ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

Ανάπτυξη Εφαρμογής για Αναγνώριση Ήχων μέσα σε ένα Έξυπνο Σπίτι

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

της

ΚΥΡΙΑΚΗΣ – ΔΑΦΝΗΣ ΓΑΛΕΤΖΑ

Επιβλέπουσα : Θεοδώρα Βαρβαρίγου
Καθηγήτρια Ε.Μ.Π.

Αθήνα, Ιούνιος 2017

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ
ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

Ανάπτυξη Εφαρμογής για Αναγνώριση Ήχων μέσα σε ένα Έξυπνο Σπίτι

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΚΥΡΙΑΚΗ - ΔΑΦΝΗ ΓΑΛΕΤΖΑ

Επιβλέπουσα : Θεοδώρα Βαρβαρίγου
Καθηγήτρια Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή στις 06 / 06 / 2017.

.....	.....	.....
Θεοδώρα Βαρβαρίγου	Εμμανουήλ Βαρβαρίγος	Δημήτριος Ασκούνης
Καθηγήτρια Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.

.....

Κυριακή – Δάφνη Γαλετζά

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

Περίληψη

Ο σκοπός της παρούσας διπλωματικής εργασίας είναι η μελέτη, ο σχεδιασμός και η ανάπτυξη μιας εφαρμογής για την αναγνώριση των ήχων που αναπαράγονται σε όποιο Έξυπνο Σπίτι διαμένουν άτομα με ιδιαιτερότητες στην ακοή. Αποτελεί μια πρώτη προσπάθεια ενσωμάτωσης της προσβασιμότητας των ατόμων με ειδικές ανάγκες στο Διαδίκτυο των Πραγμάτων. Δίνεται έμφαση στην διευκόλυνση της καθημερινής τους ζωής μέσω της ειδοποίησής τους σχετικά με το είδος των ήχων που δεν μπορούν να αντιληφθούν στο περιβάλλον της κατοικίας τους.

Αρχικά, γίνεται προσπάθεια διείσδυσης στο όραμα του Διαδικτύου των Πραγμάτων σε συνδυασμό με τον όρο του Έξυπνου Σπιτιού. Πραγματοποιείται έρευνα για τον καθορισμό των παραμέτρων της διπλωματικής και παρουσιάζεται μια ολοκληρωμένη πρόταση η οποία αποσκοπεί στην πλήρη αντιμετώπιση των προβλημάτων που αντιμετωπίζουν βαρήκοα / κωφά άτομα σχετικά με την αντίληψη των ήχων του σπιτιού. Στη συνέχεια, γίνεται ανάλυση της τρέχουσας τεχνολογικής κατάστασης και επισημαίνονται οι νέες προκλήσεις και περιοχές έρευνας.

Επιπροσθέτως, παρουσιάζεται μια πρώτη απόπειρα ανάπτυξης εφαρμογής που θέτει τα βασικά θεμέλια για την επίτευξη του σκοπού της προαναφερθείσας ολοκληρωμένης πρότασης. Για την εφαρμογή αυτή χρησιμοποιήθηκαν τα δύο ευρέως γνωστά λογισμικά Android Studio και Node-RED και παρουσιάζονται τόσο η ανάπτυξη της Πλευράς του Εξυπηρετητή όσο και η ανάπτυξη της Πλευράς του Χρήστη. Τέλος, παρατίθονται τα συμπεράσματα που εξάχθησαν κατά την εκπόνηση της διπλωματικής καθώς και προτάσεις για την βελτίωση της εφαρμογής που δημιουργήθηκε.

Λέξεις Κλειδιά: Διαδίκτυο των Πραγμάτων, Έξυπνο Σπίτι, Αναγνώριση Ήχων, Android Studio, Node-RED, Κώφωση, Βαρηκοΐα, Ειδοποίηση Ήχων, Planner

Abstract

The purpose of this thesis is to research, design and develop an application to recognise any sounds that are produced in a Smart Home where people with hearing-impairement live in. This is a first attempt to integrate accessibility of people with disabilities into the Internet of Things. Emphasis is placed on facilitating their daily lives by alerting them about the kind of sounds they are unable to hear in their home environment.

To begin with, there is an attempt to enter in the vision of the Internet of Things in combination with Smart Homes, by studying their key characteristics and referencing in all the technologies that could support them. Research is being carried out to determine the requirements of the thesis and a comprehensive and extensive proposal is presented, aiming to fully address the problems encountered by hearing-impaired people while living in their own houses. Then, there follows a state- of-the-art analysis as well as the identification of new potential challenges and research areas.

Moreover, there is a first attempt to develop an application that sets the foundations for achieving the ultimate goal of the aforementioned integrated proposal. The application is running on Android Studio and Node-RED, two very well-known operating systems. Afterwards, both Back-End and Front-End sides are presented. Finally, conclusions are drawn and a presentation of suggestions for improving the already existing implementation, follows.

Key Words: Internet of Things, Smart Home, Sound Recognition, Android Studio, Node-RED, Deafness, Hearing Impairment, Sound Notification, Planner

Ευχαριστίες

Η παρούσα διπλωματική εκπονήθηκε στη σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Εθνικού Μετσοβίου Πολυτεχνείου, στα πλαίσια των δραστηριοτήτων του Τομέα Επικοινωνιών, Ηλεκτρονικής και Συστημάτων Πληροφορικής. Ωστόσο, η πραγματοποίησή της δε θα ήταν εφικτή χωρίς την καθοριστική βοήθεια κάποιων ανθρώπων.

Θα ήθελα να ευχαριστήσω πολύ θερμά την καθηγήτρια Ε.Μ.Π. και επιβλέπουσα της διπλωματικής μου εργασίας κ. Θεοδώρα Βαρβαρίγου, για την ευκαιρία που μου έδωσε να εργαστώ σε ένα τόσο σύγχρονο και ταυτόχρονα ενδιαφέρον αντικείμενο και για την εμπιστοσύνη που μου έδειξε.

Επίσης, θα ήθελα να ευχαριστήσω τους Γιώργο Κουσιουρή και Ορφέα Βουτυρά για την αμέριστη υπομονή τους και βοήθειά τους κατά τη διάρκεια εκπόνησης της διπλωματικής καθώς και για τον χρόνο που αφιέρωσαν για να συζητήσουμε τους όποιους προβληματισμούς.

Τέλος, θα ήθελα να εκφράσω την ευγνωμοσύνη μου στους κοντινούς μου ανθρώπους οι οποίοι με υποστήριξαν, συμπαραστάθηκαν και μου έδωσαν δύναμη κατά τη διάρκεια των ακαδημαϊκών χρόνων μου. Ιδιαίτερες ευχαριστίες στην μητέρα μου και τον πατέρα μου οι οποίοι με έκαναν να είμαι αυτή που είμαι, στον αδερφό μου που πάντα πίστευε σε εμένα και φυσικά στον Τάσο χωρίς τον οποίο δεν θα βρισκόμουν στο σημείο που είμαι τώρα.

Κυριακή – Δάφνη Γαλετζά

Ιούνιος 2017

Περιεχόμενα

1 Εισαγωγή

Διανύουμε μια περίοδο όπου η σύγχρονη τεχνολογία έχει πλέον εισχωρήσει στην ζωή όλων των ανθρώπων και την έχει επηρεάσει βαθύτατα σε ποικίλες πτυχές της. Η εξέλιξή της προχωράει με ταχύτατους ρυθμούς και αυτό επιβεβαιώνεται από το γεγονός πως πράγματα που τα θεωρούμε σήμερα δεδομένα, μόλις λίγες δεκαετίες πριν αποτελούσαν σημαντικά τεχνολογικά επιτεύγματα, όπως στην περίπτωση της ασπρόμαυρης τηλεόρασης. Σήμερα, οι επιστήμονες επικεντρώνονται στην τεχνολογική ανάπτυξη συγκεκριμένων τομέων που θα συμβάλλουν στην ποιοτική βελτίωση της καθημερινότητας των πολιτών όπως η Ιατρική, τα Ευφυή Αυτοκίνητα, η Τεχνητή Νοημοσύνη, η Εικονική Πραγματικότητα, κτλ.

Από την αρχή της ιστορίας, το ανθρώπινο γένος αναζητούσε ένα φυσικό καταφύγιο στο οποίο θα αισθανόταν ασφαλές από τους εξωτερικούς κινδύνους και τα ακραία καιρικά φαινόμενα. Με την πάροδο του χρόνου, το φυσικό καταφύγιο αυτό μετατράπηκε στην έννοια του σπιτιού το οποίο πέρα από την ασφάλεια, πλέον αποσκοπεί στην εξυπηρέτηση ανέσεων παρά ασφάλειας. Οι ρυθμοί, όμως, που επιβάλλει η σύγχρονη κοινωνία σε συνδυασμό με την ραγδαία ανάπτυξη της τεχνολογίας δημιούργησαν την ανάγκη της αυτοματοποίησης των λειτουργιών ενός σπιτιού. Έτσι, γεννήθηκε ο όρος «Έξυπνο Σπίτι» (Smart Home).

1.1 Το Έξυπνο Σπίτι

Ο όρος «Έξυπνο Σπίτι» περιγράφει τις ηλεκτρολογικές και μηχανολογικές εγκαταστάσεις που τοποθετούνται σε σπίτια με σκοπό να προσφέρουν άνεση, ασφάλεια και εξοικονόμηση ενέργειας και χρημάτων στους ενοίκους/ιδιοκτήτες [1]. Οι έξυπνες εγκαταστάσεις αλληλοεπιδρούν με το περιβάλλον χρησιμοποιώντας ασύρματα ή ενσύρματα μέσα επικοινωνίας μέσω των οποίων ανταλλάσσουν δεδομένα προκειμένου να διεξάγουν κάποιες λειτουργίες, όπως η ρύθμιση του φωτισμού και της θερμοκρασίας ενός χώρου.

Τα έξυπνα συστήματα, εκτός από τις ηλεκτρολογικές και μηχανολογικές εγκαταστάσεις, δύναται να ελέγχουν και οικιακές συσκευές καθώς και συσκευές πολυμέσων, δημιουργώντας έτσι ένα ενοποιημένο σύστημα. Συνδυάζοντας όλες αυτές τις ανεξάρτητες, αρχικά, εγκαταστάσεις σε μια κοινή βάση, καθίσταται εφικτός ο πλήρης έλεγχος της κατοικίας.

Η τελευταία γενιά των Έξυπνων Σπιτιών είναι πλέον σε θέση να ενσωματώνει μεγάλες ποσότητες υπολογιστικής ισχύος για να παρακολουθεί τις δραστηριότητες των ενοίκων του και να προβλέπει τις ανάγκες τους. Μια υποκατηγορία των κατοίκων που επωφελούνται σε μεγάλο βαθμό από τις αυξημένες παροχές ενός Smart Home είναι οι άνθρωποι με ειδικές ανάγκες.

1.2 Ανεξάρτητη διαβίωση ατόμων με ειδικές ανάγκες

Σύμφωνα με έρευνα που πραγματοποιήθηκε από την Υπηρεσία Απογραφής των ΗΠΑ το 2011 [2], ο πληθυσμός των ΗΠΑ με ηλικίες άνω των 65 ετών έφτασε το 14% του πληθυσμού το 2010, με περίπου το 16% των ενηλίκων να έχουν κάποια αναπηρία. Είναι αναμφισβήτητο πως τα ποσοστά αυτά είναι πολύ υψηλά με αποτέλεσμα πολλές κυβερνήσεις να εκλαμβάνουν τα Έξυπνα Σπίτια ως βιώσιμη εναλλακτική λύση για τη μείωση της οικονομικής επιβάρυνσης της υποστήριξης των ατόμων με ειδικές ανάγκες καθώς και των ηλικιωμένων.

Πιο συγκεκριμένα, η Αμερικάνικη Ακαδημία Ακουολογίας δημοσίευσε ενημερωτική ανάρτηση όπου έδειχνε πως η έλλειψη ακοής αποτελεί ένα σημαντικό ζήτημα δημόσιας υγείας μιας και είναι το τρίτο πιο συχνό πρόβλημα που εμφανίζεται σε ανθρώπους, μετά από την αρθρίτιδα και την καρδιοπάθεια [3].

Ακόμη, παρατηρήθηκε μέσω μελέτης από τον τομέα Γεροντολογίας της Οξφόρδης, πως τα άτομα με θέματα ακοής δυσκολεύονται στην καθημερινότητά τους να ανταποκριθούν στα ηχητικά ερεθίσματα του περιβάλλοντος με αποτέλεσμα να δυσχεραίνεται η ποιότητα της ζωής τους σε κάποιο βαθμό [4]. Η παρούσα ομάδα-στόχος (target-group) επωφελείται, λοιπόν, σε μεγάλο βαθμό από την εφαρμογή που αναπτύχθηκε στα πλαίσια της διπλωματικής, όπως θα δείξουμε στη συνέχεια.

1.3 Σκοπός Διπλωματικής

Σκοπός της διπλωματικής είναι η οικοδόμηση ενός προσιτού συστήματος που θα μπορεί να ενσωματωθεί σε ένα Έξυπνο Σπίτι και θα ειδοποιεί άμεσα κωφούς / βαρήκοους ανθρώπους σχετικά με τους ήχους που λαμβάνουν χώρα γύρω τους, ώστε να ανταποκριθούν καταλλήλως.

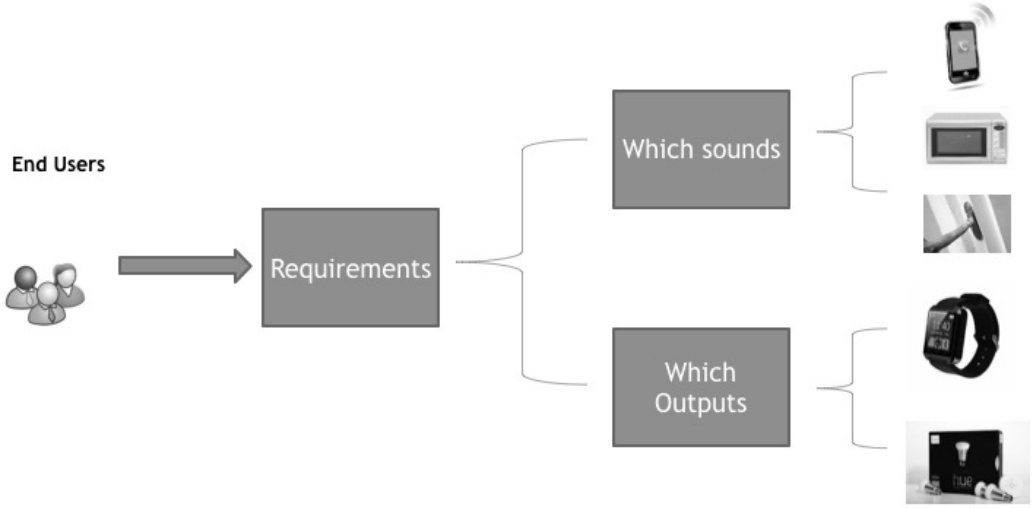
Μελετώνται οι τρόποι ανίχνευσης των ήχων (monitoring), επεξεργασίας / αναγνώρισής τους καθώς και οι μέθοδοι επικοινωνίας τους στον τελικό χρήστη μέσω κάποιας άλλης μορφής, είτε οπτικώς (visualization) είτε μέσω αφής.

Η βασική ιδέα του συστήματος αυτού είναι να τοποθετηθούν λάμπες LED σε εμφανή σημεία σε κάθε δωμάτιο του σπιτιού και, αναλόγως της κατηγορίας του ήχου που ηχογραφείται από μικρόφωνα / αισθητήρες που έχουν τοποθετηθεί μέσα στο σπίτι, θα δονούνται το Έξυπνο Κινητό (Smart Phone) και το Έξυπνο Ρολόι (Smart Watch), τα οποία κουβαλάει πάνω του ο χρήστης έτσι ώστε να ειδοποιείται για το όποιο ηχητικό συμβάν.

Αξιοποιούνται εργαλεία του Διαδικτύου των Πραγμάτων (Internet of Things) βάσει των οποίων αναπτύχθηκε μία εφαρμογή για κινητά με λογισμικό Android η οποία εξυπηρετεί τους παραπάνω σκοπούς. Γίνεται χρήση γνωστών μεθόδων επεξεργασίας και ανάλυσης ήχων, όπως ο μετασχηματισμός FFT (Fast Fourier Transform), καθώς και γνωστών αλγορίθμων σύγκρισης (Bray – Curtis Distance).

Αρχικά, καθορίστηκαν οι παράμετροι της διπλωματικής οι οποίες χωρίζονται σε δύο κατηγορίες. Η πρώτη κατηγορία είναι οι ήχοι που είναι σημαντικότερο να εντοπιστούν μέσα στο σπίτι και

μετέπειτα να αναλυθούν μέσω της εφαρμογής (Στάδιο Αναγνώρισης). Η δεύτερη κατηγορία είναι οι τρόποι με τους οποίους μπορεί να ειδοποιηθεί ο χρήστης για την αναπαραγωγή των ήχων αυτών (Στάδιο Επικοινωνίας).



Σχήμα 1.1: Παράμετροι διπλωματικής

Στη συνέχεια, δημιουργήθηκε η εφαρμογή μέσω του λογισμικού Android Studio στην οποία αναπτύχθηκε το περιβάλλον χρήστη (Front-end) το οποίο συνδέεται μέσω του πρωτοκόλλου HTTP με το περιβάλλον του διακομιστή (Back-end). Για την φιλοξενία του Back-end χρησιμοποιήθηκε το λογισμικό Node-RED, ένα από τα πιο γνωστά εργαλεία για την εικονική διασύνδεση μεταξύ συσκευών στο IoT.

Τέλος, στην εφαρμογή προστέθηκε η δυνατότητα για τον χρήστη να προσθέτει σε μια προκαθορισμένη Βάση Δεδομένων τους τοπικούς ήχους του σπιτιού του οποίους ηχογραφεί εκείνη την ώρα ο ίδιος, έτσι ώστε να βελτιωθεί η αποδοτικότητά της και η ακρίβειά της.

1.4 Συνοπτική Παρουσίαση Προβλήματος και Λύσης

Από την στιγμή που ένας άνθρωπος με θέμα βαρηκοΐας αδυνατεί να ανταποκριθεί άμεσα και επαρκώς σε ηχητικά ερεθίσματα του περιβάλλοντος, είναι αναμφισβήτητο πως θα επηρεαστεί και η καθημερινότητά του μέσα στο σπίτι που κατοικεί.

Κάποια από τα πιθανά σενάρια που θα λάβουν χώρα μέσα στο σπίτι και θα παρουσιαστεί πρόβλημα ανταπόκρισης από το βαρήκοο / κωφό άτομο είναι:

- Ήχος κουδουνιού ή χτύπημα πόρτας
- Συναγερμός σπιτιού ή φωτιάς
- Κλάμα παιδιού ή μωρού
- Ξυπνητήρι

Τα παραπάνω σενάρια που παρουσιάζονται αποτελούν δεδομένες καταστάσεις της καθημερινότητας ενός ακούοντα αλλά για τα άτομα με θέμα ακοής απαιτείται επιπρόσθετη βοήθεια. Μέρος αυτής της επιπρόσθετης βοήθειας μπορεί να αποτελέσει η εφαρμογή που αναπτύχθηκε στα πλαίσια της διπλωματικής.

Η εφαρμογή μας εκμεταλλεύεται ηχητικούς αισθητήρες οι οποίοι βρίσκονται τοποθετημένοι στην οικία του χρήστη και ηχογραφούν συνεχώς ήχους, οι οποίοι στη συνέχεια επεξεργάζονται και αναγνωρίζονται από την εφαρμογή μας. Εν τέλει, ο χρήστης ενημερώνεται για το είδος του ήχου μέσω λαμπτήρων που αναβοσβήνουν στο κάθε δωμάτιο με διαφορετικό χρώμα και συχνότητα, καθώς και μέσω ειδοποιήσεων στο Smart Phone και Smart Watch του.

1.5 Δομή της Διπλωματικής

Η διπλωματική εργασία δομείται ως εξής:

Κεφάλαιο 2: Στο κεφάλαιο αυτό περιγράφεται η διαδικασία με την οποία καθορίστηκαν οι παράμετροι της διπλωματικής καθώς και η ολοκληρωμένη πρόταση που αποσκοπεί στην πλήρη αντιμετώπισή τους. Στην συνέχεια, γίνεται μια εισαγωγή στα είδη των αισθητήρων και τα χαρακτηριστικά τους. Τέλος, παρουσιάζονται οι ενεργοποιητές και μέθοδοι οπτικοποίησης που επιλέχθησαν.

Κεφαλαιο 3: Στο κεφάλαιο αυτό πραγματοποιειται ανάλυση των όρων "Διαδίκτυο των Πραγμάτων" και "Έξυπνο Σπίτι". Περιγράφεται η αρχιτεκτονική του Έξυπνου Σπιτιού καθώς και η δικτύωση που πραγματοποιείται μεταξύ των συσκευών. Στην συνέχεια θίγεται το θέμα της

προστασίας των προσωπικών δεδομένων και εν τέλει, παραθέτονται οι υπηρεσίες που προσφέρει ένα Έξυπνο Σπίτι.

Κεφάλαιο 4: Στο παρόν κεφάλαιο γίνεται ανάλυση της τρέχουσας τεχνολογικής κατάστασης παρουσιάζοντας τις ήδη υπάρχουσες εφαρμογές και έξυπνες συσκευές για ανθρώπους με ιδιαιτερότητα στην ακοή. Επίσης, επισημαίνονται οι νέες προκλήσεις και περιοχές έρευνας.

Κεφάλαιο 5: Το κεφάλαιο αυτό αφορά την ανάπτυξη της Πλευράς του Εξυπηρετητή. Πραγματοποιείται περιγραφή των εργαλείων Node-RED και Planner και στην συνέχεια αναλύεται εκτενώς η αρχιτεκτονική του συστήματος της εφαρμογής μας.

Κεφάλαιο 6: Το έκτο κεφάλαιο είναι αφιερωμένο στην ανάπτυξη Πλευράς του Χρήστη. Περιγράφονται οι βασικές αρχές κατασκευής του Περιβάλλοντος Εργασίας και παρουσιάζεται λεπτομερώς η διεπαφή της εφαρμογής μας.

Τέλος, παρουσιάζονται τα τελικά συμπεράσματα τις εργασίας καθώς και ορισμένες ιδέες για μελλοντικές επεκτάσεις της.

2 Ανάλυση Προβλήματος

2.1 Καθορισμός Ήχων προς αναγνώριση στην οικία

Το βασικό πρόβλημα που αντιμετωπίστηκε εξ' αρχής για την ανάπτυξη της εφαρμογής ήταν ο καθορισμός των ήχων που θα έπρεπε να οριστούν προς αναγνώριση μέσα στην Βάση Δεδομένων (Database).

Ένας από τους αποδοτικότερους τρόπους για να επιτεχθεί ο καθορισμός των ήχων αυτών ήταν να οργανωθεί μια συνάθροιση «καταιγισμού ιδεών» (brainstorming) ατόμων με μερική ή ολική έλλειψη ακοής (και άρα οικείων προς τα καθημερινά προβλήματα που μας ενδιαφέρει να επιλύσουμε), ώστε να μοιραστούν τις εμπειρίες τους και ιδέες τους. Έτσι, θα λάμβανε χώρα η καταγραφή εκτενούς και έγκυρης πληροφορίας της οποίας η μετέπειτα ανάλυση σε βάθος θα οδηγούσε στο επιθυμητό αποτέλεσμα.

Αρχικά, να αναφερθεί πως και εγώ είμαι βαρήκοη οπότε είμαι ήδη εξοικειωμένη σχετικά με τις ανάγκες που παρουσιάζουν άτομα με την ίδια ιδιαιτερότητα με εμένα και έχω εμπειρία. Για πληρότητα, όμως, αποφάσισα να οργανώσω ένα brainstorming όπου θα συμμετείχαν γνωστοί μου που αντιμετωπίζουν τις ίδιες συνθήκες. Ενημερώθηκαν εξαρχής για τον σκοπό της συνάντησης, την μορφή της καθώς και για την επιθυμία μου να υπάρξει συλλογή πληροφοριών σχετικά με τους ήχους για τους οποίους θα ήθελαν να έχουν την δυνατότητα να ειδοποιούνται από ένα Έξυπνο Σπίτι. Η πρόταση για το brainstorming έγινε δεκτή από πέντε άτομα, μιας και η προοπτική δημιουργίας αυτού του είδους εφαρμογής τους φάνηκε ελκυστική και πως θα ωφελούσε εμάς τους ίδιους μιας και θα βελτίωνε την καθημερινότητά μας.

Η συνάθροιση έλαβε χώρο σε κατάλληλα διαμορφωμένο χώρο και οι πέντε συμμετέχοντες είχαν μέσο όρο ηλικίας είκοσι πέντε ετών. Αρχικά, υπενθύμισα τον σκοπό της συνάντησης, παρουσίασα τις λειτουργίες που μπορούν να υποστηριχθούν από προγραμματιστική σκοπιά και στην συνέχεια

καθοδήγησα τον καταιγισμό ιδεών. Ο κάθε συμμετέχοντας με την σειρά του περιέγραφε τις δυσκολίες που αντιμετωπίζει καθημερινά από την έλλειψη αντίληψης των ήχων μέσα στην κατοικία του καθώς και ποιούς θεωρεί σημαντικότερους να συμπεριληφθούν στην εφαρμογή της παρούσας διπλωματικής.

Στην συνέχεια, αφού καθορίστηκαν οι απαιτούμενοι ήχοι, τέθηκε από εμένα το ερώτημα για το πώς θα επιθυμούσαν οι ίδιοι να ειδοποιούνται όταν ένας ήχος θα ακουγόταν. Το τελικό συμπέρασμα ήταν πως η καταλληλότερη προσέγγιση θα ήταν να τοποθετηθούν αισθητήρες στο κάθε δωμάτιο της κατοικίας του χρήστη, οι οποίοι θα ηχογραφούν συνεχόμενα το περιβάλλον, καθώς και επίσης συγκεκριμένος αριθμός λαμπτήρων οι οποίοι θα αναβοσβήνουν σε διαφορετικό χρώμα καθώς και συχνότητα, ανάλογα με το είδος και την σημαντικότητα του ήχου που θα λαμβάνει χώρα. Επιπροσθέτως, παράλληλα με το αναβόσβημα των λαμπτήρων, εξέφρασαν την επιθυμία τους να υπάρχει και ειδοποίηση μέσω δόνησης συσκευών και μηνυμάτων στις οθόνες τους.

Όλα τα δεδομένα αυτά που συλλέχθηκαν κατά την διάρκεια της τρίωρης συζήτησης καταγράφθηκαν και ομαδοποιήθηκαν.

Τα δεδομένα που συγκεντρώθηκαν παρουσιάζονται στον ακόλουθο πίνακα:

Ήχοι	Χρώμα Led	Συχνότητα αναβόσβησης (φορές/s)	Περαιτέρω περιγραφή
Κουδούνι Εξώπορτας	Άσπρο	1	
Χτύπημα Πόρτας	Άσπρο	1	
Συναγερμός Σπιτιού	Πορτοκαλί	Πολύ γρήγορα, με δυνατό φώς	
Συναγερμός φωτιάς	Κόκκινο	Πολύ γρήγορα, με δυνατό φώς	
Κλάμα μωρού	Πράσινο	Σχετικά γρήγορα	
Χτύπημα Τηλεφώνου ή μηνύματος	Μωβ	1	
Ξυπνητήρι	Γαλάζιο	Όχι φως, αλλά με δόνηση στην συσκευή	Μιας και ο χρήστης της συσκευής θα κοιμάται και δεν θα μπορεί να προσέξει τις λάμπες, θα ήταν προτιμότερο να χρησιμοποιηθεί σκέτο δόνηση της συσκευής στην συγκεκριμένη περίπτωση.
Τρεχούμενο νερό	Μπλε	1	Κάποιες φορές οι χρήστες, μιας και δεν ακούνε το νερό όταν πλένουν τα χέρια τους, τους αποσπά την προσοχή κάτι άλλο και ξεχνάνε την βρύση ανοιχτή. Επίσης, δεν μπορούν να αντιληφθούν τυχόν διαρροές.
Κλιματιστικό	Γκρι	1	Οι χρήστες μπορεί να αφήσουν το κλιματιστικό ανοιχτό όταν φεύγουν απο το σπίτι, χωρίς να έχουν προσέξει ότι λειτουργεί ακόμα.
Ανοιχτή πόρτα ψυγείου	Κίτρινο	1	Κάποια ψυγεία έχουν προειδοποιητικό θόρυβο άμα αφήσεις την πόρτα τους ανοιχτή.
Νερό που βράζει	Κίτρινο	1	Αυτή η δραστηριότητα έχει το ίδιο χρώμα (Κίτρινο) με την προηγούμενη γιατί και οι δύο διαδραματίζονται στον χώρο της κουζίνας, οπότε ο χρήστης θα πάει αυτοπροσώπως στον χώρο αυτό να δει ποιο από τα δύο συμβαίνει.

Πίνακας 2.1: Δεδομένα Σταδίου Αναγνώρισης της εφαρμογής

2.2 Αισθητήρες

Κατά την διάρκεια της εκπόνησης της διπλωματικής, έγινε αντιληπτό πως αποτελεί μείζονος σημασίας η ενδελεχής εξέταση και αξιολόγηση όλων των ειδών αισθητήρων που υπάρχουν στην αγορά. Έτσι μόνο θα ήταν εφικτό να επιλεχθεί ο πιο ωφέλιμος και αποδοτικός αισθητήρας, όσον αφορά την ανάλυση των ηχητικών ερεθισμάτων που θα ηχογραφεί η εφαρμογή μας.

2.2.1 Ορισμός του αισθητήρα

Ο αισθητήρας (sensor) είναι μια συσκευή η οποία μπορεί να μετατρέψει σε ένα σήμα οποιαδήποτε φυσική ποσότητα που μπορεί να μετρηθεί. Το προκύπτον σήμα μπορεί να διαβαστεί, να εμφανιστεί, να αποθηκευτεί ή να χρησιμοποιηθεί ώστε να έχει υπό έλεγχο κάποια άλλη ποσότητα. Γενικότερα, οι αισθητήρες χρησιμοποιούνται για την μέτρηση ενός συγκεκριμένου χαρακτηριστικού οποιουδήποτε αντικειμένου ή συσκευής. Αξίζει να σημειωθεί πως είναι πολύ σημαντική η ρύθμιση του κάθε αισθητήρα με βάση ένα σημείο αναφοράς ή μια τυποποιημένη συσκευή, έτσι ώστε να επιτευχθεί η κατά το δυνατότερο ακριβέστερη μέτρηση [5].

2.2.2 Χαρακτηριστικά των αισθητήρων

Ένας αποδοτικός αισθητήρας θα πρέπει να απαρτίζεται από τα εξής χαρακτηριστικά [6]:

- Υψηλή ευαισθησία. Η ευαισθησία υποδηλώνει κατά πόσο η έξοδος που παράγεται από την συσκευή αλλάζει ανά μονάδα μεταβολής της εισόδου. Για παράδειγμα, η τάση ενός αισθητήρα θερμοκρασίας αλλάζει κατά 1mV για κάθε αλλαγή 1°C στο περιβάλλον. Έτσι, η ευαισθησία του αισθητήρα χαρακτηρίζεται ως 1mV / °C.
- Γραμμικότητα. Η έξοδος θα πρέπει να αλλάζει γραμμικά με την είσοδο. Ένας αισθητήρας είναι μια συσκευή που ουσιαστικά μετατρέπει κάποια ποσότητα εισόδου σε μια αναλογική έξοδο, επομένως είναι σημαντικό να αναπαράγει ο αισθητήρας την ακριβή συμπεριφορά του σήματος εξόδου βάση της παραμέτρου εισόδου.
- Υψηλή ανάλυση. Η ανάλυση είναι η μικρότερη αλλαγή στην είσοδο που μπορεί να ανιχνεύσει η συσκευή.
- Μειωμένο θόρυβο.
- Μειωμένη κατανάλωση ενέργειας.

2.2.3 Τύποι αισθητήρων

Οι αισθητήρες ταξινομούνται με βάση την φύση της ποσότητας που μετράνε. Παρακάτω παρατίθενται οι κύριες κατηγορίες των αισθητήρων που έχουν εφευρεθεί μέχρι σήμερα και χρησιμοποιούνται ευρέως στον τεχνολογικό τομέα [7]:

Αισθητήρες Ακουστικοί και Ήχου (Sound Sensors): Οι αισθητήρες ήχου ανιχνεύουν ακουστικά κύματα μέσω της χρήσης μικροφώνων ή φίλτρα άλλου τύπου. Γενικότερα, η λειτουργία τους βασίζεται στον τρόπο που διαμορφώνονται τα επιφανειακά ακουστικά κύματα και στον τρόπο επαφής των κυμάτων αυτών με την επιφάνεια του αισθητήρα.

Αυτοκινούμενοι αισθητήρες: Οι αισθητήρες τέτοιου τύπου κυρίως καταγράφουν δεδομένα για αυτοκινούμενα είδη όπως πίεση, θερμοκρασία αέρα και ψυκτικού υγρού, ταχύτητα, επιτάχυνση, γραμμική κίνηση, χρόνο και εν τέλει μετατρέπουν τα δεδομένα αυτά σε ηλεκτρικά σήματα. Στη συνέχεια τα σήματα αυτά ερμηνεύονται από τον "εγκέφαλο" του αυτοκινήτου.

Χημικοί αισθητήρες: Οι χημικοί αισθητήρες είναι συσκευές που μετατρέπουν διάφορες χημικές πληροφορίες σε αναλυτικό σήμα. Οι χημικές πληροφορίες αυτές έχουν παρθεί είτε από την συγκέντρωση ενός δείγματος που έχει συλλεχθεί από συγκεκριμένο συστατικό, είτε από την ανάλυση ολικής σύνθεσης ενός στοιχείου. Πιο συγκεκριμένα, εντοπίζουν τις φυσικές παραμέτρους του δείγματος και τις συγκρίνουν με τα σημεία αναφοράς που έχουν καθοριστεί και η μεταβολή αυτή αναφέρεται με την βοήθεια ενός ολοκληρωμένου μετατροπέα που παράγει το σήμα εξόδου.

Ηλεκτρομαγνητικοί αισθητήρες: Οι ηλεκτρομαγνητικοί αισθητήρες ανιχνεύουν την ροή του ρεύματος κατά μήκος ενός ηλεκτρικού σύρματος και μετράνε το μαγνητικό πεδίο που δημιουργείται από την παρουσία του ηλεκτρικού πεδίου.

Περιβαλλοντικοί αισθητήρες: Ένας περιβαλλοντικός αισθητήρας είναι μια συσκευή η οποία μετρά ή ανιχνεύει καταστάσεις στον πραγματικό κόσμο, όπως η υγρασία, το φυσικό αέριο ή

οτιδήποτε άλλο περιέχεται στο περιβάλλον και μετατρέπει την κατάσταση αυτή σε μια αναλογική ή ψηφιακή αναπαράσταση.

Οπτικοί αισθητήρες: Οι οπτικοί αισθητήρες είναι σχεδιασμένοι έτσι ώστε να ανταποκρίνονται στα ερεθίσματα φωτός. Συνήθως, ορίζονται λειτουργίες που πρέπει να λαμβάνουν χώρα ανάλογα με την παρουσία ή όχι του φωτός.

Μηχανικοί Αισθητήρες: Οι μηχανικοί αισθητήρες είναι γνωστοί και ως αισθητήρες δύναμης. Μετατρέπουν την είσοδό τους, η οποία είναι ουσιαστικά μηχανική δύναμη, σε ηλεκτρονικό σήμα.

Θερμικοί αισθητήρες: Είναι συσκευές οι οποίες ανιχνεύουν τις μεταβολές της θερμοκρασίας. Χρησιμοποιούνται ευρέως σε πολλούς φορητούς ή επιτραπέζιους υπολογιστές έτσι ώστε να βγάζουν προειδοποιητικό ήχο όταν η θερμοκρασία τους ξεπεράσει ένα προκαθορισμένο όριο.

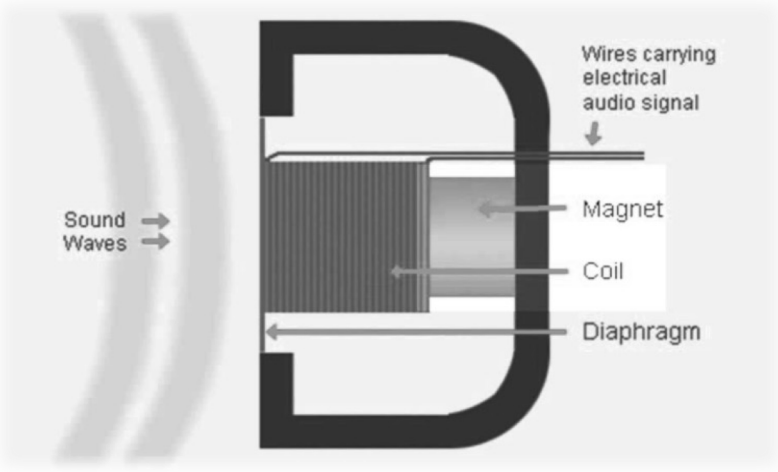
Αισθητήρες εγγύτητας και παρουσίας: Ένας αισθητήρας εγγύτητας ή παρουσίας είναι ο αισθητήρας ο οποίος είναι σε θέση να ανιχνεύσει τις παρουσίες των κοντινών αντικειμένων χωρίς καμία φυσική επαφή μαζί τους. Κατά κύριο λόγο λειτουργούν εκπέμποντας ηλεκτρομαγνητικές ακτινοβολίες και ανιχνεύουν τυχόν αλλαγές που υπάρχουν στο ανακλώμενο σήμα που επιστρέφεται στον αισθητήρα.

Από τους παραπάνω αισθητήρες επιλέχθηκαν προφανώς οι Sound Sensors για την υλοποίηση της εφαρμογής μας.

2.3 Ανάλυση Αισθητήρων Ήχου – Μικρόφωνο

Οι αισθητήρες ήχου λειτουργούν μιμούμενοι την διαδικασία μετάδοσης του ηχητικού σήματος μέσα στο ανθρώπινο σώμα, από τα αυτιά στον εγκέφαλο.

Αρχικά, υπάρχουν τα μικρόφωνα τα οποία μετατρέπουν ένα ηχητικό σήμα σε τάση ή ρεύμα τα οποία είναι μεγέθη ανάλογα με το ανιχνευόμενο σήμα εισόδου. Συνήθως έχουν ένα μικρό διάφραγμα κατασκευασμένο από μαγνήτες οι οποίοι περιβάλλονται από ένα σπειροειδές μεταλλικό σύρμα. Τα ηχητικά σήματα τα οποία έρχονται σε επαφή με το διάφραγμα, προκαλούν την δόνησή του με επακόλουθο να δονείται και ο μαγνήτης ο οποίος βρίσκεται στο εσωτερικό του διαφράγματος. Η δόνηση του μαγνήτη επάγει ρεύμα στο πηνίο το οποίο ουσιαστικά αποτελεί και το ηλεκτρικό σήμα του ήχου το οποίο μεταφέρεται μέσω καλωδίων στις όποιες συσκευές είναι συνδεδεμένες με το μικρόφωνο [8].



Σχήμα 2.1: Μικρόφωνο

Αν και τα μικρόφωνα είναι οι πιο ευρέως χρησιμοποιούμενοι αισθητήρες ήχου, υπάρχουν και οι ηλεκτροστατικοί και πιεζοηλεκτρικοί αισθητήρες οι οποίοι συμβάλλουν επίσης στην ανίχνευση του ήχου στη βιομηχανία, στην ιατρική, στην ρομποτική αλλά και για στην αναγνώριση και παρακολούθηση οντοτήτων. Αυτοί οι αισθητήρες είναι ιδανικοί για την ανίχνευση ηχητικών κυμάτων που δεν βρίσκονται εντός του ηχητικού φάσματος, κάτι που τα καθιστά κατάλληλα να αξιοποιηθούν για πολλές εφαρμογές. Για παράδειγμα, μια κατηγορία τους αποτελούν οι υπέρηχοι υψηλής συχνότητας οι οποίοι χρησιμοποιούνται στην συγκόλληση πλαστικών αντικειμένων, ενώ

υπάρχουν και οι υπέρηχοι χαμηλών συχνοτήτων οι οποίοι χρησιμοποιούνται για την επιθεώρηση λιγότερο πυκνών υλικών, όπως το ξύλο, το σκυρόδεμα και το τσιμέντο.

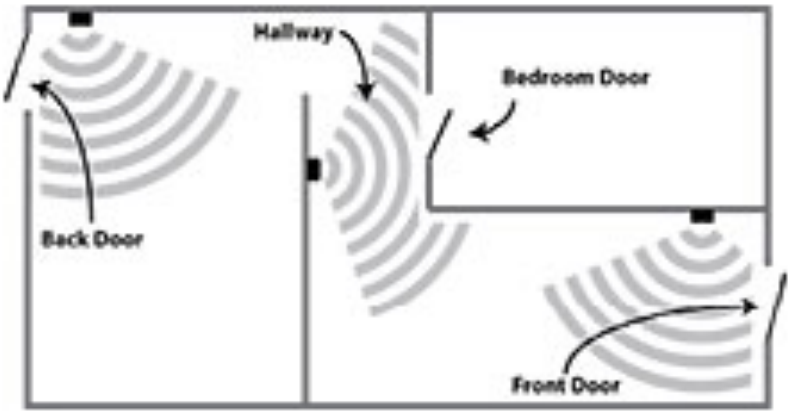
Τελικά, επιλέχθηκαν τα μικρόφωνα σαν αισθητήρες που θα επικοινωνούν με την εφαρμογή που αναπτύξαμε στα πλαίσια της διπλωματικής για τους εξής λόγους:

- Αξιοπιστία
- Χαμηλή τιμή
- Ανθεκτικότητα
- Μικρό Μέγεθος

2.4 Τοποθέτηση Αισθητήρων Ήχου

Πολύ σημαντικό ρόλο, για την ορθή λειτουργία της εφαρμογής, έχει το να τοποθετηθούν στρατηγικά οι αισθητήρες ήχου σε κάθε δωμάτιο. Πιο συγκεκριμένα πρέπει να ληφθεί υπόψη το μέγεθος του δωματίου, η αρχιτεκτονική του καθώς και τα αντικείμενα που βρίσκονται μέσα σε αυτό, έτσι ώστε να επιτευχθεί η καλύτερη το δυνατόν ηχογράφηση όλων των ήχων που μπορεί να αναπαραχθούν μέσα στο περιβάλλον του σπιτιού. Παρακάτω παρατίθενται μερικά σημεία-κλειδιά που αφορούν την σωστή τοποθέτηση αισθητήρων ήχου μέσα στο σπίτι [9].

- Οι αισθητήρες πρέπει να τοποθετηθούν περίπου 10 με 15 μέτρα μακριά από θερμαντικά σώματα, από σημεία στα οποία πέφτουν οι ακτίνες του ήλιου αλλά και αέρας από κλιματιστικά. Σε αντίθετη περίπτωση μπορεί να υπάρξει φθορά του μικροφώνου αλλά και μπορεί να φύγει από την βάση τοποθέτησής του λόγω ρευμάτων θερμότητας ή αέρα.
- Τα μικρόφωνα είναι προτιμότερο να τοποθετηθούν σε σημεία των δωματίων όπου λαμβάνουν χώρα οι περισσότερες δραστηριότητες πχ. αν υπάρχει συναγερμός στο σπίτι, το μικρόφωνο να είναι κοντά στον συναγερμό ώστε να εντοπίσει αμέσως την τυχόν ενεργοποίησή του.
- Το κάθε μικρόφωνο έχει μια συγκεκριμένη εμβέλεια ηχογράφησης η οποία πρέπει να ληφθεί υπόψη κατά την διάρκεια τοποθέτησής του. Είναι πολύ σημαντικό να μην υπάρχουν σημεία μέσα στο σπίτι τα οποία είναι εκτός εμβέλειας ηχογράφησης γιατί σε τέτοια περίπτωση είναι πολύ πιθανό να μην υπάρξει αναγνώριση των ήχων που παράγονται σε αυτά τα σημεία. Στο παρακάτω παράδειγμα φαίνεται πως άμα αναπαραχθεί ήχος στο μπάνιο, δεν θα εντοπιστεί από κανέναν αισθητήρα μιας και είναι εκτός εμβέλειας.



Σχήμα 2.2: Εμβέλεια Ηχογράφησης

Οι αισθητήρες λειτουργούν καλύτερα όταν βρίσκονται σε ανοιχτούς χώρους και δεν υπάρχουν αντικείμενα στο κοντινό περίγυρο ειδάλλως τα ηχητικά σήματα ανακλώνται σε διάφορες επιφάνειες πχ. επίπλων και μειώνεται ταυτοχρόνως η ποιότητα του ηχητικού σήματος.

Παρατίθεται ένα παράδειγμα κάτοψης σπιτιού με τις ιδανικές ενδεικτικές θέσεις των αισθητήρων ήχου.

Σχήμα 2.3: Κάτοψη σπιτιού με αισθητήρες

2.5 Ενεργοποιητές – Μέθοδοι Οπτικοποίησης

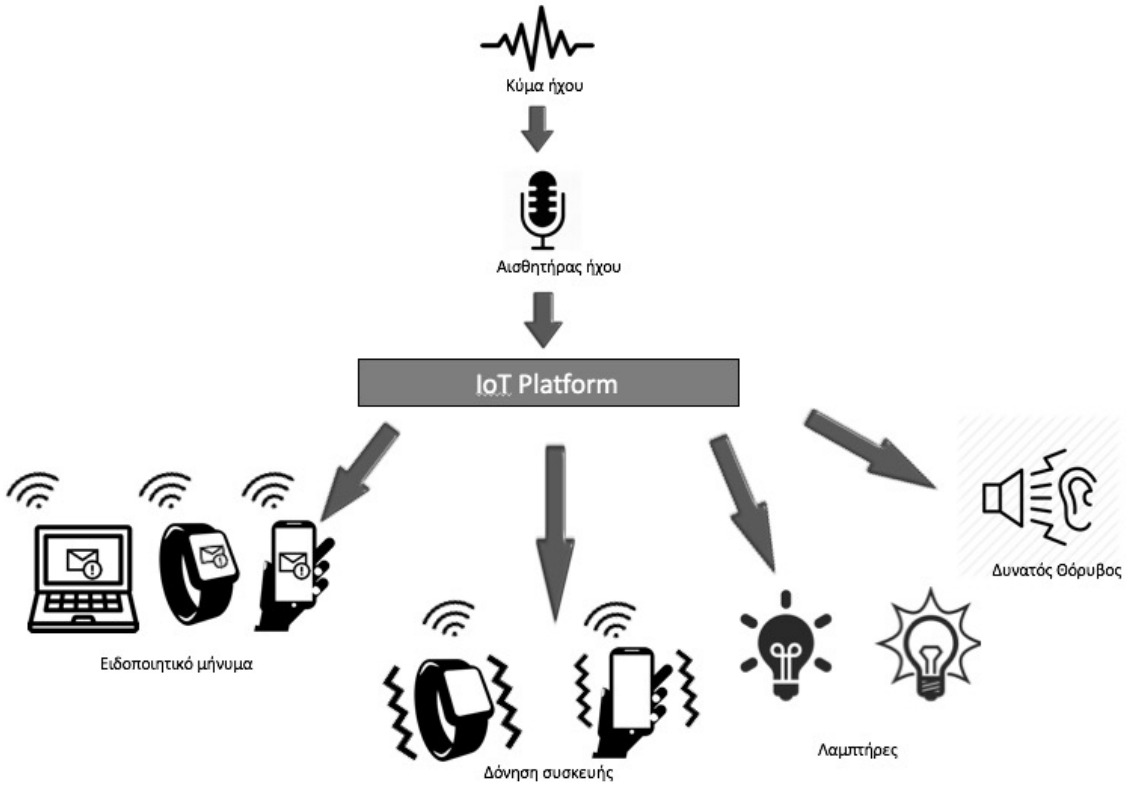
Οι κωφοί και οι βαρήκοοι αντιλαμβάνονται τον γύρω κόσμο κυρίως με την όραση και την αφή τους. Επομένως, ο ήχος θα πρέπει ουσιαστικά να απεικονιστεί με τέτοιον τρόπο έτσι ώστε να γίνεται αντιληπτός είτε μέσω κάποιου μηνύματος είτε με δόνηση. Υπάρχουν διάφορα σενάρια απεικόνισης που θα μπορούσαμε να ακολουθήσουμε στην ανάπτυξη της εφαρμογής στα πλαίσια της διπλωματικής.

Αρχικά, το ηχητικό κύμα που θα αναπαραχθεί μέσα στον χώρο του σπιτιού θα φθάσει στον πλησιέστερο αισθητήρα ο οποίος με την σειρά του θα το μετατρέψει σε ηλεκτρικό σήμα. Στη συνέχεια, το ηλεκτρικό σήμα αυτό θα επεξεργασθεί από την πλατφόρμα IoT και το αποτέλεσμα δύναται να γίνει γνωστό μέσω τεσσάρων διαφορετικών σεναρίων.

Το πρώτο σενάριο απεικόνισης ήχου (Sound Visualisation Method) αποτελεί η εμφάνιση κάποιου γραπτού μηνύματος στην οθόνη μιας συσκευής. Η συσκευή αυτή μπορεί να είναι είτε κινητό, είτε υπολογιστής, είτε Έξυπνο Ρολόι. Το δεύτερο σενάριο απεικόνισης ήχου είναι η δόνηση των συσκευών αυτών όταν λαμβάνεται το ηχητικό σήμα. Το τρίτο σενάριο είναι το έντονο φως από τις λάμπες σε διάφορους χρωματισμούς και το αναβόσβημά τους σε συχνότητες ανάλογα την κατηγορία του ήχου. Το τέταρτο και τελευταίο σενάριο απεικόνισης ήχου είναι ο δυνατός ήχος για τα βαρήκοα άτομα.

Η ιδανική συνθήκη θα ήταν κάθε φορά που εντοπίζεται και αναλύεται ο ήχος να διαδραματίζονται ταυτοχρόνως και τα τέσσερα σενάρια. Στην συγκεκριμένη διπλωματική θα αξιοποιήσουμε μόνο το σενάριο της εμφάνισης μηνύματος σε κινητό όπως και τη δόνησή του, καθώς είναι παρασάγγας δυσκολίας το εγχείρημα να υλοποιηθούν και τα τέσσερα σενάρια στα πλαίσια της παρούσας διπλωματικής.

Στο ακόλουθο σχήμα παρουσιάζονται συνοπτικά τα προαναφερθείσα σενάρια:



Σχήμα 2.4: Σενάρια απεικόνισης ήχων

3 Internet of Things – Smart Homes

3.1 Ορισμός του Internet of Things

Το Διαδίκτυο των Πραγμάτων (Internet of Things - IoT) είναι το δίκτυο των φυσικών αντικειμένων στα οποία υπάρχει πρόσβαση μέσω του Διαδικτύου, όπως ορίζεται από τους αναλυτές και τους οραματιστές της τεχνολογίας. Τα αντικείμενα αυτά περιέχουν ενσωματωμένη τεχνολογία έτσι ώστε να μπορούν να αλληλεπιδρούν με εσωτερικές καταστάσεις ή το εξωτερικό περιβάλλον [10].

Στο μέλλον, η ψηφιακή ανίχνευση, η επικοινωνία και οι δυνατότητες ψηφιακής επεξεργασίας θα έχουν ενσωματωθεί πανταχού σε αντικείμενα καθημερινής χρήσης, μετατρέποντάς τες σε συσκευές του IoT. Οι έξυπνες συσκευές αυτές θα συλλέγουν δεδομένα, θα μεταδίδουν πληροφορίες μεταξύ τους καθώς και θα επεξεργάζονται τις πληροφορίες αυτές συνεργατικά.

3.2 Έξυπνο Σπίτι

Μια πλατφόρμα IoT είναι ουσιαστικά ένα μεγάλο δίκτυο επιμέρους δικτύων στο οποίο, συνήθως, ένας τεράστιος αριθμός αντικειμένων / πραγμάτων / αισθητήρων / συσκευών συνδέονται όλα μεταξύ τους μέσω ανταλλαγής πακέτων και πληροφορίας σχετικά με την παροχή υπηρεσιών. Γίνεται ευφυής επεξεργασία των δεδομένων (intelligent data processing) καθώς και διαχείρισής τους για τις διάφορες υπηρεσίες που τρέχουν στην προκείμενη πλατφόρμα IoT.

Σε ένα Έξυπνο Σπίτι ουσιαστικά χρησιμοποιούμε μια πλατφόρμα IoT η οποία συμπεριλαμβάνει την χρήση υπολογιστών, δικτύων καθώς και ολοκληρωμένη τεχνολογία ενσύρματης ή ασύρματης δικτύωσης έτσι ώστε να επιτευχθεί η ενσωμάτωση των διαφόρων υποσυστημάτων (sub - systems), που σχηματίζονται από τις συσκευές του σπιτιού, σε ένα ενιαίο σύστημα το οποίο απαρτίζει ουσιαστικά την πλατφόρμα IoT του σπιτιού.

Πιο συγκεκριμένα, ένα Smart Home είναι εξοπλισμένο με ειδικά σχεδιασμένη και δομημένη δικτύωση που ουσιαστικά βοηθάει τους χρήστες να ελέγχουν ή να προγραμματίζουν μια σειρά από αυτοματοποιημένες λειτουργίες που αφορούν τις ηλεκτρονικές συσκευές που βρίσκονται στο σπίτι. Όλα τα παραπάνω προκύπτουν με μία μόνο εντολή, όπως για παράδειγμα με ένα κουμπί μπορεί να σβήσουν ή να ανάψουν ανάψουν όλα τα φώτα της οικίας. Αξίζει να προστεθεί πως η αυτοματοποίηση λειτουργιών στον χώρο του σπιτιού προσφέρει μεγάλο όφελος όσον αφορά την ευκολία, την ασφάλεια (security) καθώς και την ενεργειακή αποδοτικότητα (energy efficiency).

Οι παραδοσιακές αρχιτεκτονικές των Έξυπνων Σπιτιών έχουν σαν κοινό γνώρισμα πως όλες οι συσκευές που είναι συνδεδεμένες με το τοπικό δίκτυο ελέγχονται από την ονομαζόμενη "πύλη οικιακού τύπου" (home gateway) η οποία λειτουργεί σαν πάροχος υπηρεσιών για τους χρήστες. Επομένως, όλες οι συσκευές του σπιτιού ελέγχονται από τους χρήστες ενώ ταυτοχρόνως τα πρωτόκολλα του ελέγχου αυτού καθορίζονται μέσα στην Home Gateway.

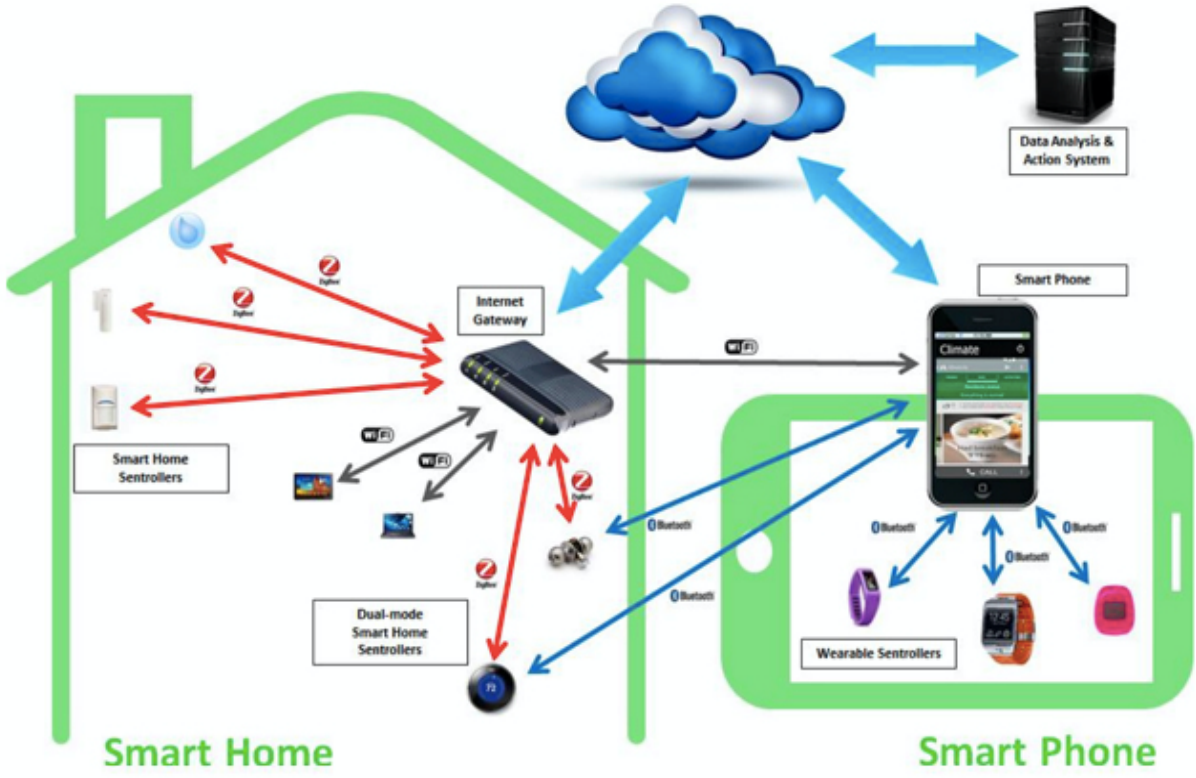
3.2.1 Λόγος ύπαρξης Home Gateway στα Smart Homes

Υπάρχουν ορισμένες συσκευές οι οποίες μπορούν να λειτουργήσουν με τον ισοδύναμο διαμοιρασμό και ανταλλαγή των πόρων τους, μεταξύ τους, μέσω ειδικά εγκαταστημένου τοπικού δικτύου (Peer-to-peer Network), χωρίς να χρησιμοποιείται ένα συγκεντρωτικό διοικητικό σύστημα (Centralized Administrative System).

Το παραπάνω, όμως, μπορεί να αποδειχτεί χρήσιμο μόνο στην περίπτωση που οι όλες οι συσκευές της οικίας χρησιμοποιούν το ίδιο πρωτόκολλο. Έτσι, καθίσταται απαραίτητη η παρουσία της προαναφερόμενης Home Gateway η οποία θα είναι ενεργοποιημένη ως πύλη υπηρεσιών (Service Gateway) και ουσιαστικά θα μεταφράζει τα πρωτόκολλα των συσκευών επιτρέποντας και διευκολύνοντας με αυτόν τον τρόπο την επικοινωνία μεταξύ τους.

3.3 *Παραδοσιακή Αρχιτεκτονική ενός Smart Home*

Στο παρακάτω σχήμα παρουσιάζεται η παραδοσιακή αρχιτεκτονική ενός Έξυπνου Σπιτιού:



Σχήμα 3.1: Αρχιτεκτονική Έξυπνου Σπιτιού

Οι βασικές συσκευές στην παραπάνω αρχιτεκτονική είναι [11]:

- η **Πύλη του Διαδικτύου** (Internet Gateway), η οποία συνδέει το Έξυπνο σπίτι με το "Νέφος" (Cloud), στο οποίο λαμβάνει χώρα η Ανάλυση των Δεδομένων (Data Analysis) και καθορίζεται ο τρόπος που θα "τρέχουν" οι αυτοματοποιημένες λειτουργίες (Action Systems) και
- το **Έξυπνο Τηλέφωνο** (Smart Phone) το οποίο ουσιαστικά λειτουργεί σαν πίνακας ελέγχου και ελέγχει τις όποιες αλληλεπιδράσεις εμφανίζονται, μέσω πχ λήψης ειδοποιήσεων. Η Internet Gateway όπως και το Smart Phone είναι και τα δύο συνδεδεμένα με το Cloud είτε μέσω καλωδίωσης (DSL) είτε ασύρματα: δεδομένα τηλεφωνίας (2G, 3G, LTE κτλ), WiFi, Bluetooth.

Εντός του Δικτύου του Smart Home, η Internet Gateway συνδέει στο Cloud όλους τους αισθητήρες / χειριστές (Smart Home ή Wearable Sentrollers) μέσω ZigBee (η οποία είναι μια

εξειδικευμένη κατηγορία ασύρματης τεχνολογίας και θα αναλυθεί παρακάτω). Έτσι, όταν οι sentrollers εντοπίσουν μια δραστηριότητα, εισέρχονται τα καταγραφέντα δεδομένα αυτά στο Cloud και αναλύονται και τέλος, καθορίζεται η αντίστοιχη ενέργεια που πρέπει να λάβει χώρα μετέπειτα.

Το Smart Phone δύναται να συνδεθεί μέσω Bluetooth με sentrollers οι οποίοι φοριούνται από τον χρήστη, είτε ο χρήστης αυτός βρίσκεται στο δρόμο είτε στο σπίτι.

3.4 Είδη Δικτυώσεων στα Έξυπνα Σπίτια

3.4.1 Ενσύρματη Λειτουργία

Σε αυτό το είδος δικτύωσης ουσιαστικά όλες οι συσκευές συνδέονται μεταξύ τους μέσω καλωδίωσης. Έτσι, το Κέντρο Ελέγχου λαμβάνει ταυτοχρόνως πάρα πολλα σήματα από τα διάφορα καλώδια που έχουν εγκατασταθεί στο Έξυπνο Σπίτι με αποτέλεσμα η λαμβανόμενη πληροφορία να είναι τεράστια. Επακόλουθο του παραπάνω είναι πως όταν προκύπτει ένα πρόβλημα στο σύστημα, καθίσταται αδύνατος ο εντοπισμός του σήματος που θα είναι υπεύθυνο, λόγω του μεγάλου όγκου δεδομένων που είναι προς επεξεργασία.

Επομένως, είναι προφανές πως αυτή η κατηγορία καλωδίωσης έχει πολλά εξέχοντα προβλήματα όπως περίπλοκη καλωδίωση, μεγάλο φόρτο εργασίας, υψηλό κόστος, δυσκολία στη συντήρηση καθώς και αυξημένη δυσκολία στην κατασκευή του διαδικτύου.

3.4.2 Ασύρματη λειτουργία

Έχοντας κατά νου τα μειονεκτήματα που προκύπτουν από την Καλωδίωση, θέλουμε το ασύρματο σύστημα που εφαρμόζεται στο Έξυπνο Σπίτι να απαρτίζεται από τα ακόλουθα χαρακτηριστικά:

- Χαμηλή κατανάλωση ενέργειας
- Σταθεροποίηση
- Ευκολία επέκτασης δικτύου
- Υψηλή ταχύτητα μετάδοσης

Σε ένα Έξυπνο Σπίτι μπορούμε να βρούμε τρία είδη ασύρματης καλωδίωσης. Το Bluetooth, το WiFi και το ZigBee. Το κάθε είδος έχει τα δικά του χαρακτηριστικά καθώς και μειονεκτήματα - πλενεκτήματα.

Bluetooth

Το Bluetooth είναι ασύρματη τεχνολογία η οποία ευνοεί την ανταλλαγή δεδομένων σε μικρές αποστάσεις. Χρησιμοποιώντας ειδικές ραδιοσυχνότητες (της τάξης από 2.4 σε 2.485 GHz) για την μετάδοση των δεδομένων, ουσιαστικά δημιουργεί ένα δίκτυο μικρής εμβέλειας. Το δίκτυο αυτό είναι πολύ ασφαλές και μπορεί να συνδεθεί ταυτοχρόνως μέχρι και με οχτώ συσκευές. Παρόλα αυτά, η εμβέλεια του δικτύου είναι περίπου μόνο 10 μέτρα με αποτέλεσμα να είναι απαραίτητη η εγγύτητα των συσκευών μεταξύ τους. Προφανώς, κάτι τέτοιο είναι αδύνατο σε ένα μεσαίου μεγέθους σπίτι. Έτσι, το Bluetooth δεν μπορεί να ανταποκριθεί στις απαιτήσεις ενός Έξυπνου Σπιτιού. Μπορεί να χρησιμοποιηθεί, όμως, πολύ αποτελεσματικώς για την ασφαλή και επιτυχή αλληλεπίδραση μεταξύ των Smart Phones και των Smart Watches του χρήστη.

WiFi

Το WiFi είναι γνωστό και σαν την οικογένεια των ασύρματων προτύπων IEEE 802.11. Έχει γνωρίσει ευρέα αποδοχή και ανάπτυξη τα τελευταία χρόνια καθώς έχει πολλά πλεονεκτήματα. Το βασικότερο πλεονέκτημα που προσφέρει το WiFi είναι η ικανοποίηση της ανάγκης που υπήρχε και υπάρχει ακόμα για την αντικατάσταση της παραδοσιακής καλωδιακής δικτύωσης των σπιτιών καθώς και άλλων χώρων. Ένα άλλο πλεονέκτημα αποτελεί το χαμηλό κόστος των προϊόντων που είναι συμβαντά με τα πρότυπα 802.11, όπως τα σημεία πρόσβασης (Access Points) και οι επαναλήπτες σήματος. Επίσης, τα πιο πρόσφατα πρότυπα της οικογένειας των IEEE 802.11 παρέχουν αρκετά μεγάλες ταχύτητες μετάδοσης δεδομένων. Για παράδειγμα, το πρότυπο 802.11n μπορεί να επιτύχει ρυθμό μετάδοσης δεδομένων πάνω από 500 Mbps.

Το τελευταίο χαρακτηριστκό των WiFi, το οποίο και αποτελεί και βασικό μειονέκτημα της συγκεκριμένης τεχνολογίας όσον αφορά την ευρεία χρησιμοποίησή της στα Έξυπνα σπίτια, είναι η μεγάλη κατανάλωση ενέργειας. Οι αισθητήρες που αποτελούν βασικό συστατικό ενός Smart Home, δεν πρέπει να καταναλώνουν μεγάλα ενεργειακά ποσά. Την λύση σε αυτό το πρόβλημα την δίνει η επόμενη τεχνολογία δικτύωσης Έξυπνων Σπιτιών, η τεχνολογία ZigBee.

ZigBee

Η τεχνολογία ZigBee είναι βασισμένη στο πρότυπο IEEE 802.15.2 και όπως και οι άλλες προαναφερθείσες ασύρματες τεχνολογίες λειτουργεί στο φάσμα των 2.4GHz και έχει εμβέλεια μετάδοσης μέχρι 100 μέτρα και με μέγιστη ταχύτητα τα 250 Kbps.

Ο βασικός στόχος του ZigBee είναι να παρέχει επικοινωνιακές δυνατότητες σε συσκευές ελέγχου και αισθήτηρες οι οποίες δεν χρειάζονται μεγάλο εύρος ζώνης, αλλά απαιτούν παρατεταμένους χρόνους αυτόνομης λειτουργίας (με χρήση μπαταρίων κατά κύριο λόγο), καθώς και ευέλικτες τοπολογίες δικτύου. Για να καταστεί δυνατή η κατασκευή των συσκευών που έχουν χαμηλότερες απαιτήσεις σε ενέργεια, οι συσκευές ZigBee βγαίνουν σε δύο ξεχωριστές εκδόσεις. Η πρώτη έκδοση είναι οι συσκευές πλήρους λειτουργικότητας (FFD - Full Function Devices) και η δεύτερη έκδοση ειναι οι συσκευές μειωμένης λειτουργικότητας (RFD - Reduced Function Devices). Οι FFD είναι πάντα ενεργοποιημένες με αποτέλεσμα να καταναλώνουν πολλή περισσότερη ενέργεια από τις δεύτερες που συνήθως τίθονται αυτόματα σε αναμονή (sleep mode) και μεταδίδουν δεδομένα μόνο στην περίπτωση που υπάρξει κάποιο συμβάν.

Η διαφορά της τεχνολογίας ZigBee με την οικογένεια WiFi είναι πως οι συσκευές RFD μπορούν να λειτουργήσουν μόνο ως τερματικά σημεία (end points) ενός δικτύου και χρειάζονται τουλάχιστον μια συσκευή FFD για να επικοινωνήσουν. Αυτό σημαίνει ότι ένα δίκτυο το οποίο έχει μία συσκευή FFD (Internal Gateway στην περίπτωσή μας) και πολλαπλές RFD, μπορεί να δημιουργηθεί μόνο σε τοπολογία αστέρα όπου όλες οι RFD συνδέονται με την κεντρική συσκευή, κάτι το οποίο και εφαρμόζεται στην περίπτωση ενός Έξυπνου Σπιτιού.

Η τεχνολογία ZigBee, λοιπόν, έρχεται να λύσει το πρόβλημα της κατανάλωσης ενέργειας και χρησιμοποιείται ευρέως στην ασύρματη δικτύωση των Smart Homes σε συνδυασμό και με τις υπόλοιπες τεχνολογίες.

IPv4 και IPv6

Το Πρωτόκολλο Διαδικτύου (IP) είναι υπεύθυνο για την δρομολόγηση των πακέτων δεδομένων ανάμεσα σε διάφορα δίκτυα, ανεξάρτητα από την υποδομή τους και αποτελεί το κύριο πρωτόκολλο πάνω στο οποίο είναι βασισμένο το Διαδίκτυο [12].

Το IPv4 είναι η τέταρτη έκδοση του πρωτοκόλλου Ίντερνετ, αλλά είναι το πρώτο που χρησιμοποιείται ευρέως. [13] Χρησιμοποιεί ένα 32 bit σύστημα που επιτρέπει να δώσει 4.294.967.296 μοναδικές διευθύνσεις IP. Επίσης, χρησιμοποιεί μια μάσκα υποδικτύου, λόγω του μεγάλου αριθμού των υπολογιστών που χρησιμοποιούνται σήμερα και βοηθά στη μείωση του αριθμού των μοναδικών IP που χορηγούνται σε επιχειρήσεις, εταιρείες, πανεπιστήμια και σε κέντρα με πολλούς υπολογιστές.

Στο IoT οι συσκευές που είναι συνδεδεμένες και επικοινωνούν μεταξύ τους, ολοένα αυξάνονται. Η ραγδιαία αύξηση αυτή δημιουργεί την ανάγκη δημιουργίας περισσότερων διευθύνσεων IP από τον αριθμό που έχουμε τώρα με το πρωτόκολλο IPv4.

Το πρόβλημα έλλειψης διαθεσιμότητας διευθύνσεων IP έρχεται να λύσει η έκτη έκδοση του πρωτοκόλλου Ίντερνετ, IPv6. Σε σύγκριση με το IPv4, το οποίο επιτρέπει μόνο 4.294.967.296 μοναδικές διευθύνσεις, το IPv6 χρησιμοποιεί ένα σύστημα 128-bit που θα μπορεί να δώσει μέχρι και 340 - ενδεκάκις εκατομμύρια διευθύνσεις.

Λόγω αυτού του βασικού πλεονεκτήματος προτιμάται η χρήση IPv6 στα Smart Homes για την διευκόλυνση δικτύωσης των διαφόρων συσκευών μέσα στην κατοικία, καθώς και για τη διεκόλυνση ανάπτυξης της εφαρμογής μας σε πραγματικές συνθήκες.

3.5 Προστασία Προσωπικών Δεδομένων και Απόρρητο για IoT [14]

Η διαφύλαξη των προσωπικών δεδομένων είναι άρρηκτα συνδεδεμένη με το Διαδίκτυο των Πραγμάτων μιας και πλέον λόγω της αλματώδους ανάπτυξης της τεχνολογίας κάθε τύπου συσκευή και τα δεδομένα της, μπορούν να ενσωματωθούν στο IoT.

Το κύριο πρόβλημα είναι ότι η ιδέα των συσκευών δικτύωσης είναι σχετικά νέα, με αποτέλεσμα το απόρρητο των προσωπικών δεδομένων να μην λαμβάνεται πάντα υπόψη στο σχεδιασμό του προϊόντος. Τα προϊόντα IoT συχνά πωλούνται με παλιά και μη δοκιμασμένα εκτενώς ενσωματωμένα λειτουργικά συστήματα και λογισμικό. Επιπλέον, οι αγοραστές συχνά αποτυγχάνουν να αλλάξουν τους προεπιλεγμένους κωδικούς πρόσβασης σε έξυπνες συσκευές - ή αν το κάνουν, αδυνατούν να επιλέξουν επαρκώς ισχυρούς κωδικούς πρόσβασης. Έτσι, οι

συσκευές που βρίσκονται μέσα στο Έξυπνο Σπίτι είναι ευάλωτες σε διαδικτυακές επιθέσεις οι οποίες απειλούν την ιδιωτικότητα και την προστασία προσωπικών δεδομένων των κατοίκων.

Κάποιοι από τους βασικότερους κινδύνους είναι:

- Χρήση προσωπικών δεδομένων για δευτερεύοντες σκοπούς
- Απουσία δυνατότητας συγκατάθεσης από το κυρίως πρόσωπο/χρήστη
- Δημιουργία/ανάλυση ατομικού προφίλ
- Λεπτομερής παρακολούθηση των χρηστών των συσκευών
- Απουσία δυνατότητας να παραμένει κανείς ανώνυμος

Σύμφωνα με έρευνα που πραγματικοποιήθηκε το 2011 από το Ευρωβαρόμετρο [103], το οποίο είναι μια σειρά ερευνών κοινής γνώμης που διεξάγονται για λογαριασμό της Ευρωπαϊκής Επιτροπής, το 70% των Ευρωπαίων πολιτών είναι ανήσυχοι για το πώς οι εταιρείες χειρίζονται τα δεδομένα τους και θεωρούν ότι έχουν μερικό ή και καθόλου έλεγχο των δεδομένων τους. Ακόμη, το 74% θέλει να δίνει ειδική συγκατάθεση πριν τα προσωπικά τους δεδομένα συλλεχθούν και υποστούν επεξεργασία μέσω του Διαδικτύου.

Βάσει των παραπάνω, καθίσταται σαφές πως είναι απαραίτητη η λήψη δραστικών μέτρων για την βελτίωση της ασφάλειας μιας συσκευής IoT που θα είναι άμεσα προσβάσιμη μέσω του Διαδικτύου.

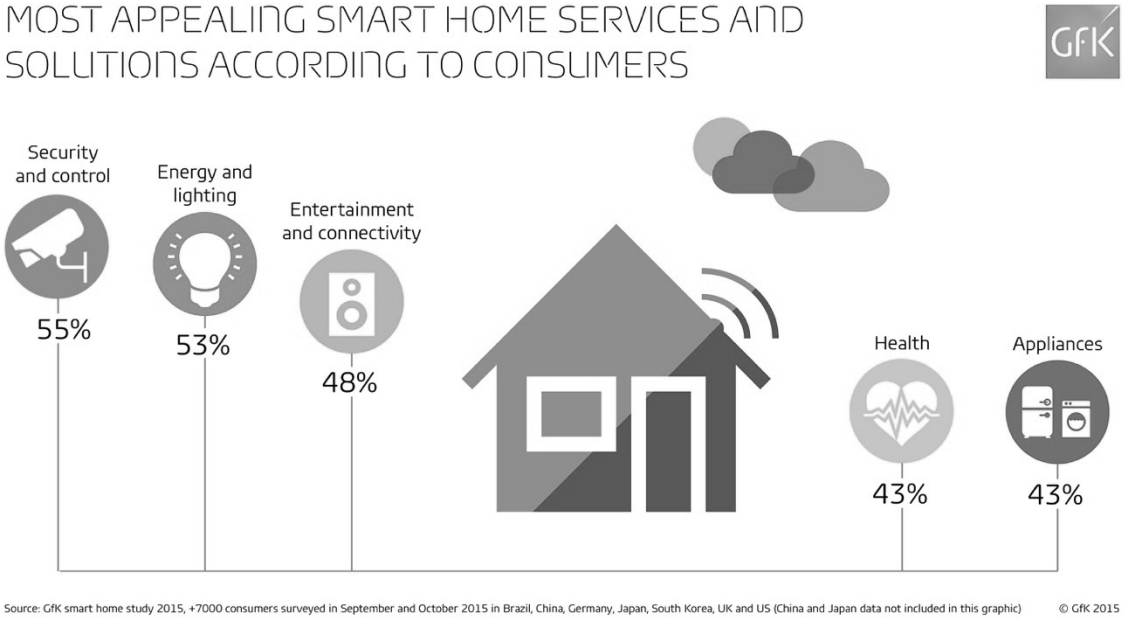
Αρχικά, η κάθε συσκευή θα πρέπει να υποδιαιρεθεί σε δικό της δίκτυο και να έχει περιορισμένη πρόσβαση στο δίκτυο. Αυτό το τμήμα του δικτύου θα πρέπει στη συνέχεια να παρακολουθείται για τον εντοπισμό ενδείξεων ύποπτης διαδικτυακής κυκλοφορίας και οι ύποπτες ενδείξεις θα πρέπει να εξαλειφθούν αν υπάρξει κάποιο πρόβλημα. Ακόμη, θα πρέπει να εφαρμοστεί η κρυπτογράφηση των δεδομένων που ανταλλάζονται μεταξύ των συσκευών για ακόμη μεγαλύτερη ενίσχυση της ασφάλειάς τους.

Λαμβάνοντας υπόψη το γεγονός πως μέχρι το 2020 θα υπάρχουν 24 δισεκατομμύρια συσκευές συνδεδεμένες με το διαδίκτυο, γίνεται εμφανές πως η ασφάλεια πλέον πρέπει να θεωρηθεί προτεραιότητα από όλους τους κατασκευαστές λογισμικού / λειτουργικών συστημάτων καθώς και από τους προγραμματιστές που ασχολούνται με την ανάπτυξη του Διαδικτύου των Πραγμάτων.

3.6 Ενδεικτικές υπηρεσίες που προσφέρει το Smart Home

Ένα Έξυπνο Σπιτι μπορεί να προσφέρει αυτοματοποιημένες λύσεις σε πολλούς τομείς, χρησιμοποιώντας αισθητήρες και ενεργοποιώντας συγκεκριμένες ενέργειες ανάλογα με τις ρυθμίσεις / προτιμήσεις του ιδιοκτήτη.

Μια εταιρεία ονόματι GFK (Εταιρεία Έρευνας Καταναλωτών) πραγματοποίησε μια έρευνα τον Σεπτέμβριο του 2015 [16] η οποία συμπεριλάμβανε 7.000 ενήλικες ηλικίας 16 ετών και άνω στη Γερμανία, Ηνωμένο Βασίλειο, ΗΠΑ, Βραζιλία, Νότια Κορέα, την Κίνα και την Ιαπωνία, με συνεντεύξεις σε απευθείας διαδικτυακή σύνδεση. Το αποτέλεσμα της έρευνας ήταν να διευκρινιστούν ποιες υπηρεσίες θα προτιμούσαν οι ίδιοι να είναι διαθέσιμες από ένα Smart Home.



Σχήμα 3.2: Υπηρεσίες στο Smart Home

Όπως φαίνεται από το παραπάνω σχήμα, οι υπηρεσίες αυτές χωρίζονται σε πέντε κατηγορίες [17].

- Ασφάλεια και παρακολούθηση
- Εξοικονόμηση ενέργειας
- Διασκέδαση και ψυχαγωγία
- Υποστήριξη στον τομέα της υγείας
- Συσκευές

3.6.1 Ασφάλεια και παρακολούθηση

Τα διάφορα προϊόντα ασφαλείας είναι σήμερα μέρος του Διαδικτύου των Πραγμάτων. Τα συστήματα βιντεοεπιτήρησης, οι φορητές συσκευές, ο έλεγχος πρόσβασης και οι συναγερμοί συνδέονται με το IoT ως «αισθητήρες» με τον ίδιο τρόπο που συνδέονται άλλες συσκευές – κινητά τηλέφωνα, αυτοκίνητα, ψυγεία – αναδύοντας στοιχεία που διευκολύνουν τη ζωή των ανθρώπων. Το σημείο επαφής όλων αυτών των τεχνολογιών και των συνδεδεμένων στο διαδίκτυο συσκευών, είναι η κοινή τους λειτουργία που τις καθιστά ένα αποτελεσματικό και ολοκληρωμένο σύνολο με κοινό στόχο.

Κάποιες πρακτικές εφαρμογές των συσκευών που αφορούν την ασφάλεια είναι η εγκατάσταση καμερών στους εσωτερικούς και εξωτερικούς χώρους του σπιτιού που θα επιτρέπουν στον ιδιοκτήτη να παρακολουθεί ζωντανά οτιδήποτε διαδραματίζεται μέσα σε αυτό. Ακόμη, μπορούν να εγκατασταθούν αισθητήρες κίνησης σε ολόκληρο το σπίτι εξασφαλίζοντας με αυτόν τον τρόπο προστασία σε επιλεγμένα δωμάτια απέναντι σε εισβολείς. Παράλληλα, η εγκατάσταση ηλεκτρονικών κλειδαριών σε όλο το σπίτι ή σε επιλεγμένα δωμάτια είναι ένα επιπλέον θετικό στοιχείο, καθώς πολλές κλειδαριές αυτού του τύπου απαιτούν κωδικό για να ξεκλειδώσουν και να προσφέρουν πρόσβαση σε ένα χώρο, διαφυλάσσοντας με αυτόν τον τρόπο πολύτιμα αντικείμενα.

3.6.2 Εξοικονόμηση ενέργειας

Το Έξυπνο Σπίτι μπορεί να ταυτιστεί και με το Πράσινο Σπίτι. Λόγω της υψηλής ενεργειακής επάρκειας και αυτονομίας ενός Έξυπνου Σπιτιού, η κατανάλωση ηλεκτρισμού και ορυκτών καυσίμων είναι μειωμένη, γεγονός που συντελεί στην προστασία του περιβάλλοντος και τη διαφύλαξη των μη ανανεώσιμων πηγών ενέργειας. Συν τοις άλλοις, τα έξυπνα σπίτια ενσωματώνουν καινοτόμες τεχνολογίες, όπως τα ηλιακά πάνελ, που αποσκοπούν στην περαιτέρω ελάττωση των αναγκών για συμβατικές πηγές ενέργειες, όπως τα ορυκτά καύσιμα.

Κάποιες πρακτικές εφαρμογές αποτελούν η διαχείριση της θερμοκρασίας του Smart Home έχοντας παράλληλα την δυνατότητα να ρυθμίζεται το αυτόματο πότισμα του κήπου βασιζόμενο στις καιρικές συνθήκες. Επίσης, είναι δυνατή η παρακολούθηση της κατανάλωσης του σπιτιού σε ηλεκτρικό ρεύμα, έτσι ώστε να είναι δυνατό να εκπονηθεί κάποιο πλάνο για τη μείωση των

ενεργειακών λογαριασμών του ιδιοκτήτη και να περιοριστεί η κατανάλωση του ηλεκτρικού ρεύματος για συγκεκριμένες χρήσεις. Παράλληλα, υπάρχει η δυνατότητα να αλλαχθεί ο φωτισμός στο σπίτι και να αντικατασταθεί από ηλιακά προϊόντα που θα συντελέσουν στην εξοικονόμηση ενέργειας, την προστασία του περιβάλλοντος και την ελάττωση των ενεργειακών εξόδων.

3.6.3 Διασκέδαση και Ψυχαγωγία

Τα χαρακτηριστικά ενός Έξυπνου Σπιτιού προσφέρουν στους ιδιοκτήτες τη δυνατότητα να ελέγχουν με ψηφιακό τρόπο τη λειτουργία όλων των συσκευών. Έτσι, είναι εφικτό να ανοιγοκλείνουν απομακρυσμένα τα στόρια των παραθύρων της οικίας ή να τίθεται το ηχοσύστημά της υπό λειτουργία κάποια ορισμένη ώρα. Επιπλέον, οι έξυπνες ηλεκτρικές συσκευές στην κουζίνα προσφέρουν πολλαπλά οφέλη. Τα ψυγεία μπορούν να συμβάλλουν στον προγραμματισμό των ψώνιων και οι φούρνοι μικροκυμάτων να είναι εφοδιασμένοι με προκαθορισμένους χρόνους μαγειρέματος, γλυτώνοντας τον ιδιοκτήτη από σπατάλη χρόνου. Ακόμη, μερικοί υπολογιστές επιτρέπουν στους χρήστες να συνδέουν οποιοδήποτε μέσο απευθείας με την τηλεόραση, προσφέροντας τη δυνατότητα να μοιραστούν οτιδήποτε με άλλα άτομα ή να παρακολουθήσουν κάτι σε μεγαλύτερη οθόνη.

3.6.4 Υποστήριξη στον τομέα της υγείας και συσκευές

Πολλοί άνθρωποι έχουν ήδη υιοθετήσει wearable συσκευές για να παρακολουθούν την φυσική τους άσκηση, τον ύπνο ή άλλες συνήθειες τους – και αυτά είναι το πιο απλό δείγμα του πώς το IoT συνδυάζεται με τον κλάδο της υγείας. Συσκευές παρακολούθησης ασθενών, ηλεκτρονικά αρχεία και άλλα έξυπνα αξεσουάρ μπορούν να σώσουν ζωές.

4 Τελευταία λέξη της Τεχνολογίας για υπηρεσίες προς Βαρήκοους

Στο παρόν κεφάλαιο παρουσιάζεται η τελευταία αιχμή της τεχνολογίας (State of the Art) όσον αφορά την υποστήριξη για κωφούς και βαρήκοους ανθρώπους. Καθίσταται αναμφισβήτητο πως το υψηλό κόστος φροντίδας της υγείας αποτελεί μεγάλο θέμα ανησυχίας για σχεδόν οποιαδήποτε κυβέρνηση και οικογένεια. Το γεγονός αυτό δεν έχει βγει εκτός υπόληψης των κυβερνήσεων και των μεγιστάνων της τεχνολογίας, οι οποίοι διαρκώς προσπαθούν να αναπτύξουν νέα μοντέλα και χρήσιμες πατέντες / συσκευές που αφορούν την βελτίωση της ηλεκτρονικής υγείας (e-healthcare). Υπάρχει μεγάλος αριθμός ερευνητικών έργων που διεξάγονται στην Ευρώπη και έχουν ως κύριο στόχο τους την μέγιστη βοήθεια ανθρώπων με αναπηρίες με το μικρότερο δυνατό κόστος επιβάρυνσης προς αυτούς.

Πιο συγκεκριμένα, οι άνθρωποι με θέματα ακοής έχουν μόνο τέσσερις αισθήσεις από τις πέντε βασικές γεγονός που καθιστά προφανές πως υπάρχει μεγάλο εύρος συσκευών και εφαρμογών που θα μπορέσει να τους βοηθήσει.

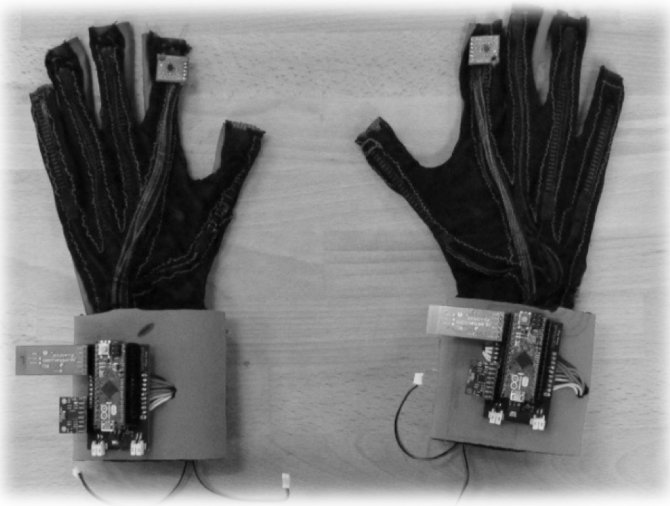
4.1 Έξυπνες Συσκευές

Την τελευταία δεκαετία έχουν κατασκευαστεί με μεγάλη επιτυχία πολλές συσκευές οι οποίες έχουν σαν σκοπό την εξυπηρέτηση και διευκόλυνση ανθρώπων που έχουν πρόβλημα ακοής. Η κάθε συσκευή έχει διαφορετικό σκοπό, τρόπο χρήσης, τρόπο ειδοποίησης αλλά ο στόχος όλων είναι ο ίδιος, να κάνουν εμφανή την παρουσία του ήχου σε άλλες μορφές, κείμενο ή δόνηση, τις οποίες ένα βαρήκοο ή κωφό άτομο είναι σε θέση να αντιληφθεί. Επίσης, υποβοηθούνε και όσον αφορά την διευκόλυνση της επικοινωνίας μέσω νοηματικής.

Παρακάτω παρουσιάζονται state-of-the-art συσκευές οι οποίες καλύπτουν ένα ευρύ φάσμα αναγκών.

Έξυπνα Γάντια (Smart Gloves)

Τα έξυπνα γάντια που παρουσιάζονται στην εικόνα δημιουργήθηκαν από δύο φοιτητές του Πανεπιστημίου της Γουάσινγκτον. Η εφεύρεσή τους, η οποία είναι και γνωστή ως "SignAloud", είναι ένα ζευγάρι γάντια το οποίο έχει την δυνατότητα να αναγνωρίσει χειρονομίες που αντιστοιχούν σε λέξεις και φράσεις της Αμερικάνικης Νοηματικής Γλώσσας [18].



Σχήμα 4.1: Έξυπνα Γάντια

Κάθε γάντι περιέχει αισθητήρες που καταγράφουν την θέση του χεριού καθώς και την κίνησή του. Στη συνέχεια, οι αισθητήρες αυτοί στέλνουν ασυρμάτως, μέσω Bluetooth, τα καταγραφέντα δεδομένα σε έναν κεντρικό υπολογιστή, ο οποίος εξετάζει τα δεδομένα αυτά μέσω διαφόρων διαδοχικών στατιστικών παλινδρομήσεων, όπως γίνεται και στα νευρωνικά διαδίκτυα. Εν τέλει, αν τα στοιχεία ταιριάζουν με μια χειρονομία, τότε η αντίστοιχη λέξη ή φράση ακούγεται μέσω των ηχείων.

Η παραπάνω εφεύρεση αποτελεί ένα τεράστιο άλμα προς την τελευταία λέξη της τεχνολογίας μιας και τα γάντια αυτά είναι ελαφρά, συμπαγή, προσαρμόζονται εύκολα στα χέρια και πάνω από όλα οικονομικά. Εύκολα μπορεί να τα φορέσει κανείς σαν καθημερινό εξάρτημα, όπως τα ακουστικά βοηθήματα ή τους φακούς επαφής και ουσιαστικά αποτελούν μια γέφυρα επικοινωνίας μεταξύ των φυσικών ομιλητών της Αμερικής Νοηματικής Γλώσσας και του υπόλοιπου κόσμου.

Vibering

Η συσκευή Vibering είναι μια καινοτόμα εφεύρεση η οποία επιτρέπει να φορεθεί ένα σύστημα ανίχνευσης και ταυτοποίησης ήχου ως ένα μοντέρνο ζευγάρι δαχτυλίδια και ρολόι χειρός. Ο κάθε δακτύλιος φοριέται στο κάθε χέρι και ουσιαστικά προσομοιώνουν τα αυτιά του χρήστη μιας και μπορούν όχι μόνο να εντοπίσουν τους ήχους που αναπαράγονται πίσω από τον χρήστη αλλά επίσης μπορούν να καθορίσουν την απόσταση του ήχου και την θέση του. Όταν εντοπίσουν τα δαχτυλίδια έναν ήχο, αρχικά δονούνται με ένταση ανάλογης της απόστασης του ήχου πχ. αν είναι δεξιά ο ήχος, το δεξί δαχτυλίδι δονείται περισσότερο. Στη συνέχεια, το ρολόι αναλύει το είδος του ήχου και παρουσιάζει τις τελικές πληροφορίες στον χρήστη. Έτσι, ο χρήστης είναι σε θέση να καταλάβει από ποια κατεύθυνση ήρθε ο ήχος αλλά και να αντιδράσει σχεδόν άμεσα στο ερέθισμά του [19].

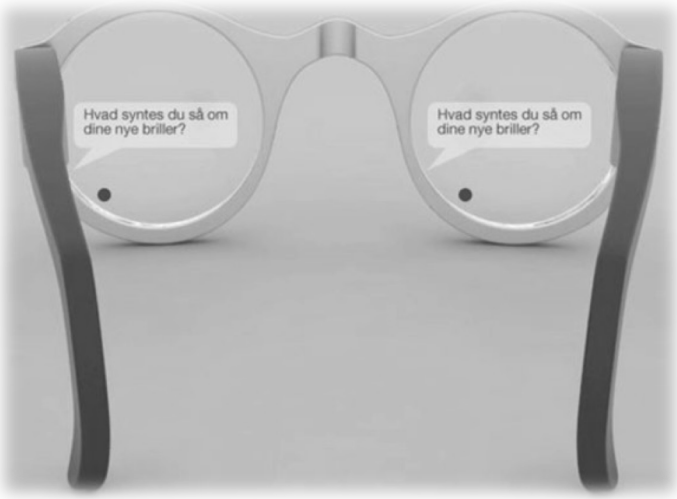
Σχήμα 4.2: Vibering

Σχήμα 4.3: Λειτουργία Vibering

Το Vibering έχει προγραμματιστεί ώστε να αναγνωρίζει ορισμένες βασικές φράσεις από ανθρώπους, όπως “Με συγχωρείτε…”, το όνομα του χρήστη και οποιοδήποτε αριθμό από θορύβους αυτοκινήτων συμπεριλαμβανομένου του πιο σημαντικού, κόρνα αυτοκινήτου. Είναι προφανές πως η συσκευή αυτή έχει ακόμα μεγάλο περιθώριο βελτίωσης μιας και η εμβέλεια του ήχου είναι περιορισμένη καθώς και το εύρος των ήχων που αναγνωρίζονται.

Γυαλιά BabelFisk

Τα επονομαζόμενα γυαλιά BabelFisk έχουν προσαρμοσμένα δύο μικρόφωνα τα οποία καταγράφουν κάθε ομιλία που διαδραματίζεται στον έξω κόσμο. Στη συνέχεια, μεταφράζουν την ομιλία και την μετατρέπουν σε κείμενο το οποίο και εμφανίζεται άμεσα στην οθόνη των γυαλιών (live speech to text). Τα μικρόφωνα, επιπροσθέτως, έχουν την ικανότητα να εντοπίσουν προς ποιά κατεύθυνση βρίσκεται η πηγή του ήχου και εκτυπώνουν τα μηνύματα αναλόγως. Είναι δυνατή η ενσωμάτωση και μίας κάρτας μνήμης στην οποία αποθηκεύονται οι συνομιλίες καθώς και τα μεταφρασμένα μηνύματα, έτσι ώστε ο χρήστης να μπορεί να τα επανεξετάσει σε μεταγενέστερο χρόνο. [20]



Σχήμα 4.4: Γυαλιά BabelFisk

Αυτή η συσκευή αποδεικνύεται ιδιαιτέρως χρήσιμη για τα άτομα με θέματα ακοής που διανύουν τα χρόνια φοίτησής τους καθώς μπορούν ταυτοχρόνως να παρακολουθούν τις διαλέξεις μέσα στην τάξη και να κρατάνε εικονικές σημειώσεις τις οποίες μπορούν να διαβάσουν αργότερα.

Τηλέφωνα με λεζάντες (Captioned Telephones)

Τα σταθερά τηλέφωνα τα οποία εμφανίζουν λεζάντες κατά τη διάρκεια μιας τηλεφωνικής κλήσης προσφέρουν υπέρμετρη υποστήριξη σε άτομα που είναι βαρήκοα και κατέχουν την ικανότητα της ομιλίας, όπως οι ηλικιωμένοι.

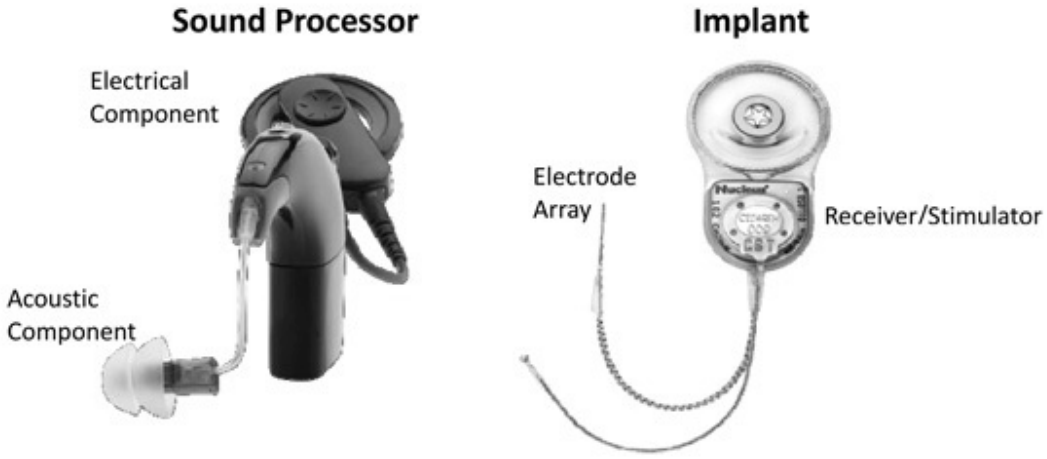
Σχήμα 4.5: Τηλέφωνο με λεζάντες

Όταν μια εισερχόμενη ή εξερχόμενη κλήση λαμβάνει χώρα, η κλήση συνδέεται αυτόματα στο Τηλεφωνικό Κέντρο Υποτιτλισμού (ΤΚΥ) . Στο ΤΚΥ παρακολουθεί την συνομιλία ένας ειδικά εκπαιδευμένος χειριστής ο οποίος επαναλαμβάνει τα λόγια του ακούοντα που θέλει να μιλήσει στο βαρήκοο συνομιλητή του. Ταυτοχρόνως, η τεχνολογία Αναγνώρισης Ομιλίας μεταγράφει αυτόματα την φωνή του χειριστή σε κείμενο (λεζάντες) οι οποίες εμφανίζονται στην οθόνη της τηλεφωνικής συσκευής ώστε να μπορεί να τα διαβάσει ο βαρήκοος συνομιλητής. [21]

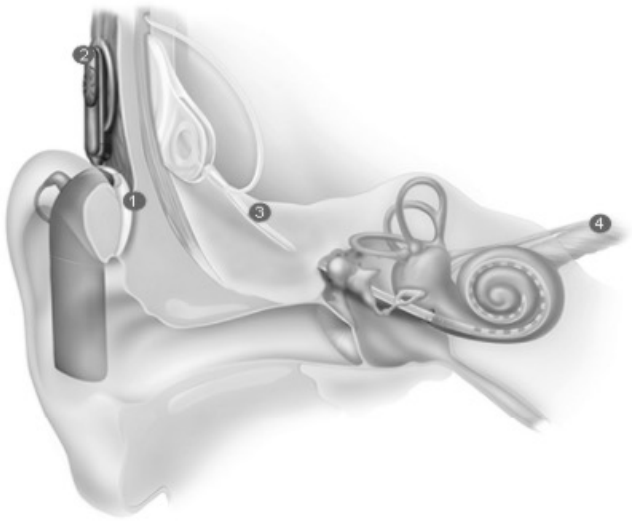
Κοχλιακά Εμφυτεύματα

Μια μεγάλη μερίδα των ανθρώπων με μεγάλη απώλεια ακοής αντιμετωπίζουν το πρόβλημα αυτό επειδή τα τριχοφόρα αγγεία μέσα στον κοχλία τους είναι κατεστραμμένα. Ο κοχλίας, το τύμπανο του αυτιού και όλα τα υπόλοιπα όργανα τα οποία συνεργάζονται μεταξύ τους για την μετάδοση του ήχου στον εγκέφαλο, λειτουργούν κανονικά. Τα κοχλιακά εμφυτεύματα αποτελούν την τελευταία λέξη της τεχνολογίας όσον αφορά τα ακουστικά βοηθήματα και βρίσκονται στην αγορά μόνο δέκα χρόνια. Ο ρόλος τους είναι να παρακάμπτουν τα τριχοειδή αγγεία, εκμεταλλευόμενα τα υπόλοιπα όργανα και να βοηθάνε στην μετάδοση του ήχου απευθείας στον εγκέφαλο.

Ένα κοχλιακό εμφύτευμα αποτελείται από δύο μέρη. Το εμφύτευμα και τον επεξεργαστή ήχου. Το εμφύτευμα τοποθετείται χειρουργικά στον χρήστη όπως φαίνεται στην παρακάτω εικόνα. Ο επεξεργαστής φοριέται εξωτερικώς.



Σχήμα 4.6: Κοχλιακό Εμφύτευμα



Σχήμα 4.7: Κοχλιακό Εμφύτευμα στον Κοχλία

Η μετάδοση του ήχου επιτυγχάνεται ως εξής: [22]

1. Ο επεξεργαστής ήχου, ο οποίος βρίσκεται τοποθετημένος πίσω από το αυτί, συλλαμβάνει τον ήχο και τον μετατρέπει σε ψηφιακή πληροφορία.
2. Στην συνέχεια, μεταφέρει την ψηφιακή πληροφορία αυτή στον μαγνήτη που βρίσκεται εξωτερικά της κεφαλής.

3. Η ψηφιακή πληροφορία εισέρχεται στον εσωτερικό μαγνήτη του κοχλιακού εμφυτεύματος, μέσω του μαγνητικού πεδίου που έχει σχηματιστεί μεταξύ του εσωτερικού και εξωτερικού μαγνήτη. Το εμφύτευμα, έπειτα, μετατρέπει αυτή την πληροφορία σε ηλεκτρικά ερεθίσματα τα οποία και φτάνουν μέχρι μια σειρά ηλεκτροδίων που έχουν τοποθετηθεί στρατηγικά μέσα στον κοχλία (εσωτερικό μέρος αυτιού).
4. Εν τέλει, τα ηλεκτρόδια του εμφυτεύματος διεγείρουν τα νεύρα του κοχλία τα οποία με την σειρά τους στέλνουν τα ερεθίσματα αυτά στον εγκέφαλο και ερμηνεύονται ως ήχος. Έτσι, έχει μεταδοθεί ο ήχος στον εγκέφαλο παρακάμπτοντας τελείως τα κατεστραμμένα τριχοειδή αγγεία του κοχλία.

Τα κοχλιακά εμφυτεύματα θεωρούνται από πολλούς η νέα επανάσταση στον χώρο των ακουστικών βοηθημάτων. Έχουν διεξαχθεί έρευνες που δείξανε πως όσοι τα χρησιμοποιούν έχουν επιτύχει κατά μέσο όρο 80% ποσοστό κατανόησης στην ομιλία, σε αντίθεση με το 10% που παρουσιάζουν όσοι φοράνε τα απλά ακουστικά βοηθήματα. Ακόμη, βοηθάει στην καλύτερη εστίαση ακρόασης του ήχου σε θορυβώδη περιβάλλοντα με αποτέλεσμα ο χρήστης να μπορεί να επικοινωνεί πιο άνετα σε πολυσύχναστα μέρη. Πρέπει να σημειωθεί πως πολύ σημαντικό είναι το γεγονός πως ο χρήστης αισθάνεται πιο ασφαλής στο περιβάλλον του μιας και μπορεί να ανταποκριθεί έγκαιρα σε εξωτερικά ερεθίσματα όπως συναγερμούς, αυτοκίνητα και παρεμφερή συμβάντα.

VV-Talker

Τα παιδιά που παρουσιάζουν έλλειψη ακοής λόγω του ότι σε μικρές ηλικίες δεν έχουν τα ερεθίσματα του ήχου να τα βοηθήσουν στις γλωσσικές τους εξελίξεις, δεν αναπτύσσουν την ομιλία τους στον ίδιο βαθμό με τα υπόλοιπα. Παρά το γεγονός πως το εκπαιδευτικό σύστημα για τους κωφούς έχει βελτιωθεί σημαντικά προσφέροντας ειδική λογοθεραπευτική στήριξη, τα παιδιά αυτά εξακολουθούν να δυσκολεύονται να επικοινωνήσουν με ευφράδεια ή να αφομοιώσουν πληροφορίες γρηγορότερα. Προκειμένου να ξεπεραστεί αυτό το πρόβλημα, δημιουργήθηκε η συσκευή VV-Talker [23].

Η συσκευή αυτή αποτελείται από δύο μέρη, τον προσομοιωτή δόνησης και τον καταγραφέα της δόνησης που προκαλείται από τις φωνητικές χορδές. Αρχικά, ο καταγραφέας τοποθετείται στο λαιμό του εκπαιδευτή, που έχει αναπτύξει καθαρή ομιλία, πάνω από τις φωνητικές του χορδές.

Στη συνέχεια, καταγράφονται ορισμένες φράσεις ή λέξεις οι οποίες αποθηκεύονται στην εσωτερική μνήμη της συσκευής. Από την άλλη, όταν το παιδί που εκπαιδεύεται βάζει τους ήχους αυτούς να αναπαράγονται, νιώθει τις δονήσεις του μηχανήματος σαν να ακουμπάει τις φωνητικές χορδές του εκπαιδευτή του την ώρα που αναπαρήγαγε τις φράσεις / λέξεις αυτές. Έτσι, το παιδί μπορεί στη συνέχεια να δεi τον ηχητικό παλμό που παράγει το ίδιο καθώς και την διαφορά που έχει με το επιθυμητό που πρέπει να φτάσει. Είναι προφανές, λοιπόν, πως αυτή η συσκευή αποτελεί ένα πολύ σημαντικό εργαλείο αυτοβελτίωσης για το ίδιο το παιδί αλλά και για την αποτελεσματικότερη γλωσσική εκπαίδευσή του.

Σχήμα 4.8: VV-Talker

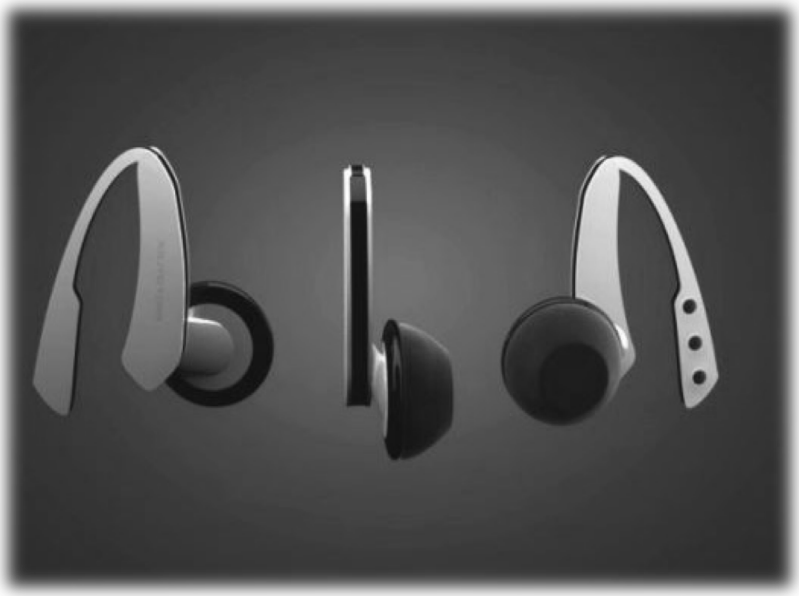
Ακουστικά Μουσικής

Μια πολύ ενδιαφέρουσα και καινοτόμα εφεύρεση αποτελούν τα ακουστικά τα οποία επιτρέπουν σε έναν κωφό να αντιλαμβάνεται την μουσική μέσω δονήσεων. Είναι τα μοναδικά στο είδος τους μιας και επιτυγχάνουν η μουσική να γίνεται αισθητή παρά να ακουστεί, εκμεταλλευόμενα την πολυδιάστατη δυναμική της.

Αρχικά, τα ακουστικά αυτά είναι ασύρματα και έτσι ο χρήστης δεν επηρεάζεται από επιπρόσθετη καλωδίωση. Μπορεί εύκολα να κινείται καθώς και να κάνει άλλα πράγματα ενώ ακούει την μουσική που επιθυμεί. Επιπροσθέτως, τροφοδοτούνται από κυψέλη καυσίμου η οποία είναι γνωστή για την μετατροπή καυσίμου σε ηλεκτρική ενέργεια και φημίζεται για την φιλικότητά της προς το περιβάλλον, προσφέροντας ταυτόχρονα παρατεταμένη χρήση των ακουστικών αυτών.

Από τεχνική άποψη, μέσα στα ακουστικά αυτά έχουν ενσωματωθεί πιεζοηλεκτρικά υλικά τα οποία αναδημιουργούν τους διάφορους ρυθμούς και τα αρμονικά στοιχεία που συστήνουν την μουσική. Όταν τα πιεζοηλεκτρικά υλικά αυτά ενεργοποιούνται, δονούνται σειριακά και μεθοδικά

με τρόπο τέτοιον που να προσομοιώνουν την αναπαραχθείσα μελωδία και έτσι ο κωφός αισθάνεται τον παλμό της.



Σχήμα 4.9: Ακουστικά μουσικής με δόνηση

Δονούμενο Ξυπνητήρι

Ένα καθημερινό πρόβλημα που αντιμετωπίζουν όλα τα κωφά άτομα κατά τη διάρκεια του ύπνου είναι το ξύπνημά τους. Από την στιγμή που δεν μπορούν να ανταποκριθούν στα παραδοσιακά ξυπνητήρια, εφευρέθηκαν τα δονούμενα ξυπνητήρια χειρός. Είναι ελαφρά, λειτουργούν με μπαταρίες και επιτυγχάνουν αποτελεσματικά τον σκοπό τους.

Σχήμα 4.10: Δονούμενο ξυπνητήρι χειρός

4.2 Εφαρμογές για Smart Phone

Στην συνέχεια, έγινε έρευνα, στα πλαίσια της διπλωματικής, των εφαρμογών που υπάρχουν στην αγορά με σκοπό την υποστήριξη χρηστών που παρουσιάζουν έλλειψη ακοής. Οι εφαρμογές αυτές χωρίζονται στις εξής κατηγορίες [24]:

1. Εφαρμογές για Νοηματική

Οι εφαρμογές που ανήκουν σε αυτή την κατηγορία παρακινούν και ενθαρρύνουν τους ανθρώπους να μάθουν την νοηματική γλώσσα. Περιέχουν μια μεγάλη βιβλιοθήκη στην οποία έχουν κατηγοριοποιηθεί τα διάφορα νοήματα και διευκολύνουν τον χρήστη να περιηγηθεί και να μάθει τις κατηγορίες για τις οποίες ενδιαφέρεται περισσότερο. Το μεγάλο πλεονέκτημα των εφαρμογών αυτών είναι κυρίως η προσιτή τους τιμή, η οποία παίζει καθοριστικό ρόλο, καθώς λίγοι έχουν τον χρόνο και τους πόρους που απαιτούνται για μια μακροπρόθεσμη δέσμευση για την εκμάθηση αυτής της γλώσσας.

2. Αναγνώριση ομιλίας και μετατροπή του ήχου σε κείμενο

Αυτού του τύπου οι εφαρμογές μετατρέπουν ζωντανά τον λόγο ενός ατόμου σε κείμενο αναλύοντας το φάσμα της φωνής του. Η ομιλία δημιουργεί δονήσεις στον αέρα και ο Αναλογικός Ψηφιακός Μετατροπέας (ADC) μεταφράζει αυτό το αναλογικό κύμα που προκύπτει σε ψηφιακά δεδομένα τα οποία ο υπολογιστής μπορεί να κατανοήσει και επεξεργαστεί. Στην συνέχεια, το σύστημα φιλτράρει τον ψηφιοποιημένο ήχο ώστε να καταργηθεί ο ανεπιθύμητος θόρυβος αλλά και να χωριστεί η ομιλία σε διαφορετικές ζώνες συχνοτήτων. Εν τέλει, ο ήχος ομαλοποιείται και μετατρέπεται σε κείμενο στην οθόνη όπου και το διαβάζει ο χρήστης. Γενικά, υπάρχει μεγάλο περιθώριο βελτίωσης στις Speech Recognition εφαρμογές αλλά μπορούν να διευκολύνουν σημαντικά την συνεννόηση μεταξύ ενός μη ακούοντα ανθρώπου με έναν ακούοντα.

3. Προσαρμογή ήχου στις ασθενείς συχνότητες του ατόμου

Ο χρήστης έχει την δυνατότητα να κάνει ένα ακουόγραμμα - τεστ ακοής που είναι ενσωματωμένο μέσα στην εφαρμογή και να διαπιστώσει αν η ακοή του είναι εντός των φυσιολογικών ορίων ή έχει πιθανές απώλειες που του δημιουργούν δυσκολίες στην καθημερινή του ζωή. Στην συνέχεια, ανάλογα τις συχνότητες στις οποίες προέκυψε πως ο χρήστης ακούει λιγότερο, η εφαρμογή βελτιώνει και δυναμώνει την έξοδο του ήχου στις συχνότητες αυτές. Τέτοιου είδους εφαρμογές

βοηθάνε πολύ τα άτομα που είναι μεγαλύτερης ηλικίας και δεν θέλουν ή δεν έχουν τα χρήματα να αγοράσουν ακουστικά βοηθήματα.

4. Ειδοποίηση μέσω δόνησης ή μηνύματος για την ύπαρξη ήχου

Μια ιδιαίτερα αναγκαία κατηγορία εφαρμογών αποτελούν αυτές που μόλις ακουστεί κάποιος ήχος, ο χρήστης θα ειδοποιηθεί μέσω δόνησης στην συσκευή επιλογής του και με την εμφάνιση αντίστοιχου μηνύματος. Δυστυχώς, τέτοιου είδους εφαρμογές είναι σε πολύ πρώιμο στάδιο μιας και οι περισσότερες μπορούν να αναγνωρίσουν μόνο τους πολύ βασικούς ήχους (συναγερμός, κουδούνι) και χωρίς μεγάλη επιτυχία. Επίσης, κάποιες ήδη υπάρχουσες εφαρμογές απλώς ειδοποιούν τον χρήστη μέσω δόνησης όταν κάποιος ήχος ακούγεται χωρίς να υποδεικνύεται ποιος είναι.

5. Συνδυασμός των παραπάνω

Τέλος, εννοείται πως συμπεριλαμβάνονται οι εφαρμογές οι οποίες αποτελούν μείγμα των προαναφερθέντων κατηγοριών. Ένα απλό παράδειγμα μπορεί να αποτελέσει μια εφαρμογή η οποία μετατρέπει ηχητικά μηνύματα (voice mails) σε κείμενο το οποίο και στέλνει σαν γραπτό (SMS) ή ηλεκτρονικό μήνυμα (e-mail) στον χρήστη.

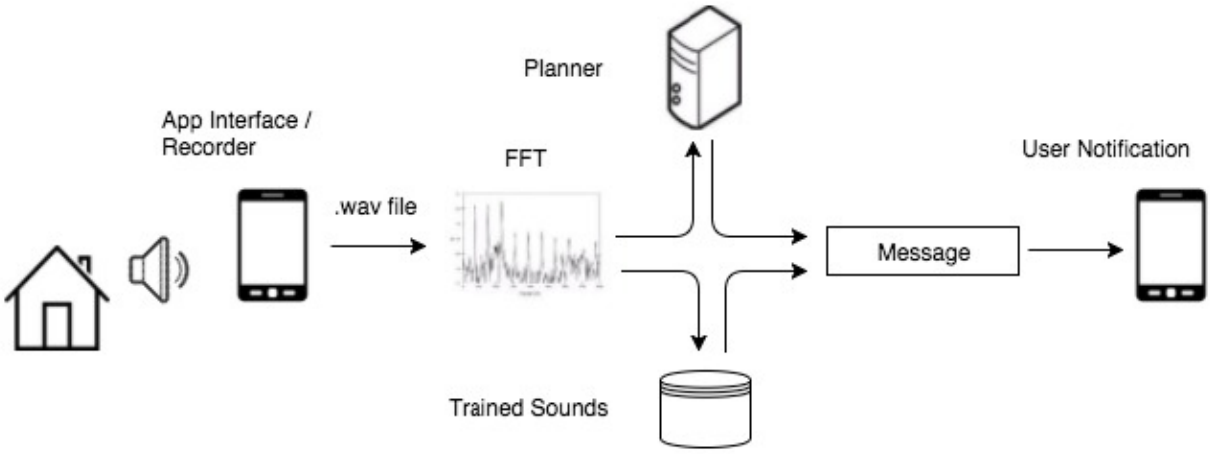
Από τα παραπάνω καθίσταται εμφανές ένα κενό στην υποστηρικτικές εφαρμογές για τα άτομα με θέματα ακοής και αυτό ακριβώς το κενό συμπληρώνει η παρούσα διπλωματική.

5 Ανάπτυξη Πλευράς Εξυπηρετητή (Back-End Development)

5.1 Περιγραφή και Υλοποίηση Back-end

Στην εφαρμογή που δημιουργήθηκε στα πλαίσια της διπλωματικής, αξιοποιήσαμε το **ολοκληρωμένο προγραμματιστικό περιβάλλον (IDE) Android Studio** το οποίο χρησιμοποιείται για την ανάπτυξη εφαρμογών στην πλατφόρμα Android.

Η εφαρμογή μας έχει ως σκοπό να αναγνωρίζει τους ήχους που λαμβάνουν χώρα στο σπίτι. Για να το επιτύχει αυτό, ηχογραφεί κομμάτια ήχου τα οποία αναλύονται και υποβάλλονται σε ανάλυση FFT προκειμένου να ληφθούν τα αναγκαία δεδομένα. Στην συνέχεια, χρησιμοποιώντας την έξοδο της προαναφερθείσας ανάλυσης στέλνουμε τα απαραίτητα δεδομένα μέσω του πρωτοκόλλου HTTP στο λογισμικό Node-Red. Μέσα στο Node-Red πραγματοποιείται εκτέλεση αναζήτησης σε δύο βάσεις δεδομένων (Planner και Trained Sounds) η οποία αναγνωρίζει και ταυτοποιεί τον ήχο που ηχογραφήθηκε. Τέλος, η απάντηση εμφανίζεται στην διεπαφή χρήστη. Η περιγραφείσα σύνθεση εμφανίζεται στην παρακάτω εικόνα:



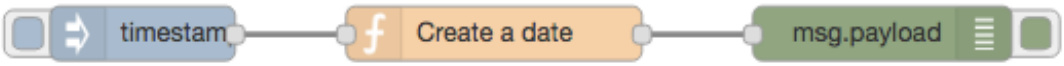
Σχήμα 5.1: Σύνθεση Back-End της εφαρμογής μας

5.1.1 Node-Red

Κομμάτι της υλοποίησης του περιβάλλοντος του διακομιστή αποτελεί το λογισμικό **Node-Red** [25]. Το λογισμικό Node-Red είναι ένα εικονικό εργαλείο που καθιστά δυνατή την διασύνδεση συσκευών μεταξύ τους καθώς και την χρήση της διεπαφής προγραμματισμού εφαρμογών (APIs) και online υπηρεσιών.

Το Node-Red ουσιαστικά παρέχει έναν επεξεργαστή ροής (flow editor) ο οποίος φιλοξενείται στο πρόγραμμα περιήγησης (browser). Ο flow editor διευκολύνει την διασύνδεση ροών μεταξύ τους εκμεταλλευόμενος το μεγάλο εύρος των κόμβων (nodes) που προσφέρονται μέσα στην παλέτα, η οποία έχει αναπτυχθεί ειδικά γι' αυτόν τον σκοπό. Ακόμη, υπάρχει η δυνατότητα να δημιουργηθούν JavaScript συναρτήσεις, εντός των κόμβων, χρησιμοποιώντας πλούσιο επεξεργαστή κειμένου, οι οποίες δύναται στην συνέχεια να αποθηκευτούν στην βιβλιοθήκη του Node-Red ώστε να επαναχρησιμοποιηθούν στο μέλλον. Όλα τα flows που δημιουργούνται μέσα στο Node-Red έχουν την δυνατότητα να εξαχθούν σε μορφή JSON κάτι που καθιστά πολύ εύκολο τον διαμοιρασμό τους.

Παρακάτω παρατίθεται, για καλύτερη κατανόηση του Node-Red, ένα απλό διάγραμμα ροής στο οποίο υπάρχουν τρείς κόμβοι και εμφανίζεται σαν έξοδος η τρέχουσα ημερομηνία.

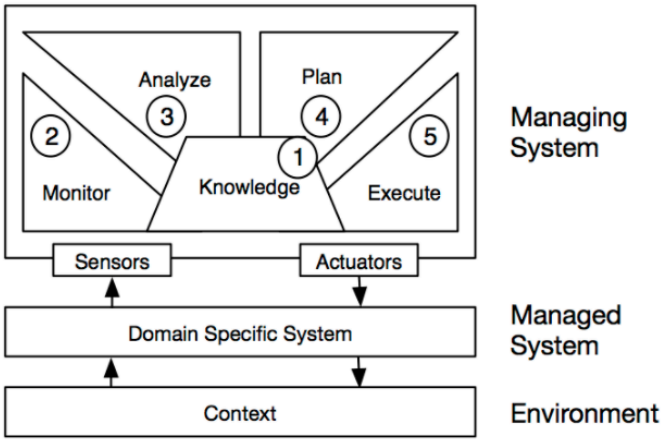


Σχήμα 5.2: Απλό παραδειγματικό διάγραμμα ροής Node-Red

5.1.2 Planner

Για την κατηγοριοποίηση των ήχων χρησιμοποιείται ο Planner [26] ο οποίος είναι ένας διακομιστής (server) που λαμβάνει αιτήσεις HTTP και μετά από αναζήτηση σε προϋπάρχουσα βάση δεδομένων (Case Base), χρησιμοποιώντας την μέθοδο CBR (Case – Based Reasoning), επιστρέφει την πιθανή κατηγορία στην οποία ανήκει ο ήχος.

Όσον αφορά τον προγραμματισμό του Planner, ειναι αυτοδιαχειρίσιμος χάρη στον υψηλού επιπέδου σχεδιασμό του, που βασίζεται στις αρχές του βρόχου MAPE-K [27], ο οποίος αποτελεί μια βασική εννοιολογική πτυχή του πεδίου της Αυτόνομης Πληροφορικής. Ο αυτόνομος βρόχος MAPE-K (Monitor, Analyze, Plan, Execute and Knowledge) αποτελεί ένα πρότυπο για το σχεδιασμό αυτόνομων συστημάτων, όπου ένα διαχειριζόμενο στοιχείο συντονίζεται από ένα βρόχο ο οποίος είναι δομημένος σε τέσσερις φάσεις και μια κοινή γνώση.



Σχήμα 5.3: MAPE-K Βρόχος

Για να εισπνευστεί νοημοσύνη στα Πράγματα, πρέπει να είναι εξοπλισμένα με μια λειτουργικότητα που θα τους επιτρέψει να έχουν συλλογισμό. Έτσι, το πρώτο βήμα για την ανάπτυξη ενός συστήματος που θα παίρνει αποφάσεις είναι να επιλεχθεί μια τεχνική συλλογισμού την οποία και θα ακολουθεί το σύστημα αυτό.

Η τεχνική συλλογισμού που επιλέχθηκε για τον Planner της εφαρμογής μας είναι η Αιτιολογική Σκέψη που βασίζεται στην Υπόθεση (Case – Based Reasoning). Η τεχνική αυτή βασίζεται κατά κύριο λόγο στην διαίσθηση ότι ένα μεγάλο μέρος της ανθρώπινης εμπειρογνωμοσύνης λειτουργεί

με την ανάκτηση και τροποποίηση προηγούμενων προβλημάτων που έχουν αποθηκευτεί στην μνήμη.

Επομένως, ο Planner είναι ένα αυτόνομο, δυναμικά διαμορφωμένο στοιχείο λογισμικού, το οποίο κάνει πλήρη χρήση του παραδείγματος CBR και υλοποιεί τον κύκλο CBR. Αποτελείται από λειτουργίες παρασκηνίου καθώς και απο περιφερειακές λειτουργίες, οι οποίες μπορούν να προσεγγιστούν με παραμετρικές και δυναμικές λειτουργίες. Ο Planner ρυθμίζει την ροή της Γνώσης από και προς την Βάση Δεδομένων (Case Base) και υλοποιεί τα σχέδια ενεργοποίησης που προκύπτουν από τον Συλλογισμό σχετικά με τις διαθέσιμες πληροφορίες εισόδου.

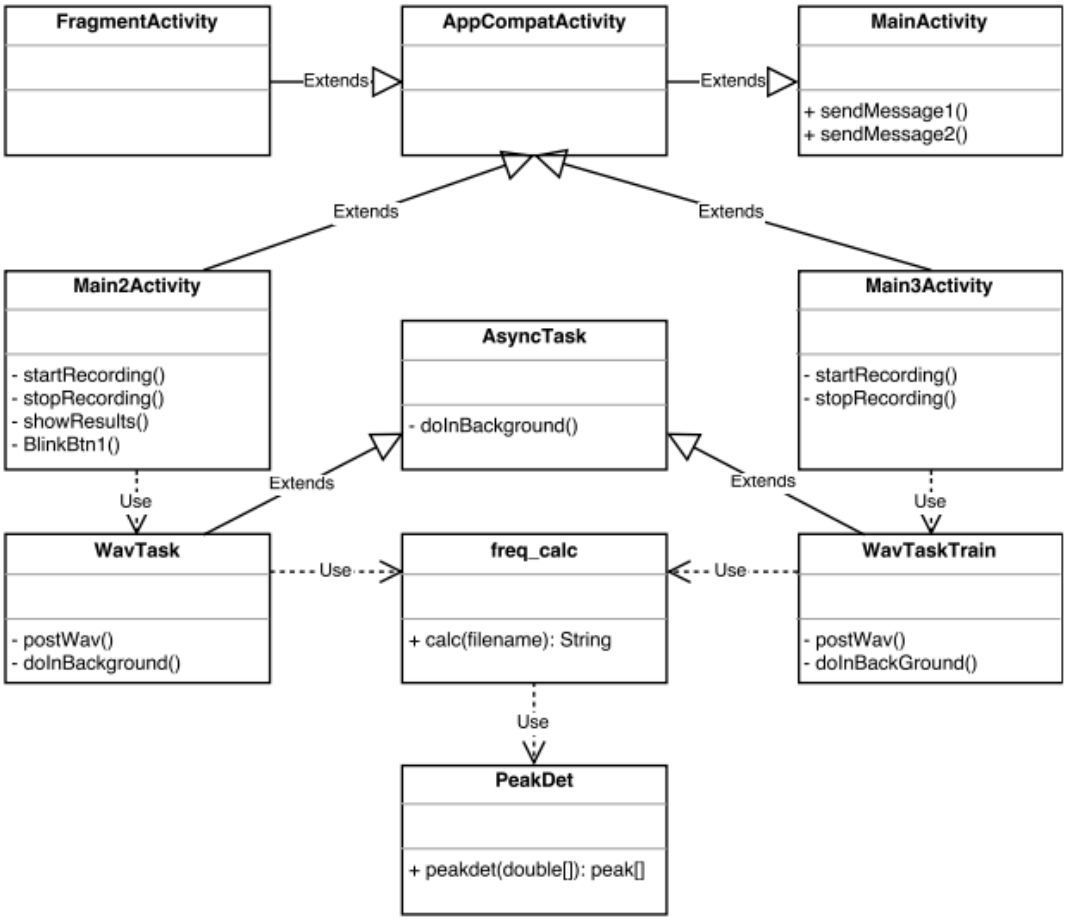
Μια υπόθεση (case) μπορεί να θεωρηθεί συνδυασμός ενός προβλήματος με την λύση του, ενώ ένα πρόβλημα μπορεί να αποτελείται από ένα ή περισσότερα Συμβάντα (Events) ή Στόχους (Goals). Με άλλα λόγια, μια υπόθεση μπορεί να αποτελέσει και ένα είδος κανόνα για ένα σχέδιο ενεργοποίησης, το οποίο ενεργοποιείται όταν εντοπιστούν συγκεκριμένα Συμβάντα.

Στην συγκεκριμένη περίπτωση, οι αισθητήρες (Sensors) που θα είναι τοποθετημένοι στο περιβάλλον του σπιτιού θα λειτουργούν και ως ενεργοποιητές (Actuators) του Planner. Όταν θα παράγεται ήχος μέσα στην κατοικία, οι αισθητήρες (μικρόφωνα) θα εντοπίζουν τον ήχο αυτόν, θα τον ηχογραφούν και θα τον στέλνουν σε κατάλληλη μορφή (Γνώση) στον Planner ο οποίος και θα ενεργοποιείται εκτελώντας ορισμένες ενέργειες (Events). Μετά την ενεργοποίηση του Planner, πραγματοποιείται σύγκριση του εισερχόμενου ήχου με αυτούς που έχουν αποθηκευτεί στην οντολογία του (Knowledge). Τέλος, έχοντας περάσει όλα τα βήματα του βρόχου MAPE-K, ο Planner δίνει σαν έξοδο την κατηγορία του ήχου που ηχογραφήθηκε από τους αισθητήρες.

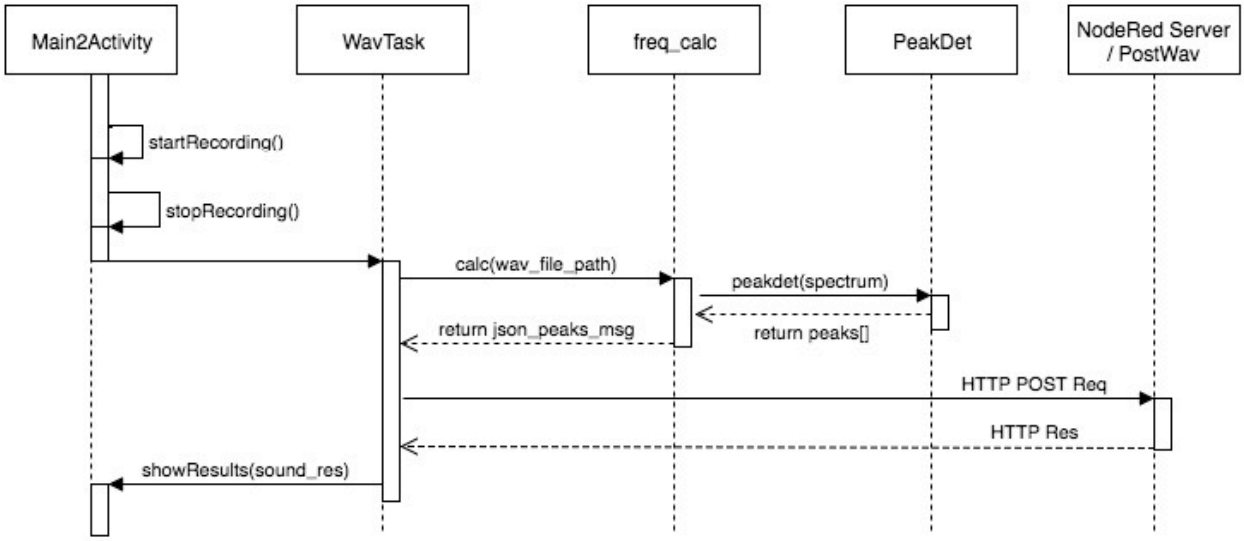
Να σημειωθεί πως ο εμπλουτισμός της οντολογίας του Planner πραγματοποιήθηκε με προσεκτικά επιλεγμένους ήχους από κάθε μία από τις ακόλουθες κατηγορίες: Ήχος Τηλεφώνου, Συναγερμός Φωτιάς, Χτύπος Πόρτας, Κουδούνι πόρτας και τέλος Κλάμα Μωρού. Οι ήχοι είναι είκοσι σε κάθε κατηγορία και έχουν ευρεθεί από το Διαδίκτυο έπειτα από επιλεκτική αναζήτηση και αξιολόγηση.

5.2 Αρχιτεκτονική Συστήματος σε Επίπεδο Κλάσεων

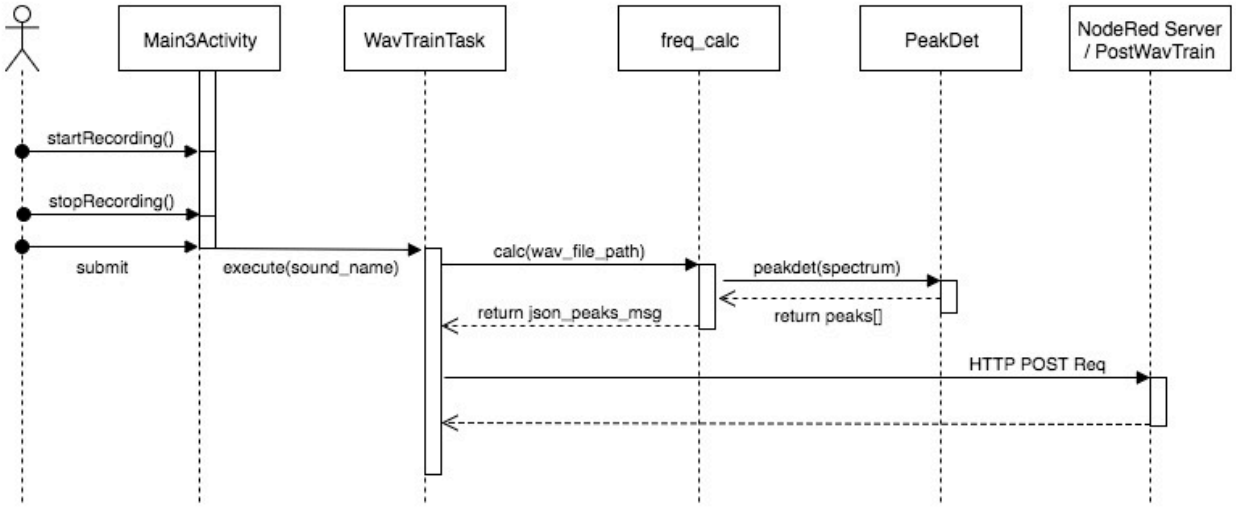
Στην παρούσα ενότητα θα παρουσιάσουμε τρία UML διαγράμματα των κλάσεων που έχουν δημιουργηθεί για την εφαρμογή και δείχνουν τη μεταξύ τους σχέση, συμπεριλαμβάνοντας όποια πληροφορία είναι απαραίτητη για την πλήρη κατανόηση της συνοχής του κώδικα της εφαρμογής. Το πρώτο διάγραμμα αποτελεί ένα απλό διάγραμμα κλάσεων. Τα επόμενα δύο είναι διαγράμματα ακολουθίας (sequence diagrams). Στην συνέχεια θα αναλυθεί η χρήση της κάθε κλάσης ξεχωριστά.



Σχήμα 5.4: Απλό διάγραμμα Κλάσεων της εφαρμογής



Σχήμα 5.5: Sequence διάγραμμα Κλάσεων για real-time επεξεργασία



Σχήμα 5.6: Sequence διάγραμμα Κλάσεων για Trained Sounds

5.2.1 MainActivity()

Η κλάση MainActivity αποτελεί το κύριο activity της εφαρμογής, δηλαδή το activity που εμφανίζεται στον χρήστη όταν ξεκινάει η εφαρμογή. Η κλάση αυτή κληρονομεί την AppCompatActivity η οποία αποτελεί μέρος του πακέτου android.support.v7.app. Είναι παρόμοια της γνωστής και παλαιότερης κλάσης Activity αλλά περιέχει επιπλέον βιβλιοθήκες οι οποίες βοηθάνε στην προσθήκη και τον χειρισμό του Action Bar.

Η MainActivity αποτελείται από τρείς μεθόδους:

Μέθοδος	Περιγραφή
OnCreate()	Η μέθοδος αυτή χρησιμοποιείται για την εκκίνηση κάποιας δραστηριότητας. Στην δική μας περίπτωση απλά θέτει το αρχικό γραφικό περιβάλλον του χρήστη και του εμφανίζει δύο επιλογές εκ των οποίων η καθεμία ενεργοποιεί διαφορετική κλάση.
sendMessage1()	Η sendMessage1() μέθοδος καλείται όταν ο χρήστης επιλέξει το κουμπί «Start» και ενεργοποιείται η MainActivity2() κλάση.
sendMessage2()	Η μέθοδος αυτή καλείται όταν ο χρήστης επιλέξει το κουμπί «Start» και καλείται η κλάση MainActivity3().

Ουσιαστικά στην MainActivity ο χρήστης καλείται να διαλέξει μεταξύ δύο επιλογών και αναλόγως την επιλογή καλείται η αντίστοιχη κλάση που πραγματοποιεί την επιθυμία του. (Σχήμα 6.1)

5.2.2 MainActivity2()

Η κλάση MainActivity2() αποτελεί και την πιο βασική κλάση της εφαρμογής μας, μιας και λαμβάνουν χώρα σε αυτήν οι πιο βασικές λειτουργίες της. Απαρτίζεται από τις εξής μεθόδους:

Μέθοδος	Περιγραφή
onCreate()	Η μέθοδος onCreate() στην παρούσα περίπτωση εκτελεί δύο λειτουργίες. Αρχικά, ορίζει το γραφικό περιβάλλον του χρήστη. Στην συνέχεια καλεί την μέθοδο

	startRecording() και μπαίνει σε ατέρμων βρόχο καλώντας ένα νήμα παράλληλο νήμα εκτέλεσης (Thread) το οποίο “κοιμάται” για 5 δευτερόλεπτα και έπειται καλεί την μέθοδο stopRecording(). Με αυτήν την τεχνική επιτυγχάνεται η ηχογράφηση αρχείων .wav διάρκειας 5 δευτερολέπτων.
startRecording()	Η μέθοδος αυτή αξιοποιεί την κλάση AudioRecord η οποία ειναι ενσωματωμένη στο Android Studio και όταν καλείται, το κινητό αρχίζει να ηχογραφεί.
writeAudioDataToFile()	Η writeAudioDataToFile() καλείται κατά την διάρκεια της startRecording και ουσιαστικά γράφει τα δεδομένα της τρέχουσας ηχογράφησης σε ένα αρχείο.
stopRecording()	Η μέθοδος stopRecording() δίνει εντολή στην συσκευή να σταματήσει να ηχογραφεί. Στην συνέχεια, καλούνται οι μέθοδοι copyWaveFile() και deleteTempFile().
copyWaveFile()	Η copyWaveFile() μέσω της μεθόδου getTempFileName() διαβάζει το τρέχον προσωρινά ηχογραφημένο αρχείο ήχου, που έχει διάρκεια 5 δευτερολέπτων, και το μετατρέπει σε αρχείο .wav το οποίο είναι αποθηκευμένο τοπικά στην συσκευή ηχογράφησης, μέσω της getFilename().
getFilename()	Επιστρέφεται το όνομα του αρχείου που είναι τοπικά αποθηκευμένο στην συσκευή και αποθηκεύονται εκεί όλες οι ηχογραφήσεις σε μορφή .wav.
getTempFilename()	Επιστρέφεται η τοποθεσία στην οποία έχει αποθηκευτεί η προσωρινή ηχογράφηση μέσα στο κινητό.
deleteTempFile()	Διαγράφεται το προσωρινό αρχείο το οποίο περιέχει μόνο τις υπό εξέλιξη ηχογραφήσεις.
WriteWaveFileHeader()	Η μέθοδος αυτή καλείται από την copyWaveFile() και ο ρόλος της είναι να προσθέσει την επικεφαλίδα (header) των .wav στα αρχεία ήχου που αποθηκεύονται στην συσκευή του χρήστη.
showResults()	Σε αυτή τη μέθοδο λαμβάνεται σαν είσοδος σε string το είδος του ήχου που αναγνωρίστηκε. Στην συνέχεια, μέσω ενός if-else βρόχου καλεί τις συναρτήσεις BlinkBtn1-4 οι οποίες ειδοποιούν τον χρήστη.

BlinkBtn1()	Αν ο ήχος που έχει αναγνωριστεί είναι κουδούνι πόρτας, τότε καλείται αυτή η μέθοδος η οποία προκαλεί δόνηση της συσκευής για μισό δευτερόλεπτο δευτερόλεπτο και αναβοσβήνει η αντίστοιχη ένδειξη έως ότου ο χρήστης ανταποκριθεί ότι έλαβε γνώση της ειδοποίησης.
BlinkBtn2()	Αν ο ήχος που έχει αναγνωριστεί είναι κουδούνισμα τηλεφώνου, τότε καλείται αυτή η μέθοδος η οποία προκαλεί δόνηση της συσκευής για μισό δευτερόλεπτο δευτερόλεπτο και αναβοσβήνει η αντίστοιχη ένδειξη έως ότου ο χρήστης ανταποκριθεί ότι έλαβε γνώση της ειδοποίησης.
BlinkBtn3()	Αν ο ήχος που έχει αναγνωριστεί είναι κλάμα μωρού, τότε καλείται αυτή η μέθοδος η οποία προκαλεί δόνηση της συσκευής για μισό δευτερόλεπτο δευτερόλεπτο και αναβοσβήνει η αντίστοιχη ένδειξη έως ότου ο χρήστης ανταποκριθεί ότι έλαβε γνώση της ειδοποίησης.
BlinkBtn4()	Αν ο ήχος που έχει αναγνωριστεί είναι συναγερμός φωτιάς, τότε καλείται αυτή η μέθοδος η οποία προκαλεί δόνηση της συσκευής για μισό δευτερόλεπτο δευτερόλεπτο και αναβοσβήνει η αντίστοιχη ένδειξη έως ότου ο χρήστης ανταποκριθεί ότι έλαβε γνώση της ειδοποίησης.

Συνοπτικά, μόλις κληθεί η παρούσα κλάση, η εφαρμογή ηχογραφεί συνεχόμενα το περιβάλλον και αποθηκεύει τις ηχογραφήσεις σε αρχεία τύπου .wav των 5 δευτερολέπτων. Για να αποφευχθεί η μεγάλη δέσμευση της μνήμης του κινητού έχει επιλεχθεί η συνεχής αντικατάσταση κάθε παλιού αρχείου .wav με το νέο που έχει ηχογραφηθεί, ώστε να υπάρχει μόνο ένα αρχείο .wav αποθηκευμένο στην συσκευή του κινητού.

Γενικά όταν ένα αρχείο .wav αποθηκευτεί, το επόμενο βήμα είναι η κλήση της κλάσης WavTask() η οποία επιστρέφει το είδος του ήχου και εν τέλει εμφανίζεται αντίστοιχη ειδοποίηση στον χρήστη. (Σχήματα 6.2-6.7)

5.2.3 WavTask()

Η κλάση WavTask() κληρονομεί την κλάση AsyncTask() και χρησιμοποιείται κυρίως για την επικοινωνία της εφαρμογής στο κινητό με το Node-Red και αντιστοίχως με τον Server του Planner. Ο λόγος που χρησιμοποιήθηκε η AsyncTask() κλάση είναι επειδή λειτουργεί στο background της Διεπαφής Χρήστη (UI) και μας επιτρέπει να ανταλλάξουμε πληροφορίες με το διαδίκτυο ενώ τρέχει παράλληλα το UI. Έτσι, δεν γίνεται αισθητή στον χρήστη η καθυστέρηση που προκαλείται λόγω της κίνησης των δεδομένων από και προς το διαδίκτυο.

Η WavTask(), εκτός από τις ρουτίνες που αναφέρθηκαν στην προηγούμενη παράγραφο και εκτελούν τις αντίστοιχες διεργασίες, περιέχει μόνο δύο μεθόδους. Όταν καλείται έχει σαν είσοδο ένα απλό αρχείο .wav το οποίο στην συνέχεια αναλύει και αποθηκεύει σε ένα διάνυσμα (vector) τις πέντε επικρατέστερες συχνότητες που εμφανίζονται στο φάσμα ήχου του .wav. Για να επιτευχθεί προαναφερθείσα η διεργασία καλείται η κλάση freq_calc() η οποία θα αναλυθεί στην επόμενη υποενότητα. Η freq_calc επιστρέφει ένα JSON Object (JavaScript Object Notation) το οποίο έχει προσαρμοστεί σε κατάλληλη μορφή ώστε να διαβαστεί μετέπειτα από τον Planner και να συγκριθεί με τους ήδη υπάρχοντες ήχους που είναι αποθηκευμένοι στην Οντολογία (Ontology) του Planner. Τέλος, μέσω της postWav() στέλνει μέσω HTTP πρωτοκόλλου το vector στο Node-red.

5.2.3.1 AsyncTask()[28]

Σε γενικές γραμμές πρόκειται για ένα νήμα (thread) εντός μιας κλάσης το οποίο επιτρέπει την επικοινωνία με το UI thread (το κεντρικό thread) της εφαρμογής με απλό και σίγουρο τρόπο, δίχως να απασχολούν τον προγραμματιστή ζητήματα συγχρονισμού καθώς όλες αυτού του είδους διεργασίες αναλαμβάνονται από την AsyncTask() κλάση. Αυτό επιτυγχάνεται επειδή η AsyncTask() παρέχει μια σειρά από ιδιότητες (properties), συναρτήσεις και διαδικασίες πάνω στις οποίες μπορεί κανείς να χτίσει (να κάνει δηλαδή extend) την δική του κλάση (thread).

Οι ρουτίνες που προσφέρει η AsyncTask() είναι πολλές αλλά τέσσερις θεωρούνται βασικές. Η πρώτη είναι η onPreExecute() η οποία εκτελείται εντός του UI Thread και στην οποία δημιουργείται ένα ProgressDialog που ενημερώνει τον χρήστη ότι η εφαρμογή ξεκινά μια χρονοβόρα διαδικασία.

Η δεύτερη ρουτίνα της AsyncTask() είναι η doInBackground(), στην οποία τοποθετούνται οι εντολές που επιθυμεί ο προγραμματιστής να εκτελεστούν στο Thread της AsyncTask() κλάσης. Η συνάρτηση αυτή είναι τύπου μεταβλητού αριθμού παραμέτρων, δηλαδή το είδος της επιστρεφόμενης τιμής καθορίζεται από τον προγραμματιστή κατά την δήλωση της AsyncTask().

Η τρίτη ρουτίνα είναι η onPostExecute(). Ομοίως με την onPreExecute() εκτελείται εντός του UI Thread. Επίσης, εδώ μπορεί ο προγραμματιστής να ενημερώσει το UI Thread για τα αποτελέσματα της AsyncTask, μιας και η onPostExecute() δέχεται ως όρισμα της το αποτέλεσμα της doInBackground()).

Η τέταρτη είναι η onProgressUpdate() η οποία και αυτή όπως η προηγούμενη διαδικασία τρέχει εντός του UI Thread. Καθιστά εφικτή την ενημέρωση του υπόλοιπου προγράμματος για την εξέλιξη των εντολών που εκτελούνται εντός της doInBackground κάθε φορά που η τελευταία εκτελεί την συνάρτηση publishProgress.

Τέλος, υπάρχει και η onCancelled() που κυρίως διαχειρίζεται περιπτώσεις ακύρωσης του AsyncTask από τον χρήστη.

5.2.3.2 PostWav()

Στην μέθοδο PostWav() ουσιαστικά λαμβάνει χώρα η URL σύνδεση του Node-Red (Server) με την συσκευή μας (Client). Για να επιτευχθεί η σύνδεση αυτή, κληρονομεί από την HttpURLConnection() η οποία είναι μια γνωστή κλάση που βρίσκεται στην βιβλιοθήκη της Java και υποστηρίζει το πρωτόκολλο επικοινωνίας HTTP.

Η χρήση της κλάσης στην εφαρμογή μας ακολουθεί το εξής μοτίβο [29]:

- Δημιουργία νέας σύνδεσης HttpURLConnection (http://localhost:1880/PostWav) μέσω της κλήσης της μεθόδου url.openConnection() και χύτευση του αποτελέσματος στην μεταβλητή(?) HttpURLConnection.
- Προετοιμασία της αίτησης (request). Για να γίνει εφικτή η αποδοχή της αίτησης από τον Server είναι απαραίτητο ο Client να την προσαρμόσει κατάλληλα. Συγκεκριμένα στην

παρούσα περίπτωση στέλνουμε αρχεία JSON οπότε στον header ορίζουμε RequestMethod ως “POST” και RequestProperty ως “Content-Type”.

- Εκφράζουμε μετά την επιθυμία μας για μία output connection με “connection.setDoOutput(true);” . Τούτο πρακτικά σημαίνει ότι θα σταλεί και σώμα από τον client, το οποίο θα πρέπει να διαβάσει ο server. Το σώμα αυτό πρέπει να γραφεί στην πλευρά του client σε ένα output stream.
- Τα δεδομένα προς μεταφορά μεταφορτώνονται μέσω εγγραφής στο ρεύμα (stream) που επιστρέφεται από την προαναφερθείσα μέθοδο getOutputStream().
- Όταν ο Server επεξεργαστεί τις συχνότητες και τις συγκρίνει, επιστρέφει το είδος του ήχου που έχει ακουστεί. Η συσκευή μας λαμβάνει την απάντηση (response) του Server η οποία είναι String και λαμβάνεται ως επιστροφή του stream της μεθόδου getInputStream().
- Εν τέλει, το τελευταίο βήμα της σύνδεσης HttpURL είναι η αποσύνδεση. Αφού έχει διαβαστεί η απάντηση η εφαρμογή καλείται να κλείσει την σύνδεση HttpURL μέσω της κλήσης της μεθόδου disconnect(). Με την αποσύνδεση απελευθερώνονται όλοι οι πόροι που έχουν δεσμευτεί από την σύνδεση που διακόπηκε και έτσι μπορούν να επαναχρησιμοποιηθούν σε μελλοντική σύνδεση.

Παρακάτω παρατίθεται ως παράδειγμα μια POST Request σε μορφή JSON από εμάς προς τον Planner η οποία περιέχει τις 5 κύριες συχνότητες των .wav που αναλύουμε καθώς και τα 5 κύρια βάρη τους (weights).

{ "fName":
"{%payload%:{%message_type%:%1%,%problem_attributes%:%freq1#freq2#freq3#freq4#freq5%,%problem_values%:%428.3088684082031#1.3458251953125#2.355194091796875#0.5046844482421875#160.9943389892578%,%solution_attributes%:%hasSoundof%,%forSharing%:%true%,%weights%:%0.5#0.06066846304312529#0.056644159149927244#0.04854882221897655#0.04537751630253077%,%threshold%:%0.6%}}" }

Η POST Response που επιστρέφεται είναι String και αντιστοιχεί στο μήνυμα που θα εμφανιστεί μετέπειτα στον χρήστη: ”The sound is DoorKnock.”. (Σχήμα 6.6)

5.2.3.3 getWavPath()

Η getWavPath() εκτελεί την λειτουργία που είναι και η προφανής. Επιστρέφει την τοποθεσία στην οποία βρίσκεται αποθηκευμένο στην μνήμη του κινητού το αρχείο .wav το οποίο θέλουμε να θέσουμε προς επεξεργασία και να στείλουμε μετέπειτα τις συχνότητές του μέσω JSON στο Node-Red.

5.2.4 PeakDet()

Η PeakDet() είναι η κλάση στην οποία υλοποιείται η συνάρτηση PeakDet της Matlab. Η συνάρτηση αυτή ουσιαστικά υπολογίζει τις θέσεις και τις τιμές των κορυφών (peaks) σε ένα μονοδιάστατο σήμα. Αφού φτιάξεις ένα αντικείμενο της κλάσης (με new), καλείς την συνάρτηση PeakDet (δηλαδή η συνάρτηση που πραγματοποιεί την λειτουργία είναι η PeakDet.peakdet()). Σαν παραμέτρους, η συνάρτηση δέχεται το σήμα και μία παράμετρο delta (δέλτα). Ως κορυφή (peak) θεωρείται το σημείο του σήματος που αποτελεί τοπικό μέγιστο και το προηγούμενο ακριβώς σημείο είναι μικρότερό του, τουλάχιστον κατά την τιμή του delta.

5.2.5 freq_calc()

Στην παρούσα κλάση αξιοποιείται η βιβλιοθήκη “Musicg” η οποία έχει δημιουργηθεί από την Google. Ουσιαστικά αποτελεί μια απλή βιβλιοθήκη ανάλυσης ήχου η οποία έχει γραφεί σε Java και έχει σαν σκοπό την εξαγωγή χαρακτηριστικών των ήχων τόσο σε υψηλό επίπεδο όσο και σε χαμηλό.

Το δοθέν API (Application Programming Interface) της Musicg επιτρέπει στον προγραμματιστή να εξάγει χαρακτηριστικά ήχων και να επεξεργάζεται τα δεδομένα τους αποτελεσματικά, όπως πχ. την ανάγνωσή τους ή την αποκοπή ορισμένων σημείων. Επίσης, παρέχονται κατάλληλα εργαλεία που βοηθάνε στην επεξεργασία του ψηφιακού σήματος και καθιστούν την κυματομορφή (waveform) των αρχείων .wav και το φασματογράφημά (spectrum) τους κατάλληλα για επεξεργασία από τον προγραμματιστή.

Ο κύριος λόγος που επιλέχθηκε αυτή η βιβλιοθήκη κατά την διάρκεια της δημιουργίας της εφαρμογής μας είναι λόγω της συμβατότητάς της με το Android Studio καθώς και λόγω του ότι

παρέχει εξαγωγή τιμών σήματος και FFT (Fast Fourier Transform) καθαρά για ήχους χωρίς να χρειάζονται άλλες κλάσεις ή μιγαδικοί αριθμοί.

Στην κλάση αυτή χρησιμοποιούνται οι δύο προαναφερθείσες (PeakDet, Musicg) για να επιτευχθεί η επιτυχής αναγνώριση των πέντε βασικών συχνοτήτων ενός αρχείου .wav και η επιστροφή τους στον χρήστη.

Η συνάρτηση που παίζει τον βασικότερο ρόλο είναι λεγόμενη calc(). Δέχεται το όνομα ενός .wav αρχείου και επιστρέφει ένα String με JSON περιεχόμενο που περιέχει τις πέντε συχνότητες που αποτελούν επικρατέστερα peaks και τα ύψη τους.

Αρχικά, ανοιγεί το αρχείο και παίρνει τις τιμές του σήματος χρησιμοποιώντας την musicg (συνάρτηση Wave.getNormalizedAmplitudes()). Στην συνέχεια, φτιάχνει ένα σήμα με μέγεθος που αποτελεί δύναμη του δύο (μεγαλύτερο από το αρχικό) που αποτελεί κυκλική επανάληψη του αρχικού. Έπειτα χρησιμοποιώντας πάλι την musicg παίρνει τις τιμές του FFT αυτού του σήματος (συνάρτηση FastFourierTransform.getMagnitutes (signalArray)) και σύμφωνα με το μέγεθος του πίνακα και το sampleRate του αρχείου (το SampleRate από την Wave.getWaveHeader().getSampleRate() της musicg), δημιουργείται ο άξονας με τις συχνότητες του FFT πχ:

index=0 → 0Hz ,
index=1 → 2.1Hz , ...
index=465 → 3475Hz, κλπ.

Στην συνέχεια, ξεκινώντας από ένα υψηλό delta, καλεί την συνάρτηση PeakDet.peakdet πολλές φορές, μειώνοντας κάθε φορά το delta, έως ότου τα peak που θα ληφθούν θα είναι τουλάχιστον πέντε. Αν το delta φτάσει στο 0 και τα peaks είναι λιγότερα από 5, δημιουργείται αποτέλεσμα με όσα peaks έχουν βρεθεί και συμπληρώνονται peaks στην μηδενική συχνότητα, με μηδενικό ύψος, ώστε να δοθούν πέντε (συνάρτηση Silence).

Τέλος, αφού επιλεγούν τα πέντε μεγαλύτερα (αν είναι παραπάνω από πέντε), τα peak και ο άξονας των συχνοτήτων δίνονται στην συνάρτηση getJsonString, η οποία δημιουργεί το κατάλληλο Json String (κείμενο), το οποίο έχει τις συχνότητες των peaks και τα ύψη τους.

5.2.6 MainActivity3()

Η παρούσα κλάση είναι πολύ παρόμοια της MainActivity2() και μιμείται αρκετά τον σκελετό της. Η βασική διαφορά είναι πως η MainActivity3() χρησιμοποιείται για την ηχογράφηση ενός μόνο τοπικού αρχείου .wav, με ενέργεια του χρήστη, το οποίο έπειτα στέλνεται στο Node-RED μέσω HTTP POST Request και εν τέλει, αποθηκεύεται σε ένα τοπικό αρχείο στον υπολογιστή μας το οποίο αντιπροσωπεύει την Database των Trained Sounds.

Ο κύριος λόγος ύπαρξής της είναι η ανάγκη βελτίωσης της αποτελεσματικότητας της αναγνώρισης των ήχων. Η οντολογία του Planner ουσιαστικά έχει αποθηκευμένους είκοσι ήχους σε κάθε κατηγορία οι οποίοι έχουν ευρεθεί στο Διαδίκτυο. Προφανώς οι ήχοι που μπορούν να λάβουν χώρα στο σπίτι μπορεί να εμφανίζουν κάποιες διαφοροποιήσεις με τους ήχους της οντολογίας με αποτέλεσμα η ταυτοποίησή τους να μην είναι ιδιαίτερα ακριβής. Για να λυθεί αυτό το πρόβλημα προστέθηκε η λειτουργία να μπορεί ο χρήστης να γράφει σε ένα αρχείο τα δεδομένα ήχων τα οποία ηχογραφεί στο χώρο του σπιτιού (πρακτικά «εκπαίδευση» (train) νέων ήχων της εφαρμογής) και στην συνέχεια να γίνεται σύγκριση και με αυτούς όταν γίνεται η ηχογράφηση πραγματικού χρόνου.

Στον παρακάτω πίνακα εμφανίζονται οι μέθοδοι οι οποίες εμφανίζονται μέσα στην κλάση αυτή:

Μέθοδοι	Περιγραφή
onCreate()	Η μέθοδος αυτή χρησιμοποιείται για την εκκίνηση κάποιας δραστηριότητας. Στην δική μας περίπτωση απλά θέτει το αρχικό γραφικό περιβάλλον του χρήστη και του εμφανίζει ένα αναπτυσσόμενο μενού (drop-down menu) που περιλαμβάνει τις κατηγορίες των διαθέσιμων ήχων καθώς και δύο κουμπιά εκ των οποίο το ένα σηματοδοτεί την έναρξη ηχογράφησης και το άλλο την λήξη της.
enableButton()	Για να υπάρχει κάποιο αποτέλεσμα κάνοντας κλικ στα κουμπιά που εμφανίζονται στην διεπαφή του χρήστη χρειάζεται να τα ενεργοποιήσουμε πρώτα. Η παρούσα μέθοδος πραγματοποιεί

	αναζήτηση στο σύστημα με βάσει το ID του κουμπιού και ελέγχει αν είναι ενεργοποιημένο.
enableButtons()	Αν στην αναζήτηση της προηγούμενης μεθόδου προκύψει ψευδές αποτέλεσμα τότε καλείται η παρούσα μέθοδος και ενεργοποιεί τα απαραίτητα κουμπιά για την αλληλεπίδραση με τον χρήστη.
addListenerOnButton()	Για να μπορέσει ο προγραμματιστής να επεξεργαστεί κάποιο γεγονός (event) αρχικά πρέπει να δηλώσει έναν ακροατή γεγονότων (event listener) που θα ακούει τέτοιου τύπου events, δηλαδή το πότε ο χρήστης πατάει τα αντίστοιχα κουμπιά.
setButtonHandlers()	Στην συνέχεια, υλοποιείται με την μέθοδο setButtonHandlers() ο χειριστής γεγονότων (event handler) ο οποίος «πιάνει» τα events που προκύπτουν από την μέθοδο addListenerOnButton() και εκτελεί τις ανάλογες ενέργειες για τα οποία είναι προορισμένα.
addListenerOnSpinnerItemSelection()	Η παρούσα μέθοδος είναι αντίστοιχος χειριστής γεγονότων με τον προηγούμενο που όμως που αφορά το drop-down menu.
startRecording()	Η μέθοδος αυτή αξιοποιεί την κλάση AudioRecord η οποία ειναι ενσωματωμένη στο Android Studio και όταν καλείται, το κινητό αρχίζει να ηχογραφεί.
writeAudioDataToFile()	Η writeAudioDataToFile() καλείται κατά την διάρκεια της startRecording και ουσιαστικά γράφει τα δεδομένα της τρέχουσας ηχογράφησης σε ένα αρχείο.
stopRecording()	Η μέθοδος stopRecording() δίνει εντολή στην συσκευή να σταματήσει να ηχογραφεί. Στην συνέχεια, καλούνται οι μέθοδοι copyWaveFile() και deleteTempFile().
copyWaveFile()	Η copyWaveFile() μέσω της μεθόδου getTempFileName() διαβάζει το αρχείο ήχου του οποίου ο σκοπός είναι να γίνει train, και το μετατρέπει σε αρχείο .wav το οποίο είναι αποθηκευμένο τοπικά στην συσκευή ηχογράφησης, μέσω της getFilename().
getFilename()	Επιστρέφεται το όνομα του αρχείου που είναι τοπικά αποθηκευμένο στην συσκευή και αποθηκεύονται εκεί όλες οι ηχογραφήσεις σε μορφή .wav.

getTempFilename()	Επιστρέφεται η τοποθεσία στην οποία έχει αποθηκευτεί η προσωρινή ηχογράφηση μέσα στο κινητό.
deleteTempFile()	Διαγράφεται το προσωρινό αρχείο το οποίο περιέχει μόνο τις υπό εξέλιξη ηχογραφήσεις.
WriteWaveFileHeader()	Η μέθοδος αυτή καλείται από την copyWaveFile() και ο ρόλος της είναι να προσθέσει την επικεφαλίδα (header) των .wav στα αρχεία ήχου που αποθηκεύονται στην συσκευή του χρήστη.

Συνελόντι ειπείν, μόλις κληθεί η παρούσα κλάση ο χρήστης καλείται να επιλέξει μέσω ενός αντικειμένου, μορφής Spinner, την κατηγορία που ανήκει ο ήχος που πρόκειται να ηχογραφήσει. Στην συνέχεια, καλείται η μέθοδος startRecording() και μετά η stopRecording() ώστε να τερματιστεί η ηχογράφηση. Το αρχείο .wav που προκύπτει από την ηχογράφηση αποθηκεύεται τοπικά στην μνήμη του κινητού. Να σημειωθεί πως για να αποφευχθεί η μεγάλη δέσμευση της μνήμης του κινητού έχει επιλεχθεί η συνεχής αντικατάσταση κάθε παλιού αρχείου trainsound.wav με το νέο που έχει ηχογραφηθεί, ώστε να υπάρχει μόνο ένα αρχείο trainsound.wav αποθηκευμένο στην συσκευή του κινητού.

Γενικά όταν το τρέχον αρχείο .wav αποθηκευτεί, το επόμενο βήμα είναι η κλήση της κλάσης WavTaskTrain() έχοντας σαν παράμετρο την κατηγορία του ήχου που έχει επιλεχθεί μέσω ενός αντικειμένου Spinner (ουσιαστικά ένα drop-down menu το οποίο περιέχει όλες τις διαθέσιμες κατηγορίες ήχου). (Σχήματα 6.8-6.9)

5.2.7 WavTaskTrain()

Η WavTaskTrain() εκτελεί ακριβώς την ίδια λειτουργία με την WavTask() με την διαφορά ότι η HttpURLConnection δημιουργεί νέα σύνδεση με άλλο URL (http://localhost:1880/PostWavTrain) έτσι ώστε να σταλθεί το απαραίτητο JSON αντικείμενό μας στο αντίστοιχο flow το οποίο προορίζεται για την επεξεργασία και αποθήκευση των Train Sounds.

Το JSON αρχείο που στέλνεται στο Node-RED περιέχει την κατηγορία του ηχογραφημένου ήχου καθώς και τις πέντε κυρίαρχες συχνότητές του:

{ "TrainSound": "PhoneRing", "SoundResults":
"{%payload%:{%message_type%:%1%,%problem_attributes%:%freq1#freq2#freq3#freq4#freq5%,%problem_values%:%4.0374755859375#1.682281494140625#3.36456298828125#7.40203857421875#11.43951416015625%,%solution_attributes%:%hasSoundof%,%forSharing%:%true%,%weights%:%0.5#0.4320651487720298#0.3045866042085763#0.2482471312445156#0.24800343844664013%,%threshold%:%0.6%}}" }

5.2.8 CustomOnItemSelectedListener()

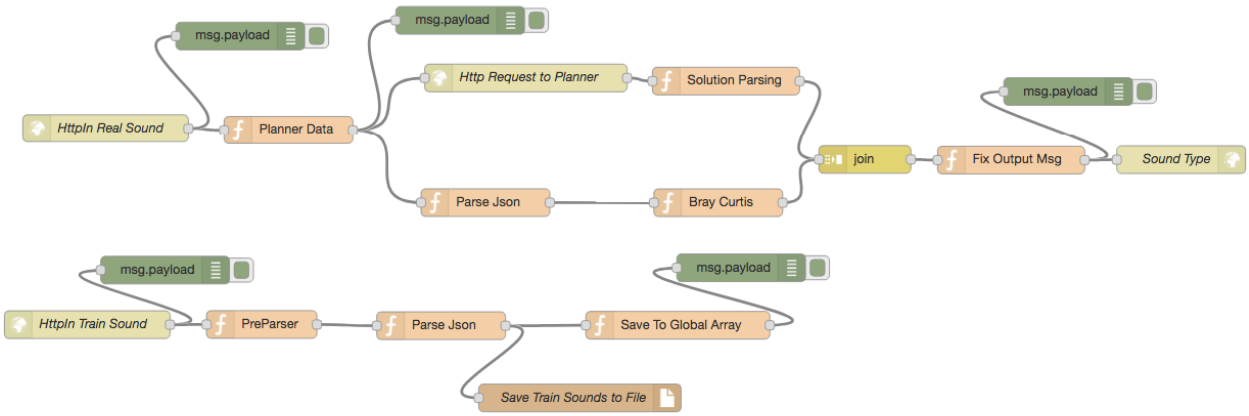
Η παρούσα κλάση χρησιμεύει στην εμφάνιση κειμένου στην οθόνη του χρήστη το οποίο επιβεβαιώνει την επιλογή μιας από τις πέντε κατηγορίες ήχου που εμφανίζονται στο drop-down menu.

5.2.9 AppLog()

Η AppLog() κλάση επιτρέπει στον προγραμματιστή να εμφανίσει μηνύματα (strings) με την ένδειξη “Audio Recorder: Message” (όπου Message το αντίστοιχο μήνυμα που έχουμε ορίσει να εμφανίζεται κάθε φορά) στο Android Studio κατά την διάρκεια της χρήσης της εφαρμογής ώστε να υπάρχει καλύτερη εικόνα σχετικά με την ροή των διεργασιών, όπως το μέγεθος των αρχείων .wav που αναλύονται.

5.3 Διάγραμμα Ροής (Flow) Επεξεργασίας Ήχων στο Node-RED

Στο παρακάτω διάγραμμα παρατίθεται το κομμάτι επεξεργασίας ήχου που λαμβάνει χώρα στο λογισμικό Node-RED. Κάθε κόμβος (node) έχει αναλάβει και μια λειτουργία η οποία συμβάλλει στην επεξεργασία καθώς και την ανάλυση και ταυτοποίηση των ήχων .wav που στέλνονται από συσκευή με λογισμικό Android. Η μεταξύ τους επικοινωνία επιτυγχάνεται με το πέρασμα JSON μηνυμάτων.



Σχήμα 5.7: Το διάγραμμα ροής της εφαρμογής μας στο Node-RED

Στο παραπάνω flow έχουμε ως input αρχεία .wav, τα οποία έχουν σταλθεί από μια συσκευή λογισμικού Android, και ως output το είδος του ήχου που αναγνωρίστηκε στο .wav.

Το flow χωρίζεται σε δύο υπο-διαγράμματα ροής. Το πρώτο διάγραμμα, το άνω, αφορά την επεξεργασία και αναγνώριση ήχων που εισέρχονται σε αληθινό χρόνο και το δεύτερο διάγραμμα, το κάτω, αφορά την επεξεργασία και αποθήκευση ήχων που ηχογραφούνται στην κατοικία του χρήστη.

Παρακάτω θα αναλυθεί η λειτουργία κάθε κόμβου με την σειρά και θα επεξηγηθούν τα inputs και outputs τους, χρησιμοποιώντας παραδείγματα από πραγματικές τιμές που εμφανίστηκαν κατά την διάρκεια χρήσης της εφαρμογής.

5.3.1 Flow Επεξεργασίας .wav ηχογραφημένων σε πραγματικό χρόνο

Κόμβος HttpIn Real Sound

Ο κόμβος HttpIn Real Sound ουσιαστικά αποτελεί έναν κόμβο εισόδου ο οποίος δέχεται αιτήματα HTTP επιτρέποντας την δημιουργία απλών διαδικτυακών υπηρεσιών. Στην παρούσα περίπτωση, μιας και τρέχουμε το Node-RED τοπικά στον υπολογιστή μας, επιλέγουμε να στέλνει η συσκευή του χρήστη HTTP μηνύματα με την μέθοδο POST στην διεύθυνση http://localhost:1880/PostWav.

Σχήμα 5.8: Κόμβος HttpIn Real Sound στο Node-RED

Ο λόγος που επιλέξαμε την μέθοδο POST αντί για GET είναι επειδή με την μέθοδο POST ο client δεν στέλνει μόνο επικεφαλίδα, αλλά και σώμα (body) μηνύματος. Αυτό μπορεί να είναι οσονδήποτε μεγάλο. Έτσι, μεταξύ των άλλων, το μήνυμα POST επιλύει περιορισμούς μεγίστου μήκους πού ενίοτε τίθενται για το μέγεθος μίας επικεφαλίδας του HTTP και που ανακύπτουν στην περίπτωση του GET.

Το σώμα του POST εκτείνεται σύμφωνα με το μήκος του, που δηλώνεται με το υποχρεωτικό Content-Length: ... πεδίο της επικεφαλίδας. Η πρώτη γραμμή της επικεφαλίδας περιέχει υποχρεωτικά το πεδίο Content-Type: type, όπου type έχει οριστεί στην εφαρμογή μας ως application/x-www-form-urlencoded. Επομένως, το μήνυμα POST είναι και το ενδεικνυόμενο για την αποστολή δεδομένων από τον client στον server.

Έτσι, «ανεβάζουμε» ένα XML το οποίο περιέχει τις 5 κύριες συχνότητες των .wav αρχείων, στον web server και το λαμβάνει το Node-RED το οποίο και το αποθηκεύει στην μεταβλητή msg.req και έτσι περνιέται το περιεχόμενο του κόμβου στον επόμενο ο οποίος ειναι ο Planner Data.

Στην προηγούμενη υποενότητα αναφέραμε πως το μήνυμα που στέλνεται από την Android συσκευή μας προς το Node-RED ειναι το παρακάτω. Το ίδιο μήνυμα ακριβώς είναι και αυτό που λαμβάνεται στον κόμβο HttpIn Real Sound:

{ "fName":
"{%payload%:{%message_type%:%1%,%problem_attributes%:%freq1#freq2#freq3#freq4#freq5%,%problem_values%:%428.3088684082031#1.3458251953125#2.355194091796875#0.5046844482421875#160.994389892578%,%solution_attributes%:%hasSoundof%,%forSharing%:%true%,%weights%:%0.5#0.06066846304312529#0.056644159149927244#0.04854882221897655#0.04537751630253077%,%threshold%:%0.6%}}" }

Κόμβος Planner Data

Ο κόμβος Planner Data ανήκει στην κατηγορία των κόμβων στους οποίους μπορείς να γράψεις συναρτήσεις σε κώδικα JavaScript. Λαμβάνει σαν είσοδο ένα αντικείμενο (object) το οποίο ονομάζεται msg. Στην συνέχεια, κατά σύμβαση θα έχει μία ιδιότητα (property) η οποία θα ονομάζεται msg.payload και θα περιέχει το σώμα του μηνύματος που εισήλθε σε αυτόν.

Ο σκοπός του node "Planner Data" είναι να προσαρμόσει το μήνυμά μας σε μορφή κατάλληλη ώστε να μπορέσει να το επεξεργαστεί αργότερα ο Planner και να πραγματοποιηθεί επιτυχώς η ταυτοποίηση ήχου. Παρακάτω εμφανίζεται ο κώδικας JavaScript ο οποίος έχει γραφτεί εσωτερικά του node:

```
var str = msg.payload.fName;
str=str.replace(new RegExp('%', 'g'), '"');
var result = str;
msg.payload = result;
return msg;
```

Στον παραπάνω κώδικα ουσιαστικά λαμβάνουμε το μήνυμα που στάλθηκε από τον κόμβο HttpIn Real Sound και αντικαθιστούμε τα % με " μέσα σε αυτό. Στην συνέχεια, φορτώνουμε το νέο string στο msg το οποίο και περνάμε στο επόμενο node. Το output, λοιπόν, θα είναι της μορφής:

{"payload":{"message_type":"1","problem_attributes":"freq1#freq2#freq3#freq4#freq5","problem_values":"428.3088684082031#1.3458251953125#2.355194091796875#0.5046844482421875#160.9943389892578","solution_attributes":"hasSoundof","forSharing":"true","weights":"0.5#0.06066846304312529#0.056644159149927244#0.04854882221897655#0.04537751630253077","threshold":"0.6"}}

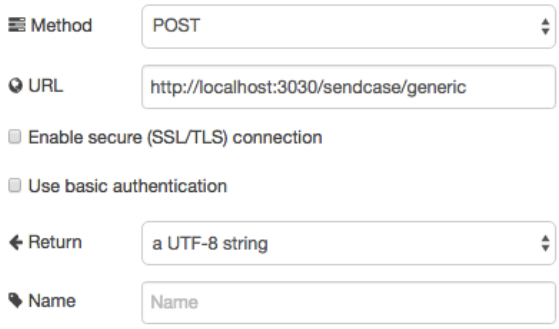
Διακλάδωση

Πλέον έχοντας σε κατάλληλη μορφή, στην μεταβλητή msg.payload, τις πέντε κύριες συχνότητες του ήχου ηχογραφημένου σε πραγματικό χρόνο, προχωράμε σε δύο ειδών συγκρίσεις. Παρατηρούμε ότι το flow μας διακλαδώνεται μετά τον κόμβο ονόματι Planner Data και η κάθε διακλάδωση οδηγεί σε μια διαφορετική κατηγορία σύγκρισης. Στην πάνω διακλάδωση, ο real-time ήχος συγκρίνεται με ήχους που βρίσκονται προ-αποθηκευμένοι στην Database του Planner ενώ στην κάτω διακλάδωση, ο real-time ήχος συγκρίνεται με τους ήχους που έχει κάνει ο χρήστης train στην συσκευή και έχουν αποθηκευτεί σε αρχείο ειδικά προορισμένο για αυτόν τον σκοπό.

Σύγκριση με Ήχους του Planner

Προχωράμε στην ανάλυση των δύο κόμβων των οποίων ο σκοπός είναι η σύγκριση του real-time ήχου με τους ήχους που βρίσκονται στην Database του Planner.

Κόμβος Http Request to Planner



Σχήμα 5.9: Κόμβος Http Request to Planner στο Node-RED

Ο παρών κόμβος παρέχει την υπηρεσία υποβολής αιτήσεων HTTP (http request). Έχουμε θέσει σαν μέθοδο την POST για τους ίδιους λόγους που την επιλέξαμε στον κόμβο "HttpIn Real Sound" αλλά αυτή τη φορά έχουμε αλλάξει το URL σε http://localhost:3030/sendcase/generic. Με αυτόν τον τρόπο επιτυγχάνουμε να στείλουμε το JSON αντικείμενο, που ήρθε σαν έξοδος από το προηγούμενο node, στον Planner. Στην συνέχεια, λαμβάνεται ως απάντηση από τον Planner το είδος του ήχου και περνιέται στον επόμενο κόμβο.

{"hasMessage":"It also handles 2 variables","answer_values":"DoorKnock","similarity":"0.6869644809901648","hasURI":"","VE":"N\/A"}

Κόμβος Solution Parsing

Ο Planner έχει πλέον επιστρέψει την κατηγορία ήχου στην οποία ανήκει η ηχογράφησή μας αλλά χρειάζεται κατάλληλη μορφοποίηση ώστε να μπορέσει ο Node-RED να στείλει στην συσκευή μας την ανάλογη απάντηση.

Με τον κόμβο Solution Parsing η απάντηση του Planner χωρίζεται σε έναν πίνακα ονόματι result και φορτώνεται στο αντικείμενο msg.payload το οποίο περνιέται στον επόμενο κόμβο:

```
var str = msg.payload;
var result = str.split(' " ');
msg.payload = result[7];
return msg;
```

Το output του κόμβου Solution Parsing είναι απλά η κατηγορία του ήχου:

DoorKnock

Σύγκριση με Ήχους που έχουν γίνει Train

Οι δύο προηγούμενοι κόμβοι που αναλύθηκαν αφορούσαν την σύγκριση του real-time ήχου με τους ήχους που βρίσκονται στο Database του Planner. Τώρα προχωράμε στην ανάλυση των κόμβων που αφορούν την σύγκριση του real-time ήχου με την Database που έχει χτίσει ο χρήστης προσθέτοντας ήχους που παράγονται μέσα στην κατοικία του.

Κόμβος Parser Json

Ο κύριος σκοπός του κόμβου Parser Json είναι να απομονώσει τις 5 κύριες συχνότητες μέσα στο XML σώμα που ήλθε μέσω του HTTP Request. Για να επιτευχθεί η απομόνωση αυτή, ζητείται να απομονωθεί το string το οποίο ακολουθεί μετά από την λέξη problem_values από το πρώτο doublequotes έως το δεύτερο doublequotes:

```
var str = msg.payload;
str = str.substring(str.indexOf("problem_values"));
str = str.substring(str.indexOf(' " ')+1);
str = str.substring(str.indexOf(' " ')+1);
var freqs = str.substring(0,str.indexOf(' " '));
freqs = freqs.replace(new RegExp('#','g'),' ');
msg.payload = freqs;
return msg;
```

Το output του κόμβου Parser Json είναι ένα διάνυσμα το οποίο περιέχει τις 5 κύριες συχνότητες διαχωρισμένες με ένα κενό ανάμεσά τους:

428.3088684082031 1.3458251953125 2.355194091796875 0.5046844482421875 160.9943389892578

Σύγκριση με Ήχους που έχουν γίνει Train - Κόμβος Bray Curtis

Σε αυτόν τον κόμβο πραγματοποιείται η σύγκριση του real-time ήχου με τους τοπικούς ήχους που έγιναν train στην εφαρμογή μας. Για να επιτευχθεί η σύγκριση γίνεται χρήση ενός λογισμικού

πακέτου (software package) το οποίο υπολογίζει αυτόματα την Bray – Curtis απόσταση [30]. Αρχικά, παίρνει το τρέχον διάνυσμα που περιέχει τις 5 κύριες συχνότητες, διασχίζει γραμμή – γραμμή μία γενική μεταβλητή σε μορφή πίνακα, με τα αποθηκευμένα διανύσματα, και σε κάθε γραμμή υπολογίζει την απόσταση Bray Curtis. Εν τέλει, δίνεται σαν έξοδος ο ήχος του οποίου το διάνυσμα έφερε σαν έξοδο την μικρότερη απόσταση Bray Curtis η οποία θεωρείται και η βέλτιστη / καλύτερη (μεγαλύτερο similarity).

```
var BrayCurtis = global.get('braycurtis');

var tokens = msg.payload.split(' ');
var sound = [];
sound.push(parseFloat(tokens[0]));
sound.push(parseFloat(tokens[1]));
sound.push(parseFloat(tokens[2]));
sound.push(parseFloat(tokens[3]));
sound.push(parseFloat(tokens[4]));

var vectors = [];
var names = [];
vectors.push(sound);
var array = global.get('TrainSounds');
for (var i=0; i<array.length; i++){
   vectors.push(array[i].freqs);
   names[array[i].freqs.join(' ')] = array[i].sound;
}
var nn = BrayCurtis(vectors);
var neighbours = nn.knn(sound,2);
var res = names[neighbours[1].vector.join(' ')];
msg.payload = res;
return msg;
```

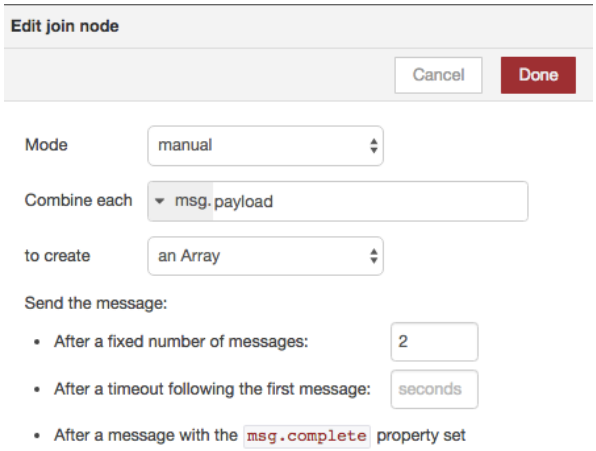
Το output του κόμβου Bray - Curtis είναι απλά η κατηγορία του ήχου:

DoorKnock

Κόμβος join

Αφότου στους δύο προηγούμενους κόμβους (Solution Parsing και Bray – Curtis) έχει γίνει επιλογή της αντίστοιχης κατηγορίας ήχου στην οποία ανήκει το τρέχον ηχογραφημένο αρχείο .wav, είναι απαραίτητη η αποστολή του αντίστοιχου μηνύματος στην συσκευή μας. Κάτι τέτοιο καθίσταται δυνατό με την καταχώρηση του αντίστοιχο μηνύματος σε μορφή string στο msg.payload.

Μιας και υπήρξαν δύο συγκρίσεις του real-time ήχου, μπορεί να υπάρχουν ενδεχομένως δύο πιθανά σενάρια σχετικά με το είδος του ήχου. Για να μπορέσουμε να εμφανίσουμε κατάλληλο μήνυμα στο χρήστη το οποίο θα καλύπτει όλα τα ενδεχόμενα, είναι απαραίτητο να ενώσουμε τα δύο outputs των προαναφερθέντων διακλαδώσεων:



Σχήμα 5.10: Κόμβος join στο Node-RED

Παράδειγμα output του κόμβου join:

```
[ "PhoneRing", "DoorKnock" ]
```

Κόμβος Fix Output Msg

Ο παρών κόμβος ελέγχει αν τα δύο outputs που αφορούν το είδος του ήχου είναι ίδια ή διαφορετικά και μορφοποιεί αναλόγως το μήνυμα εξόδου.

```
var res1 = msg.payload[0];
var res2 = msg.payload[1];
var res = "The sound is ";
if (res1.localeCompare(res2)==0)
    res = res + res1;
else
    res = res + res1 + " or " + res2;
res = res + ".";
msg.payload = res;
return msg;
```

Παρακάτω δίνονται δύο ειδών αποτελέσματα. Στο πρώτο είναι η περίπτωση στην οποία οι δύο συγκρίσεις (Database Planner, Database Trained Sounds) έχουν βγάλει διαφορετικό αποτέλεσμα.

Στο δεύτερο παράδειγμα εμφανίζεται η περίπτωση στην οποία το είδος του ήχου θα είναι ίδιο και στις δύο περιπτώσεις.

The sound is PhoneRing or DoorKnock.

The sound is DoorKnock.

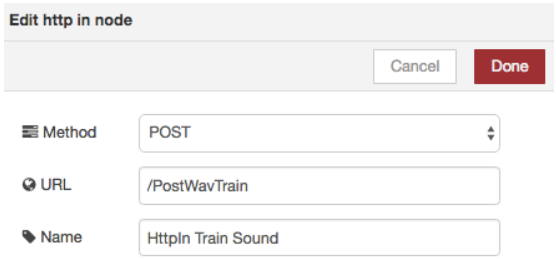
Κόμβος Sound Type

Εν τέλει, αφότου έχει γίνει η ανάλυση του του .wav αρχείου, έχει γίνει ταυτοποίησή του και το msg.payload έχει φορτωθεί με το αντίστοιχο μήνυμα ειδοποίησης προς τον χρήστη, στέλνεται η απάντηση πίσω στο αίτημα HTTP που έχει ληφθεί από το κόμβο HttpIn Real Sound. Το HTTP Response που στέλνεται πίσω στον Client (η εφαρμογή μας Android), θα εμφανίσει ένα μήνυμα και θα ανάψει τα αντίστοιχα εικονίδια (Σχήματα 6.3-6.7)

5.3.2 Flow Επεξεργασίας και αποθήκευσης .wav για Train

Κόμβος HttpIn Train Sound

Ο κόμβος HttpIn Train Sound αποτελεί έναν κόμβο εισόδου ο οποίος δέχεται αιτήματα HTTP επιτρέποντας την δημιουργία απλών διαδικτυακών υπηρεσιών, όπως ο HttpIn Real Sound κόμβος. Η διαφορά είναι πως επιλέγουμε να στέλνει η συσκευή του χρήστη HTTP μηνύματα με την μέθοδο POST σε άλλη διεύθυνση, η οποία είναι η http://localhost:1880/PostWavTrain.



Σχήμα 5.11: Κόμβος HttpIn Train Sound στο Node-RED

Το XML αρχείο που λαμβάνει ο κόμβος HttpIn Train Sound έχει ενσωματωθεί καταλλήλως στην MainActivity3() έτσι ώστε να περιέχει την κατηγορία στην οποία ανήκει ο ήχος ο οποίος γίνεται train καθώς και τις 5 κύριες συχνότητές του:

{ "TrainSound": "PhoneRing", "SoundResults":
"{%payload%:{%message_type%:%1%,%problem_attributes%:%freq1#freq2#freq3#freq4#freq5%,%problem_values%:%4.0374755859375#1.682281494140625#3.36456298828125#7.40203857421875#11.43951416015625%,%solution_attributes%:%hasSoundof%,%forSharing%:%true%,%weights%:%0.5#0.4320651487720298#0.3045866042085763#0.2482471312445156#0.24800343844664013%,%threshold%:%0.6%}}" }

Κόμβος PreParser

Ο σκοπός του κόμβου PreParser είναι να μετατρέψει το XML αρχείο που δέχεται από το HttpIn Train Sound σε JSON και στην συνέχεια να φέρει το JSON σε κατάλληλη μορφή, αντικαθιστώντας τα % με “, τέτοια ώστε να γίνει επεξεργάσιμο στον επόμενο κόμβο και να απομονωθούν τα επιθυμητά τμήματα (strings) του.

```
var str = JSON.stringify(msg.payload);
str=str.replace(new RegExp('%', 'g'), '"');
var result = str;
msg.payload = result;
return msg;
```

{"TrainSound":"PhoneRing","SoundResults":"{"payload":{"message_type":"1","problem_attributes":"freq1#freq2#freq3#freq4#freq5","problem_values":"4.0374755859375#1.682281494140625#3.36456298828125#7.40203857421875#11.43951416015625","solution_attributes":"hasSoundof","forSharing":"true","weights":"0.5#0.4320651487720298#0.3045866042085763#0.2482471312445156#0.24800343844664013","threshold":"0.6"}}"}

Κόμβος Parse Json

Ο κόμβος αυτός λαμβάνει από το HttpIn Train Sound το είδος του ήχου κι από το XML τις συχνότητες. Δίνουμε εντολή ώστε όταν διαβάσει το XML και συναντήσει την λέξη Trainsound, στο δευτερο doublequote και μεχρι το επόμενο, απομονώνει και αποθηκεύει το string που αφορά το είδος του ήχου. Στην συνέχεια επαναλαμβάνει την ίδια διαδικασία αποθηκεύοντας και απομονώντας σε ένα διάνυσα τις 5 συχνότητας αποθηκεύοντας το string που ακολουθεί το doublequote μετά την λέξη problem_values. Έν τέλει, στο επόμενο node δίνεται σαν input ένα string που περιέχει το όνομα ήχου και τις 5 κύριες συχνότητες διαχωρισμένες με κενό:

```
var str = msg.payload;//.replace(new RegExp('%', 'g'), ' " ');

str = str.substring(str.indexOf("TrainSound"));
str = str.substring(str.indexOf(' " ')+1);
str = str.substring(str.indexOf(' " ')+1);
```

```
var sound = str.substring(0,str.indexOf(' " '));
sound = sound.replace(new RegExp(' ','g'),'-');

str = str.substring(str.indexOf("problem_values"));
str = str.substring(str.indexOf(' " ')+1);
str = str.substring(str.indexOf(' " ')+1);
var freqs = str.substring(0,str.indexOf(' " '));
freqs = freqs.replace(new RegExp('#','g'),' ');

msg.payload = sound+' '+freqs;
return msg;
```

Το output του κόμβου Parse Json είναι ένα διάνυσμα το οποίο περιέχει την κατηγορία στην οποία ανήκει ο ήχος καθώς και τις 5 κύριες συχνότητές τους, διαχωρισμένα με κενά:

```
PhoneRing 4.0374755859375 1.682281494140625 3.36456298828125 7.40203857421875
11.43951416015625
```

Κόμβος Save to Global Array

Στον κόμβο αυτό αρχικά δημιουργείται ένας πίνακας (array) ο οποίος είναι global, δηλαδή οι τιμές του είναι διαθέσιμες σε όλους τους κόμβους που δημιουργούνται στο παρόν flow του Node-RED. Στη συνέχεια δημιουργείται ένα αντικείμενο entry. Το αντικείμενο entry έχει την ιδιότητα sound η οποία είναι το είδος / όνομα του ήχου και την ιδιότητα freqs η οποία ουσιαστικά είναι ένας πίνακας με τις 5 κύριες συχνότητες (τα στοιχεία αυτά τα έχει λάβει απο το προηγούμενο node – Parse Json). Εν τέλει, το entry αυτό το καταχωρεί στον array και τον array, με τη σειρά του, τον αποθηκεύει στην global μεταβλητή TrainSounds. Ο σκοπός της global μεταβλητής TrainSounds είναι να κληθεί μετά από τον κόμβο Bray Curtis, τον οποίο αναλύσαμε προηγουμένως, ώστε να γίνει διάσχιση του πίνακα array που εμπεριέχει τις 5 κύριες συχνότητες των Train Sounds και να υπολογίσει το βέλτιστο Bray Curtis distance σε σχέση με τον real-time ήχο.

Ο κώδικας του κόμβου είναι ο εξής:

```
var array = global.get('TrainSounds');
if (!array) array = [];

//Empty array
//array = [];

var tokens = msg.payload.split(' ');
var entry = [];
```

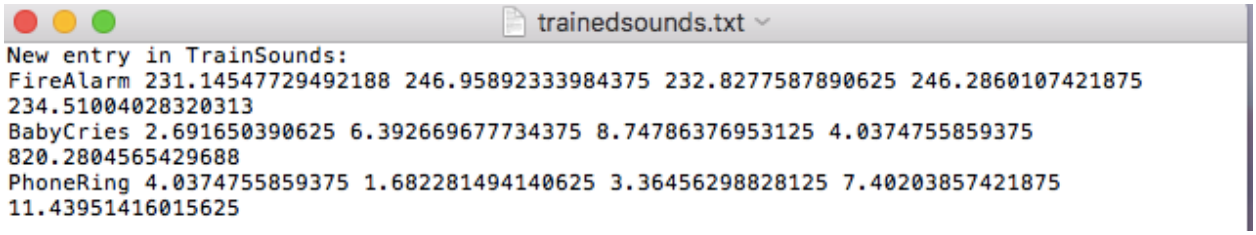
```
entry["sound"] = tokens[0].replace(new RegExp('-','g'),' ');
var freqs = [];
freqs.push(parseFloat(tokens[1]));
freqs.push(parseFloat(tokens[2]));
freqs.push(parseFloat(tokens[3]));
freqs.push(parseFloat(tokens[4]));
freqs.push(parseFloat(tokens[5]));
entry["freqs"] = freqs;

array.push(entry);
global.set('TrainSounds',array);

//----Debug-----
msg.payload = "New entry in TrainSounds: \n" +entry.sound + " - Array : " +
JSON.stringify(array);
return msg;
```

Το output του κόμβου αυτού ουσιαστικά είναι η αποθήκευση του βέλτιστου ήχου στην γενική μεταβλητή Train Sounds η οποία είναι σε μορφή array.

Κόμβος Save Train Sounds to File

Αφότου έχουμε απομονώσει σε JSON το είδος του ήχου τον οποίο επιθυμεί ο χρήστης να κάνει train καθώς και τις 5 κύριες συχνότητες που εμφανίζονται, αποθηκεύουμε όλες τους νέους ήχους – καταχωρήσεις σε ένα τοπικό αρχείο στον υπολογιστή μας ονόματι trainedsounds.txt. Ουσιαστικά, στα πλαίσια της παρούσας διπλωματικής, το τοπικό αρχείο αντιπροσωπεύει την Database των Trained Sounds.

Παρακάτω εμφαίνεται ένα στιγμιότυπο από το αρχείο trainedsounds.txt με μερικές καταχωρήσεις:



Σχήμα 5.12: Database των Trained Sounds

6 Ανάπτυξη Πλευράς Χρήστη (Front-End Development)

6.1 Βασικές αρχές κατασκευής Περιβάλλοντος Εργασίας

Πριν το σχεδιασμό του περιβάλλοντος εργασίας της εφαρμογής ήταν απαραίτητη η έρευνα και συλλογή πληροφοριών σχετικά με το σχεδιασμό του UI ώστε να είναι φιλικό προς το target group στο οποίο στοχεύει. Προγραμματιστές της Microsoft έχουν καθορίσει κάποιες πολύ καλές βασικές αρχές για το σχεδιασμό της διάταξης του Περιβάλλοντος Εργασίας [31]. Παρακάτω παρουσιάζονται αυτές αρχές:

Χρώμα: Ο προγραμματιστής θα πρέπει να εξετάσει το χρώμα το οποίο θα απαρτίζει το UI προσεχτικά γιατί τα σωστά χρώματα συλλαμβάνουν επιτυχώς την προσοχή του χρήστη. Επίσης, πολλοί χρήστες αντιλαμβάνονται τα χρώματα με διαφορετικό τρόπο οπότε δεν είναι συνετό να βασιστεί η μετάδοση πληροφορίας σε αυτό. Τέλος, παρασάγγας σημαντικό ρόλο παίζει και η αντίθεση των χρωμάτων του παρασκηνίου (foreground) με το προσκήνιο (backround).

Τυπογραφία: Ένα επίσης πολύ σημαντικό χαρακτηριστικό που ένας προγραμματιστής πρέπει να λάβει υπόψη κατά το σχεδιασμό του UI. Οι προγραμματιστές της Microsoft συνιστούν «Το μέγεθος της γραμματοσειράς, το βάρος της γραμματοσειράς και οι αποστάσεις μεταξύ των γραμμάτων, λέξεων και παραγράφων είναι επίσης πολύ σημαντικοί παράγοντες. Πρέπει να αποφεύγονται γραμματοσειρές των οποίων το μέγεθος είναι πολύ μεγάλο ή πολύ μικρό και πάντα πρέπει να λαμβάνεται υπόψη το μέγεθος της οθόνης του χρήστη ώστε να μην αναγκάζεται να μετακινηθεί προς τα κάτω (scroll).»

Ισορροπία και συμμετρία: Η παρούσα κατηγορία βασίζεται στην διανομή του οπτικού βάρους και στην εμφάνιση συμμετρίας/ασυμμετρίας στην επιφάνεια διεπαφής. Η συμμετρία γενικά

προτιμάται όταν οι χρήστες είναι πιο παραδοσιακοί και επιθυμούν να αλληλοεπιδρούν με την εφαρμογή με σταθερότητα. Η ασυμμετρία καθίσταται κατάλληλη όταν οι χρήστες αποτελούν ένα σύγχρονο ακροατήριο ή έχουμε μια εφαρμογή που αποσκοπεί σε ψυχαγωγία επειδή επικρατεί μια πιο «ανεπίσημη» ισορροπία.

Συνέπεια: Εδώ συμπεριλαμβάνονται η διάταξη της σελίδας, το χρώμα καθώς και η τυπογραφία. Όλα αυτά θα πρέπει στο UI να συνεργάζονται και να "δένουν" μεταξύ τους εναρμονικά ώστε να είναι ξεκάθαρο πως θα πρέπει να αλληλοεπιδράσουν οι χρήστες με την διεργασία επιφάνειας της εφαρμογής.

Απλότητα: Είναι η απλή και λογική διάταξη του UI που επιτρέπει στους χρήστες να εκτελούν σημαντικά διεργασίες μέσα στην εφαρμογή. Αυτό επιτυγχάνεται με τον περιορισμό των κινούμενων εικόνων, των ειδικών εφέ, πολλαπλών χρωμάτων καθώς και άλλων διαβαθμίσεων που τροφοδοτούν τον χρήστη με περιττή πληροφορία.

6.2 Απαιτήσεις σχεδίασης της εφαρμογής μας

Το target group της εφαρμογής μας είναι άνθρωποι με ιδιαίτερες ανάγκες που αλληλεπιδρούν διαφορετικά με τον κόσμο όσον αφορά τους ήχους. Επομένως, ήταν απαραίτητο στην εφαρμογή μας να προσαρμόσουμε την αλληλοεπίδρασή της με το παρόν target group.

Το interface της αρχικής οθόνης είναι απλό και σαφές. Ο χρήστης καλείται να επιλέξει μεταξύ δύο επιλογών. Ανάλογα την επιλογή κατευθύνεται σε δεύτερη οθόνη η οποία εκτελεί την λειτουργία για την οποία προορίζεται.

Στο σχεδιασμό του interface λήφθηκαν υπόψη όλες οι βασικές αρχές σχεδιασμό που αναφέρθηκαν στην προηγούμενη ενότητα.

Χρώμα: Η εφαρμογή μας απαρτίζεται κυρίως από μπλε,μαύρο και άσπρο. Το μπλε είναι ένα χρώμα που εμπνέει επαγγελματισμό και εμπιστοσύνη, κάτι που αποδεικνύεται από το γεγονός πως κολοσσοί – διαδικτυακές επιχειρήσεις το χρησιμοποιούνε εκτενώς στις εφαρμογές τους (Facebook, LinkedIn, Flickr κτλ). Επίσης, το άσπρο και το μπλέ έχουν ευχάριστη αντίθεση μεταξύ τους, μιας

και το ένα πρόκειται για χρώμα ανοιχτής απόχρωσης και το άλλο σκοτεινής απόχρωσης, οπότε ο χρήστης μπορεί εύκολα να ξεχωρίσει την όποια γραμματοσειρά. Τέλος, υπάρχει μια κατηγορία ανθρώπων η οποία εμφανίζει αχρωματοψία και το μπλε βοηθάει σε μεγάλο βαθμό στην διευκόλυνσή τους όσον αφορά την ανάγνωση της οθόνης.

Τυπογραφία: Στην εφαρμογή μας χρησιμοποιούμε μια απλή γραμματοσειρά μεγάλου μεγέθους και άσπρου χρώματος, έτσι ώστε να είναι εύκολο να διαβαστεί το κείμενο από τον χρήστη.

Ισορροπία και συμμετρία: Το interface της εφαρμογής ακολουθεί τον κανόνα της συμμετρίας πιστά. Στις δύο από τις τρεις οθόνες όλα τα αντικείμενα σχεδιασμού ειναι διατεταγμένα στο κέντρο τους, το ένα κάτω από το άλλο. Στην τρίτη οθόνη, τα τέσσερα κύρια εικονίδια είναι οργανωμένα συμμετρικά μεταξύ τους.

Συνέπεια: Τα χρώματα, η γραμματοσειρά και η συμμετρία δένουν αρμονικά μεταξύ τους με αποτέλεσμα να ικανοποιείται ο κανόνας σχεδιασμού της συνέπειας.

Απλότητα: Η εφαρμογή μας αποτελείται απο λίγα κουμπιά και περιέχει ελάχιστο κείμενο. Η πληροφορία που δίνεται στον χρήστη είναι τόση όση είναι απαραίτητη ώστε να αντιληφθεί ποιος είναι ο χειρισμός της. Αποφεύγεται ο κατακλυσμός του χρήστη με πολλή πληροφορία καθώς και η περίπλοκη περιήγηση.

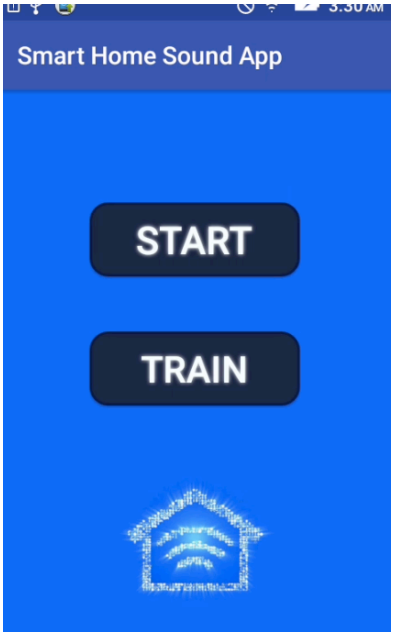
6.3 To GUI

Το GUI (Graphical User Interface) της εφαρμογής μας επικεντρώνεται στην απλότητα και απαρτίζεται από τρεις οθόνες διεπαφής. Την αρχική οθόνη, την οθόνη που αφορά την ειδοποίηση του χρήστη σχετικά με το είδος του ήχου που αναπαράγεται στο κοντινό περιβάλλον και την οθόνη που αφορά την εκπαίδευση της εφαρμογής των νέων τοπικών ήχων που ηχογραφεί ο χρήστης. Η περιήγηση είναι σαφής μιας και έχουμε μόνο τρεις οθόνες οι οποίες εναλάσσονται μεταξύ τους κατα τη διάρκεια περιήγησης του χρήστη.

Οι βασικές λειτουργίες είναι δύο. Στην αρχική οθόνη ο χρήστης έχει την επιλογή είτε να πατήσει “START” είτε “TRAIN”. Αν επιλέξει “START”, τότε ουσιαστικά επιλέγει να αρχίσει η εφαρμογή να ηχογραφεί συνεχόμενα το περιβάλλον και να τον ειδοποιεί αναλόγως. Αν επιλέξει ο χρήστης

“TRAIN”, τότε οδηγείται σε μια νέα οθόνη στην οποία έχει την δυνατότητα να ηχογραφήσει ένα ήχο τοπικά και να τον προσθέσει στο Trained Sounds Database, την οποία αναφέραμε στην ενότητα 5.3.2.

Παρακάτω παρατίθεται η αρχική οθόνη της εφαρμογής και παρατηρούμε ότι αντικατοπτρίζει όσα αναφέρθηκαν προηγουμένως:



Σχήμα 6.1: Η αρχική οθόνη της εφαρμογής μας

6.3.1 Κουμπί Start

Αν πατηθεί το κουμπί “START” τότε μέσω της onCreate() καλείται η κλάση MainActivity2() και ο χρήστης κατευθύνεται σε δεύτερη οθόνη η οποία συμπεριλαμβάνει τέσσερα εικονίδια εκ των οποίων το καθένα αντιπροσωπεύει μια από τις πέντε κατηγορίες ήχων που υποστηρίζονται ως προς αναγνώριση από την εφαρμογή.

Σχήμα 6.2: Οθόνη ειδοποίησης ήχων σε πραγματικό χρόνο

Όταν μεταβεί ο χρήστης σε αυτή την οθόνη, όπως αναφέραμε, καλείται η MainActivity2() η οποία ξεκινάει να ηχογραφεί το περιβάλλον του χρήστη συνεχόμενα και αποθηκεύει αρχεία .wav χρονικών διαστημάτων των 5 δευτερολέπτων. Αν αναγνωριστεί επιτυχώς κάποιος από τους πέντε ήχους τότε συμβαίνουν 3 σενάρια ταυτόχρονα:

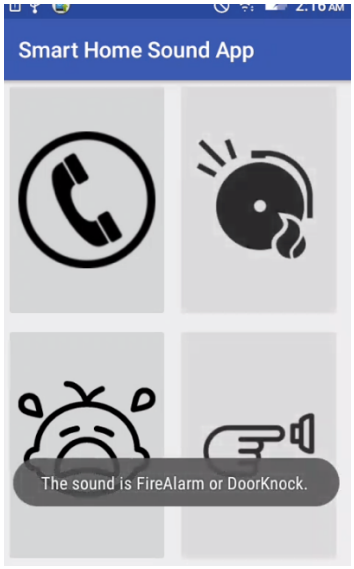
Μέσω των μεθόδων BlinkBtn δημιουργείται το αντικείμενο v (Vibrator v = (Vibrator) this.getSystemService(Context.VIBRATOR_SERVICE)) το οποίο αξιοποιεί την κλάση Vibrator της βιβλιοθήκης του Android Studio. Στην συνέχεια, μέσω της συνάρτησης v.vibrate(500) στέλνεται εντολή να δονηθεί το κινητό για 500 milliseconds.

Αμέσως, αξιοποιείται η κλάση Animation που επίσης ανήκει στην βιβλιοθήκη του Android Studio και επιτρέπει την επ' αόριστο ξεθωρίαση ενός εικονιδίου καθώς και την επανεμφάνισή του. Έτσι, ο χρήστης αντιλαμβάνεται αμέσως ποιον ήχο αφορά η δόνηση του κινητού, χωρίς να διαβάσει κάποιο μήνυμα, παρά μόνο προσέχοντας ποιο εικονίδιο αναβοσβήνει. Εν τέλει, η εφαρμογή μόλις λάβει μια ειδοποίηση για έναν ήχο θα δονείται συνεχώς έως ότου ο χρήστης πατήσει στο αντίστοιχο εικονίδιο δηλώνοντας έτσι ότι έλαβε γνώση της ύπαρξης της ειδοποίησης και η εφαρμογή θα περιμένει έως ότου αναγνωριστεί ο επόμενος ήχος.

Ταυτόχρονα με την δόνηση και το αναβόσβημα των εικονιδίων εμφανίζεται επίσης στον χρήστη ένα αναδυόμενο μήνυμα στην οθόνη με την βοήθεια της κλάσης Toast (επίσης βιβλιοθήκη του Android Studio) το οποίο εξαφανίζεται μετά από λίγα δευτερόλεπτα.

Να σημειωθεί πως στην ειδοποίηση Toast εμφανίζονται δύο πιθανά σενάρια. Με προτεραιότητα εμφανίζεται ο ήχος με τον οποίο έγινε ταυτοποίηση μέσα στην Database των Trained Sounds και δεύτερος εμφανίζεται ο ήχος με τον οποίο έγινε ταυτοποίηση στον Planner. Θεωρήθηκε στα πλαίσια της διπλωματικής πως οι ήχοι που βρίσκονται στην Database των Trained Sounds αντιπροσωπεύουν καλύτερα τον χώρο της κατοικίας, μιας και ηχογραφήθηκαν στο περιβάλλον της, οπότε το αποτέλεσμα θα είναι εγκυρότερο κατά πάσα πιθανότητα.

Στα ακόλουθα στιγμιότυπα εμφαίνονται παραδείγματα των παραπάνω εις πράξην:



Σχήμα 6.3: Ειδοποίηση ήχου Συναγερμού Φωτιάς – Χτύπος σε Πόρτα

Για τους ήχους που ανήκουν στην κατηγορία DoorKnock και DoorBell αναβοσβήνει το ίδιο εικονίδιο, το κάτω δεξιά, μιας και οι δύο ήχοι υποδηλώνουν δραστηριότητα στην είσοδο της κατοικίας και καλείται ο χρήστης να ανοίξει την πόρτα.

Σχήμα 6.4: Ειδοποίηση ήχου Συναγερμού Φωτιάς – Κουδούνι Πόρτας

Παρατίθεται επίσης στιγμιότυπο της εφαρμογής και σε άλλο ενδεχόμενο σενάριο ειδοποίησης ήχων:

Σχήμα 6.5: Ειδοποίηση ήχου Χτύπος σε Πόρτα– Κλάμα μωρού

Έχει ληφθεί υπόψη και το σενάριο να έχουμε την ίδια ταυτοποίηση και στην Trained Sounds Database και στην Database του Planner. Σε αυτην την περίπτωση το μήνυμα που εμφανίζεται εν μέσω του Toast είναι αλλαγμένο και επίσης αναβοσβήνει μόνο ένα εικονίδιο:

Σχήμα 6.6: Ειδοποίηση Χτυπήματος Πόρτας – Όμοιο σενάριο

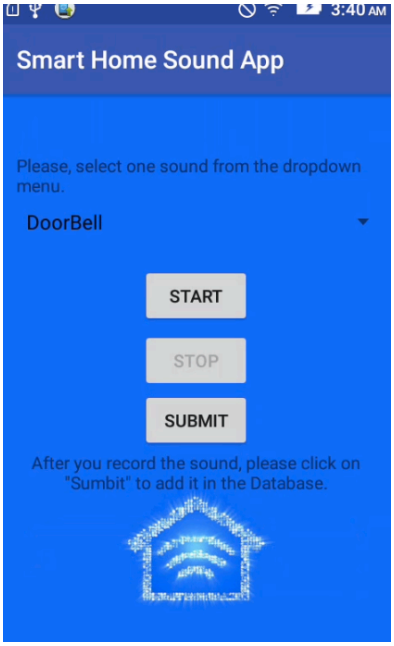
Παρατίθεται επίσης στιγμιότυπο στο οποίο εμφαίνεται το οτι τα εικονίδια που υποδεικνύουν τους ταυτοποιημένους ήχους, αναβοσβήνουν ταυτόχρονα με την δόνηση της συσκευής:

Σχήμα 6.7: Στιγμιότυπο ένδειξης αναβόσβησης εικονιδίων

Παρατηρούμε, λοιπόν, πως ο χρήστης ειδοποιείται επιτυχώς με δόνηση της συσκευής, εμφανίζονται αντίστοιχα μηνύματα και αναβοσβήνουν τα κατάλληλα εικονίδια, έως ότου τα πατήσει ο χρήστης για να δηλώσει ότι έλαβε γνώση σχετικά με την ειδοποίηση. Άρα, η εφαρμογή μας καλύπτει πλήρως όλους τους πιθανούς τρόπους με τους οποίους ένα άτομο με θέμα ακοής μπορεί να ειδοποιηθεί για την παρουσία ενός ήχου.

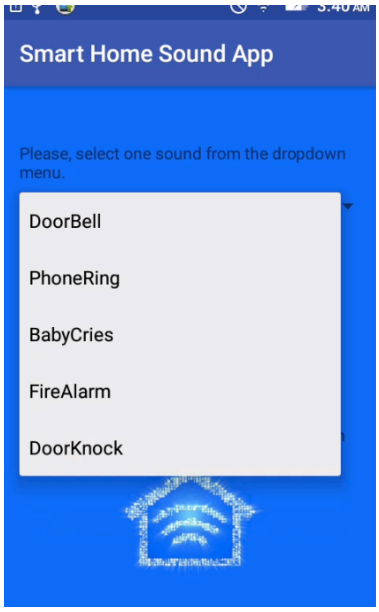
6.3.2 Κουμπί Train

Αν πατηθεί το κουμπί "TRAIN" στην αρχική οθόνη, τότε ο χρήστης θα οδηγηθεί στην τρίτη οθόνη που αποτελεί την εφαρμογή μας και στην οποία εκτελείται η κλάση MainActivity3(). Ο ρόλος της παρούσας οθόνης είναι να προσφέρει στον χρήστη την δυνατότητα να κάνει train την εφαρμογή μας τοπικούς ήχους που ακούγονται στην οικία του. Για παράδειγμα, το κουδούνι της πόρτας του χρήστη να έχει ιδιαίτερο ήχο και να μην αναγνωρίζεται επιτυχώς από τους προεπιλεγόμενους του Planner. Για να επιλυθεί αυτό το πρόβλημα και να βελτιωθεί η ακρίβεια αναγνώρισης της εφαρμογής, ο χρήστης θα ηχογραφήσει τον ήχο του κουδουνιού της οικίας και θα το αποθηκεύσει στο Database των Trained Sounds. Το interface στο οποίο λαμβάνουν χώρα τα παραπάνω είναι το εξής:



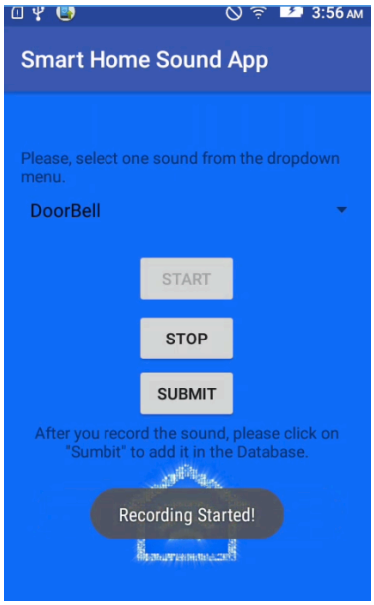
Σχήμα 6.8: Οθόνη για τοπικό Sound Training

Παρατίθεται λιτό κείμενο με οδηγίες ώστε ο χρήστης να καθοδηγηθεί καθώς και να διασαφηνιστεί το περιεχόμενο και ο σκοπός της οθόνης αυτής. Αρχικά, του δίνεται η δυνατότητα να επιλέξει την κατηγορία στην οποία θα ανήκει ο ήχος που θα ηχογραφήσει σε έναν Spinner ο οποίος είναι ένα αντικείμενο που έχει δημιουργηθεί μέσα στην μέθοδο addListenerOnButton() της κλάσης MainActivity3():



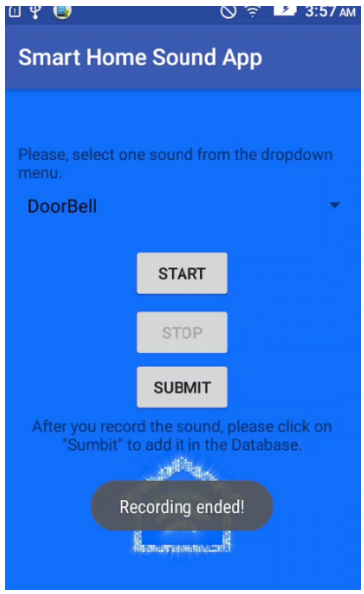
Σχήμα 6.9: Επιλογές του Spinner της εφαρμογής

Αφότου ο χρήστης επιλέξει την επιθυμητή κατηγορία ήχου στον Spinner, καλείται να πατήσει το κουμπί "START" ώστε να ηχογραφήσει τον ήχο που επιθυμεί να προσθέσει στην Database των Trained Sounds. Επίσης, ένα μήνυμα εμφανίζεται ώστε να επιβεβαιώσει πως η ηχογράφηση ξεκίνησε:



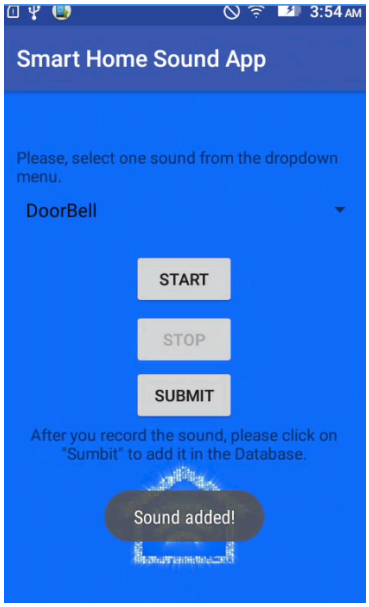
Σχήμα 6.10: Έναρξη ηχογράφησης τοπικού ήχου

Όταν ο χρήστης ηχογραφήσει πλέον τον επιθυμητό ήχο, πατάει το κουμπί “STOP” και η ηχογράφηση τερματίζεται:



Σχήμα 6.11: Λήξη ηχογράφησης τοπικού ήχου

Εν τέλει, αφότου ο χρήστης πατήσει το κουμπί Submit, ο ήχος στέλνεται στην διεύθυνση URL http://localhost:1880/PostWavTrain και προστίθεται μέσω του Node-RED στην Database των Trained Sounds. Εμφανίζεται και αντίστοιχη οπτική ειδοποίηση:



Σχήμα 6.12: Επιβεβαίωση προσθήκης τοπικού ήχου στην Database

Με τα παραπάνω δείγματα γίνεται προφανές πως με την δυνατότητα των Trained Sounds αυξάνεται ποιοτικά η ακρίβεια αναγνώρισης ήχων, ειδικά όσον αφορά την αναγνώριση συσκευών μέσα στην κατοικία του χρήστη, στην οποία ίσως οι ήχοι που αναπαράγονται δεν ταυτοποιούνται με τους ήχους που υπάρχουν στην προεπιλεγμένη Database του Planner.

7 Συμπεράσματα και Μελλοντικές Επεκτάσεις

Στο παρόν κεφάλαιο παρουσιάζονται τα συμπεράσματα που προκύπτουν από την εξέταση της ορθής λειτουργίας της εφαρμογής μας σε πραγματικές συνθήκες. Γίνεται αναφορά στα θετικά και τα αρνητικά στοιχεία της εφαρμογής και εν τέλει, παρουσιάζονται προτάσεις για την επέκτασή της, βελτιώσεις και τρόπους αντιμετώπισης των αδύναμων σημείων της.

Με την παρούσα εργασία έγινε μια κατά το δυνατόν πλήρης μελέτη όλων των παραμέτρων που θα πρέπει να συμπεριλαμβάνει ένα Έξυπνο Σπίτι. Στην συνέχεια, έγινε μελέτη των απαιτήσεων που παρουσιάζονται από τους κάτοικους ενός Έξυπνου Σπιτιού που εμφανίζουν βαρηκοΐα ή κώφωση. Οι απαιτήσεις αυτές καταγράφηκαν και παρουσιάστηκαν λεπτομερώς ως προς την ενσωμάτωσή τους στο σύστημα του Έξυπνου Σπιτιού καθώς και ως προς την αλληλουχία της λειτουργίας τους. Τέλος, δόθηκε ιδιαίτερη έμφαση στην προσαρμογή του ιδιαίτερου τρόπου ειδοποίησης της ομάδας των χρηστών για τους οποίους προορίζεται η εργασία.

Το πλαίσιο που παρουσιάσαμε αποτελεί μία πρώτη προσέγγιση για την υλοποίηση της ολοκληρωμένης εφαρμογής που περιγράφεται στην Ενότητα 2.1. Σχεδιάσαμε μια εφαρμογή η οποία εγκαθίσταται σε συσκευές με λογισμικό Android και ηχογραφεί ήχους του κοντινού περιβάλλοντος. Στην συνέχεια, αφότου εντοπίσει το είδος του ήχου, εμφανίζεται ειδοποίηση στον χρήστη μέσω δόνησης ή μηνυμάτων, στην οθόνη της συσκευής.

Η εφαρμογή μας σε πραγματικές συνθήκες εμφανίζει ικανοποιητικά στατιστικά τα οποία κυμαίνονται σε ποσοστά ~60% επιτυχημένης αναγνώρισης ήχων. Αν ο χρήστης προσθέσει μέσω training στην οντολογία του συστήματος τοπικούς ήχους, που εμφανίζονται ειδικά στην κατοικία του, τότε τα ποσοστά επιτυχίας αυξάνονται και κυμαίνονται σε περίπου ~70%.

Επομένως, καθίσταται σαφές πως η εφαρμογή, που αναπτύχθηκε στα πλαίσια της διπλωματικής, έχει φτάσει σε ένα στάδιο αποτελεσματικότητας που θα μπορούσε να χρησιμοποιηθεί από ένα άτομο με βαρηκοΐα και να το ειδοποιεί για τους ήχους που παράγονται μέσα στην ηχητική εμβέλεια του κινητού, κάτι που αποσκοπούσαμε εξ αρχής.

Το πρώτο επόμενο βήμα για βελτίωση της εφαρμογής μας θα ήταν εφαρμοστούν οι ακόλουθες προτεινόμενες επεκτάσεις:

- Εμπλουτισμός της οντολογίας του Planner με πρόσθετους ήχους. Η παρούσα οντολογία περιέχει δέκα (10) ήχους από κάθε κατηγορία με τους οποίους συγκρίνεται κάθε νέος εισερχόμενος ήχος στον Planner. Εμπλουτισμός της οντολογίας σημαίνει και μεγαλύτερο εύρος παρατηρήσεων ως προς σύγκριση και εν τέλει, μεγαλύτερη ακρίβεια στην κατηγοριοποίηση του ήχου.
- Εμπλουτισμός των κατηγοριών ήχων τόσο στον Planner όσο και ελευθέρια στους ήχους που μπορεί να κάνει προσθήκη ο χρήστης (σωλήνες νερού, φωνές ανθρώπων κτλ.).
- Βελτίωση του αλγορίθμου σύγκρισης των ήχων. Ο αλγόριθμος που χρησιμοποιούμε στην εφαρμογή μας είναι ο Brays – Curtis Distance. Ο αλγόριθμος αυτός θα βγάζει πάντα κάποιο αποτέλεσμα, ακόμα και σε κατάσταση ησυχίας, μιας και θα βγαίνει ταυτοποίηση με τον πλησιέστερο σε ησυχία ήχο των Βάσεων Δεδομένων. Ένας βέλτιστος και γρηγορότερος αλγόριθμος μπορεί να βοηθήσει σε επεξεργασία μεγαλύτερου όγκου δεδομένων καθώς και αποτελεσματικότερης σύγκρισης.
- Επέκταση της εφαρμογής μας και εγκαθίδρυση επικοινωνίας της με λάμπες LED και αισθητήρες ήχου, τοποθετημένα μέσα στα δωμάτια της κατοικίας, τα οποία θα λειτουργούν αναλόγως των συμβάντων που λαμβάνουν χώρα. Ακόμη, θα μπορούσαμε να εγκαθιδρύσουμε επικοινωνία της εφαρμογής μας με Smart Watch το οποίο θα φοράει ο χρήστης και θα λαμβάνει μηνύματα / ειδοποιήσεις.
- Τέλος, η σύγκριση των ήχων λαμβάνει χώρα στον Planner ο οποίος ουσιαστικά είναι Server. Θα μπορούσαμε να ενσωματώσουμε μέσα στην εφαρμογή μας την ίδια λειτουργία χρησιμοποιώντας Java. Έτσι θα μειωθεί ο απαιτούμενος χρόνος σύγκρισης ήχων, μιας και θα αποφευχθεί η ανάγκη δημιουργίας σύνδεσης HTTP.

Βιβλιογραφία

[1] Χ. Τζανετοπούλου, «Έξυπνο Σπίτι με χρήση του προτύπου Konnex και εξοικονόμηση ενέργειας», ΕΜΠ - ΗΜΜΥ, Αθήνα, Ιούλιος 2010

[2] Linda A. Jacobsen, Mary Kent, Marlene Lee and Mark Mather, "America's Aging Population", Population Reference Bureau, Vo. 66, no.1, February 2011

[3] American Academy of Audiology, "Myth VS Fact, The truth about hearing loss", September 2015, http://www.audiology.org/sites/default/files/AAM Poster (24x36).pdf

[4] Dayna S. Dalton, Karen J. Cruickshanks, Barbara E. K. Klein, Ronald Klein, Terry L. Wiley and David M. Nondahl, "The Impact of Hearing Loss on Quality of Life in Older Adults", The Gerontologist, Vol. 43, No. 5, 661-668, October 2003

[5] Bennett, S. (1993). *A History of Control Engineering 1930–1955*. London: Peter Peregrinus Ltd. on behalf of the Institution of Electrical Engineers. ISBN 0-86341-280-7

[6] Characteristics of Sensor, Online Electrical Engineering Study Site, Retrieved December 2016 from https://www.electrical4u.com/sensor-types-of-sensor/

[7] Types of Sensors, SunMan Engineering Inc. , Retrieved January 2017 from http://www.sunmantechnology.com/system-ip/acoustic-sensors.html

[8] Woodford Chris, "How Microphones Work: Explain That Stuff", 26 May 2011. http://www.explainthatstuff.com/microphones.html

[9] "Applications of MP Motion Sensor 'Napion' ", Panasonic, Retrieved January 2017 from http://media.digikey.com/pdf/data sheets/panasonic electric works pdfs/amn design manual.pdf

[10] "An Introduction to the Internet of Things (IoT)" *Cisco.com*. San Francisco, California: Lopez Research. November 2013.

[11] "After Big Data, it is now Small Data's turn to change our world.", Quorvo Company, Retrieved March 2017 from http://www.hometoys.com/article/2014/06/sentrollers-and-the-world-of-small-data/2164/

[12] Vinton G. Cerf, Robert E. Kahn, "A Protocol for Packet Network Intercommunication", IEEE Transactions on Communications, Vol. 22, No. 5, May 1974 pp. 637–648

[13] "IPv4 Address Exhaustion, Mitigation Strategies and Implications for the U.S", Committee on Communications Policy of the Institute of Electrical and Electronics Engineers-United States of America, White Paper, 2009

[14] Γ. Παναγοπούλου, "Διαδίκτυο των Πραγμάτων. Ζητήματα Προσωπικών Δεδομένων", Τμήμα Ελεγκτών, Αρχή Προστασίας Δεδομένων, Ιανουάριος 2014

[15] Ειδικό Ευρωβαρόμετρο 359, Συμπεριφορές στην Προστασία Δεδομένων και την Ηλεκτρονική Ταυτότητα στην Ευρωπαϊκή Ένωση, Ιούνιος 2011

[16] GFK, "Smart Home Study 2015", September and October 2015

[17] "Characteristics and Advantages of a Smart Home", Retrieved April 2016 from http://www.ezines.gr/ipiresies-gia-to-spiti/ta-xaraktiristika-kai-ta-pleonektimata-tou-eksipnou-spitiou.html

[18] Navid Azodi, Thomas Pryor, "SignAloud Gloves", University of Washington, April 2016

[19] Kwang Jeonh, Min-hee Kim and Hyun Kim, "Vibering Sensor: For The Deaf", June 2008

[20] Mads Sukhdev Hindhede, "BabelFisk: Speech Recognition Software", January 2011

[21] "Captioned Telephone Service", National Association of the Deaf, Retrieved April 2016 from https://www.nad.org/resources/technology/telephone-and-relay-services/captioned-telephone-service-cts/

[22] Arndt P,S Arcaroli J, Hines A, Ebinger K, "Within Subject Comparison of Advanced Coding Strategies in the Nucleus 24 Cochlear Implant" Cochlear Corporation,1999

[23] Cui Chen, Wang Qi, Shi Kaiyuan, Huang Jianbo, Geng Kun, Wang Zhi, Chen Zhen and Qiu Shuang, "VV-Talker Speech Device", Industrial Designers Society of America, July 2011

[24] "Useful Applications for Hearing Loss", Hearing Link, Retrieved December 2016 from https://www.hearinglink.org/living/loops-equipment/useful-apps-for-hearing-loss/

[25] Nick Heath, "How IBM's Node-Red is hacking together the internet of things", Tech Republic, Retrieved May 2017 from http://www.techrepublic.com/article/node-red/

[26] O. Voutyras, P. Bourelos, D. Kyriazis, and T. Varvarigou, "An Architecture supporting Knowledge flow in Social Internet of Things systems", 10th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), pp. 45--50, 2014.

[27] IBM. "An architectural blueprint for autonomic computing.", Autonomic Computing White Paper, June 2005, Third Edition.

[28] AsyncTask, Android.os.AsyncTask.Android.Development, Retrieved March 2016 from https://developer.android.com/reference/android/os/AsyncTask.html

[29] HttpURLConnection, java.net.HttpURLConnection.Android.Development, Retrieved February 2016 from https://developer.android.com/reference/java/net/HttpURLConnection.html

[30] https://www.npmjs.com/package/bray-curtis

[31] Microsoft Developer Network Website, June 13, 2014, http://msdn.microsoft.com/en-us/library/

8 Παράρτημα

8.1 Κώδικας για Back-End

MainActivity()

```
import android.content.Intent;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.View;

public class MainActivity extends AppCompatActivity {

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

    public void sendMessage1(View view) {
        Intent intent = new Intent(this, Main2Activity.class);
        startActivity(intent);
    }

    public void sendMessage2(View view) {
        Intent intent = new Intent(this, Main3Activity.class);
        startActivity(intent);
    }
}
```

Main2Activity()

```
import android.content.Context;
import android.media.AudioFormat;
import android.media.AudioRecord;
import android.media.MediaRecorder;
import android.os.Bundle;
import android.os.Environment;
import android.os.Vibrator;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.view.animation.AlphaAnimation;
import android.view.animation.Animation;
import android.view.animation.LinearInterpolator;
import android.widget.ImageButton;
import android.widget.Toast;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
```

```
public class Main2Activity extends AppCompatActivity {
    private static final int RECORDER_BPP = 16;
    private static final String AUDIO_RECORDER_FILE_EXT_WAV = ".wav";
    private static final String AUDIO_RECORDER_FOLDER = "AudioRecorder";
    private static final String AUDIO_RECORDER_TEMP_FILE = "record_temp.raw";
    private static final int RECORDER_SAMPLERATE = 44100;
    private static final int RECORDER_CHANNELS = AudioFormat.CHANNEL_IN_STEREO;
    private static final int RECORDER_AUDIO_ENCODING = AudioFormat.ENCODING_PCM_16BIT;

    private AudioRecord recorder = null;
    private int bufferSize = 0;
    private Thread recordingThread = null;
    private boolean isRecording = false;
    private boolean readingFile = false;


    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main2);



        bufferSize = AudioRecord.getMinBufferSize(RECORDER_SAMPLERATE,
RECORDER_CHANNELS, RECORDER_AUDIO_ENCODING);

        //ksekinaei h hxografhsh me to pou arxisei to programma
        startRecording();

        //kanei infinite loop kalwntas thread
        Thread TimerThread = new Thread(new Runnable() {

            @Override
            public void run() {
                while (true) {
                    try {
                        Thread.sleep(5000);
                        while (readingFile) Thread.sleep(1000);
                        stopRecording();

                        startRecording();
                    } catch (InterruptedException e) {
                        e.printStackTrace();
                    }
                }
            }
        });
        TimerThread.start();
    }




    private String getFilename() {
        String filepath = Environment.getExternalStorageDirectory().getPath();
        File file = new File(filepath, AUDIO_RECORDER_FOLDER);

        if (!file.exists()) {
            file.mkdirs();
        }
        return (file.getAbsolutePath() + "/" + "realsound" +
AUDIO_RECORDER_FILE_EXT_WAV);
```

```
    }

    private String getTempFilename() {
        String filepath = Environment.getExternalStorageDirectory().getPath();
        File file = new File(filepath, AUDIO_RECORDER_FOLDER);

        if (!file.exists()) {
            file.mkdirs();
        }

        File tempFile = new File(filepath, AUDIO_RECORDER_TEMP_FILE);

        if (tempFile.exists())
            tempFile.delete();

        return (file.getAbsolutePath() + "/" + AUDIO_RECORDER_TEMP_FILE);
    }

    private void startRecording() {
        recorder = new AudioRecord(MediaRecorder.AudioSource.MIC,
                RECORDER_SAMPLERATE, RECORDER_CHANNELS, RECORDER_AUDIO_ENCODING,
bufferSize);

        recorder.startRecording();

        isRecording = true;

        recordingThread = new Thread(new Runnable() {

            @Override
            public void run() {
                writeAudioDataToFile();
            }
        }, "AudioRecorder Thread");

        recordingThread.start();
    }

    private void writeAudioDataToFile() {
        byte data[] = new byte[bufferSize];
        String filename = getTempFilename();
        FileOutputStream os = null;

        try {
            os = new FileOutputStream(filename);
        } catch (FileNotFoundException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }

        int read = 0;

        if (null != os) {
            while (isRecording) {
                read = recorder.read(data, 0, bufferSize);

                if (AudioRecord.ERROR_INVALID_OPERATION != read) {
                    try {
                        os.write(data);
                    } catch (IOException e) {
                        e.printStackTrace();
                    }
                }
```

```
            }

            try {
                os.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }

    private void stopRecording() {
        if (null != recorder) {
            isRecording = false;

            recorder.stop();
            recorder.release();

            recorder = null;
            recordingThread = null;
        }

        copyWaveFile(getTempFilename(), getFilename());
        deleteTempFile();


        //stelnontas tis frequencies san http message
        readingFile = true;
        WavTask wavPostTask = new WavTask();
        wavPostTask.execute(this);
    }

    public void showResults(String response) {
        readingFile = false;
        Context context = getApplicationContext();
        CharSequence text = response;
        AppLog.logString("Minima: " + response);

        if (response.contains("DoorBell")) {
            BlinkBtn1();
        }
        if (response.contains("PhoneRing")) {
            BlinkBtn2();
        }
        if (response.contains("BabyCries")) {
            BlinkBtn3();
        }
        if (response.contains("FireAlarm")) {
            BlinkBtn4();
        }
        if (response.contains("DoorKnock")) {
            BlinkBtn1();
        }
        int duration = Toast.LENGTH_SHORT;
        Toast toast = Toast.makeText(context, text, duration);
        toast.show();

    }


    private void deleteTempFile() {
        File file = new File(getTempFilename());

        file.delete();
```

```
}

private void copyWaveFile(String inFilename, String outFilename) {
    FileInputStream in = null;
    FileOutputStream out = null;
    long totalAudioLen = 0;
    long totalDataLen = totalAudioLen + 36;
    long longSampleRate = RECORDER_SAMPLERATE;
    int channels = 2;
    long byteRate = RECORDER_BPP * RECORDER_SAMPLERATE * channels / 8;

    byte[] data = new byte[bufferSize];

    try {
        in = new FileInputStream(inFilename);
        out = new FileOutputStream(outFilename);
        totalAudioLen = in.getChannel().size();
        totalDataLen = totalAudioLen + 36;

        AppLog.logString("File size: " + totalDataLen);

        WriteWaveFileHeader(out, totalAudioLen, totalDataLen,
                longSampleRate, channels, byteRate);

        while (in.read(data) != -1) {
            out.write(data);
        }

        in.close();
        out.close();
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
}

private void WriteWaveFileHeader(
        FileOutputStream out, long totalAudioLen,
        long totalDataLen, long longSampleRate, int channels,
        long byteRate) throws IOException {

    byte[] header = new byte[44];

    header[0] = 'R';  // RIFF/WAVE header
    header[1] = 'I';
    header[2] = 'F';
    header[3] = 'F';
    header[4] = (byte) (totalDataLen & 0xff);
    header[5] = (byte) ((totalDataLen >> 8) & 0xff);
    header[6] = (byte) ((totalDataLen >> 16) & 0xff);
    header[7] = (byte) ((totalDataLen >> 24) & 0xff);
    header[8] = 'W';
    header[9] = 'A';
    header[10] = 'V';
    header[11] = 'E';
    header[12] = 'f';  // 'fmt ' chunk
    header[13] = 'm';
    header[14] = 't';
    header[15] = ' ';
    header[16] = 16;  // 4 bytes: size of 'fmt ' chunk
    header[17] = 0;
    header[18] = 0;
```

```
        header[19] = 0;
        header[20] = 1;  // format = 1
        header[21] = 0;
        header[22] = (byte) channels;
        header[23] = 0;
        header[24] = (byte) (longSampleRate & 0xff);
        header[25] = (byte) ((longSampleRate >> 8) & 0xff);
        header[26] = (byte) ((longSampleRate >> 16) & 0xff);
        header[27] = (byte) ((longSampleRate >> 24) & 0xff);
        header[28] = (byte) (byteRate & 0xff);
        header[29] = (byte) ((byteRate >> 8) & 0xff);
        header[30] = (byte) ((byteRate >> 16) & 0xff);
        header[31] = (byte) ((byteRate >> 24) & 0xff);
        header[32] = (byte) (2 * 16 / 8);  // block align
        header[33] = 0;
        header[34] = RECORDER_BPP;  // bits per sample
        header[35] = 0;
        header[36] = 'd';
        header[37] = 'a';
        header[38] = 't';
        header[39] = 'a';
        header[40] = (byte) (totalAudioLen & 0xff);
        header[41] = (byte) ((totalAudioLen >> 8) & 0xff);
        header[42] = (byte) ((totalAudioLen >> 16) & 0xff);
        header[43] = (byte) ((totalAudioLen >> 24) & 0xff);

        out.write(header, 0, 44);
    }

    public void BlinkBtn1() {
        Vibrator v = (Vibrator) this.getSystemService(Context.VIBRATOR_SERVICE);
        // Vibrate for 500 milliseconds
        v.vibrate(500);
        final Animation animation = new AlphaAnimation(1, 0); // Change alpha from
fully visible to invisible
        animation.setDuration(500); // duration - half a second
        animation.setInterpolator(new LinearInterpolator()); // do not alter animation
rate
        animation.setRepeatCount(Animation.INFINITE); // Repeat animation infinitely
        animation.setRepeatMode(Animation.REVERSE); // Reverse animation at the end so
the button will fade back in
        final ImageButton btn = (ImageButton) findViewById(R.id.doorbtn);
        btn.startAnimation(animation);
        btn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(final View view) {
                view.clearAnimation();
            }
        });
    }

    public void BlinkBtn2() {
        Vibrator v = (Vibrator) this.getSystemService(Context.VIBRATOR_SERVICE);
        // Vibrate for 500 milliseconds
        v.vibrate(500);
        final Animation animation = new AlphaAnimation(1, 0); // Change alpha from
fully visible to invisible
        animation.setDuration(500); // duration - half a second
        animation.setInterpolator(new LinearInterpolator()); // do not alter animation
rate
        animation.setRepeatCount(Animation.INFINITE); // Repeat animation infinitely
        animation.setRepeatMode(Animation.REVERSE); // Reverse animation at the end so
the button will fade back in
```

```
        final ImageButton btn = (ImageButton) findViewById(R.id.phonebtn);
        btn.startAnimation(animation);
        btn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(final View view) {
                view.clearAnimation();
            }
        });
    }

    public void BlinkBtn3() {
        Vibrator v = (Vibrator) this.getSystemService(Context.VIBRATOR_SERVICE);
        // Vibrate for 500 milliseconds
        v.vibrate(500);
        final Animation animation = new AlphaAnimation(1, 0); // Change alpha from
fully visible to invisible
        animation.setDuration(500); // duration - half a second
        animation.setInterpolator(new LinearInterpolator()); // do not alter animation
rate
        animation.setRepeatCount(Animation.INFINITE); // Repeat animation infinitely
        animation.setRepeatMode(Animation.REVERSE); // Reverse animation at the end so
the button will fade back in
        final ImageButton btn = (ImageButton) findViewById(R.id.babybtn);
        btn.startAnimation(animation);
        btn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(final View view) {
                view.clearAnimation();
            }
        });
    }

    public void BlinkBtn4() {
        Vibrator v = (Vibrator) this.getSystemService(Context.VIBRATOR_SERVICE);
        // Vibrate for 500 milliseconds
        v.vibrate(500);
        final Animation animation = new AlphaAnimation(1, 0); // Change alpha from
fully visible to invisible
        animation.setDuration(500); // duration - half a second
        animation.setInterpolator(new LinearInterpolator()); // do not alter animation
rate
        animation.setRepeatCount(Animation.INFINITE); // Repeat animation infinitely
        animation.setRepeatMode(Animation.REVERSE); // Reverse animation at the end so
the button will fade back in
        final ImageButton btn = (ImageButton) findViewById(R.id.alarmbtn);
        btn.startAnimation(animation);
        btn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(final View view) {
                view.clearAnimation();
            }
        });
    }

}
```

Main3Activity()

```
package com.example.corrina.thesis_app;

import android.content.Context;
import android.media.AudioFormat;
```

```
import android.media.AudioRecord;
import android.media.MediaRecorder;
import android.os.Bundle;
import android.os.Environment;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.Spinner;
import android.widget.Toast;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;

import static android.R.attr.duration;

public class Main3Activity extends AppCompatActivity {

    public String TrainSound;
    private static final int RECORDER_BPP = 16;
    private static final String AUDIO_RECORDER_FILE_EXT_WAV = ".wav";
    private static final String AUDIO_RECORDER_FOLDER = "AudioRecorder";
    private static final String AUDIO_RECORDER_TEMP_FILE = "record_temp.raw";
    private static final int RECORDER_SAMPLERATE = 44100;
    private static final int RECORDER_CHANNELS = AudioFormat.CHANNEL_IN_STEREO;
    private static final int RECORDER_AUDIO_ENCODING = AudioFormat.ENCODING_PCM_16BIT;
    short[] audioData;

    private AudioRecord recorder = null;
    private int bufferSize = 0;
    private Thread recordingThread = null;
    private boolean isRecording = false;
    private Spinner spinner1;
    private Button btnSubmit;
    private View.OnClickListener btnClick = new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            switch (v.getId()) {
                case R.id.btnStart: {
                    AppLog.logString("Start Recording");

                    enableButtons(true);

                    startRecording();

                    break;
                }
                case R.id.btnStop: {
                    AppLog.logString("Stop Recording");

                    enableButtons(false);

                    stopRecording();

                    break;
                }
            }
        }
    };
```

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main3);
    setButtonHandlers();
    enableButtons(false);
    addListenerOnButton();
    addListenerOnSpinnerItemSelection();

    bufferSize = AudioRecord.getMinBufferSize
            (RECORDER_SAMPLERATE, RECORDER_CHANNELS, RECORDER_AUDIO_ENCODING) * 3;

    audioData = new short[bufferSize]; //short array that pcm data is put into.
}

private void setButtonHandlers() {
    ((Button) findViewById(R.id.btnStart)).setOnClickListener(btnClick);
    ((Button) findViewById(R.id.btnStop)).setOnClickListener(btnClick);
}

public void addListenerOnSpinnerItemSelection() {
    spinner1 = (Spinner) findViewById(R.id.spinner1);
    spinner1.setOnItemSelectedListener(new CustomOnItemSelectedListener());
}

// get the selected dropdown list value
public void addListenerOnButton() {

    spinner1 = (Spinner) findViewById(R.id.spinner1);
    btnSubmit = (Button) findViewById(R.id.btnSubmit);

    btnSubmit.setOnClickListener(new OnClickListener() {


        @Override
        public void onClick(View v) {

            TrainSound = String.valueOf(spinner1.getSelectedItem());
            //stelnontas to wav san http message
            WavTaskTrain wavPostTask = new WavTaskTrain();
            wavPostTask.execute(TrainSound);
        }

    });
}

private void enableButton(int id, boolean isEnable) {
    ((Button) findViewById(id)).setEnabled(isEnable);
}

private void enableButtons(boolean isRecording) {
    enableButton(R.id.btnStart, !isRecording);
    enableButton(R.id.btnStop, isRecording);
}

private String getFilename() {
    String filepath = Environment.getExternalStorageDirectory().getPath();
    File file = new File(filepath, AUDIO_RECORDER_FOLDER);

    if (!file.exists()) {
        file.mkdirs();
    }
```

```
        return (file.getAbsolutePath() + "/" + "trainsound" +
AUDIO_RECORDER_FILE_EXT_WAV);
    }

    private String getTempFilename() {
        String filepath = Environment.getExternalStorageDirectory().getPath();
        File file = new File(filepath, AUDIO_RECORDER_FOLDER);

        if (!file.exists()) {
            file.mkdirs();
        }

        File tempFile = new File(filepath, AUDIO_RECORDER_TEMP_FILE);

        if (tempFile.exists())
            tempFile.delete();

        return (file.getAbsolutePath() + "/" + AUDIO_RECORDER_TEMP_FILE);
    }

    private void startRecording() {
        recorder = new AudioRecord(MediaRecorder.AudioSource.MIC,
                RECORDER_SAMPLERATE,
                RECORDER_CHANNELS,
                RECORDER_AUDIO_ENCODING,
                bufferSize);
        int i = recorder.getState();
        if (i == 1)
            recorder.startRecording();

        isRecording = true;

        recordingThread = new Thread(new Runnable() {
            @Override
            public void run() {
                writeAudioDataToFile();
            }
        }, "AudioRecorder Thread");

        recordingThread.start();
    }

    private void writeAudioDataToFile() {
        byte data[] = new byte[bufferSize];
        String filename = getTempFilename();
        FileOutputStream os = null;

        try {
            os = new FileOutputStream(filename);
        } catch (FileNotFoundException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }

        int read = 0;
        if (null != os) {
            while (isRecording) {
                read = recorder.read(data, 0, bufferSize);
                if (read > 0) {
                }

                if (AudioRecord.ERROR_INVALID_OPERATION != read) {
                    try {
```

```
                    os.write(data);
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }

        try {
            os.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}

private void stopRecording() {
    if (null != recorder) {
        isRecording = false;

        int i = recorder.getState();
        if (i == 1)
            recorder.stop();
        recorder.release();

        recorder = null;
        recordingThread = null;
    }

    copyWaveFile(getTempFilename(), getFilename());
    deleteTempFile();

}

private void deleteTempFile() {
    File file = new File(getTempFilename());
    file.delete();
}

private void copyWaveFile(String inFilename, String outFilename) {
    FileInputStream in = null;
    FileOutputStream out = null;
    long totalAudioLen = 0;
    long totalDataLen = totalAudioLen + 36;
    long longSampleRate = RECORDER_SAMPLERATE;
    int channels = 2;
    long byteRate = RECORDER_BPP * RECORDER_SAMPLERATE * channels / 8;

    byte[] data = new byte[bufferSize];

    try {
        in = new FileInputStream(inFilename);
        out = new FileOutputStream(outFilename);
        totalAudioLen = in.getChannel().size();
        totalDataLen = totalAudioLen + 36;

        AppLog.logString("File size: " + totalDataLen);

        WriteWaveFileHeader(out, totalAudioLen, totalDataLen,
                longSampleRate, channels, byteRate);

        while (in.read(data) != -1) {
            out.write(data);
        }
```

```
            in.close();
            out.close();
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    private void WriteWaveFileHeader(
            FileOutputStream out, long totalAudioLen,
            long totalDataLen, long longSampleRate, int channels,
            long byteRate) throws IOException {
        byte[] header = new byte[44];

        header[0] = 'R';  // RIFF/WAVE header
        header[1] = 'I';
        header[2] = 'F';
        header[3] = 'F';
        header[4] = (byte) (totalDataLen & 0xff);
        header[5] = (byte) ((totalDataLen >> 8) & 0xff);
        header[6] = (byte) ((totalDataLen >> 16) & 0xff);
        header[7] = (byte) ((totalDataLen >> 24) & 0xff);
        header[8] = 'W';
        header[9] = 'A';
        header[10] = 'V';
        header[11] = 'E';
        header[12] = 'f';  // 'fmt ' chunk
        header[13] = 'm';
        header[14] = 't';
        header[15] = ' ';
        header[16] = 16;  // 4 bytes: size of 'fmt ' chunk
        header[17] = 0;
        header[18] = 0;
        header[19] = 0;
        header[20] = 1;  // format = 1
        header[21] = 0;
        header[22] = (byte) channels;
        header[23] = 0;
        header[24] = (byte) (longSampleRate & 0xff);
        header[25] = (byte) ((longSampleRate >> 8) & 0xff);
        header[26] = (byte) ((longSampleRate >> 16) & 0xff);
        header[27] = (byte) ((longSampleRate >> 24) & 0xff);
        header[28] = (byte) (byteRate & 0xff);
        header[29] = (byte) ((byteRate >> 8) & 0xff);
        header[30] = (byte) ((byteRate >> 16) & 0xff);
        header[31] = (byte) ((byteRate >> 24) & 0xff);
        header[32] = (byte) (2 * 16 / 8);  // block align
        header[33] = 0;
        header[34] = RECORDER_BPP;  // bits per sample
        header[35] = 0;
        header[36] = 'd';
        header[37] = 'a';
        header[38] = 't';
        header[39] = 'a';
        header[40] = (byte) (totalAudioLen & 0xff);
        header[41] = (byte) ((totalAudioLen >> 8) & 0xff);
        header[42] = (byte) ((totalAudioLen >> 16) & 0xff);
        header[43] = (byte) ((totalAudioLen >> 24) & 0xff);

        out.write(header, 0, 44);
```

```
    }
}
```

WavTask()

```
import android.os.AsyncTask;
import android.os.Environment;

import org.json.JSONException;

import java.io.BufferedReader;
import java.io.DataOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;
import java.net.URLEncoder;


public class WavTask extends AsyncTask<Main2Activity, String, String> {

    private static final String URL_POST_WAV = "http://192.168.1.5:1880/PostWav";
    ;
    private Main2Activity window;

    @Override
    /* Executed when the async task progress is updated (ex in the doInBackground
method) */
    protected void onProgressUpdate(String... values) {
        String response = values[0];
        AppLog.logString("onProgressUpdate SUCCESS, consuming result: " + response);
        window.showResults(response);
    }

    @Override
    protected String doInBackground(Main2Activity... windows) {
        window = windows[0];
        String response = null;
        try {
            response = postWav();
        } catch (IOException | JSONException e) {
            AppLog.logString("POST FAILED: " + e.getMessage());
        }
        /* Update the progress the async task - this triggers the onProgressUpdate
method */
        publishProgress(response);
        return response;
    }

    private String postWav() throws IOException, JSONException {
        String targetURL = URL_POST_WAV;

        //kanontas metatroph se fft kai briskontas ta peaks mesw twn 2 klasewn
        //freq_calc kai PeakDeat

        freq_calc calculator = new freq_calc();
        String result = calculator.calc(getWavPath());

        String urlParameters =
                "fName=" + URLEncoder.encode(result, "UTF-8");
```

```
        URL url;
        HttpURLConnection connection = null;
        try {
            //Create connection
            url = new URL(targetURL);
            connection = (HttpURLConnection) url.openConnection();
            connection.setRequestMethod("POST");
            connection.setRequestProperty("Content-Type",
                    "application/x-www-form-urlencoded");

            connection.setRequestProperty("Content-Length", "" +
                    Integer.toString(urlParameters.getBytes().length));
            connection.setRequestProperty("Content-Language", "en-US");

            connection.setUseCaches(false);
            connection.setDoInput(true);
            connection.setDoOutput(true);

            //Send request
            DataOutputStream wr = new DataOutputStream(
                    connection.getOutputStream());
            wr.writeBytes(urlParameters);
            wr.flush();
            wr.close();

            //Get Response
            InputStream is = connection.getInputStream();
            BufferedReader rd = new BufferedReader(new InputStreamReader(is));
            String line;
            StringBuffer response = new StringBuffer();
            while ((line = rd.readLine()) != null) {
                response.append(line);
                response.append('\r');
            }
            rd.close();
            return response.toString();

        } catch (Exception e) {

            e.printStackTrace();
            return null;

        } finally {

            if (connection != null) {
                connection.disconnect();
            }
        }
    }


    private String getWavPath() {
        final String wavpath =
Environment.getExternalStorageDirectory().getAbsolutePath()
                + "/AudioRecorder/" + "realsound" + ".wav";
        return wavpath;
    }

}
```

WavTaskTrain()

```
import android.os.AsyncTask;
import android.os.Environment;

import org.json.JSONException;

import java.io.BufferedReader;
import java.io.DataOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;
import java.net.URLEncoder;


public class WavTaskTrain extends AsyncTask<String, String, String> {

    private static final String URL_POST_WAV = "http://192.168.1.5:1880/PostWavTrain";
    private String TrainSound;

    @Override
    /* Executed when the async task progress is updated (ex in the doInBackground
method) */
    protected void onProgressUpdate(String... values) {
        String response = values[0];
        AppLog.logString("onProgressUpdate SUCCESS, consuming result: " + response);
    }

    @Override
    protected String doInBackground(String... strings) {
        TrainSound = strings[0];
        String response = null;
        try {
            response = postWav();
        } catch (IOException | JSONException e) {
            AppLog.logString("POST FAILED: " + e.getMessage());
        }
        /* Update the progress the async task - this triggers the onProgressUpdate
method */
        publishProgress(response);
        return response;
    }

    private String postWav() throws IOException, JSONException {
        String targetURL = URL_POST_WAV;

        //kanontas metatroph se fft kai briskontas ta peaks mesw twn 2 klasewn
        //freq_calc kai PeakDeat

        freq_calc calculator = new freq_calc();
        String result = calculator.calc(getWavPath());



        String urlParameters ="TrainSound="+ URLEncoder.encode(TrainSound, "UTF-8") +
"&SoundResults=" + URLEncoder.encode(result, "UTF-8");

        URL url;
        HttpURLConnection connection = null;
        try {
```

```
            //Create connection
            url = new URL(targetURL);
            connection = (HttpURLConnection) url.openConnection();
            connection.setRequestMethod("POST");
            connection.setRequestProperty("Content-Type",
                    "application/x-www-form-urlencoded");

            connection.setRequestProperty("Content-Length", "" +
                    Integer.toString(urlParameters.getBytes().length));
            connection.setRequestProperty("Content-Language", "en-US");

            connection.setUseCaches(false);
            connection.setDoInput(true);
            connection.setDoOutput(true);

            //Send request
            DataOutputStream wr = new DataOutputStream(
                    connection.getOutputStream());
            wr.writeBytes(urlParameters);
            wr.flush();
            wr.close();

            //Get Response
            InputStream is = connection.getInputStream();
            BufferedReader rd = new BufferedReader(new InputStreamReader(is));
            String line;
            StringBuffer response = new StringBuffer();
            while ((line = rd.readLine()) != null) {
                response.append(line);
                response.append('\r');
            }
            rd.close();
            return response.toString();

        } catch (Exception e) {

            e.printStackTrace();
            return null;

        } finally {

            if (connection != null) {
                connection.disconnect();
            }
        }
    }

    private String getWavPath() {
        final String wavpath =
Environment.getExternalStorageDirectory().getAbsolutePath()
                + "/AudioRecorder/" + "trainsound" + ".wav";
        return wavpath;
    }

}
```

PeakDet()

```
import java.util.ArrayList;
import java.util.List;

public class PeakDet {
```

```
    /*public static void main(String[] args) {
        long[] vector = new long[] {
            0, 1, 2, 3, 4, 5, 1, 2, 3, 4, 8, 7, 6, 5, 4, 3, 2, 1
        };
        List<Peak> peaks = new PeakDet().peakdet(vector, 1);

        for (Peak peak : peaks) {
            System.out.println(peak);
        }
    }*/

    List<Peak> peakdet(double[] vector, double triggerDelta) {
        return peakdet(vector, 0, vector.length, triggerDelta);
    }

    /*
     !PEAKDET Detect peaks in a vector
     !
     !        call PEAKDET(MAXTAB, MINTAB, N, V, DELTA) finds the local
     !        maxima and minima ("peaks") in the vector V of size N.
     !        MAXTAB and MINTAB consists of two columns. Column 1
     !        contains indices in V, and column 2 the found values.
     !
     !        call PEAKDET(MAXTAB, MINTAB, N, V, DELTA, X) replaces the
     !        indices in MAXTAB and MINTAB with the corresponding X-values.
     !
     !        A point is considered a maximum peak if it has the maximal
     !        value, and was preceded (to the left) by a value lower by
     !        DELTA.
     !
     ! Eli Billauer, 3.4.05 (http://billauer.co.il)
     ! Translated into Fortran by Brian McNoldy
(http://andrew.rsmas.miami.edu/bmcnoldy)
     ! This function is released to the public domain; Any use is allowed.*/

    List<Peak> peakdet(double[] vector, int offset, int length, double triggerDelta) {
        double mn = Double.POSITIVE_INFINITY;
        double mx = Double.NEGATIVE_INFINITY;
        double mnpos = Double.NaN;
        double mxpos = Double.NaN;
        int lookformax = 1;

        List<Peak> maxtab_tmp = new ArrayList<>();
        //List<Valley> mintab_tmp = new ArrayList<>();

        for (int i = offset; i < length; i++) {
            double a = vector[i];
            if (a > mx) {
                mx = a;
                mxpos = vector[i];
            }
            if (a < mn) {
                mn = a;
                mnpos = vector[i];
            }
            if (lookformax == 1) {
                if (a < mx - triggerDelta) {
                    maxtab_tmp.add(new Peak(mxpos, i));
                    mn = a;
                    mnpos = vector[i];
                    lookformax = 0;
                }
```

```
            } else {
                if (a > mn + triggerDelta) {
                    //mintab_tmp.add(new Valley(mnpos, i));
                    mx = a;
                    mxpos = vector[i];
                    lookformax = 1;
                }
            }
        }

        return maxtab_tmp;
    }

    static class Peak {

        public final double height;
        public final int index;

        Peak(double height, int index) {
            this.height = height;
            this.index = index;
        }

        @Override
        public String toString() {
            return "Peak{" + "height=" + height + ", index=" + index + '}';
        }

    }

    static class Valley {

        public final double height;
        public final int index;

        private Valley(double height, int index) {
            this.height = height;
            this.index = index;
        }

        @Override
        public String toString() {
            return "Valley{" + "height=" + height + ", index=" + index + '}';
        }

    }
}
```

freq_calc()

```
import com.example.corrina.thesis_app.PeakDet.Peak;
import com.musicg.dsp.FastFourierTransform;
import com.musicg.wave.Wave;

import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Arrays;
import java.util.Collections;
import java.util.Comparator;
import java.util.List;
```

```
public class freq_calc {
    public static void write(String filename, double[] x) throws IOException {
        BufferedWriter outputWriter = null;
        outputWriter = new BufferedWriter(new FileWriter(filename));
        outputWriter.write(Arrays.toString(x));
        outputWriter.flush();
        outputWriter.close();
    }

    public String calc(String filename) {
        //Open file and create FFT transformer
        Wave file = new Wave(filename);
        FastFourierTransform FftEngine = new FastFourierTransform();

        //Get sound data array, trans->normalize around -1.0
        double[] signal = trans((file.getNormalizedAmplitudes()));

        //FFT takes array with length = 2^N
        //Make signal y = [signal signal ... signal(0:...)],
        //cyclical repetition of signal with length=2^N
        int len = (int) Math.pow(2, Math.ceil(Math.log(signal.length * 1.0) /
Math.log(2.0)));
        double[] y = new double[len];
        for (int i = 0; i < len; i++) y[i] = signal[i % signal.length];

        //Debugging
        //log(Arrays.toString(y));
        //System.out.println(y.length+" "+glen);

        //Get FFT abs heights of y
        //Normalization around -1.0 creates a zero frequency. Remove it
        double[] fft = FftEngine.getMagnitudes(y);
        fft[0] = 0;

        //Create frequency axis
        double[] axis = new double[fft.length];
        len = fft.length;
        int sampleRate = file.getWaveHeader().getSampleRate();
        for (int i = 0; i < fft.length; i++) axis[i] = (i * 1.0 / len) * (sampleRate /
2);

        //Debugging
        //Plot.show(axis, fft);

        //Try to get peaks until you have at least 5
        double delta = getMaxValue(fft);
        double step = delta / 100;
        boolean silence = false;
        PeakDet detector = new PeakDet();
        List<Peak> peaks;
        do {
            peaks = detector.peakdet(fft, delta);
            delta = delta - step;
            //System.out.println(delta+" "+peaks.size());
            if (delta < 0) {
                silence = true;
                break;
            }
        } while (peaks.size() < 5);
        if (silence) {
            return Silence(peaks, axis);
```

```
    }

    //Sort peaks by heights
    Collections.sort(peaks, new Comparator<Peak>() {
        @Override
        public int compare(final Peak object1, final Peak object2) {
            return (int) Math.signum(object2.height - object1.height);
        }
    });

    //Get frequencies and heights of the heighest 5 peaks
    /*System.out.println("Peaks: "+peaks.size());
    for (int i=0; i<Math.min(20, peaks.size()); i++){
     System.out.println(axis[peaks.get(i).index]+" Hz, "+(peaks.get(i).height));
  }*/
    return getJsonString(peaks.toArray(new Peak[peaks.size()]), axis);
}


private String Silence(List<Peak> curPeaks, double[] axis) {
    Peak[] peaks = {new Peak(0, 0), new Peak(0, 1),
            new Peak(0, 2), new Peak(0, 3),
            new Peak(0, 4)};
    int len = curPeaks.size();
    for (int i = len; i < 5; i++) curPeaks.add(peaks[i - len]);
    return getJsonString(curPeaks.toArray(new Peak[5]), axis);
}

private String getJsonString(Peak[] peaks, double[] axis) {
    String msg = "{%payload%:{%message_type%:%1%,";
    msg += "%problem_attributes%:%freq1#freq2#freq3#freq4#freq5%,";
    msg += "%problem_values%:%";
    double max = peaks[0].height;
    if (max == 0) max = 1;
    for (int i = 0; i < 4; i++) msg += axis[peaks[i].index] + "#";
    msg += axis[peaks[4].index];
    msg += "%,%solution_attributes%:%hasSoundof%";
    msg += ",%forSharing%:%true%";
    msg += ",%weights%:%";
    for (int i = 0; i < 4; i++) msg += (peaks[i].height / (2 * max)) + "#";
    msg += (peaks[4].height / (2 * max));
    msg += "%,%threshold%:%0.6%}}";
    return msg;
}

private void log(String txt) {
    try {
        PrintWriter writer = new PrintWriter("logT.txt", "UTF-8");
        writer.println(txt);
        writer.close();
    } catch (IOException e) {
        // do something
    }
}

// getting the maximum value
private double getMaxValue(double[] array) {
    double maxValue = array[0];
    for (int i = 1; i < array.length; i++) {
        if (array[i] > maxValue) {
            maxValue = array[i];
        }
    }
```

```
        return maxValue;
    }

    private double[] get_first_channel(double[] input) {
        double[] output = new double[input.length / 4];
        for (int i = 0; i < output.length; i++) output[i] = input[4 * i];
        return output;
    }

    private double[] trans(double[] input) {
        double[] output = new double[input.length];
        for (int i = 0; i < output.length; i++) {
            output[i] = (input[i] * Math.pow(2, -7.0)) - 1;
            //output[i]= input[i];
        }
        return output;
    }

}
```

CustomOnItemSelectedListener()

```
import android.view.View;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemSelectedListener;
import android.widget.Toast;

public class CustomOnItemSelectedListener implements OnItemSelectedListener {

    public void onItemSelected(AdapterView<?> parent, View view, int pos, long id) {
        Toast.makeText(parent.getContext(),
                "OnItemSelectedListener : " +
parent.getItemAtPosition(pos).toString(),
                Toast.LENGTH_SHORT).show();
    }

    @Override
    public void onNothingSelected(AdapterView<?> arg0) {
        // TODO Auto-generated method stub
    }

}
```

AppLog()

```
import android.util.Log;

public class AppLog {
    private static final String APP_TAG = "AudioRecorder";

    public static int logString(String message) {
        return Log.i(APP_TAG, message);
    }
}
```

8.2 Κώδικας για Front-End

Activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:background="@drawable/backround"

    tools:context="com.example.corrina.thesis_app.MainActivity">

    <Button
        android:id="@+id/start_btn"

        android:text="Start"
        android:textColor="#F7FEFF"
        android:textSize="31sp"
        android:onClick="sendMessage1"
        android:layout_height="60dp"
        android:background="@drawable/buttonshape"
        android:shadowColor="#FFFFFF"
        android:shadowDx="0"
        android:shadowDy="0"
        android:shadowRadius="5"
        android:layout_width="170dp"
        android:layout_marginTop="75dp"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true" />

    <Button
        android:id="@+id/configure_btn"

        android:text="Train"
        android:textColor="#F7FEFF"
        android:textSize="30sp"
        android:layout_height="60dp"
        android:background="@drawable/buttonshape"
        android:shadowColor="#FFFFFF"
        android:shadowDx="0"
        android:shadowDy="0"
        android:shadowRadius="5"
        android:layout_width="170dp"
        android:onClick="sendMessage2"
        android:layout_centerVertical="true"
        android:layout_alignLeft="@+id/start_btn"
        android:layout_alignStart="@+id/start_btn" />

</RelativeLayout>
```

Activity_main2.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:layout_marginBottom="5sp"
    android:layout_marginLeft="2sp"
    android:layout_marginRight="5sp"
    android:layout_marginTop="0sp"     >

    <LinearLayout
        android:layout_height="0dp"
        android:layout_width="match_parent"
        android:layout_weight="1"
        android:orientation="horizontal">

        <ImageButton
            android:layout_height="match_parent"
            android:layout_width="0dp"
            android:layout_weight="1"
            android:id="@+id/phonebtn"
            android:scaleType="fitCenter"
            android:src="@drawable/phone"
            android:layout_marginBottom="5sp"
            android:layout_marginLeft="2sp"
            android:layout_marginRight="5sp"
            android:layout_marginTop="0sp"
            />

        <ImageButton
            android:layout_height="match_parent"
            android:layout_width="0dp"
            android:layout_weight="1"
            android:src="@drawable/firealarm"
            android:id="@+id/alarmbtn"
            android:scaleType="fitCenter"
            android:layout_marginBottom="5sp"
            android:layout_marginLeft="2sp"
            android:layout_marginRight="5sp"
            android:layout_marginTop="0sp"
            />


    </LinearLayout>
    <LinearLayout
        android:layout_height="0dp"
        android:layout_width="match_parent"
        android:layout_weight="1"
        android:orientation="horizontal">

        <ImageButton
            android:layout_height="match_parent"
            android:layout_width="0dp"
            android:layout_weight="1"
            android:src="@drawable/babycrying"
            android:id="@+id/babybtn"
            android:scaleType="fitCenter"
            android:layout_marginBottom="5sp"
            android:layout_marginLeft="2sp"
            android:layout_marginRight="5sp"
```

```
            android:layout_marginTop="0sp"
            />
        <ImageButton
            android:src="@drawable/doorbell"
            android:id="@+id/doorbtn"
            android:layout_height="match_parent"
            android:layout_width="0dp"
            android:layout_weight="1"
            android:scaleType="fitCenter"
            android:layout_marginBottom="5sp"
            android:layout_marginLeft="2sp"
            android:layout_marginRight="5sp"
            android:layout_marginTop="0sp"
            />
    </LinearLayout>

</LinearLayout>
```

Activity_main3.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_main3"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.example.corrina.thesis_app.Main3Activity">

    <Button
        android:text="Stop"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/btnStop"
        android:layout_alignParentBottom="true"
        android:layout_alignLeft="@+id/btnStart"
        android:layout_alignStart="@+id/btnStart"
        android:layout_marginBottom="68dp" />

    <TextView
        android:text="Please, select one sound from the dropdown menu."
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/textView"
        android:layout_alignParentTop="true"
        android:layout_alignParentRight="true"
        android:layout_alignParentEnd="true"
        android:layout_marginTop="73dp" />

    <Spinner
        android:id="@+id/spinner1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:entries="@array/sound_arrays"
        android:prompt="@string/sound_prompt"
        android:layout_alignTop="@+id/textView"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:layout_marginTop="43dp" />
```

```
    <Button
        android:id="@+id/btnSubmit"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Submit"
        android:layout_below="@+id/textView"
        android:layout_alignLeft="@+id/btnStart"
        android:layout_alignStart="@+id/btnStart"
        android:layout_marginTop="49dp" />

    <Button
        android:text="Start"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/btnStart"
        android:layout_marginBottom="31dp"
        android:layout_above="@+id/btnStop"
        android:layout_centerHorizontal="true" />

</RelativeLayout>
```