";

    public final static String My_array2 =

        "com.example.nakama.myapplication_new.myarray2";

    public final static String My_array3 =

        "com.example.nakama.myapplication_new.myarray3";

    public final static String My_choice =

        "com.example.nakama.myapplication_new.mychoice";

    //δημιουργία λέξεων κλειδιών –αναγνωριστικών για πέρασμα ως παραμέτρων στην
    //επόμενη Activity των τιμών αναφοράς των μεγεθών που επέλεξε ο χρήστης

    public final static String My_ref1 =

        "com.example.nakama.myapplication_new.myref1";

    public final static String My_ref2 =

```

```

        "com.example.nakama.myapplication_new.myref2";

public final static String My_ref3 =

        "com.example.nakama.myapplication_new.myref3";

int choice=0; //μετράει πόσα από τα τρία μεγέθη (θερμοκρασία, υγρασία, ποσοστό
//co2) έχει επιλέξει ο χρήστης να απεικονίσει και το γνωστοποιεί στην plotActivity

String temp_array;

String hum_array;

String co2_array;

@Override

protected void onCreate(Bundle savedInstanceState) {

    super.onCreate(savedInstanceState);

    setContentView(R.layout.activity_sensors_ui);

    txtview1=(TextView)findViewById(R.id.textView4);

    txtview1.setVisibility(View.INVISIBLE);

    //Δημιουργία των αντικειμένων spinner που είναι είδος dropdown menu και //καθορισμός
ενός συνόλου τιμών για το καθένα, από τις οποίες μπορεί να επιλέξει ο //χρήστης μία
συγκεκριμένη ως τιμή αναφοράς για το αντίστοιχο μέγεθος. Τα //spinners θέτονται αρχικά
αόρατα και ανενεργά έως ότου ο χρήστης να επιλέξει το //αντίστοιχο μέγεθος

    mspin=(Spinner) findViewById(R.id.spinner);

    //καθορισμός συνόλου τιμών αναφοράς για επιλογή για τη θερμοκρασία

    Integer[] items = new Integer[]{22,24,26,28,30};

    ArrayAdapter<Integer> adapter = new
ArrayAdapter<Integer>(this,android.R.layout.simple_spinner_item, items);

    mspin.setAdapter(adapter);

    //mspin.setEnabled(false);

    mspin.setVisibility(View.INVISIBLE);

    txtview2=(TextView)findViewById(R.id.textView7);

```

```

txtview2.setVisibility(View.INVISIBLE);

mspin2=(Spinner) findViewById(R.id.spinner2);

//καθορισμός συνόλου τιμών αναφοράς για επιλογή για την υγρασία

Integer[] items2 = new Integer[]{45,50,55,60,65};

ArrayAdapter<Integer> adapter2 = new
ArrayAdapter<Integer>(this,android.R.layout.simple_spinner_item, items2);

mspin2.setAdapter(adapter2);

//mspin2.setEnabled(false);

mspin2.setVisibility(View.INVISIBLE);

txtview3=(TextView)findViewById(R.id.textView6);

txtview3.setVisibility(View.INVISIBLE);

mspin3=(Spinner) findViewById(R.id.spinner3);

//καθορισμός συνόλου τιμών αναφοράς για επιλογή για το ποσοστό του co2

Integer[] items3 = new Integer[]{3,4,5,6,7};

ArrayAdapter<Integer> adapter3 = new
ArrayAdapter<Integer>(this,android.R.layout.simple_spinner_item, items3);

mspin3.setAdapter(adapter3);

//mspin3.setEnabled(false);

mspin3.setVisibility(View.INVISIBLE);
}

@Override

public boolean onCreateOptionsMenu(Menu menu) {

// Inflate the menu; this adds items to the action bar if it is present.

getMenuInflater().inflate(R.menu.menu_sensors_ui, menu);

return true;

}

@Override

```

```

public boolean onOptionsItemSelected(MenuItem item) {

    int id = item.getItemId();

    if (id == R.id.action_settings) {

        return true;

    }

    return super.onOptionsItemSelected(item);

}

Intent intent = getIntent();

```

*//Η μέθοδος αυτή εκτελείται όταν ο χρήστης επιλέξει κάποιο (ένα η περισσότερα) από //τα μεγέθη που επιθυμεί να μετρήσει και να απεικονίσει γραφικά διαλέγοντας το //αντίστοιχο radiobutton. Στην αρχή βρίσκουμε ποιο (ή ποια) radiobutton επέλεξε ο //χρήστης, αποθηκεύουμε την επιλογή του και εμφανίζουμε το αντίστοιχο spinner //ώστε να επιλέξει τιμή αναφοράς για το μέγεθος*

```

public void onRadioButtonClicked(View view)

{

//αν επιλέχθηκε το radiobutton που αντιστοιχεί στη θερμοκρασία

    if ((view.getId() == R.id.temperature) && ((RadioButton) view).isChecked()) {

        choice++;

        RadioButton option1 = (RadioButton) findViewById(R.id.temperature);

        option1.setClickable(false);

//αποθήκευσε την επιλογή

        temp_array="temperature";

        //mspin.setEnabled(true);

//εμφάνισε το dropdown menu για επιλογή τιμής αναφοράς

        txtview1.setVisibility(View.VISIBLE);

        mspin.setVisibility(View.VISIBLE);

    }

//αν επιλέχθηκε το radiobutton που αντιστοιχεί στην υγρασία

    if ((view.getId() == R.id.humidity) && ((RadioButton) view).isChecked()) {

```

```

        choice++;

        RadioButton option2 = (RadioButton) findViewById(R.id.humidity);

        option2.setClickable(false);

        //αποθήκευσε την επιλογή

        hum_array="humidity";

        //mspin2.setEnabled(true);

        //εμφάνισε το dropdown menu για επιλογή τιμής αναφοράς

        txtview2.setVisibility(View.VISIBLE);

        mspin2.setVisibility(View.VISIBLE);

    }

    //αν επιλέχθηκε το radiobutton που αντιστοιχεί στο ποσοστό του co2

    if ((view.getId() == R.id.co2) && ((RadioButton) view).isChecked()) {

        choice++;

        RadioButton option3 = (RadioButton) findViewById(R.id.co2);

        option3.setClickable(false);

        //αποθήκευσε την επιλογή

        co2_array="co2";

        //mspin3.setEnabled(true);

        //εμφάνισε το dropdown menu για επιλογή τιμής αναφοράς

        txtview3.setVisibility(View.VISIBLE);

        mspin3.setVisibility(View.VISIBLE);

    }

}

//Η παρακάτω μέθοδος αυτή εκτελείται όταν πατήσει ο χρήστης το κουμπί για τη
//σχεδίαση της γραφικής παράστασης και ξεκινάει η αντίστοιχη Activity //(plotActivity)

public void OnButtonPlot(View view) {

    //clear –reset radiobuttons

```

```

RadioButton rb1 = (RadioButton) findViewById(R.id.temperature);
RadioButton rb2 = (RadioButton) findViewById(R.id.humidity);
RadioButton rb3 = (RadioButton) findViewById(R.id.co2);
rb1.setChecked(false);
rb2.setChecked(false);
rb3.setChecked(false);

//δημιουργία intent για να μπορούμε να ξεκινήσουμε την επόμενη Activity //(plotActivity)
Intent intent3 = new Intent(this, plotActivity.class);

//διάβασμα τιμών αναφοράς που επέλεξε ο χρήστης από τα αντικείμενα spinner
ref1=(Integer)mspin.getSelectedItemAt();
ref2=(Integer)mspin2.getSelectedItemAt();
ref3=(Integer)mspin3.getSelectedItemAt();

//πέρασμα παραμέτρων – επιλογών χρήστη στην επόμενη Activity
intent3.putExtra(My_choice, choice);
intent3.putExtra(My_array1, temp_array);
intent3.putExtra(My_array2, hum_array);
intent3.putExtra(My_array3, co2_array);

//πέρασμα τιμών αναφοράς που επέλεξε ο χρήστης στην επόμενη Activity
intent3.putExtra(My_ref1,ref1);
intent3.putExtra(My_ref2,ref2);
intent3.putExtra(My_ref3,ref3);

//εκκίνηση Activity για σχεδίαση γραφικής παράστασης
Intent intent = getIntent();
startActivity(intent3);
}

//επιστροφή πίσω στην οθόνη της προηγούμενης Activity (MainActivity)
@Override

```

```
public void onBackPressed() {  
    Intent intent4 = new Intent(this, MainActivity.class);  
    startActivity(intent4);  
}  
}
```

### **PlotSettingsActivity.java**

```
import android.content.Intent;  
import android.support.v7.app.ActionBarActivity;  
import android.os.Bundle;  
import android.view.Menu;  
import android.view.MenuItem;  
import android.view.View;  
import android.widget.ArrayAdapter;  
import android.widget.NumberPicker;  
import android.widget.Spinner;  
  
public class PlotSettingsActivity extends ActionBarActivity {  
  
    public final static String temp_Ref=  
        "com.example.nakama.myapplication_new.temp_Ref";  
    public final static String tempMx =  
        "com.example.nakama.myapplication_new.tempMx";  
    public final static String tempMn =  
        "com.example.nakama.myapplication_new.tempMn";  
}
```

```
public final static String humRef =
    "com.example.nakama.myapplication_new.humRef";
public final static String humMx =
    "com.example.nakama.myapplication_new.humMx";
public final static String humMin =
    "com.example.nakama.myapplication_new.humMin";
public final static String co2R =
    "com.example.nakama.myapplication_new.co2R";
public final static String co2Mx =
    "com.example.nakama.myapplication_new.co2Mx";
public final static String co2Mn =
    "com.example.nakama.myapplication_new.co2Mn";
public final static String timeChoice =
    "com.example.nakama.myapplication_new.timeChoice";
```

```
NumberPicker np1;
```

```
NumberPicker np2;
```

```
NumberPicker np3;
```

```
NumberPicker np4;
```

```
NumberPicker np5;
```

```
NumberPicker np6;
```

```
NumberPicker np7;
```

```
NumberPicker np8;
```

```
NumberPicker np9;
```

```
Spinner mspin;
```

```
Integer tempRef;
```

```
Integer tempMax;
```

```
Integer tempMin;
```

```
Integer humidRef;  
Integer humidMax;  
Integer humidMin;  
Integer co2Ref;  
Integer co2Max;  
Integer co2Min;  
String timeRef;
```

```
@Override
```

```
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_plot_settings);
```

```
//numberpickers for temperature
```

```
    np1 = (NumberPicker) findViewById(R.id.numberPicker);  
    np1.setMaxValue(40);  
    np1.setMinValue(18);  
    np1.setValue(36);  
    np2 = (NumberPicker) findViewById(R.id.numberPicker2);  
    np2.setMaxValue(45);  
    np2.setMinValue(10);  
    np2.setValue(37);  
    np3 = (NumberPicker) findViewById(R.id.numberPicker3);  
    np3.setMaxValue(34);  
    np3.setMinValue(0);  
    np3.setValue(28);
```

```
//numberpickers for humidity
```

```
np4 = (NumberPicker) findViewById(R.id.numberPicker4);
np4.setMaxValue(90);
np4.setMinValue(10);
np4.setValue(60);
np5 = (NumberPicker) findViewById(R.id.numberPicker5);
np5.setMaxValue(100);
np5.setMinValue(0);
np5.setValue(70);
np6 = (NumberPicker) findViewById(R.id.numberPicker6);
np6.setMaxValue(40);
np6.setMinValue(0);
np6.setValue(40);
//numberpickers for co2
np7 = (NumberPicker) findViewById(R.id.numberPicker7);
np7.setMaxValue(12);
np7.setMinValue(2);
np7.setValue(5);
np8 = (NumberPicker) findViewById(R.id.numberPicker8);
np8.setMaxValue(20);
np8.setMinValue(5);
np8.setValue(7);
np9 = (NumberPicker) findViewById(R.id.numberPicker9);
np9.setMaxValue(10);
np9.setMinValue(0);
np9.setValue(3);

mspin=(Spinner) findViewById(R.id.spinner);
```

```

String[] items = new String[]{"secs","hours","days"};

    ArrayAdapter<String> adapter = new
ArrayAdapter<String>(this,android.R.layout.simple_spinner_item, items);

    mspin.setAdapter(adapter);

    //mspin.setEnabled(false);

    //mspin.setVisibility(View.INVISIBLE);

}

@Override

public boolean onCreateOptionsMenu(Menu menu) {

    // Inflate the menu; this adds items to the action bar if it is present.

    getMenuInflater().inflate(R.menu.menu_plot_settings, menu);

    return true;

}

@Override

public boolean onOptionsItemSelected(MenuItem item) {

    // Handle action bar item clicks here. The action bar will

    // automatically handle clicks on the Home/Up button, so long

    // as you specify a parent activity in AndroidManifest.xml.

    int id = item.getItemId();

    //noinspection SimplifiableIfStatement

    if (id == R.id.action_settings) {

        return true;

    }

    return super.onOptionsItemSelected(item);

```

```

}

public void SavePlotSettings(View view) {
//λαμβάνουμε τις επιλογές του χρήστη για τη γραφική απεικόνιση
//δηλαδή τιμή αναφοράς, μέγιστη και ελάχιστη τιμή διαστήματος απεικόνισης
    tempRef=np1.getValue();
    tempMax=np2.getValue();
    tempMin=np3.getValue();
    humidRef=np4.getValue();
    humidMax=np5.getValue();
    humidMin=np6.getValue();
    co2Ref=np7.getValue();
    co2Max=np8.getValue();
    co2Min=np9.getValue();
    timeRef= (String)mspin.getSelectedItem();

//και τις περνάμε ως παραμέτρους
    Intent intent = new Intent(this, sensorsUiActivity.class);
    intent.putExtra(temp_Ref,tempRef);
    intent.putExtra(tempMx,tempMax);
    intent.putExtra(tempMn,tempMin);
    intent.putExtra(humRef,humidRef);
    intent.putExtra(humMx,humidMax);
    intent.putExtra(humMin,humidMin);
    intent.putExtra(co2R,co2Ref);
    intent.putExtra(co2Mx,co2Max);
    intent.putExtra(co2Mn,co2Min);
    intent.putExtra(timeChoice,timeRef);

```

```
        startActivity(intent);
    }

}
```

### **plotActivity.java**

```
package com.example.nakama.myapplication_new;

import java.io.DataOutputStream;
import java.io.File;
import java.io.FileOutputStream;
import java.io.ObjectOutputStream;
import java.io.OutputStream;
import java.nio.ByteBuffer;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.HashMap;
import java.util.Iterator;
import java.util.Random;

import org.achartengine.ChartFactory;
import org.achartengine.GraphicalView;
import org.achartengine.chart.PointStyle;
import org.achartengine.model.SeriesSelection;
import org.achartengine.model.XYMultipleSeriesDataset;
```

```
import org.achartengine.model.XYSeries;
import org.achartengine.renderer.XYMultipleSeriesRenderer;
import org.achartengine.renderer.XYSeriesRenderer;
import org.achartengine.tools.PanListener;
import org.achartengine.tools.ZoomEvent;
import org.achartengine.tools.ZoomListener;
import android.app.Activity;
import android.app.PendingIntent;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.graphics.Bitmap;
import android.graphics.Bitmap.CompressFormat;
import android.graphics.Color;
import android.graphics.PointF;
import android.hardware.usb.UsbConstants;
import android.hardware.usb.UsbDevice;
import android.hardware.usb.UsbDeviceConnection;
import android.hardware.usb.UsbEndpoint;
import android.hardware.usb.UsbInterface;
import android.hardware.usb.UsbManager;
import android.hardware.usb.UsbRequest;
import android.os.AsyncTask;
import android.os.Bundle;
import android.os.Environment;
import android.os.PowerManager;
import android.util.Log;
```

```

import android.view.MotionEvent;

import android.view.View;

import android.view.ViewGroup.LayoutParams;

import android.view.WindowManager;

import android.widget.Button;

import android.widget.EditText;

import android.widget.LinearLayout;

import android.widget.Toast;

public class plotActivity extends Activity {

    public final static String My_array1 =

        "com.example.nakama.myapplication_new.myarray1";

    public final static String My_array2 =

        "com.example.nakama.myapplication_new.myarray2";

    public final static String My_array3 =

        "com.example.nakama.myapplication_new.myarray3";

    public final static String My_choice =

        "com.example.nakama.myapplication_new.mychoice";

    public int j;

    private static GraphicalView view;

    //δημιουργία αντικειμένου (object) γραφικής παράστασης

    private LineGraph line=new LineGraph();

    //private LineGraph2 line2=new LineGraph2();

    //private LineGraph3 line3=new LineGraph3();

    private static Thread thread;

```

```

private boolean running;

//ορισμός σταθερών πρωτοκόλλου επικοινωνίας με το μικροελεγκτή μέσω usb
private static final byte IGNORE_00 = (byte) 0x00;
private static final byte SYNC_WORD = (byte) 0xFF;
private static final int CMD_OFF = 2;
private static final int CMD_ON = 1;
private static final int CMD_TEXT = 3;
private static final int MAX_TEXT_LENGTH = 16;

int indx;

//μεταβλητές για τα σημεία κάθε μεγέθους
int y1;
int y2;
int y3;

//Μετρητές για μέτρηση του αριθμού των σημείων που σχεδιάζονται στη γραφική
//παράσταση για κάθε μέγεθος
int count;
int count2;
int count3;

//τα αναγνωριστικά που αντιστοιχούν στη συσκευή του μικροελεγκτή που έχουμε
private static final int targetVendorID = 4292;
private static final int targetProductID = 1;
private static final String ACTION_USB_PERMISSION =
"com.android.example.USB_PERMISSION";

PendingIntent mPermissionIntent;
private UsbManager usbManager;
private UsbDevice deviceFound;
private UsbDeviceConnection usbDeviceConnection;
private UsbInterface usbInterfaceFound = null;

```

```

private UsbEndpoint endpointOut = null;
private UsbEndpoint endpointIn = null;
private UsbEndpoint endpointIn2 = null;
private UsbEndpoint endpointIn3 = null;
private Button startbt;
private Button stopbt;
UsbInterface usbInterface;

//private String file1 = "temperature";
//private String file2 = "humidity";
//private String file3 = "Co2";

//ορισμός και αρχικοποίηση των αρχείων για αποθήκευση των δεδομένων των
//αισθητήρων που διαβάζονται και σχεδιάζονται

File file1;
File file2;
File file3;
File fileTemperature;
File filehumidity;
File fileCo2;
FileOutputStream stream1 = null;
FileOutputStream stream2 = null;
FileOutputStream stream3 = null;

@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_plot);

```

```

Intent intent4 = getIntent();

Intent intent = getIntent();

String action = intent.getAction();

startbt=(Button) findViewById(R.id.start);

stopbt=(Button) findViewById(R.id.stop);

startbt.setEnabled(true);

stopbt.setEnabled(false);

//δημιουργία UsbManager που ελέγχει και διαχειρίζεται την κατάσταση της //σύνδεσης
Usb και επικοινωνεί με τη συνδεδεμένη συσκευή

usbManager = (UsbManager) getSystemService(Context.USB_SERVICE);

//εντολή για να παραμένει η οθόνη ενεργή και να μη σβήνει όσο εκτελείται το thread //για
τη λήψη των μετρήσεων και τη γραφική απεικόνισή τους σε πραγματικό χρόνο

getWindow().addFlags(WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON);

//final int j = intent4.getIntExtra(sensorsUiActivity.My_choice, 0);

//Διάβασμα παραμέτρων (επιλογές χρήστη) από την προηγούμενη Activity
//(sensorsUiActivity)

final String rb1name = intent4.getStringExtra(sensorsUiActivity.My_array1);

final String rb2name = intent4.getStringExtra(sensorsUiActivity.My_array2);

final String rb3name = intent4.getStringExtra(sensorsUiActivity.My_array3);

final Integer ref1 = intent4.getIntExtra(sensorsUiActivity.My_ref1, 0);

final Integer ref2 = intent4.getIntExtra(sensorsUiActivity.My_ref2,0);

final Integer ref3 = intent4.getIntExtra(sensorsUiActivity.My_ref3,0);

//καταχώρηση του mUsbReceiver που όταν συνδέεται μια συσκευή usb ζητάει άδεια //από
το χρήστη για να επικοινωνήσει μαζί της

```

```

mPermissionIntent = PendingIntent.getBroadcast(this, 0, new Intent(
    ACTION_USB_PERMISSION), 0);

IntentFilter filter = new IntentFilter(ACTION_USB_PERMISSION);
registerReceiver(mUsbReceiver, filter);

//καταχώρηση του mUsbDeviceReceiver για να ανταποκρίνεται ανάλογα όταν //συνδέεται
ή αποσυνδέεται μια συσκευή usb

registerReceiver(mUsbDeviceReceiver, new IntentFilter(
    UsbManager.ACTION_USB_DEVICE_ATTACHED));
registerReceiver(mUsbDeviceReceiver, new IntentFilter(
    UsbManager.ACTION_USB_DEVICE_DETACHED));

connectUsb();

running=true;

//μορφοποίηση για την εμφάνιση ημερομηνίας και ώρας
final SimpleDateFormat formatter = new
SimpleDateFormat("yyyy_MM_dd_HH_mm_ss");

//Ανταποκρίνεται όταν ο χρήστης πατήσει το κουμπί Start. Πρώτα στέλνεται στο
//μικροελεγκτή η εντολή CMD_ON (0x01 σε byte) για να ξεκινήσει τη λειτουργία //του, και
μετά ξεκινάει το νήμα που διαβάζει τις τιμές των αισθητήρων από το usb, //τις απεικονίζει
γραφικά σε πραγματικό χρόνο και τις αποθηκεύει σε αρχεία.

startbt.setOnClickListener(new View.OnClickListener() {

    //PowerManager pm = (PowerManager)
getSystemService(Context.POWER_SERVICE);

    // PowerManager.WakeLock wakeLock = pm.newWakeLock(
    // pm.FULL_WAKE_LOCK, "My wakelook");

    //wakeLock.acquire();

```

```

@Override

public void onClick(View v) {

    //εντολή για να ξεκινήσει ο μικροελεγκτής να λαμβάνει μετρήσεις από τους //αισθητήρες
    και να τις στέλνει στο Android

        sendMcuCommand(CMD_ ON);

    //δημιουργία φακέλου που θα αποθηκεύονται τα αρχεία με τις μετρήσεις

        file1=new
File(Environment.getExternalStorageDirectory(),"MEASUREMENTS");

        if (!file1.exists()) {

            file1.mkdirs();

        }

        //file2=new File(Environment.getExternalStorageDirectory(),"HUMIDITY");

    //λήψη και αποθήκευση της τρέχουσας χρονικής στιγμής (ημερομηνία και ώρα) που
    //ξεκίνησαν να λαμβάνονται και να καταγράφονται οι μετρήσεις

        Date now = new Date();

        String fileDate = formatter.format(now) ;

    //ονομασία των αρχείων που αποθηκεύονται τα δεδομένα και πρόσθεση στο όνομά //τους
    της χρονικής στιγμής που ξεκίνησε η λήψη και καταγραφή των μετρήσεων

        // δημιουργία txt αρχείων

        fileTemperature=new File(file1,"TEMPERATURE "+fileDate+".txt");

        filehumidity=new File(file1,"HUMIDITY "+fileDate+".txt");

        fileCo2=new File(file1,"Co2 "+fileDate+".txt");

    // δημιουργία binary αρχείων

        fileTemperatureBin=new File(file1,"TEMPERATURE "+fileDate);

        filehumidityBin=new File(file1,"HUMIDITY "+fileDate);

        fileCo2Bin=new File(file1,"Co2 "+fileDate);

```

```

        startbt.setEnabled(false);

        stopbt.setEnabled(true);

//εκκίνηση νήματος
Thread threadMcuDataRead =
        new Thread(new Runnable() {
                @Override
                public void run() {

//δημιουργία των τριών buffer μεγέθους 64 bytes –ένας για κάθε endpoint//
                ByteBuffer buffer = ByteBuffer.allocate(64);
                ByteBuffer buffer2 = ByteBuffer.allocate(64);
                ByteBuffer buffer3 = ByteBuffer.allocate(64);

                //δημιουργία και εκκίνηση των ασύγχρονων αιτήσεων Usb (πακέτα UsbRequest) στα
                //αντίστοιχα endpoints, για διάβασμα των δεδομένων
                UsbRequest request = new UsbRequest();
                UsbRequest request2 = new UsbRequest();
                UsbRequest request3 = new UsbRequest();
                request.initialize(usbDeviceConnection, endpointIn);
                request2.initialize(usbDeviceConnection, endpointIn2);
                request3.initialize(usbDeviceConnection, endpointIn3);

//μετρητές για καταμέτρηση των σημείων που έχουν σχεδιαστεί στη γραφική //παράσταση
                count = 0;
                count2 = 0;
                count3 = 0;

                while (true) {

//τοποθέτηση σε ουρά των αιτήσεων Usb
                request.queue(buffer, 64);
                request2.queue(buffer2, 64);
                request3.queue(buffer3, 64);

```

*//αφαίρεση των παλιών σημείων όταν συμπληρωθεί ένας καθορισμένος αριθμός*

```
if ((count==1280)||(count2==1280)||(count3==1280)){  
    for(int k=0;k<1280;k++) {  
        line.clearOldPoints(0);  
    }  
    // try {  
        //Thread.sleep(1000);  
    // } catch (InterruptedException e) {  
        //e.printStackTrace();  
    //}
```

*//μηδενισμός των μετρητών του πλήθους των σημείων*

```
count=0;  
count2=0;  
count3=0;  
}  
// try {  
    // Thread.sleep(1000);  
// } catch (InterruptedException e) {  
    //e.printStackTrace();  
//}
```

*//διάβασμα των δεδομένων από το πρώτο endpoint-αν έχει ολοκληρωθεί επιτυχώς η αίτηση  
//αίτηση Usb request και η μεταφορά - και τοποθέτησή τους στο buffer*

```
if (usbDeviceConnection.requestWait() == request) {  
    if (rb3name != null) {  
        byte[] dataR = new byte[64];  
        for (indx = 0; indx < 64; indx++) {  
            dataR[indx] = buffer.get(indx);  
            if (dataR[indx] != IGNORE_00) {
```

*//σχεδίαση της σταθερής τιμής αναφοράς που έχει επιλέξει ο χρήστης για το πρώτο  
//μέγεθος προς μέτρηση*

```
Point pref1=new Point(count,ref3);
```

```
line.addNewPointsCn1(pref1);
```

*//διάβασμα με τη σειρά των στοιχείων του buffer, μετατροπή τους από μορφή byte  
//πρώτα σε αριθμό και μετά σε σημείο (Point),μαθηματική επεξεργασία - //κανονικοποίηση  
και πρόσθεση-σχεδίαση του σημείου στην αντίστοιχη γραφική //παράσταση*

```
y1 = (int) dataR[indx];
```

```
//if (rb1name != null) {
```

```
Point p = new Point(count, normalization  
(y1,max1,min1,axisMax1,axisMin1));
```

```
line.addNewPoints(p);
```

*//καταμέτρηση σημείων που σχεδιάζονται*

```
count++;
```

```
//}
```

*//γράψιμο-αποθήκευση κάθε σημείου στο αντίστοιχο αρχείο*

```
try {
```

```
stream1=new FileOutputStream(fileTemperature,true);
```

```
stream1.write(dataR[indx]);
```

```
stream1.flush();
```

```
stream1.close();
```

```
// Toast.makeText(getBaseContext(),"file  
saved",Toast.LENGTH_SHORT).show();
```

```
}
```

```
catch (Exception e) {
```

```
e.printStackTrace();
```

```
}
```

```
}
```

```
}
```

```

    }
}
// try {
    // Thread.sleep(1000);
//} catch (InterruptedException e) {
    // e.printStackTrace();
// }

//διάβασμα των δεδομένων από το δεύτερο endpoint-αν έχει ολοκληρωθεί επιτυχώς η
//αίτηση Usb request και η μεταφορά - και τοποθέτησή τους στο buffer

if (usbDeviceConnection.requestWait() == request2) {
    if (rb2name != null) {
        byte[] dataR2 = new byte[64];
        for (indx = 0; indx < 64; indx++) {
            dataR2[indx] = buffer2.get(indx);
            if (dataR2[indx] != IGNORE_00) {

//σχεδίαση της σταθερής τιμής αναφοράς που έχει επιλέξει ο χρήστης

                Point pref2=new Point(count,ref2);

                line.addNewPointsCn2(pref2);

//διάβασμα με τη σειρά των στοιχείων του buffer, μετατροπή τους από μορφή byte
//πρώτα σε αριθμό και μετά σε σημείο (Point), και πρόσθεση-σχεδίαση του σημείου //στην
αντίστοιχη γραφική παράσταση

                y2 = (int) dataR2[indx];

                Point p2 = new Point(count2, normalazation
(y2,max2,min2,axisMax2,axisMin2));

                line.addNewPoints2(p2);

                count2++;

//γράψιμο-αποθήκευση κάθε σημείου στο αντίστοιχο αρχείο

                try {

                    stream2=new FileOutputStream(filehumidity,true);

```

```

        stream2.write(dataR2[indx]);

        stream2.flush();

        stream2.close();

        // Toast.makeText(getBaseContext(),"file
saved",Toast.LENGTH_SHORT).show();

    }

    catch (Exception e) {

        e.printStackTrace();

    }

}

}

}

}

//try {

    // Thread.sleep(1000);

//} catch (InterruptedException e) {

    // e.printStackTrace();

//}

//διάβασμα των δεδομένων από το τρίτο endpoint-αν έχει ολοκληρωθεί επιτυχώς η
//αίτηση Usb request και η μεταφορά - και τοποθέτησή τους στο buffer

    if (usbDeviceConnection.requestWait() == request3) {

        if (rb1name != null) {

            byte[] dataR3 = new byte[64];

            for (indx = 0; indx < 64; indx++) {

                dataR3[indx] = buffer3.get(indx);

                if (dataR3[indx] != IGNORE_00) {

//σχεδίαση της σταθερής τιμής αναφοράς που έχει επιλέξει ο χρήστης

                    Point pref3=new Point(count,ref1);

```



```

        view.repaint();
    } else {
        if ((count > 128 && count <= 192)||count2 > 128 && count2 <=
192)||count3 > 128 && count3 <= 192)) {
            line.adjust1();
            line.adjust2();
            line.adjust4();
        } else {
            line.adjust2();
        }
    }
}
try {
    Thread.sleep(1000);
} catch (InterruptedException e) {
    e.printStackTrace();
}

//τερματισμός του νήματος (thread) όταν σταματήσει η εφαρμογή από τον χρήστη ή
//αποσυνδεθεί το usb

    if(!running) return;
}
}
});

//κλείσιμο των αρχείων που αποθηκεύτηκαν τα δεδομένα
try {
    stream1.close();
    stream2.close();
    stream3.close();
}
catch (Exception e) {

```

```

        e.printStackTrace();
    }

    //εκκίνηση του νήματος
        threadMcuDataRead.start();

    });

    //view=line.getView(this);

    //setContent View(view);

    //startTestThread();
}

@Override

public void onResume() {
    super.onResume();
    //if (view == null) {

    //ρυθμίζει το layout ώστε να εμφανίζεται στην οθόνη η γραφική παράσταση
        LinearLayout layout = (LinearLayout) findViewById(R.id.chart);

    //εμφανίζει τη γραφική παράσταση
        view=line.getView(this);

        layout.addView(view, new LayoutParams
            (LayoutParams.FILL_PARENT, LayoutParams.FILL_PARENT));
    }

    //Εναλλακτική μέθοδος για στήσιμο και εγκατάσταση της σύνδεσης Usb, εύρεση των
    //endpoints και της αντίστοιχης interface, καθώς και αποστολή μηνύματος τύπου //control
    transfer για ρύθμιση των παραμέτρων της γραμμής επικοινωνίας

```

```

//private void setDevice(UsbDevice device) {

    //usbInterfaceFound = null;

    // endpointOut = null;

    //endpointIn = null;

    //for (int i = 0; i < device.getInterfaceCount(); i++)

    //{

//εύρεση των endpoints

        //UsbInterface usbif = device.getInterface(0);

        //UsbEndpoint[] tIn = new UsbEndpoint[3];

        //UsbEndpoint tOut = null;

        //int tEndpointCnt = usbif.getEndpointCount();

        //if (tEndpointCnt >= 2) {

            //for (int j = 0; j < tEndpointCnt; j++) {

                //if (usbif.getEndpoint(j).getType()

                    // == UsbConstants.USB_ENDPOINT_XFER_BULK) {

                    //if (usbif.getEndpoint(j).getDirection()

                        // == UsbConstants.USB_DIR_OUT) {

                            // tOut = usbif.getEndpoint(j);

                        //} else if (usbif.getEndpoint(j).getDirection()

                            // == UsbConstants.USB_DIR_IN) {

                                // tIn[j] = usbif.getEndpoint(j);

                            // }

                        //}

                    //}

                //}

            //}

// εύρεση της interface

        //if (tOut != null && tIn != null) {

            // This interface have both USB_DIR_OUT

            // and USB_DIR_IN of USB_ENDPOINT_XFER_BULK

```

```

//usbInterfaceFound = usbif;

//endpointOut = tOut;

//endpointIn = tIn[0];

//endpointIn2 = tIn[1];

// endpointIn3 = tIn[2];

//}

// }

//}

//if (usbInterfaceFound == null) {

//return;

//}

//deviceFound = device;

//άνοιγμα της σύνδεσης

//if (device != null) {

//UsbDeviceConnection connection =

//usbManager.openDevice(device);

//if (connection != null &&

//connection.claimInterface(usbInterfaceFound, true)) {

//αποστολή μηνύματος τύπου controlTransfer για καθορισμό των παραμέτρων της

//σύνδεσης

//connection.controlTransfer(0x21, 34, 0, 0, null, 0, 0);

//connection.controlTransfer(0x21, 32, 0, 0,

// new byte[] { (byte) 0x80, 0x25, 0x00,

// 0x00, 0x00, 0x00, 0x08 },

// 7, 0);

//usbDeviceConnection = connection;

```

```

// Thread thread = new Thread();

// thread.start();

///} else {

    //usbDeviceConnection = null;

//}

//}

//}

```

*//Μέθοδος για αποστολή εντολών ελέγχου στο μικροελεγκτή. Στέλνεται στον //μικροελεγκτή ένα μήνυμα που αποτελείται από δύο bytes. Το πρώτο byte είναι η //τιμή '0xFF' που χρησιμοποιείται για το συγχρονισμό της επικοινωνίας και για να δηλώσει //την έναρξη των δεδομένων, και το δεύτερο byte είναι η κύρια εντολή ελέγχου που //μπορεί να έχει τιμή **0x01** σε byte , που είναι η εντολή για να αρχίσει ο μικροελεγκτής να //παίρνει μετρήσεις από τους αισθητήρες και να τις στέλνει, και την τιμή **0x03** σε byte , η //οποία είναι η εντολή για να σταματήσει ο μικροελεγκτής και να τερματίσει τη σύνδεση //usb με το Android*

```

private void sendMcuCommand(int control) {

    synchronized (this) {

        if (usbDeviceConnection != null) {

            byte[] message = new byte[2];

//byte για το συγχρονισμό της επικοινωνίας

            message[0] = SYNC_WORD;

//εντολή ελέγχου του μικροελεγκτή

            message[1] = (byte)control;

//αποστολή του μηνύματος στον μικροελεγκτή στο endpoint Out

            usbDeviceConnection.bulkTransfer(endpointOut,

                message, message.length, 0);

            Log.d(TAG, "sendMcuCommand: " + String.valueOf(control));

        }
    }
}

```

```

    }
}

@Override
protected void onStart() {
    super.onStart();
    view=line.getView(this);

    //setContentView(view);
}

//επιστροφή πίσω στην οθόνη της προηγούμενης Activity (sensorsUiActivity)
@Override
public void onBackPressed() {
    //μηδενισμός –reset της μεταβλητής choice για επαναχρησιμοποίηση στην
//sensorsUiActivity
    int choice = 0;

    Intent intent6 = new Intent(this, sensorsUiActivity.class);

    intent6.putExtra(My_choice, choice);

    startActivity(intent6);
}

```

*//Μέθοδος που εκτελείται όταν ο χρήστης πατήσει το κουμπί Stop. Στέλνεται στο //μικροελεγκτή εντολή (συγκεκριμένα η εντολή CMD\_ OFF που έχει τιμή σε byte //0x03) για να σταματήσει να λαμβάνει και να στέλνει μετρήσεις από τους αισθητήρες και //να τερματίσει την επικοινωνία usb με το Android.Ταυτόχρονα σταματάει και το //thread που διαβάζει διαρκώς τις τιμές των αισθητήρων και τις σχεδιάζει ( //threadMcuDataRead) ,αλλάζοντας τη τιμή της μεταβλητής running σε false.*

```

public void OnButtonStop(View view2)
{
    sendArduinoCommand(CMD_ OFF);
}

```

```

        //releaseUsb();

        //wakeLock.release();

        //σταματάει την οθόνη του Android από το να παραμένει ενεργή
getWindow().clearFlags(WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON);

        running=false;

        releaseUsb();

    }

```

*//δημιουργούμε έναν BroadcastReceiver για τα //ACTION\_USB\_DEVICE\_ATTACHED και //ACTION\_USB\_DEVICE\_DETACHED , που θα ανταποκρίνεται ανάλογα στο αν //συνδέεται ή αποσυνδέεται μια συσκευή usb στο Android.*

```

private final BroadcastReceiver mUsbDeviceReceiver = new BroadcastReceiver() {

    @Override

    public void onReceive(Context context, Intent intent) {

        String action = intent.getAction();

        //Όταν συνδεθεί μια συσκευή usb

        if (UsbManager.ACTION_USB_DEVICE_ATTACHED.equals(action)) {

            deviceFound = (UsbDevice) intent

                .getParcelableExtra(UsbManager.EXTRA_DEVICE);

            //επιχείρησε να συνδεθείς με τη συσκευή

            connectUsb();

            //Αν αποσυνδεθεί η συνδεδεμένη συσκευή

        } else if (UsbManager.ACTION_USB_DEVICE_DETACHED.equals(action)) {

            UsbDevice device = (UsbDevice) intent

                .getParcelableExtra(UsbManager.EXTRA_DEVICE);

            if (device != null) {

                //και είναι η συσκευή usb που επικοινωνούμε μαζί της

                if (device == deviceFound) {

                    //αποσύνδεσε τη συσκευή και κλείσε τη σύνδεση

                    releaseUsb();

```

```

        }
    }
}
};

```

*//δημιουργούμε έναν BroadcastReceiver που όταν συνδεθεί στο σύστημα Android μια //συσκευή usb, εμφανίζει ένα παράθυρο διαλόγου στον χρήστη ζητώντας του άδεια //για να επικοινωνήσει και να συνδεθεί μαζί της και λαμβάνει το intent που περιέχει //το EXTRA\_PERMISSION\_GRANTED, το οποίο είναι μια boolean τιμή που //αντιπροσωπεύει την απάντηση.*

```

private final BroadcastReceiver mUsbReceiver = new BroadcastReceiver() {

    @Override

    public void onReceive(Context context, Intent intent) {

        String action = intent.getAction();

        if (ACTION_USB_PERMISSION.equals(action)) {

            synchronized (this) {

                //λαμβάνουμε την απάντηση του χρήστη

                UsbDevice device = (UsbDevice) intent

                    .getParcelableExtra(UsbManager.EXTRA_DEVICE);

                //αν έχουμε άδεια από το χρήστη και επιτρέπεται να επικοινωνήσουμε με τη συσκευή

                if (intent.getBooleanExtra(

                    UsbManager.EXTRA_PERMISSION_GRANTED, false)) {

                    //τότε συνδεόμαστε με τη συσκευή

                    if (device != null) {

                        connectUsb();

                    }

                } else {

                    //εμφάνιση μηνύματος σφάλματος

                    Log.d(TAG, "permission denied for device " + device);

```

```

        }
    }
}
};

private boolean setupUsbComm()
{
    //Προκειμένου να εγκατασταθεί η σύνδεση πρέπει να καθορίσουμε κάποιες
    //συγκεκριμένες παραμέτρους οι οποίες ρυθμίζουν τα χαρακτηριστικά της σύνδεσης. //Οι
    //παραμέτροι αυτοί στέλνονται με μεταφορά τύπου control transfer στο endpoint 0 //από τον
    //host στο device.

    final int RQSID_SET_LINE_CODING = 0x20;           //επιτρέπει στον host να
    //καθορίσει τις ιδιότητες μορφοποίησης (formatting) χαρακτήρων τυπικής //ασύγχρονης
    //γραμμής. Πιο συγκεκριμένα το SET_LINE_CODING είναι τύπος //request που ρυθμίζει τις
    //παραμέτρους της γραμμής επικοινωνίας όπως είναι η //ταχύτητα επικοινωνίας ,το μέγεθος
    //των δεδομένων, το bit ελέγχου ισοτιμίας (parity //bit), το μέγεθος του stop bit

    final int RQSID_SET_CONTROL_LINE_STATE = 0x22;    // Το
    //SET_CONTROL_LINE_STATE καθορίζει τις ρυθμίσεις του σήματος ελέγχου //(RTS,
    //DTR) της γραμμής επικοινωνίας.

    boolean success = false;

    //δημιουργία UsbManager που διαχειρίζεται τη σύνδεση και την κατάστασή της
    UsbManager manager = (UsbManager) getSystemService(Context.USB_SERVICE);
    Boolean permitToRead = manager.hasPermission(deviceFound);

    if (permitToRead) {
        // «άνοιγμα» της σύνδεσης
        usbDeviceConnection = manager.openDevice(deviceFound);
        if (usbDeviceConnection != null) {

```

```

usbDeviceConnection.claimInterface(usbInterfaceFound, true);

//μεταφορά τύπου control transfer στο endpoint 0 για στήσιμο της σύνδεσης και //ρύθμιση
των παραμέτρων της

usbDeviceConnection.controlTransfer(0x21, // τύπος αιτήματος //(requestType)
για τη μεταφορά - καθορίζει την κατεύθυνση της μεταφοράς

        RQSID_SET_CONTROL_LINE_STATE, //request ID:
//SET_CONTROL_LINE_STATE για κατηγορία συσκευών CDC

        0, // value

        0, // index, ο δείκτης που δείχνει από ποια θέση του buffer ξεκινάει η
//μεταφορά δεδομένων

        null, // buffer, είναι μηδέν γιατί δεν θέλουμε να λάβουμε ή να
//στείλουμε δεδομένα για να αποθηκευτούν σε αυτή τη μεταφορά

        0, // length, το μέγεθος των δεδομένων που στέλνουμε ή λαμβάνουμε
//σε αυτή τη μεταφορά

        0); // timeout σε milliseconds

// baud rate = 9600, καθορισμός ταχύτητας σύνδεσης

// 8 data bit

// 1 stop bit

byte[] encodingSetting = new byte[] { (byte) 0x80, 0x25, 0x00,

        0x00, 0x00, 0x00, 0x08 };

usbDeviceConnection.controlTransfer(0x21, // requestType

        RQSID_SET_LINE_CODING, // SET_LINE_CODING

        0, // value

        0, // index

        encodingSetting, // buffer

        7, // length, έχει την τιμή 7 γιατί ο buffer έχει 7 bytes

        0); // timeout

//εκκίνηση νέου νήματος που αναλαμβάνει τη λειτουργία της επικοινωνίας, της
//εγκατάστασης της σύνδεσης usb και τη ρύθμιση των παραμέτρων της

Thread thread = new Thread();

```

```

        thread.start();
    }
} else {
    manager.requestPermission(deviceFound, mPermissionIntent); //ζητάει
//προσωρινή άδεια για να μπορέσει το συγκεκριμένο πακέτο να έχει πρόσβαση στη //συσκευή
}
return success;
}

private void connectUsb() {
// βρίσκουμε τα διαθέσιμα endpoints και την αντίστοιχη interface
    searchEndPoint();

    if (usbInterfaceFound != null) {
// και μετά εγκαθιστούμε τη σύνδεση
        setupUsbComm();
    }
}

private void releaseUsb() {
    if (usbDeviceConnection != null) {
        if (usbInterfaceFound != null) {
            usbDeviceConnection.releaseInterface(usbInterface);
            usbInterfaceFound = null;
        }
//κλείνει η σύνδεση usb
        usbDeviceConnection.close();
        usbDeviceConnection = null;
    }
}

```

*//θέτουμε null τα στοιχεία που απαρτίζουν τη σύνδεση (endpoints, interface,κτλ), //ώστε όταν συνδεθεί κάποια καινούργια συσκευή usb (ή ξανά η ίδια), να μπορούμε //να εγκαταστήσουμε νέα σύνδεση αν θέλουμε.*

```
        deviceFound = null;

        usbInterfaceFound = null;

        endpointIn = null;

        endpointOut = null;

        endpointIn2 = null;

        endpointIn3 = null;

    }

private void searchEndPoint()
{
    usbInterfaceFound = null;

    endpointOut = null;

    endpointIn = null;

    // Αναζητούμε συσκευή με συγκεκριμένο VendorID και ProductID

    if (deviceFound == null) {

        UsbManager          manager          =          (UsbManager)
        getSystemService(Context.USB_SERVICE);

        //απαριθμούμε τις συνδεδεμένες συσκευές usb στον δίαυλο και παίρνουμε ένα //HashMap (δομή δεδομένων της Java) που περιέχει όλες τις συνδεδεμένες συσκευές //usb και αναζητούμε αυτή που μας ενδιαφέρει ψάχνοντας για συγκεκριμένο //VendorID και ProductID

        HashMap<String, UsbDevice> deviceList = manager.getDeviceList();

        Iterator<UsbDevice> deviceIterator = deviceList.values().iterator();

        while (deviceIterator.hasNext()) {

//εξετάζουμε μία προς μία τις συνδεδεμένες συσκευές usb

            UsbDevice device = deviceIterator.next();

            if (device.getVendorId() == targetVendorID) {
```

```

        if (device.getProductId() == targetProductID) {

            //βρέθηκε η συσκευή usb που μας ενδιαφέρει να επικοινωνήσουμε μαζί της ( δηλαδή //το
            κύκλωμα του μικροελεγκτή)

                deviceFound = device;

            }

        }

    }

}

if (deviceFound == null) {

} else {

    // Αναζητούμε τώρα-αφού βρέθηκε η συσκευή- UsbInterface με Endpoint
    //USB_ENDPOINT_XFER_BULK (δηλαδή τύπου bulk) , με αμφίδρομη κατεύθυνση
    //δηλαδή USB_DIR_OUT και USB_DIR_IN

        //for (int i = 0; i < deviceFound.getInterfaceCount(); i++) {

            UsbInterface usbif = deviceFound.getInterface(0);

            UsbEndpoint[] tIn = new UsbEndpoint[3];

            UsbEndpoint tOut = null;

            int tEndpointCnt = usbif.getEndpointCount();

            if (tEndpointCnt >= 2) {

                for (int j = 0; j < tEndpointCnt; j++) {

                    //βρίσκουμε το είδος του endpoint

                        if (usbif.getEndpoint(j).getType()

                            == UsbConstants.USB_ENDPOINT_XFER_BULK) {

                                //βρίσκουμε την κατεύθυνση του endpoint

                                    if (usbif.getEndpoint(j).getDirection()

                                        == UsbConstants.USB_DIR_OUT) {

                                            tOut = usbif.getEndpoint(j);

                                        } else if (usbif.getEndpoint(j).getDirection()

                                            == UsbConstants.USB_DIR_IN) {

```

```

        tIn[j] = usbif.getEndpoint(j);
    }
}
}

//if (tOut != null && tIn != null) {

    // Η συγκεκριμένη interface έχει κατεύθυνση και USB_DIR_OUT και //USB_DIR_IN
    (αμφίδρομη) για USB_ENDPOINT_XFER_BULK και είναι αυτή //που θέλουμε για να
    επικοινωνήσουμε

    //Ορίζουμε και τα αντίστοιχα endpoints που βρέθηκαν

        usbInterfaceFound = usbif;

        endpointOut = tOut;

        endpointIn = tIn[0];

        endpointIn2 = tIn[1];

        endpointIn3 = tIn[2];

        //}

    // }

}

}

}

//μέθοδος για κανονικοποίηση των σημείων πριν την απεικόνισή τους
private float normalazation (float value,int Max,int Min, int axisMax, int axisMin){
    float norm_value= (((value-Min)*(axisMax-axisMin))/(Max-Min))+axisMin;

    return norm_value;
}
}

```

### **LineGraph.java**

```
import android.content.Context;
```

```

import android.graphics.Color;

import com.jjoe64.graphview.GraphView;

import org.achartengine.ChartFactory;

import org.achartengine.GraphicalView;

import org.achartengine.chart.PointStyle;

import org.achartengine.model.TimeSeries;

import org.achartengine.model.XYMultipleSeriesDataset;

import org.achartengine.renderer.XYMultipleSeriesRenderer;

import org.achartengine.renderer.XYSeriesRenderer;

//Δημιουργεί ένα αντικείμενο (object) που αναπαριστά τη συνολική γραφική //παράσταση
και ρυθμίζει, με τις μεθόδους που διαθέτει, όλες τις λεπτομέρειες της //γραφικής απεικόνισης,
όπως είναι για παράδειγμα το είδος του γραφήματος, οι //άξονες, το χρώμα, το μέγεθος των
γραμμών, το στυλ των σημείων, τα περιθώρια, τη //δυνατότητα ζουμ κτλ

public class LineGraph {

//Το αντικείμενο απεικόνισης View που περικλείει τη γραφική παράσταση

    private GraphicalView view;

//δημιουργούμε για κάθε μέγεθος ξεχωριστά τις σειρές για γραφήματα χρόνου
//TimeSeries

    private TimeSeries dataset=new TimeSeries("CO2");

    private TimeSeries dataset2=new TimeSeries("humidity");

    private TimeSeries dataset3=new TimeSeries("temperature");

//δημιουργεί μια συνολική σειρά που να περιλαμβάνει και τις τρεις παραπάνω σειρές
χρόνου

    private XYMultipleSeriesDataset mDataset= new XYMultipleSeriesDataset();

//δημιουργούμε για κάθε σειρά ένα renderer που ρυθμίζει τη μορφή και τον τρόπο με
//τον οποίο σχεδιάζεται και παρουσιάζεται η κάθε σειρά

    private XYSeriesRenderer renderer=new XYSeriesRenderer();

    private XYSeriesRenderer renderer2=new XYSeriesRenderer();

    private XYSeriesRenderer renderer3=new XYSeriesRenderer();

```

*//δημιουργούμε σειρές χρόνου για την απεικόνιση των σταθερών τιμών αναφοράς που  
//έχει επιλέξει ο χρήστης για κάθε μέγεθος*

```
private TimeSeries datasetRef1=new TimeSeries("");
```

```
private TimeSeries datasetRef2=new TimeSeries("");
```

```
private TimeSeries datasetRef3=new TimeSeries("");
```

*//δημιουργούμε έναν renderer για καθεμία από τις σειρές των τιμών αναφοράς*

```
private XYSeriesRenderer rendererRef1=new XYSeriesRenderer();
```

```
private XYSeriesRenderer rendererRef2=new XYSeriesRenderer();
```

```
private XYSeriesRenderer rendererRef3=new XYSeriesRenderer();
```

*//δημιουργούμε ένα συνολικό renderer για όλη τη γραφική απεικόνιση στο σύνολό //της  
(που περιλαμβάνει όλες τις σειρές) και καθορίζει τη συνολική παρουσίαση*

```
private XYMultipleSeriesRenderer mRenderer=new XYMultipleSeriesRenderer();
```

*//μέθοδος που δημιουργεί το αντικείμενο LineGraph (constructor method)*

```
public LineGraph()
```

```
{
```

*//προσθέτουμε τις σειρές χρόνου για κάθε μέγεθος στη συνολική σειρά*

```
mDataset.addSeries(dataset);
```

```
mDataset.addSeries(dataset2);
```

```
mDataset.addSeries(dataset3);
```

*//προσθέτουμε τις σειρές των τιμών αναφοράς στη συνολική σειρά*

```
mDataset.addSeries(datasetRef1);
```

```
mDataset.addSeries(datasetRef2);
```

```
mDataset.addSeries(datasetRef3);
```

*//καθορίζουμε τις λεπτομέρειες της γραφικής παρουσίασης κάθε σειράς (για κάθε  
//μέγεθος) όπως το χρώμα, το στυλ των σημείων, το μέγεθος των γραμμών, τα //περιθώρια  
κτλ με τη βοήθεια των μεθόδων κάθε renderer*

```
renderer.setColor(Color.GREEN);
```

```
renderer.setPointStyle(PointStyle.CIRCLE);
```

```
renderer.setFillPoints(true);
```

```
renderer.setDisplayBoundingPoints(true);
```

```
renderer.setLineWidth(3);

renderer2.setColor(Color.BLUE);
renderer2.setPointStyle(PointStyle.CIRCLE);
renderer2.setFillPoints(true);
renderer2.setDisplayBoundingPoints(true);
renderer2.setLineWidth(3);

renderer3.setColor(Color.RED);
renderer3.setPointStyle(PointStyle.CIRCLE);
renderer3.setFillPoints(true);
renderer3.setDisplayBoundingPoints(true);
renderer3.setLineWidth(3);

// καθορίζουμε τις λεπτομέρειες της γραφικής παρουσίασης των σειρών των τιμών
//αναφοράς
rendererRef1.setColor(Color.GREEN);
rendererRef1.setPointStyle(PointStyle.POINT);
rendererRef1.setFillPoints(false);
rendererRef1.setDisplayBoundingPoints(false);

rendererRef2.setColor(Color.BLUE);
rendererRef2.setPointStyle(PointStyle.POINT);
rendererRef2.setFillPoints(false);
rendererRef2.setDisplayBoundingPoints(false);

rendererRef3.setColor(Color.RED);
rendererRef3.setPointStyle(PointStyle.POINT);
rendererRef3.setFillPoints(false);
rendererRef3.setDisplayBoundingPoints(false);
```

```
//rendererRef3.setLineWidth(3);
```

*//καθορίζουμε τις λεπτομέρειες της γραφικής παρουσίασης της συνολικής σειράς και  
//ρυθμίζουμε τη μορφή της γραφικής απεικόνισης στο σύνολό της (τους άξονες, τα  
//περιθώρια, τις διαστάσεις κτλ)*

```
//renderer.
```

```
//enable zoom
```

```
//mRenderer.setMarginsColor(Color.argb(0x00, 0xff, 0x00, 0x00));
```

```
mRenderer.setZoomButtonsVisible(true);
```

```
mRenderer.setXTitle("time");
```

```
mRenderer.setGridColor(Color.WHITE);
```

```
mRenderer.setShowGrid(true);
```

```
mRenderer.setXLabels(25);
```

```
mRenderer.setYLabels(10);
```

```
mRenderer.setApplyBackgroundColor(true);
```

```
mRenderer.setBackgroundColor(Color.GRAY);
```

```
mRenderer.setYAxisMax(100);
```

```
//mRenderer.setXAxisMin(mRenderer.getXAxisMin() + 1);
```

```
// mRenderer.setXAxisMax(mRenderer.getXAxisMax() + 1);
```

*//προσθέτουμε τον renderer κάθε σειράς στο συνολικό renderer που αντιστοιχεί στην  
//συνολική σειρά και καθορίζει τη μορφή της γραφικής απεικόνισης στο σύνολό της*

```
mRenderer.addSeriesRenderer(renderer);
```

```
mRenderer.addSeriesRenderer(renderer2);
```

```
mRenderer.addSeriesRenderer(renderer3);
```

*//προσθέτουμε τον renderer κάθε σειράς τιμών αναφοράς στο συνολικό renderer*

```
mRenderer.addSeriesRenderer(rendererRef1);
```

```
mRenderer.addSeriesRenderer(rendererRef2);
```

```
mRenderer.addSeriesRenderer(rendererRef3);
```

```

}

public GraphicalView getView(Context context)
{
    view= ChartFactory.getLineChartView(context,mDataset,mRenderer);
    //view.repaint();

    //επιστρέφει τη View που περικλείει τη συνολική γραφική απεικόνιση ώστε να
    //εμφανιστεί στην οθόνη του χρήστη

    return view;
}

//προσθέτει ένα καινούργιο σημείο στη σειρά που αντιστοιχεί στο ποσοστό co2
public void addNewPoints(Point p)
{
    dataset.add(p.getX(),p.getY());
}

//προσθέτει ένα καινούργιο σημείο στη σειρά που αντιστοιχεί στην υγρασία
public void addNewPoints2(Point p)
{
    dataset2.add(p.getX(),p.getY());
}

//προσθέτει ένα καινούργιο σημείο στη σειρά που αντιστοιχεί στη θερμοκρασία
public void addNewPoints3(Point p)
{
    dataset3.add(p.getX(),p.getY());
}

//προσθέτει ένα καινούργιο σταθερό σημείο σε κάθε σειρά που αντιστοιχεί στην
//επιλεγμένη από το χρήστη τιμή αναφοράς καθενός μεγέθους
public void addNewPointsCn1(Point p)

```

```

{
    datasetRef1.add(p.getX(),p.getY());
}

public void addNewPointsCn2(Point p)
{
    datasetRef2.add(p.getX(),p.getY());
}

public void addNewPointsCn3(Point p)
{
    datasetRef3.add(p.getX(),p.getY());
}

```

*//οι παρακάτω μέθοδοι προσαρμόζουν τη συνολική γραφική απεικόνιση στην οθόνη  
//ώστε να ανανεώνεται και να μετακινείται αυτόματα προς τα δεξιά (auto scrolling) //καθώς  
σχεδιάζονται καινούργια σημεία*

```

public GraphicalView adjust1()
{

```

*//καθορίζουμε ένα διάστημα εντός του οποίου προβάλλεται ο μέγιστος αριθμός των  
//σημείων που θέλουμε στην οθόνη*

```

    double newMax = mDataset.getSeriesAt(0).getMaxX();
    double newMin= mDataset.getSeriesAt(0).getMinX();
    mRenderer.setXAxisMax(newMax);
    mRenderer.setXAxisMin(newMax-128);
    view.repaint();
    return view;
}

```

```

public GraphicalView adjust2()
{

```

*//Ανανεώνει το μέγιστο και το ελάχιστο x του διαστήματος εντός του οποίου  
//προβάλλεται ο επιθυμητός αριθμός σημείων ώστε να γίνεται scrolling*

```

mRenderer.setXAxisMax(mRenderer.getXAxisMax() + 64);
mRenderer.setXAxisMin(mRenderer.getXAxisMin() + 64);
view.repaint();
return view;
}
public GraphicalView adjust3()
{
double newMax = mDataset.getSeriesAt(1).getMaxX();
double newMin= mDataset.getSeriesAt(1).getMinX();
mRenderer.setXAxisMax(newMax);
mRenderer.setXAxisMin(newMax-128);
view.repaint();
return view;
}
public GraphicalView adjust4()
{
double newMax = mDataset.getSeriesAt(2).getMaxX();
double newMin= mDataset.getSeriesAt(2).getMinX();
mRenderer.setXAxisMax(newMax);
mRenderer.setXAxisMin(newMax-128);
view.repaint();
return view;
}

```

*//μέθοδος που αφαιρεί από τη γραφική παράσταση τα παλιότερα σχεδιασμένα σημεία κάθε σειράς και την καθαρίζει*

```

public void clearOldPoints(int count){
//αφαιρούμε τα παλιότερα σημεία για κάθε σειρά ξεχωριστά
dataset.remove(count);
dataset2.remove(count);
}

```

```

        dataset3.remove(count);

//αφαιρούμε τα παλιότερα σημεία από καθεμία από τις σειρές των τιμών αναφοράς

        datasetRef1.remove(count);

        datasetRef2.remove(count);

        datasetRef3.remove(count);

    }

}

```

### **Point.java**

*//Αντιπροσωπεύει ένα σημείο της γραφικής παράστασης με συντεταγμένες x και y //στους άξονες*

```

public class Point {

    int x;

    int y;

//δημιουργεί ένα σημείο με συντεταγμένες x και y

    public Point(int x,int y){

        this.x=x;

        this.y=y;

    }

//επιστρέφει τη συντεταγμένη (τη διάσταση x) από ένα δοσμένο σημείο

    public int getX() {

        return x;

    }

//επιστρέφει τη συντεταγμένη (τη διάσταση y) από ένα δοσμένο σημείο

    public int getY() {

        return y;

    }

}

```

## B2. Κώδικας XML για σχεδιασμό των οθονών χρήστη (User Interface)

### activity\_main.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    tools:context=".MainActivity"
    android:theme="@style/Base.TextAppearance.AppCompat.Title"
    android:background="#ff9c65ff">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:text="ΕΠΙΛΟΓΗ ΛΕΙΤΟΥΡΓΙΑΣ"
        android:id="@+id/textView"
        android:layout_above="@+id/button"
        android:layout_centerHorizontal="true"
        android:layout_marginBottom="105dp" />
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Έλεγχος αισθητήρων"
        android:id="@+id/button"
        android:layout_centerVertical="true"
        android:layout_alignLeft="@+id/textView"
        android:layout_alignStart="@+id/textView"
```

```
android:layout_alignRight="@+id/textView"  
android:layout_alignEnd="@+id/textView"  
android:onClick="sensorsGui"/>
```

```
<Button
```

```
android:layout_width="wrap_content"  
android:layout_height="wrap_content"  
android:text="Έλεγχος πιεζοηλεκτρικού"  
android:id="@+id/button2"  
android:layout_below="@+id/button"  
android:layout_alignLeft="@+id/button"  
android:layout_alignStart="@+id/button"  
android:layout_marginTop="50dp"  
android:layout_alignRight="@+id/button"  
android:layout_alignEnd="@+id/button"  
android:onClick="piezoelectrGui"/>
```

```
<ImageView
```

```
android:layout_width="wrap_content"  
android:layout_height="wrap_content"  
android:id="@+id/imageView"  
android:layout_alignParentTop="true"  
android:layout_centerHorizontal="true"  
android:background="@drawable/lab_logo"/>
```

```
<TextClock
```

```
android:layout_width="wrap_content"  
android:layout_height="wrap_content"  
android:id="@+id/textClock"  
android:layout_alignParentBottom="true"  
android:layout_alignParentLeft="true"
```

```

        android:layout_alignParentStart="true" />
<Button
    style="?android:attr/buttonStyleSmall"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Exit"
    android:id="@+id/button4"
    android:layout_alignParentBottom="true"
    android:layout_alignParentRight="true"
    android:layout_alignParentEnd="true" />
</RelativeLayout>

```

### **activity\_sensors\_ui.xml**

```

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    tools:context="com.example.nakama.myapplication_new.sensorsUiActivity">
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textAppearance="?android:attr/textAppearanceLarge"
    android:text="Μετρήσεις Αισθητήρων"
    android:id="@+id/textView5"
    android:layout_alignParentTop="true"
    android:layout_alignParentLeft="true"

```

```

        android:layout_alignParentStart="true" />
<RadioButton
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Θερμοκρασία"
    android:id="@+id/temperature"
    android:onClick="onRadioButtonClicked"
    android:layout_below="@+id/textView5"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true"
    android:layout_marginTop="44dp" />
<RadioButton
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Υγρασία"
    android:id="@+id/humidity"
    android:onClick="onRadioButtonClicked"
    android:layout_below="@+id/spinner"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true" />
<RadioButton
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="co2"
    android:id="@+id/co2"
    android:onClick="onRadioButtonClicked"
    android:layout_below="@+id/spinner2"
    android:layout_alignParentLeft="true"

```

```

        android:layout_alignParentStart="true" />
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Σχεδίαση Γραφικής Παράστασης"
    android:id="@+id/button3"
    android:singleLine="true"
    android:onClick="OnButtonPlot"
    android:layout_alignParentBottom="true"
    android:layout_centerHorizontal="true" />
<ImageView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/imageView2"
    android:background="@drawable/plot"
    android:layout_above="@+id/button3"
    android:layout_below="@+id/spinner2"
    android:layout_alignParentRight="true"
    android:layout_alignParentEnd="true"
    android:layout_toRightOf="@+id/button3"
    android:layout_toEndOf="@+id/button3" />
<ImageView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/imageView3"
    android:background="@drawable/logo4"
    android:layout_alignParentTop="true"
    android:layout_above="@+id/textView4"

```

```

        android:layout_alignParentRight="true"
        android:layout_alignParentEnd="true"
        android:layout_toRightOf="@+id/button3"
        android:layout_toEndOf="@+id/button3" />
<Button
    style="?android:attr/buttonStyleSmall"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="AD Settings"
    android:id="@+id/button5"
    android:onClick= "SetSettings"
    android:layout_above="@+id/button3"
    android:layout_alignRight="@+id/textView7"
    android:layout_alignEnd="@+id/textView7" />
<Button
    style="?android:attr/buttonStyleSmall"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="PlotSettings"
    android:id="@+id/button8"
    android:layout_alignTop="@+id/button5"
    android:layout_toLeftOf="@+id/button5"
    android:layout_toStartOf="@+id/button5"
    android:onClick= "SetPlotSettings" />
</RelativeLayout>

```

### **activity\_plot\_settings.xml**

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
```

```

xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_parent"

    android:layout_height="match_parent"
android:paddingLeft="@dimen/activity_horizontal_margin"

    android:paddingRight="@dimen/activity_horizontal_margin"

    android:paddingTop="@dimen/activity_vertical_margin"

    android:paddingBottom="@dimen/activity_vertical_margin"

    tools:context="com.example.nakama.myapplication_new.PlotSettingsActivity"

    android:background="#ffdbffdd">
<NumberPicker

    android:layout_width="wrap_content"

    android:layout_height="wrap_content"

    android:id="@+id/numberPicker"

    android:layout_alignParentTop="true"

    android:layout_alignParentLeft="true"

    android:layout_alignParentStart="true"

    android:background="#fff415b" />
<NumberPicker

    android:layout_width="wrap_content"

    android:layout_height="wrap_content"

    android:id="@+id/numberPicker2"

    android:layout_below="@+id/numberPicker"

    android:layout_alignParentLeft="true"

    android:layout_alignParentStart="true"

    android:background="#fff415b" />
<NumberPicker

    android:layout_width="wrap_content"

    android:layout_height="wrap_content"

    android:id="@+id/numberPicker3"

    android:layout_below="@+id/numberPicker2"

```

```
android:layout_alignParentLeft="true"
android:layout_alignParentStart="true"
android:background="#ffff415b" />
```

```
<NumberPicker
```

```
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:id="@+id/numberPicker4"
android:layout_alignTop="@+id/numberPicker"
android:layout_toRightOf="@+id/numberPicker"
android:layout_toEndOf="@+id/numberPicker"
android:background="#ff04c0ff" />
```

```
<NumberPicker
```

```
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:id="@+id/numberPicker5"
android:layout_below="@+id/numberPicker"
android:layout_toRightOf="@+id/numberPicker2"
android:layout_toEndOf="@+id/numberPicker2"
android:background="#ff04c0ff" />
```

```
<NumberPicker
```

```
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:id="@+id/numberPicker6"
android:layout_below="@+id/numberPicker5"
android:layout_toRightOf="@+id/numberPicker3"
android:layout_toEndOf="@+id/numberPicker3"
android:background="#ff04c0ff" />
```

```
<NumberPicker
```

```

    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/numberPicker7"
    android:layout_alignParentTop="true"
    android:layout_toRightOf="@+id/numberPicker4"
    android:layout_toEndOf="@+id/numberPicker4"
    android:background="#ff5aff1b" />
<NumberPicker
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/numberPicker8"
    android:layout_below="@+id/numberPicker4"
    android:layout_toRightOf="@+id/numberPicker5"
    android:layout_toEndOf="@+id/numberPicker5"
    android:background="#ff5aff1b" />
<NumberPicker
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/numberPicker9"
    android:layout_below="@+id/numberPicker5"
    android:layout_toRightOf="@+id/numberPicker6"
    android:layout_toEndOf="@+id/numberPicker6"
    android:background="#ff5aff1b" />
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textAppearance="?android:attr/textAppearanceSmall"
    android:text="τιμή αναφοράς"

```

```

    android:id="@+id/textView17"
    android:layout_alignTop="@+id/numberPicker7"
    android:layout_toRightOf="@+id/numberPicker7"
    android:layout_toEndOf="@+id/numberPicker7"
    android:textColorHighlight="#ff000000" />
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textAppearance="?android:attr/textAppearanceSmall"
    android:text="max"
    android:id="@+id/textView18"
    android:layout_below="@+id/numberPicker7"
    android:layout_toRightOf="@+id/numberPicker8"
    android:layout_toEndOf="@+id/numberPicker8" />
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textAppearance="?android:attr/textAppearanceMedium"
    android:text="min"
    android:id="@+id/textView19"
    android:layout_below="@+id/numberPicker8"
    android:layout_toRightOf="@+id/numberPicker8"
    android:layout_toEndOf="@+id/numberPicker8" />
<Button
    style="?android:attr/buttonStyleSmall"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Πύθμιση"

```

```

        android:id="@+id/button7"

        android:layout_alignParentBottom="true"

        android:layout_centerHorizontal="true"

        android:onClick="SavePlotSettings"/>
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textAppearance="?android:attr/textAppearanceSmall"
    android:text="χρονική υποδιαίρεση άξονα x"
    android:id="@+id/textView20"
    android:layout_above="@+id/textView18"
    android:layout_toRightOf="@+id/textView17"
    android:layout_toEndOf="@+id/textView17" />
<Spinner
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/spinner4"
    android:layout_below="@+id/textView20"
    android:layout_alignLeft="@+id/textView20"
    android:layout_alignStart="@+id/textView20"
    android:layout_alignRight="@+id/textView20"
    android:layout_alignEnd="@+id/textView20" />
</RelativeLayout>

```

### **activity\_plot.xml**

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"

```

```

android:layout_height="fill_parent"

android:orientation="vertical"

android:weightSum="1">

<RelativeLayout

    android:id="@+id/relativeLayout1"

    android:layout_width="fill_parent"

    android:layout_height="wrap_content"

    android:background="@android:color/black" >

    <Button

        android:id="@+id/start"

        android:layout_width="wrap_content"

        android:layout_height="wrap_content"

        android:text="start"

        android:layout_alignParentLeft="true" />

    <Button

        android:id="@+id/stop"

        android:layout_width="wrap_content"

        android:layout_height="wrap_content"

        android:text="stop"

        android:layout_alignParentRight="true"

        android:onClick="OnButtonStop"/>

</RelativeLayout>

<LinearLayout android:id="@+id/chart"

    android:orientation="horizontal"

    android:layout_width="fill_parent" android:layout_height="0dip"

    android:layout_weight="1"/>

</LinearLayout>

```