ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

Χειρισμός Συσκευών Μέσω Smartphone

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Ιωάννης Α. Λιάκος

Επιβλέπων : Γεώργιος Καμπουράκης
Αν. Καθηγητής Ε.Μ.Π.

Αθήνα, Μάρτιος 2015

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

Χειρισμός Συσκευών Μέσω Smartphone

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Ιωάννης Α. Λιάκος

Επιβλέπων : Γεώργιος Καμπουράκης
Αν. Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 16η Μαρτίου 2015

.....	.....	.....
Ελ. Καγιάφας	Γ. Καμπουράκης	Β. Λούμος
Ομ. Καθηγητής Ε.Μ.Π.	Αν. Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.

Αθήνα, Μάρτιος 2015

.....
Ιωάννης Α. Λιάκος
Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Πρόλογος

Ανέκαθεν ο άνθρωπος έψαχνε τρόπους να στείλει μηνύματα σε μεγάλες αποστάσεις, να τηλεπικοινωνήσει. Το βασικότερο κοινό που έχει η επικοινωνία με σήματα καπνού με αυτήν που γίνεται μέσω του δικτύου κινητής επικοινωνίας, είναι η διευκόλυνση της αποστολής μηνύματος από έναν δέκτη σε έναν παραλήπτη. Εδώ και πολλές δεκαετίες, ο άνθρωπος έχει καταφέρει να τηλεπικοινωνήσει σε παραλήπτες που αποτελούν ηλεκτρονικές συσκευές, με πρωτοπόρο τον Νικολάι Τέσλα.

Ένας από τους πιο διαδεδομένους τρόπους τηλεπικοινωνίας με μια ηλεκτρονική συσκευή, είναι αυτός, που καθένας μας χρησιμοποιεί όταν θέλει να ανοίξει την τηλεόρασή του ή το κλιματιστικό του, ο τηλεχειρισμός με τηλεχειριστήριο υπερύθρων.

Με τη εξέλιξη της τεχνολογίας, έχει διαδοθεί πολύ, η χρήση "έξυπνων κινητών", με λειτουργικά συστήματα που επιτρέπουν τη δημιουργία εφαρμογών και την χρήση των συσκευών αυτών ως κάτι παραπάνω από ένα απλό κινητό τηλέφωνο. Ένα από αυτά τα λειτουργικά συστήματα είναι το λειτουργικό της Google, Android.

Καθώς τα κινητά εξελίσσονται, εξελίσσονται και οι τηλεοράσεις σε "έξυπνες τηλεοράσεις" με την δυνατότητα τηλεχειρισμού τους μέσω τοπικού δικτύου Wi-Fi. Παρόλα αυτά, το τηλεχειριστήριο υπερύθρων αποτελεί το βασικό εργαλείο τηλεχειρισμού των τηλεοράσεων.

Η παρούσα εργασία, στοχεύει στον τηλεχειρισμό συσκευών που διαθέτουν τηλεχειριστήριο υπερύθρων, όπως μια τηλεόραση, ένα ηχοσύστημα ή κάποιο κλιματιστικό, μέσω ενός "έξυπνου κινητού". Καθώς τα περισσότερα κινητά δε διαθέτουν θύρα υπερύθρων για τον τηλεχειρισμό αυτό, αλλά διαθέτουν την δυνατότητα επικοινωνίας Bluetooth, για τον σκοπό της εργασίας, δημιουργήθηκε μια συσκευή διεπαφής, μεταξύ του κινητού και της συσκευής που ο χρήστης θέλει να χειριστεί. Ο χρήστης επικοινωνεί με την συσκευή μέσα από μια εφαρμογή εγκατεστημένη σε κινητό με λειτουργικό Android.

Στο σημείο αυτό θα ήθελα να ευχαριστήσω τον καθηγητή του Εθνικού Μετσόβιου Πολυτεχνείου, κ. Καμπουράκη Γεώργιο, για την καθοδήγηση και την υποστήριξη του σε όλα τα στάδια της εργασίας, χωρίς την βοήθεια του οποίου, δε θα είχε εκπονηθεί. Θα ήθελα επίσης να ευχαριστήσω όλους τους φίλους και συγγενείς που με την υποστήριξη τους με βοήθησαν να ολοκληρώσω την εργασία.

Περιεχόμενα

Μέρος Α : Θεωρία

Μέρος Β : Υλοποίηση

1. Περιγραφή και Σκοπός Εργασίας

Η παρούσα εργασία στοχεύει στη γεφύρωση νέων τεχνολογιών με παλιών, επιτρέποντας τον τηλεχειρισμό συσκευών που διαθέτουν θύρα επικοινωνίας υπερύθρων, μέσω έξυπνου κινητού. Για τον σκοπό της εργασίας χρησιμοποιήθηκε κινητό τηλέφωνο με λειτουργικό Android, χωρίς θύρα υπερύθρων. Τελικός σκοπός ήταν ο τηλεχειρισμός τηλεόρασης μέσω υπερύθρων. Για την επικοινωνία των δύο συσκευών, δημιουργήθηκε μια μικρή φορητή συσκευή, με δυνατότητα επικοινωνίας Bluetooth, για την επικοινωνία με το κινητό, αλλά και την δυνατότητα παραγωγής σήματος υπερύθρων για την επικοινωνία με την τηλεόραση. Η συσκευή λειτουργεί ως μεσολαβητής ανάμεσα στο κινητό και την τηλεόραση, μεταφράζοντας τις εντολές που λαμβάνει μέσω του Bluetooth σε εντολές IR.

Ανάμεσα στους στόχους της εργασίας, είναι να μπορεί η συσκευή να χειριστεί όσο γίνεται μεγαλύτερο πλήθος τηλεοράσεων, ψηφιακών δεκτών και στερεοφωνικών. Η συσκευή επίσης σχεδιάστηκε ώστε με μια απλή και γρήγορη αναβάθμιση στην εφαρμογή του κινητού να μπορεί να υποστηρίξει καινούριες συσκευές.

Μια από τις δυσκολίες που εμφανίστηκαν, ήταν ότι δεν υπάρχει ένα μόνο πρωτόκολλο επικοινωνίας υπερύθρων για όλες τις συσκευές, αλλά κάθε κατασκευαστής χρησιμοποιεί το δικό του πρωτόκολλο. Αυτή η δυσκολία, μαζί με άλλες που αναλύονται παρακάτω, οδήγησαν στην κατασκευή μιας συσκευής με ένα Bluetooth σε σειριακό μόντεμ, μαζί με ένα μικροεπεξεργαστή AVR που λαμβάνει τα δεδομένα σειριακά από το μόντεμ και παράγει το απαραίτητο IR σήμα. Για τον χειρισμό της συσκευής, αναπτύχθηκε μια εφαρμογή για λειτουργικό σύστημα Android, η οποία περιέχει και τις απαραίτητες πληροφορίες για το πρωτόκολλο IR επικοινωνίας που χρησιμοποιεί ο δέκτης. Με τον τρόπο αυτό, αναβαθμίζοντας την εφαρμογή, μπορούν να προστεθούν και άλλα πρωτόκολλα υπερύθρων.

Η εργασία χωρίζεται σε δύο μέρη. Στο πρώτο μέρος αναλύονται τα εργαλεία και οι πλατφόρμες που χρησιμοποιήθηκαν, καθώς και οι μορφές ψηφιακής επικοινωνίας υπερύθρων και Bluetooth. Γίνεται μια σύντομη περιγραφή του λειτουργικού συστήματος Android και των διαφόρων εκδόσεών του. Ακολουθεί ανάλυση της πλατφόρμας Eclipse, μαζί με τα απαραίτητα εργαλεία για την ανάπτυξη της εφαρμογής. Το πρώτο μέρος κλείνει με την περιγραφή της πλατφόρμας Arduino και το περιβάλλον προγραμματισμού του.

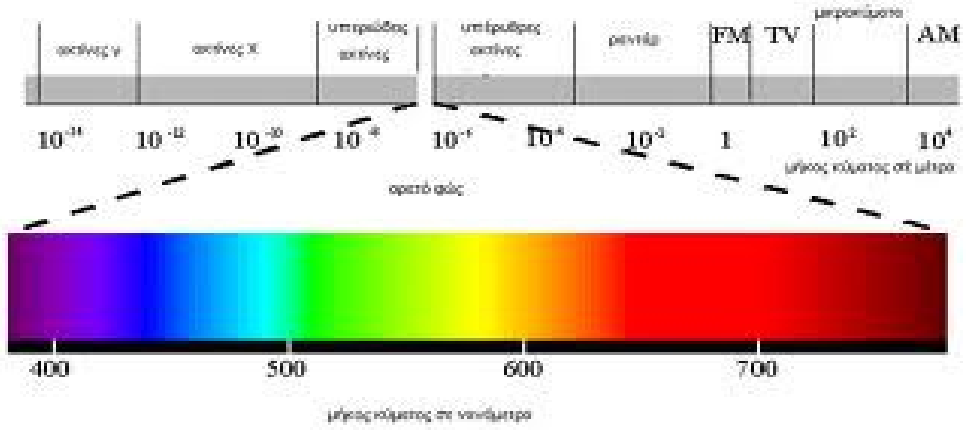
Στο δεύτερο μέρος, περιγράφονται τα στάδια σχεδιασμού της συσκευής, μαζί με τις δυσκολίες που αντιμετωπίστηκαν. Αναλύονται τα εργαλεία που χρησιμοποιήθηκαν για τον έλεγχο της συσκευής κατά τη δημιουργία της. Ακολουθεί η περιγραφή του προγράμματος που "τρέχει" στη συσκευή, αλλά και η εφαρμογή που αναπτύχθηκε για τον έλεγχό της.

Τέλος, αναλύονται τα συμπεράσματα από τη δημιουργία της συσκευής και προτείνονται λειτουργίες που μπορούν να προστεθούν για την επέκτασή της.

2. Υπέρυθρη Επικοινωνία

2.1 Εισαγωγή

Υπέρυθρη ακτινοβολία χαρακτηρίζεται η ακτινοβολία με μήκος κύματος μεταξύ 750nm και 1mm. Ονομάζεται υπέρυθρη γιατί έχει μήκος κύματος μεγαλύτερο από αυτό του ερυθρού μέρους του ορατού φάσματος ξεπερνώντας το οποίο γίνεται αόρατη στο γυμνό μάτι. Πέρα από τις τηλεπικοινωνίες η περιοχή αυτή ακτινοβολίας χρησιμοποιείται στην αστρονομία, τη μετεωρολογία και άλλους επιστημονικούς τομείς καθώς χρησιμοποιείται στη θερμική απεικόνιση αλλά και την θέρμανση σωμάτων. Στον χώρο των τηλεπικοινωνιών συναντάμε την υπέρυθρη ακτινοβολία ως τον φορέα σε συστήματα οπτικών ινών με μήκη κύματος 1330nm – 1550nm αλλά και συστήματα μετάδοσης ελεύθερου χώρου όπου απαιτείται η οπτική επαφή μεταξύ πομπού και δέκτη. Στην τελευταία κατηγορία ανήκουν συστήματα τηλεχειρισμού όπως αυτά που χρησιμοποιούνται στις περισσότερες τηλεοράσεις, στερεοφωνικά και κλιματιστικά αλλά και συστήματα μεταφοράς δεδομένων όπως για παράδειγμα κινητών τηλεφώνων για την επικοινωνία τους με υπολογιστή και ασύρματων ηχοσυστημάτων.



2.2 Πομπός

Στα τηλεχειριστήρια υπερύθρων η πηγή της ακτινοβολίας είναι συνήθως μια δίοδος φωτοεκπομπής (Ir LED) η χαμηλή κατανάλωση της οποίας την καθιστά ιδανική για φορητές συσκευές. Όπως οι κοινές δίοδοι, υλοποιείται από την ένωση ημιαγωγών τύπου n και p. Η ακτινοβολία επιτυγχάνεται κατά την επανασύνδεση των φορέων μειονότητας στην ένωση των δύο τύπων ημιαγωγών. Το ηλεκτρόνιο που ξεπερνά το ενεργειακό διάκενο και πέφτει σε χαμηλότερη ενεργειακή στάθμη αποδίδει στο περιβάλλον ενέργεια ίση με τη διαφορά της τελικής του ενέργειας από την αρχική εκπέμποντας ακτινοβολία συχνότητας που εξαρτάται από το διάκενο. Για την κατασκευή υπερύθρων διόδων φωτοεκπομπής χρησιμοποιείται Αρσενικούχο Γάλλιο (GaAs) λόγω του άμεσου ενεργειακού διάκενου και της ακτινοβολίας στη περιοχή των υπερύθρων.

Στην ψηφιακή επικοινωνία ο πομπός ανοιγοκλείνει το Ir LED με συχνότητα συνήθως (στα περισσότερα πρωτόκολλα) μεταξύ 36 και 40 kHz. Οι συχνότητες αυτές δεν συναντώνται συχνά στη φύση και έτσι στην περιοχή αυτή υπάρχει λιγότερος θόρυβος. Τα δύο ψηφία, το λογικό μηδέν και το λογικό ένα, αντιστοιχίζονται σε διάρκεια κατά την οποία υπάρχουν παλμοί και διάρκεια απουσίας παλμών. Οι εναλλαγές και η διάρκειά τους στην πλευρά του δέκτη αντιστοιχίζονται πάλι σε ψηφία. Ο τρόπος αντιστοίχισης ορίζεται από το πρωτόκολλο επικοινωνίας που χρησιμοποιείται, μαζί με τους χαρακτήρες έναρξης, λήξης και τον χαρακτήρα επανάληψης.

2.3 Ο Δέκτης

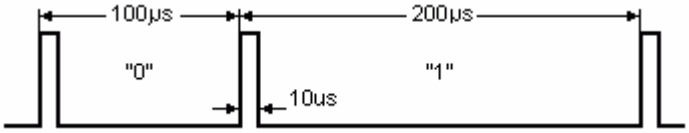
Ο δέκτης αναλαμβάνει να αποδιαμορφώσει το σήμα που λαμβάνει και να το αντιστοιχίσει στα ψηφία 0 και 1, εξάγοντας έτσι το ψηφιακό μήνυμα που έχει σταλεί. Ένα από τα πιο κοινά ολοκληρωμένα που συναντάμε στην πράξη γι αυτό τον σκοπό είναι το TSOP48XX, όπου το XX μπορεί να είναι 30, 33, 36, 38, 40 ή 56 και αντιστοιχεί σε kHz. Ανάλογα με την συχνότητα που ορίζει το πρωτόκολλο επιλέγεται ο κατάλληλος δέκτης. Το ολοκληρωμένο βασίζεται σε έναν φωτο-ανιχνευτή, που στη συνέχεια τροφοδοτεί έναν αυτόματο ρυθμιστή κέρδους (Automatic Gain Control, AGC). Το σήμα φιλτράρεται από ένα ζωνοπερατό φίλτρο και αποδιαμορφώνεται σε υψηλή στάθμη, όταν υπάρχει σήμα της κεντρικής συχνότητας του ζωνοπερατού φίλτρου και σε χαμηλή στάθμη, όταν δεν υπάρχει σήμα της συχνότητας αυτής. Το σήμα αυτό, ενισχύεται για να οδηγήσει το επόμενο στάδιο του δέκτη.

Στο επόμενο στάδιο ο δέκτης αντιστοιχίζει την έξοδο του ολοκληρωμένου TSOP48XX στο λογικό 0 και 1, ανάλογα με το πρωτόκολλο επικοινωνίας.

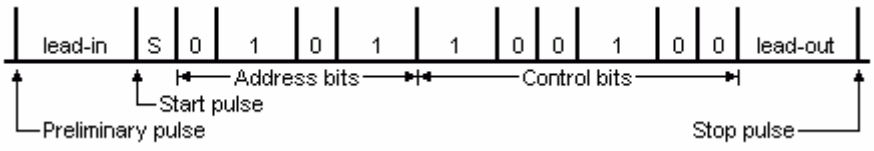
2.4 Πρωτόκολλα IR

- Το πρωτόκολλο της ITT

Το ITT είναι το πρώτο πρωτόκολλο υπέρυθρης επικοινωνίας και κατασκευάστηκε από την εταιρία κατασκευής τηλεοράσεων ITT. Στο πρωτόκολλο αυτό το σήμα πληροφορίας δεν διαμορφώνεται από κάποιο φέρον. Η εντολή αποτελείται από 14 παλμούς 10 usec ο καθένας. Ο χρόνος από την άνοδο ενός παλμού μέχρι την άνοδο του επόμενου χρησιμοποιείται για να κωδικοποιήσει τα ψηφία. Συγκεκριμένα, αν η χρονική απόσταση ανάμεσα σε δύο παλμούς είναι 100 usec το λογικό μηδέν είναι το απεσταλμένο ψηφίο ενώ για χρονική απόσταση 200 usec το ψηφίο είναι το ένα, όπως φαίνεται και στο παρακάτω σχήμα.



Για την έναρξη μιας αποστολής θα πρέπει να αποσταλεί ο "προεισαγωγικός" παλμός (preliminary pulse) ο οποίος ακολουθείται από παύση 300 usec. Το ύψος του παλμού χρησιμοποιείται από τον δέκτη για την ρύθμιση του κέρδους του ενισχυτή (AGC). Στη συνέχεια, αποστέλλεται ένα λογικό μηδέν ως ψηφίο έναρξης (start bit). Τα 4 επόμενα ψηφία αποτελούν την διεύθυνση του παραλήπτη του μηνύματος, ενώ τα επόμενα 6 αποτελούν την εντολή. Τέλος, μετά τον τελευταίο παλμό υπάρχει μια παύση 300 usec και η ακολουθία ολοκληρώνεται με τον τελικό παλμό (stop pulse).



Οι πληροφορίες αποστέλλονται με περισσότερο σημαντικό ψηφίο πρώτο και στο πεδίο της διεύθυνσης αλλά και στο πεδίο της εντολής. Με τα 4 ψηφία του πεδίου διεύθυνσης μπορούν να διευθυνσιοδοτηθούν (2^4 =) 16 συσκευές. Οι εντολές που μπορούν να αντιστοιχιστούν
στα 6 ψηφία στο πεδίο εντολής είναι (2^6=) 64. για να οριοθετηθεί χρονικά το τελευταίο ψηφίο χρειάζεται ένας ακραίος παλμός (trailing pulse) που ακολουθείται από 300 usec παύσης και τον

τελικό παλμό.

Από την πλευρά του δέκτη υπάρχει η δυνατότητα να ελεγχθεί ο τελικός παλμός ότι διαρκεί 3 φορές περισσότερο από το ψηφίο έναρξης (λογικό 0). Επίσης, θα πρέπει ο δέκτης να παρέχει περιθώριο σφάλματος ±20% για την διάρκεια των παλμών συγκριτικά με το ψηφίο έναρξης. Τέλος ο δέκτης μετά την λήψη παλμού δεν πρέπει να περιμένει περισσότερο από 360 usec αφού είναι πιθανό να υπάρχει σφάλμα στην επικοινωνία και πρέπει να είναι σε θέση να δεχτεί εκ νέου πακέτα.

Παρότι από το πεδίο διευθύνσεων μπορούν να διευθυνσιοδοτηθούν 16 συσκευές μόνο οι 8 πρώτες (0-7) είναι έγκυρες σύμφωνα με το πρωτόκολλο. Οι υπόλοιπες χρησιμοποιούνται σε περίπτωση που κάποιο πλήκτρο του χειριστηρίου κρατηθεί παρατεταμένα. Στην περίπτωση αυτή μετά την πρώτη αποστολή, που γίνεται κανονικά, το μήνυμα επαναλαμβάνεται με όλα τα ψηφία στο πεδίο της διεύθυνσης αντεστραμμένα. Με αυτόν τον τρόπο ο δέκτης αντιλαμβάνεται ότι το πλήκτρο παραμένει πατημένο. Το μήνυμα αποστέλλεται κάθε 130 msec όσο το πλήκτρο είναι πατημένο.

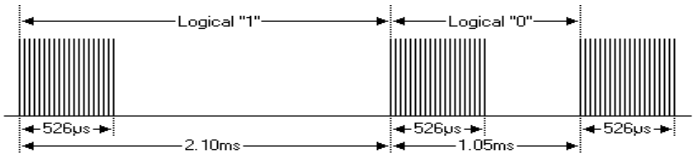
Πίνακας εντολών
Στον παρακάτω πίνακα αντιστοιχούνται οι βασικότερες εντολές που χρησιμοποιούνται στην τηλεόραση με τον αριθμό εντολής τους. Η διεύθυνση των τηλεοράσεων ITT είναι το 1.

Εντολή	Λειτουργία
26	0
17	1
18	2
19	3
20	4
21	5
22	6
23	7
24	8
25	9
8	Program +
9	Program -
47	Volume +
48	Volume -
7	Mute
2	Stand by
45	Saturation +
46	Saturation -
43	Brightness

	+
44	Brightness -

- Το πρωτόκολλο της JVC

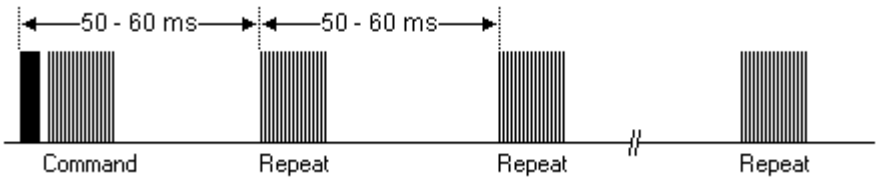
Η JVC στο πρωτόκολλό της χρησιμοποιεί διαμόρφωση του σήματος πληροφορίας γύρω από συχνότητα φέροντος 38 kHz, με προτεινόμενο duty cycle 25% (¼) ή 33% (1/3). Η κωδικοποίηση που χρησιμοποιεί είναι κωδικοποίηση απόστασης παλμών (pulse distance encoding). Πιο συγκεκριμένα το λογικό ένα αποτελείται από παλμό 526 usec και ακολουθεί παύση ώστε το ψηφίο να έχει διάρκεια 2,1 msec ενώ στο λογικό μηδέν ο παλμός είναι ο ίδιος αλλά η διάρκεια του ψηφίου είναι 1,05 msec.



Το ψηφίο έναρξης αποτελείται από παλμό διάρκειας 8,4 msec και παύση 4,2 msec που αντιστοιχούν σε περίπου 320 και 160 περιόδους της φέρουσας συχνότητας.

Μετά τον χαρακτήρα έναρξης ακολουθεί η διεύθυνση της συσκευής που παραλαμβάνει το μήνυμα. Αποτελείται από 8 bit και αποστέλλεται με το λιγότερο σημαντικό ψηφίο πρώτα (LSB). Συνολικά από το πρωτόκολλο μπορούν να διευθυνσιοδοτηθούν (2^8 =) 256 συσκευές. Τα επόμενα 8 bit αποτελούν την εντολή και όπως και η διεύθυνση αποστέλλονται με το λιγότερο σημαντικό ψηφίο πρώτα και υποστηρίζονται 256 διαφορετικές εντολές. Για να οριοθετηθεί η παύση του τελευταίου bit απαιτείται ένας παλμός (trailing pulse). Είναι και αυτός 526 usec και σηματοδοτεί το τέλος της αποστολής.

Όταν ο χειριστής κρατά πατημένο το πλήκτρο επαναλαμβάνεται το ίδιο μήνυμα αλλά χωρίς τον χαρακτήρα έναρξης. Με αυτόν τον τρόπο η συσκευή αντιλαμβάνεται ότι το πλήκτρο παρέμεινε πατημένο και δεν πατήθηκε ξανά. Η επανεκπομπή επαναλαμβάνεται κάθε 50 – 60 msec μέχρι να αφεθεί το πλήκτρο.

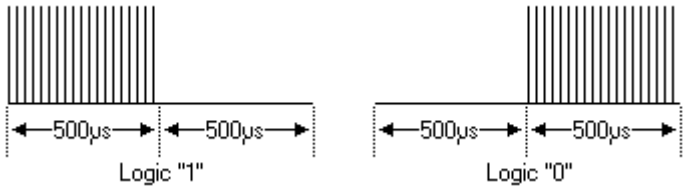


- Το πρωτόκολλο της Mitsubishi

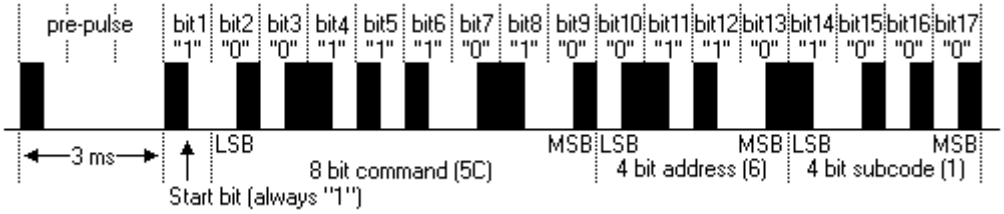
Το πρωτόκολλο αυτό δημιουργήθηκε από την Mitsubishi αλλά χρησιμοποιήθηκε και από την Γαλλική εταιρία Xcom σε δορυφορικούς δέκτες με την διαφορά ότι η πρώτη χρησιμοποιεί φέρον στα 40 kHz ενώ η δεύτερη στα 38 kHz. Το πρωτόκολλο μοιάζει πολύ με αυτό της JVC απλά όταν το πλήκτρο παραμένει κρατημένο το μήνυμα επανεκπέμπεται μαζί με τον χαρακτήρα έναρξης.

- Το πρωτόκολλο NRC17 της Nokia

Το πρωτόκολλο αυτό πήρε το όνομα του από τα αρχικά των λέξεων Nokia Remote Control και το γεγονός ότι σε κάθε πακέτο αποστέλλονται 17 bit. Η συχνότητα του φέροντος είναι 38 kHz ενώ τα δύο ψηφία κωδικοποιούνται με τρόπο ώστε να έχουν ίση διάρκεια 1 msec. Το λογικό ένα αποτελείται από παλμό 500 usec και παύση 500 usec ενώ το λογικό μηδέν από παύση 500 usec ακολουθούμενη από παλμό 500 usec. Με τον τρόπο αυτό τα μηνύματα έχουν σταθερή διάρκεια, σε αντίθεση με τα προηγούμενα πρωτόκολλα, ίση με 20 msec.



Ξεκινώντας την αποστολή ο πομπός πρέπει να στείλει έναν παλμό διάρκειας 500 usec και να περιμένει για 2.5 msec προτού στείλει το ψηφίο εκκίνησης (start bit) που είναι το λογικό 1. Με αυτόν τον τρόπο ο δέκτης συγχρονίζεται και ρυθμίζεται το αυτόματο κέρδος της ενισχυτικής του βαθμίδας. Τα επόμενα 8 bit είναι η εντολή με πρώτο το λιγότερο σημαντικό ψηφίο και ακολουθούν 4 bit διεύθυνσης τα επόμενα 4 bit που αποστέλλονται μπορούν να χρησιμοποιηθούν ως επέκταση της διεύθυνσης.



Κάθε φορά που πατιέται ένα κουμπί αποστέλλεται ένα μήνυμα έναρξης με εντολή 11111110 και διεύθυνση μαζί με τα 4 bit επέκτασης αυτής 11111111. Μετά από 40 msec αποστέλλεται η εντολή, η οποία επαναλαμβάνεται κάθε 100 msec. Όταν αφεθεί το κουμπί αποστέλλεται μήνυμα λήξης το οποίο είναι ίδιο με αυτό της έναρξης. Ο δέκτης μπορεί να αντιμετωπίσει ολόκληρη την ακολουθία σαν μία εντολή.

Το πρωτόκολλο επίσης παρέχει και την δυνατότητα αναγνώρισης από τον δέκτη ότι η μπαταρία του πομπού είναι πεσμένη. Ο δέκτης στην αρχή του μηνύματος αντί να στείλει εισαγωγικό παλμό, που μαζί με την παύση που ακολουθεί, είναι 3 msec αποστέλλει έναν με παύση που συνολικά φτάνει τα

4 msec. Στην πραγματικότητα αυτό μπορεί να γίνει μόνο στον παλμό του μηνύματος έναρξης και τα υπόλοιπα πακέτα να σταλούν κανονικά.

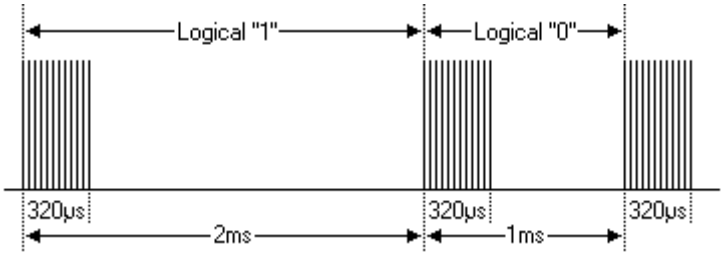
Ακολουθούν μερικές από τις βασικότερες εντολές για δορυφορικούς δέκτες.

Εντολή	SAT Διεύθυνση 1100 [0000]
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
12	Stand-by
14	Up key
15	Down key
40	Mute
41	Reveal
42	Alternate
45	Index
46	Right key
47	Left key
51	Text
53	Stop
56	Size
60	Red
61	Green
62	Yellow
112	TV/SAT

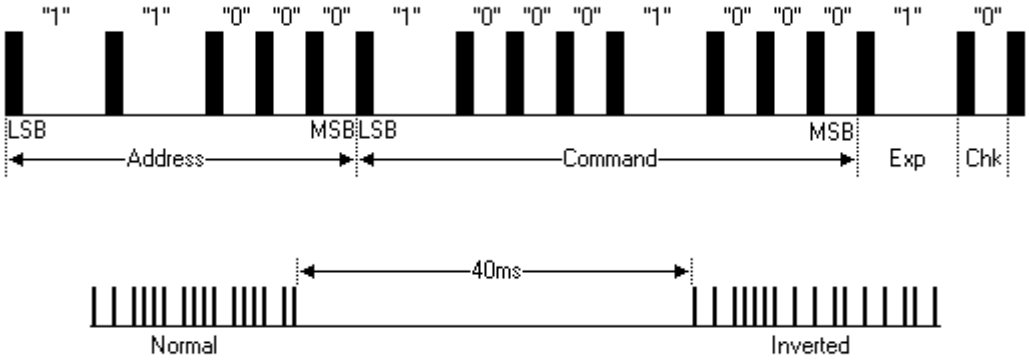
- Το πρωτόκολλο που χρησιμοποιεί η Sharp

Το πρωτόκολλο αυτό το χρησιμοποιεί η εταιρία Sharp στις συσκευές βίντεο. Το φέρον και σε αυτή την περίπτωση είναι συχνότητας 38 kHz και χρησιμοποιείται κωδικοποίηση απόστασης

παλμών. Το λογικό ένα μεταδίδεται σε 2 msec εκ των οποίων τα πρώτα 320 usec είναι διαμορφωμένος (με 38 kHz φέρον) παλμός και ο υπόλοιπος χρόνος παύση. Το λογικό μηδέν έχει διάρκεια 1 msec πάλι με παλμό 320 usec.

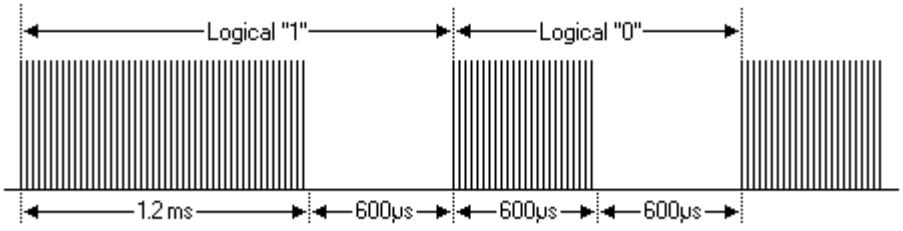


Κάθε μήνυμα μεταδίδεται δύο φορές την πρώτη κανονικά και την δεύτερη ανεστραμμένο με χρονική απόσταση 40 msec. Το μήνυμα ξεκινάει με το λιγότερο σημαντικό από τα 5 bit της διεύθυνσης του παραλήπτη. Στην συνέχεια, αποστέλλονται τα 8 bit της εντολής με το λιγότερο σημαντικό bit πρώτο. Ακολουθεί ένα bit επέκτασης και ένα bit ελέγχου που δηλώνουν το τέλος του μηνύματος και την αντιστροφή ή όχι αυτού.

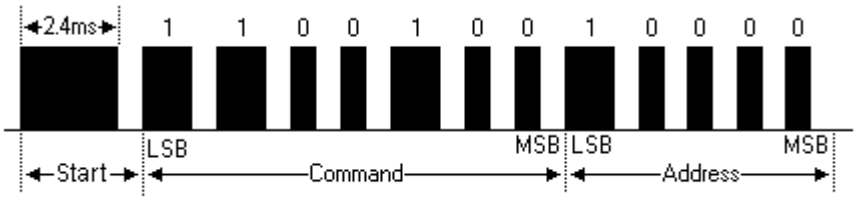


- Το πρωτόκολλο SISC της Sony

Το πρωτόκολλο αυτό έχει 3 εκδόσεις. Στην παλιότερη τα μηνύματα αποτελούνται από 12 bit ενώ αργότερα υλοποιήθηκε με 15 και 20 bit. Η συχνότητα του φέροντος του πρωτοκόλλου είναι 40 kHz και χρησιμοποιείται κωδικοποίηση διαμόρφωσης εύρους παλμών. Πιο συγκεκριμένα για το λογικό ένα ο διαμορφωμένος παλμός που αποστέλλεται έχει μήκος 1.2 msec και παύση 600 usec. Το λογικό μηδέν αποτελείται από παλμό 600 usec και παύση 600 usec.



Ο παλμός έναρξης αποτελείται από παλμό διάρκειας 2.4 msec και παύση 600 usec. Ο παλμός έναρξης μπορεί να χρησιμοποιηθεί για την ρύθμιση του κέρδους του δέκτη. Στην συνέχεια η εντολή, αποτελούμενη από 7 bit, αποστέλλεται με το λιγότερο σημαντικό bit πρώτα και τέλος τα 5 bit της διεύθυνσης, επίσης με το λιγότερο σημαντικό bit πρώτα. Το μήνυμα επανεκπέμπεται 45 msec μετά την έναρξη της προηγούμενης εκπομπής (το μήνυμα δεν είναι συγκεκριμένης διάρκειας καθώς το 1 και το 0 έχουν διαφορετική διάρκεια), όσο κρατιέται το κουμπί πατημένο.



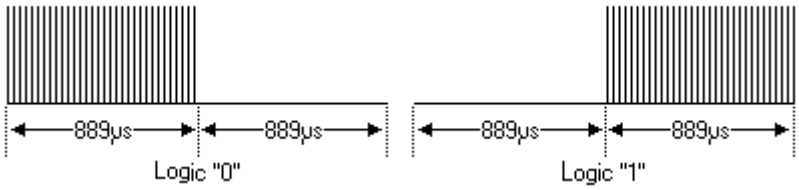
Στους πίνακες που ακολουθούν φαίνονται οι πιο συνηθισμένες διευθύνσεις και εντολές.

Εντολή	Λειτουργία
0	1
1	2
2	3
3	4
4	5
5	6
6	7
7	8
8	9
9	0
16	Channel +
17	Channel -
18	Volume +
19	Volume -
20	Mute
21	Power
22	Reset
23	Audio Mode
24	Contrast +
25	Contrast -
26	Colour +
27	Colour -
30	Brightness +
31	Brightness -
38	Balance Left
39	Balance Right
47	Standby

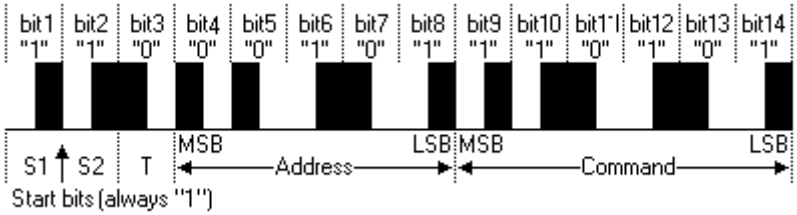
Διεύθυνση	Συσκευή
1	TV
2	VCR1
3	VCR2
6	Laser Disk Unit
12	Surround Sound
16	Cassette deck / Tuner
17	CD Player
18	Equalizer

- Το πρωτόκολλο RC-5 της Philips

Το πρωτόκολλο RC-5 είναι από τα πιο διαδεδομένα ανάμεσα σε ερασιτέχνες. Είναι φτιαγμένο ώστε να μπορεί να λειτουργήσει για ολοκληρωμένο σύστημα εικόνας και ήχου. Η συχνότητα του φέροντος είναι 36 kHz και τα ψηφία κωδικοποιούνται με κωδικοποίηση Manchester. Το λογικό μηδέν αποτελείται από παλμό διάρκειας 889 usec και παύση 889 usec ενώ το λογικό ένα το αντίθετο, από παύση και στη συνέχεια παλμό. Κάθε ψηφίο έχει σταθερή χρονική διάρκεια 1.778 msec ανεξάρτητα από την τιμή του.



Κάθε μήνυμα ξεκινάει με δύο ψηφία εκκίνησης, δύο λογικά 1. Το τρίτο ψηφίο είναι ένα ψηφίο ελέγχου, το οποίο αντιστρέφεται κάθε φορά που ένα κουμπί αφήνεται και πατιέται πάλι. Έτσι ο δέκτης μπορεί να ξεχωρίσει κατά πόσο ένα κουμπί πατιέται παρατεταμένα ή επαναλαμβανόμενα. Τα πέντε επόμενα ψηφία αποτελούν την διεύθυνση του δέκτη με το περισσότερο σημαντικό ψηφίο πρώτο. Στη συνέχεια αποστέλλονται 6 ψηφία που αποτελούν την εντολή με το περισσότερο σημαντικό ψηφίο πρώτο. Υπάρχει μια παραλλαγή του πρωτοκόλλου η οποία υποστηρίζει την χρήση του δεύτερου από τα ψηφία εκκίνησης ως το 7 ψηφίο εντολής. Με αυτόν τον τρόπο υποστηρίζονται περισσότερες εντολές. Το μήνυμα αποτελείται από 14 bit και έχει συνολική διάρκεια 25 msec. Το παράδειγμα που ακολουθεί δείχνει την εντολή 53 (110101) στην διεύθυνση 5 (00101).



Για όση διάρκεια παραμένει πατημένο κάποιο κουμπί του χειριστηρίου, το μήνυμα

επαναλαμβάνεται κάθε 114 msec, ενώ το bit ελέγχου (το τρίτο bit) παραμένει στην ίδια λογική τιμή. Ο δέκτης μπορεί έτσι να αντιληφθεί αν ένα κουμπί πατιέται επαναλαμβανόμενα ή κρατιέται πατημένο.

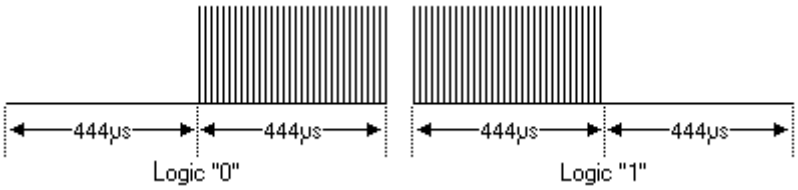
Ακολουθούν πίνακες με τις βασικότερες διευθύνσεις και εντολές.

Διεύθυνση	Συσκευή	Εντολή	Λειτουργία TV	Λειτουργία VCR
0	TV1	0	0	0
1	TV2	1	1	1
2	Teletext	2	2	2
3	Video	3	3	3
4	LV1	4	4	4
5	VCR1	5	5	5
6	VCR2	6	6	6
7	Experimental	7	7	7
8	Sat1	8	8	8
9	Camera	9	9	9
10	Sat2	10	-/--	-/--
11		12	Standby	Standby
12	CDV	13	Mute	
13	Camcorder	16	Volume +	
14		17	Volume -	
15		18	Brightness +	
16	Pre-amp	19	Brightness -	
17	Tuner	32	Program +	Program +
18	Recorder 1	33	Program -	Program -
19	Pre-amp	50		Fast Rewind
20	CD Player	52		Fast Forward
21	Phono	53		Play
22	SatA	54		Stop
23	Recorder 2	55		Recording
24				
25				
26	CDR			
27				
28				
29	Lighting			

30	Lighting			
31	Phone			

Το πρωτόκολλο RC-6 είναι ο διάδοχος του RC-5. Το RC-6 έχει διάφορες λειτουργίες, η βασικότερη των οποίων είναι η "0". Στην λειτουργία αυτή το μήνυμα χωρίζεται σε τρία μέρη, την επικεφαλίδα, το πεδίο ελέγχου και το πεδίο πληροφορίας. Η επικεφαλίδα χωρίζεται σε τρία επί μέρους τμήματα. Το πρώτο μέρος αποτελείται από έναν παλμό έναρξης 2.666 msec και παύση 889 usec και από ένα bit διπλάσιας διάρκειας από τα κανονικά. Αυτά χρησιμοποιούνται για να σηματοδοτήσουν την έναρξη του μηνύματος και να επιτρέψουν στον δέκτη να ρυθμίσει το κέρδος του. Τα επόμενα τρία bit ορίζουν την λειτουργία που χρησιμοποιείται. Τέλος, ακολουθεί ένα bit λήξης της επικεφαλίδας, το οποίο έχει διπλάσιο μήκος από τα κανονικά bit. Το πεδίο ελέγχου αποτελείται από 8 bit και αφορά την διεύθυνση του δέκτη. Το περισσότερο σημαντικό bit μεταδίδεται πρώτο. Το πεδίο πληροφορίας αποτελεί την εντολή επίσης από 8 bit με το πιο σημαντικό να αποστέλλεται πρώτο.

Το λογικό μηδέν αποτελείται από παύση 444 usec και παλμό 444 usec ενώ το λογικό ένα από παλμό 444 usec και παύση 444 usec.

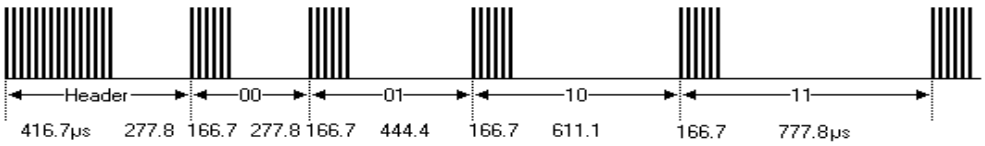


Μετά από κάθε μήνυμα πρέπει να υπάρχει παύση για τουλάχιστον 2,666 msec.

- Το Πρωτόκολλο της Philips RC-MM

Τα αρχικά MM βγαίνουν από τις λέξεις multi – media και υποδηλώνουν ότι το πρωτόκολλο σχεδιάστηκε για χρήση από πολλά μέσα. Στις βλέψεις της Philips όταν το κατασκεύαζε ήταν να χρησιμοποιηθεί από ασύρματα πληκτρολόγια, ποντίκια και χειριστήρια παιχνιδιών. Μέσα στις απαιτήσεις του πρωτοκόλλου ήταν λοιπόν η μικρή κατανάλωση και η συντομία των μηνυμάτων.

Το μήνυμα μπορεί να αποτελείται από 12 ή 24 bit ανάλογα με την λειτουργία που επιλέγεται. Σε αντίθεση με τα υπόλοιπα πρωτόκολλα χρησιμοποιούνται 4 σύμβολα με κωδικοποίηση απόστασης παλμών. Με κάθε σύμβολο αποστέλλονται 2 ψηφία. Διπλασιάζεται έτσι η ταχύτητα της επικοινωνίας και υποδιαιρείται το πλήθος των συμβόλων που αποστέλλονται με αποτέλεσμα να μειώνεται και η κατανάλωση. Η φέρουσα συχνότητα είναι τα 36 kHz και τα μηνύματα επαναλαμβάνονται ανά 28 msec. Η διάρκεια των μηνυμάτων κυμαίνεται, λόγω της μη σταθερής διάρκειας και ποσότητας ψηφίων, από 3.5 msec ως 6.5 msec. Κάθε μήνυμα πρέπει να ακολουθείται από "ησυχία" ώστε να αποφεύγονται οι παρεμβολές από άλλες συσκευές και πρωτόκολλα. Η Philips προτείνει παύση 1 msec στην κανονική λειτουργία και 3.36 msec σε περίπτωση που χρησιμοποιείται μαζί με τα πρωτόκολλα RC-5 ή RC-6. Το σχήμα που ακολουθεί εξηγεί την κωδικοποίηση. Duty cycle 1:3 ή 1:4



Κάθε μήνυμα ξεκινάει με τον χαρακτήρα εκκίνησης που αποτελείται από παλμό με διάρκεια 15 περιόδων του φέροντος (416,7 usec) και παύση 10 περιόδων (277,8 usec). Τα τέσσερα σύμβολα αντιστοιχούν σε ζευγάρια λογικών ψηφίων. Όλα τα σύμβολα ξεκινάνε με παλμό 6 περιόδων (166,7 usec), ενώ μεταβάλλεται η διάρκεια της παύσης που ακολουθεί. Το λογικό 00 αντιστοιχεί σε 10 περιόδων παύση (277,8 usec), το λογικό 01 σε 16 περιόδους (444,4 usec), το 10 σε παύση 22 περιόδων (611,1 usec) και τέλος το 11 σε παύση 28 περιόδων (777,8 usec).

Στο πρωτόκολλο ορίζονται 3 τρόποι λειτουργίας, ο καθένας από τους οποίους προορίζεται για διαφορετική εφαρμογή και διαφέρει στον αριθμό των bit που μπορούν να χρησιμοποιηθούν από την εφαρμογή. Σε όλες τις περιπτώσεις αποστέλλεται το περισσότερο σημαντικό bit πρώτο. Στην περίπτωση των 12 bit τα 2 πρώτα ορίζουν τον τρόπο λειτουργίας, τα επόμενα 2 την διεύθυνση και τα 8 τελευταία αποτελούν τα δεδομένα.

Mode 2 bits	Address 2 bits	Data 8 bits

Η κατασκευάστρια εταιρεία ορίζει 3 οικογένειες συσκευών, πληκτρολόγιο, ποντίκι και χειριστήριο παιχνιδιών με κωδικούς 01, 10 και 11 αντίστοιχα. Στην περίπτωση που το μήνυμα ξεκινάει με 00 χρησιμοποιείται η επέκταση σε 24 bit κατά την οποία χρησιμοποιούνται 4 bit για τον ορισμό της λειτουργίας και 20 για τα δεδομένα. Και σε αυτήν την περίπτωση ορίζονται οι 3 παραπάνω κατηγορίες με κωδικούς 0001, 0010 και 0011.

Mode 4 bits	Data 20 bits

Ο τρίτος τρόπος λειτουργίας αφορά την περίπτωση που τα 4 πρώτα bit είναι 0. Τα δύο επόμενα ψηφία είναι πάντοτε 1 και ακολουθούν 6 ψηφία που χαρακτηρίζουν την διεύθυνση του δέκτη ανάλογα με τον κατασκευαστή και 12 ψηφία δεδομένων.

Mode 6 bits	Customer ID 6 bits	Data 12 bits

3. Bluetooth

3.1 Εισαγωγή

Είναι σχεδόν απίθανο να ανοίξει κανείς κάποιο περιοδικό τεχνολογίας και να μην βρει τη λέξη Bluetooth. Η τεχνολογία αυτή ασύρματης δικτύωσης είναι απλή και αξιόπιστη, με αποτέλεσμα πολλά προϊόντα τεχνολογίας να την ενσωματώνουν για σκοπούς μεταφοράς δεδομένων τοπικής εμβέλειας. Το πρωτόκολλο δημιουργήθηκε το 1998 από μια ομάδα εταιρειών με τη συμμετοχή των Nokia, Ericsson, Intel και Toshiba. Το 1999 κυκλοφόρησε η πρώτη έκδοση του πρωτοκόλλου.

Το Bluetooth είναι ένα πρωτόκολλο επικοινωνίας τοπικής εμβέλειας στο οποίο η μεταφορά δεδομένων γίνεται μέσω της εγκατάστασης προσωπικού δικτύου (Personal Area Network – PAN). Η μέθοδος αυτή παρέχει υψηλή ασφάλεια για τη μεταφορά δεδομένων. Η ευρεία διάδοση του πρωτοκόλλου οφείλεται στη συμμετοχή χιλιάδων εταιρειών στη διαχείριση και θέσπιση του πρωτοκόλλου, αλλά και στην ευκολία χρήσης του. Άλλα χαρακτηριστικά που βοήθησαν στην διάδοσή του είναι η ταυτόχρονη σύνδεση πολλαπλών χρηστών και η μικρή κατανάλωση ισχύος, γεγονός που το κάνει ιδανικό για φορητές συσκευές. Ο ρυθμός μετάδοσης δεδομένων του Bluetooth είναι αρκετά χαμηλός, συγκριτικά με άλλα πρωτόκολλα επικοινωνίας, αλλά σε εφαρμογές που ο όγκος πληροφορίας είναι μικρός, ενώ η ασφάλεια και η χαμηλή κατανάλωση είναι σημαντικοί παράγοντες, γι' αυτό συχνά αποτελεί την επιλεγμένη τεχνολογία.

Η λειτουργία του Bluetooth γίνεται στην ζώνη των 2,4 Ghz που είναι ελεύθερη ζώνη του φάσματος για βιομηχανική, επιστημονική και ιατρική χρήση (ISM, Industrial Scientific and Medical). Ο μέγιστος ρυθμός μετάδοσης στην έκδοση 1.2 φτάνει τα 721 kbps ενώ στην έκδοση 2.0 ανεβαίνει στα 3Mbps και στην 3.0 στα 24 Mbps. Υπάρχουν 3 κατηγορίες ισχύος εκπομπής. Στην πρώτη η μέγιστη ισχύς εκπομπής είναι τα 100mW (20dbm), στη δεύτερη 2.5mW (4dbm) και στην τρίτη 1mW (0dbm). Η εμβέλεια είναι αντίστοιχα 100, 10 και 5 μέτρα.

Μέχρι 8 συσκευές μπορούν να συνδεθούν σε ένα δίκτυο (piconet) με μια από αυτές να αποτελεί τον “αφέντη” (master) και τις υπόλοιπες τους “δούλους” (slave). Κάθε συσκευή μπορεί να συνδεθεί σε παραπάνω από ένα piconet επιτρέποντας την επικοινωνία μεταξύ διαφορετικών “μικρών δικτύων” για την δημιουργία ενός μεγαλύτερου scatternet. Σε κάθε piconet ο master αναλαμβάνει τον έλεγχο και τον διαμοιρασμό των πόρων. Μέσω πολυπλεξίας διαμοιρασμού χρόνου (TDM Time Division Multiplex) “τεμαχίζει” τον χρόνο σε χρονοθυρίδες και εκπέμπει στις περιττές από αυτές αφήνοντας τις άρτιες για τα τερματικά slave. Η τεχνική αυτή ονομάζεται (TDD) time division duplex και οδηγεί σε αμφίδρομη “ταυτόχρονη” επικοινωνία. Για να εκπέμψει κάποια από τις slave συσκευές πρέπει να λάβει το δικαίωμα από τον master το οποίο μοιράζεται περιοδικά ανάμεσα στους διάφορους slave (TDMA Time Division Multiple Access).

Για να μπορεί μια συσκευή να συνδεθεί σε περισσότερα από ένα piconet πρέπει να υποστηρίζει τις καταστάσεις (modes) hold, park και sniff κατά τις οποίες δεν συμμετέχει ενεργά σε κάποιο piconet, για να μπορεί να δραστηριοποιείται σε κάποιο άλλο. Επίσης θα πρέπει η συσκευή να υποστηρίζει τη δυνατότητα εναλλαγής ρόλων μεταξύ αφέντη και σκλάβου κατά τη διάρκεια της σύνδεσης. Η ιδιότητα αυτή είναι απαραίτητη στο scatternet, αλλά δεν είναι υποχρεωτική για να πιστοποιηθεί μια συσκευή ως σύμφωνη με το πρωτόκολλο Bluetooth.

3.2 Πακέτα Bluetooth

Τα πακέτα αποτελούνται από 3 πεδία, τον κωδικό πρόσβασης (72 bit), την επικεφαλίδα (54 bit) και το φορτίο (μέχρι 2.745 bit). Ο κωδικός πρόσβασης είναι υπεύθυνος για τον συγχρονισμό και την ταυτοποίηση των συσκευών και αποτελείται από 3 πεδία. Το προοίμιο (4 bit) είναι η ακολουθία 1010. Ακολουθεί η “λέξη συγχρονισμού” που είναι τα επόμενα 64 bit και αποτελεί τον κωδικό που είναι υπεύθυνος για τον συντονισμό και την ταυτοποίηση. Το πεδίο κλείνει με την ουρά 4 bit αντίστοιχη ακολουθία με το προοίμιο (1010).

Η επικεφαλίδα αποτελείται από 6 πεδία. Το πρώτο είναι η διεύθυνση της συσκευής και αποτελείται από 3 bit. Το δεύτερο είναι ο τύπος του πακέτου αποτελούμενος από 4 bit. Ακολουθεί ένα bit για τον έλεγχο της ροής στο κανάλι και ένα bit ως ένδειξη σωστής μετάδοσης πακέτου. Ακολουθεί αρίθμηση των πακέτων και 8 bit για έλεγχο και διόρθωση λαθών της επικεφαλίδας.

Το πεδίο του φορτίου μπορεί να είναι διαφόρων τύπων. Κατά την αποστολή δεδομένων χρησιμοποιούνται 1 με 2 bytes (8 -16 bit) ως επικεφαλίδα. Ακολουθούν τα δεδομένα, το μήκος των οποίων έχει δηλωθεί στην επικεφαλίδα του φορτίου. Τέλος 16 bit αφιερώνονται στην εύρεση και διόρθωση λαθών. Εναλλακτική μορφή πακέτων φορτίου είναι η DV (Data – Voice) μορφή που χρησιμοποιείται για μεταφορά δεδομένων και ήχου. Σε αυτή τη μορφή τα 80 πρώτα bit είναι το πεδίο αφιερωμένο στον ήχο και ακολουθεί το πεδίο δεδομένων που αποτελείται από 32 ως 150 bit. Υπάρχουν επίσης άλλοι τύποι πακέτων που χρησιμοποιούνται για τον έλεγχο των καναλιών και του δικτύου γενικότερα.

3.3 Τεχνική Εναλλαγής Συχνότητας (Frequency hopping)

Καθώς στον ίδιο χώρο πρέπει να συνυπάρχουν πολλές συσκευές συνδεδεμένες σε διαφορετικά piconet, με διαφορετικό όγκο κίνησης και διαφορετικής ανάγκης παροχής υπηρεσιών, θα πρέπει να ελαχιστοποιηθεί η παρεμβολή που προκαλεί μια συσκευή στις άλλες. Τη λύση δίνει η τεχνική εναλλαγής συχνότητας. Το εύρος φάσματος της τεχνολογίας χωρίζεται σε κανάλια εύρους 1 Mhz. Στην Ισπανία, τη Γαλλία και την Ιαπωνία τα κανάλια είναι 23, ενώ στον υπόλοιπο κόσμο είναι 79.

Συσκευές που είναι συντονισμένες σε ένα κανάλι οφείλουν να μεταπηδήσουν σε άλλο, που ορίζεται από την αρχή της σύνδεσης από τον master με μια ψευδό-τυχαία ακολουθία. Η χρονική σχισμή, κατά την οποία οι δέκτες παραμένουν σε ένα κανάλι, είναι 625 usec. Όταν τα δεδομένα ανταλλάσσονται με πακέτα μίας σχισμής ο master εκπέμπει στις περιττές και οι slave στις άρτιες. Υπάρχουν ωστόσο πακέτα με 1, 3 ή 5 σχισμές, τα οποία έχουν ίδια επικεφαλίδα και δεδομένα ελέγχου, μεταφέροντας έτσι περισσότερο ωφέλιμο φορτίο.

Τον συγχρονισμό των λειτουργιών αυτών αναλαμβάνει ένα ρολόι πραγματικού χρόνου που αποτελείται από 28 bit. Το ρολόι μηδενίζεται κατά την έναρξη του συστήματος ή λόγω υπερχείλισης που συμβαίνει περίπου μια φορά την μέρα καθώς ο μετρητής αυξάνεται κάθε 312,5 usec. Η ύπαρξη του ρολογιού επιτρέπει τις συγχρονισμένες ανταλλαγές δεδομένων, την παραγωγή ακολουθίας καναλιών μεταπήδησης συχνότητας αλλά και την διαφοροποίηση μεταξύ αναμεταδόσεων και χαμένων πακέτων.

3.4 Σύγχρονη & Ασύγχρονη Επικοινωνία

Υπάρχουν δύο βασικές μορφές επικοινωνίας που υποστηρίζονται από το πρωτόκολλο. Η ασύγχρονη και η σύγχρονη.

Οι ασύγχρονες συνδέσεις (Asychronous Connectionless Links – ACL) χρησιμοποιούνται κυρίως για τη μεταφορά δεδομένων. Αμέσως μετά την έναρξη επικοινωνίας εγκαθίσταται ασύγχρονη ζεύξη μεταξύ των συσκευών. Μεταξύ δύο συσκευών επιτρέπεται μόνο μια σύνδεση και μόνο ο master μπορεί να συνδεθεί με παραπάνω από μια συσκευή. Ο master επιλέγει με ποιόν slave θα επικοινωνήσει και εφόσον υπάρχουν δεδομένα προς αποστολή αποστέλλονται σποραδικά. Ο slave απαντάει μόνο αν του έχει απευθυνθεί ο master στην επόμενη χρονική σχισμή. Τα περισσότερα ACL πακέτα έχουν έλεγχο λαθών και υποστηρίζουν αναμετάδοση σε περίπτωση λάθους. Τα δεδομένα μπορούν να μεταφέρονται είτε με πακέτα υψηλής μεταφοράς δεδομένων (DH, Data High Rate) με μεγαλύτερο ωφέλιμο φορτίο, είτε με πακέτα μέσης μεταφοράς δεδομένων (DM, Data Medium Rate) με καλύτερο έλεγχο και προστασία κατά των λαθών.

Η σύγχρονη ζεύξη (SCO, Synchronous Connection Oriented) αποτελεί συμμετρικό κανάλι επικοινωνίας μεταξύ master και slave. Για την εγκατάσταση σύνδεσης το επίπεδο διαχειριστή

ζεύξεων του master αποστέλλει εντολή προς τον slave. Στην εντολή αυτή περιλαμβάνονται τα χαρακτηριστικά της σύνδεσης, όπως το διάστημα που ο master απευθύνει τον λόγο στον slave αλλά και τις σχισμές που ο slave απαντάει. Οι χρονοσχισμές είναι δεσμευμένες από την εγκατάσταση της σύνδεσης και έτσι το εύρος ζώνης της επικοινωνίας είναι εγγυημένο. Πολλαπλές συνδέσεις επιτρέπονται μεταξύ δύο συσκευών με τον περιορισμό των 3 συνδέσεων για ένα master αλλά 2 έναν slave εκτός αν προέρχονται από τον ίδιο master οπότε επιτρέπονται 3 συνδέσεις. Τα πακέτα στην σύνδεση αυτή δεν αναμεταδίδονται καθώς δίνεται έμφαση στην πραγματικού χρόνου επικοινωνία. Ο master αποστέλλει πακέτα σε τακτά χρονικά διαστήματα και ο slave απαντάει στις εκχωρημένες χρονικές σχισμές.

Καθώς οι δύο μορφές συνδέσεων μπορούν να συνυπάρχουν, τα πακέτα ACL αποστέλλονται στις σχισμές που είναι ελεύθερες από τις SCO συνδέσεις που έχουν προτεραιότητα.

3.5 Η Στοίβα Πρωτοκόλλων Bluetooth

Η στοίβα του Bluetooth μπορεί να χωριστεί σε τέσσερις βασικές κατηγορίες :

- Πρωτόκολλα πυρήνα Bluetooth:
  - Στρώμα βασικής ζώνης
  - Πρωτόκολλο διαχείρισης ζεύξεων (LMP,LinkManagementProtocol)
  - Πρωτόκολλο ελέγχου και προσαρμογής λογικών ζεύξεων (L2CAP,Logical Link Control and Adaptation Protocol)
  - Πρωτόκολλο ανακάλυψης υπηρεσιών (SDP,Service Discovery Protocol)
- Πρωτόκολλο αντικατάστασης καλωδίων:
  - Επικοινωνία RF (RFCOMM, Audio Frequency Communication)
- Πρωτόκολλο τηλεφωνικού ελέγχου:
  - Προδιαγραφές τηλεφωνικού ελέγχου (TCS-Bin, Telephony Control Specification Binary)
- Υιοθετημένα πρωτόκολλα:
  - Πρωτόκολλα διαδικτύου (PPP, UDP, TCP,IP)
  - Πρωτόκολλο ανταλλαγής αντικειμένων (OBEX,Object Exchange Protocol)
  - Πρωτόκολλο ασύρματης εφαρμογής (WAP, Wireless Application Protocol)

Το επίπεδο HCI (Host Controller Interface) είναι απαραίτητο και αναλαμβάνει την διεπαφή μεταξύ του ελεγκτή host και host. Στα ανώτερα επίπεδα δεν είναι απαραίτητη η συνύπαρξη όλων των πρωτοκόλλων. Ανάλογα με την εφαρμογή που υλοποιείται επιλέγεται ένα προφίλ. Το προφίλ περιγράφει την κατακόρυφη επιλογή των πρωτοκόλλων που χρησιμοποιούνται.

3.6 Διαμόρφωση Σήματος

Το σχήμα διαμόρφωσης του Bluetooth, GFSK (Gaussian Frequency Shift Keying) οδηγεί σε μικρότερη παρεμβολή στα γειτονικά κανάλια. Η ζώνη συχνοτήτων που χρησιμοποιείται στο πρωτόκολλο είναι ανοιχτή σε βιομηχανική, επιστημονική και ιατρική χρήση είναι επομένως πολύ σημαντικό να περιορίζονται οι διακαναλικές παρεμβολές. Το γκαουσιανό φίλτρο που χρησιμοποιείται στην διαμόρφωση αυτή, είναι ένας απλός τρόπος ομαλοποίησης των παλμών και περιορισμός του φάσματος αυτών. Με δεδομένο το εύρος των καναλιών στα 1MHz, τη μετάδοση στα 1 MS/s (Mega Symbol per Second) και την κωδικοποίηση, ο ρυθμός μετάδοσης προκύπτει 1Mbps αφού σε κάθε σύμβολο αντιστοιχεί ένα bit. Το λογικό 1 αντιστοιχεί σε αύξηση της συχνότητας του φέροντος ενώ το λογικό 0 σε μείωση. Η αύξηση και μείωση της συχνότητας είναι μεταξύ 140 και 175 kHz αλλά οι μεταβολές λόγω του φίλτρου γίνονται με πιο ομαλό τρόπο από ότι

στην τυπική FSK διαμόρφωση.

Διεύθυνση συσκευών

Η διεύθυνση μιας συσκευής δημιουργείται από τον κατασκευαστή της συσκευής και δεσμεύεται, ώστε παγκοσμίως να μην υπάρχουν συσκευές με την ίδια διεύθυνση. Η διεύθυνση έχει μήκος 48 bit και αποτελείται από 3 πεδία, το χαμηλό, το υψηλό και το ασήμαντο μέρος της διεύθυνσης. Το χαμηλό μέρος (LAP, Lower Address Part) αποτελείται από τα πρώτα 24 bit ενώ το υψηλό μέρος (UAP, Upper Address Part) είναι τα επόμενα 8 bit. Τα δύο αυτά μέρη αποτελούν το σημαντικό μέρος της διεύθυνσης και δίνουν την δυνατότητα διευθυνσιοδότησης 232= 4.294.967.296 συσκευών. Τα υπόλοιπα 16 bit αποτελούν το ασήμαντο μέρος (NAP, Non – significant Address Part) της διεύθυνσης. Η διεύθυνση της συσκευής, όπως και τα υπόλοιπα δεδομένα αποστέλλονται με το λιγότερο σημαντικό bit πρώτο (Little Endian).

3.7 Πακέτα Bluetooth

Κάθε πακέτο αποτελείται από 3 βασικά μέρη, τον κώδικα πρόσβασης, την επικεφαλίδα ελέγχου και το ωφέλιμο φορτίο. Τα δεδομένα αποστέλλονται με το λιγότερο σημαντικό bit πρώτο. Τα πακέτα μπορούν να διαφέρουν πολύ ανάλογα την εφαρμογή και τον τύπο σύνδεσης, καθώς και το ωφέλιμο φορτίο, που κυμαίνεται από 0 έως 2744 bit. Ο κώδικας πρόσβασης και η επικεφαλίδα ελέγχου έχουν σταθερό μήκος 72 και 54 bit αντίστοιχα.

- Κώδικας πρόσβασης

Ο κώδικας πρόσβασης αποτελείται από τρία μέρη. Τα 4 πρώτα bit αποτελούν το πρόθεμα, τα επόμενα 64 την λέξη συντονισμού και τέλος ακολουθούν 4 bit που κλείνουν τον κώδικα πρόσβασης. Το πρόθεμα μπορεί να είναι είτε 1010, είτε 0101, ανάλογα με το πρώτο bit της λέξης συγχρονισμού. Ο ρόλος του κώδικα πρόσβασης είναι συγχρονισμός, η σωστή λειτουργία του Bluetooth Radio, αλλά και η πρόσβαση στα ανώτερα επίπεδα του πρωτοκόλλου. Καθώς εισέρχονται τα δεδομένα στον δέκτη καταχωρούνται σε έναν καταχωρητή ολίσθησης και συγκρίνονται με την αναμενόμενη λέξη συντονισμού. Η κεφαλίδα ελέγχου του πακέτου διαβάζεται μόνο αν η λέξη συγχρονισμού ταιριάξει. Σε αντίθετη περίπτωση, η μονάδα RF μπορεί να μπει σε κατάσταση αναμονής ή ακόμα και να κλείσει εξοικονομώντας ενέργεια. Η λέξη συντονισμού παράγεται από τις διευθύνσεις των συσκευών που μετέχουν σε ένα piconet με δύο τρόπους. Από το LAP (χαμηλό μέρος διεύθυνσης) του master παράγεται ο κωδικός πρόσβασης καναλιού (Channel Access Code – CAC), που χαρακτηρίζει το piconet και χρησιμοποιείται για μετάδοση δεδομένων. Από το LAP της καλούμενης συσκευής δημιουργείται ο κώδικας πρόσβασης συσκευής (Device Access Code - DAC). Το πρωτόκολλο περιγράφει και μια τρίτη μορφή της λέξης συντονισμού, τον κωδικό πρόσβασης διερεύνησης (Inquiry Access Code - IAC), που μπορεί να είναι είτε ίσος με 0χ9Ε8Β33 (γενικός κώδικας πρόσβασης διερεύνησης, General IAC – GIAC), είτε να αποτελεί αφιερωμένο κώδικα πρόσβασης διερεύνησης (Dedicated IAC – DIAC).

- Επικεφαλίδα ελέγχου

Στο πεδίο αυτό περιλαμβάνονται πληροφορίες ελέγχου του πακέτου. Τα 3 πρώτα bit της επικεφαλίδας ελέγχου αποτελούν την διεύθυνση ενεργού μέλους (Active Member Address). Κατά την σύνδεση των συσκευών σε ένα piconet ο master εκχωρεί διευθύνσεις στους slaves. Τα 3 bit δίνουν την δυνατότητα διευθυνσιοδότησης 8 συσκευών, αλλά το 0 δεσμεύεται για πακέτα εκπομπής προς όλους τους slaves. Έτσι, ο master μπορεί να έχει μέχρι 7 slaves.

Τα επόμενα 4 bit ορίζουν τον τύπο του πακέτου, γεγονός που δίνει την δυνατότητα για 16 τύπους πακέτου. Το επόμενο bit ελέγχει την ροή στα κανάλια ACL. Το 9ο bit της επικεφαλίδας χρησιμοποιείται για να επιβεβαιώσει την ορθή λήψη του τελευταίου πακέτου. Το επόμενο bit χρησιμοποιείται για την αποφυγή πολλαπλών λαμβανόμενων πακέτων ως bit ακολουθίας.

Τα 10 αυτά bit ακολουθούνται από 8 bit που αποτελούν τον κώδικα λάθους επικεφαλίδας

(Header Error Code – HEC) ώστε να προστατευτούν από λάθη. Τα συνολικά 18 bit αποστέλλονται με κωδικοποίηση ευθείας διόρθωσης λαθών (Forward Error Correction – FEC) 1/3. Σύμφωνα με την κωδικοποίηση αυτή κάθε bit πληροφορίας αντιστοιχεί σε 3 bit προς αποστολή. Έτσι προκύπτει το μήκος 54 bit της επικεφαλίδας. Με την τεχνική αυτή θυσιάζουμε εύρος ζώνης αποστέλλοντας μη απαραίτητα ψηφία, αλλά πετυχαίνουμε υψηλό επίπεδο ασφάλειας κατά την μετάδοση της επικεφαλίδας που είναι κρίσιμη για τον έλεγχο της ζεύξης.

- Ωφέλιμο φορτίο

Το μέρος αυτό του πακέτου περιέχει την προς μετάδοση πληροφορία. Χωρίζεται και αυτό σε 3 μέρη, την επικεφαλίδα φορτίου, το φορτίο και το πεδίο κυκλικού ελέγχου πλεονασμού (Cyclic Redundancy Check – CRC). Η επικεφαλίδα φορτίου αποτελείται από 8 ή 16 bit και το CRC από 16. Το φορτίο κυμαίνεται από 0 ως 2712 bit.

3.8 Ελεγκτής ζεύξεων (LC)

Σύμφωνα με το πρωτόκολλο Bluetooth για τον ελεγκτή ζεύξεων υπάρχουν 9 πιθανές καταστάσεις :

- Standby: Στην προσπάθεια να περιορισθεί η κατανάλωση ισχύος, όταν η συσκευή δεν είναι συνδεδεμένη σε ένα piconet, μπαίνει στην κατάσταση αναμονής. Για να βγει από την κατάσταση αναμονής πρέπει είτε να ξεκινήσει ένα inquiry, είτε να δεχτεί inquiry ή paging.
- Inquiry: Κατά την αναζήτηση κοντινών συσκευών ο ελεγκτής μπαίνει σε αυτήν την κατάσταση. Αποστέλλει και λαμβάνει συγκεκριμένα πακέτα (FHS) που περιλαμβάνουν απαραίτητες πληροφορίες για την σύνδεση, όπως το ρολόι του master, διευθύνσεις συσκευών, αλλά και τον τρόπο εναλλαγής συχνοτήτων.
- Inquiry Scan: Στην κατάσταση αυτή χρησιμοποιείται μια ειδική ακολουθία εναλλαγής καναλιών πιο αργή από ότι κατά την inquiry κατάσταση, ώστε κάποια στιγμή μια συσκευή που βρίσκεται σε κατάσταση inquiry (γρηγορότερης εναλλαγής συχνοτήτων) να συντονιστεί με την συσκευή που βρίσκεται σε κατάσταση inquiry scan. Η συσκευή δεν είναι απαραίτητο να εισέλθει στην κατάσταση αυτή ώστε να δεχτεί σύνδεση από άλλη, ωστόσο στην κατάσταση αυτή είναι ανιχνεύσιμη από άγνωστες συσκευές.
- Page: Η διαδικασία σύνδεσης μιας συσκευής σε ένα piconet ξεκινάει από τον master. Σε αυτήν την κατάσταση ο master αποστέλλει ειδικά πακέτα που οδηγούν στην σύνδεση της συσκευής, τον ορισμό της διεύθυνσης μέσα στο piconet και την ακολουθία εναλλαγής συχνοτήτων. Η διαδικασία αυτή ονομάζεται paging.
- Page Scan: Για να μπορεί να αντιληφθεί μια συσκευή την προσπάθεια σύνδεσης από έναν master θα πρέπει να βρίσκεται σε κατάσταση page scan. Στην κατάσταση αυτή οι συσκευές μπαίνουν κατά τακτά χρονικά διαστήματα ώστε να μπορούν να δεχτούν συνδέσεις. Σύνδεση μπορεί να επιτευχθεί με απ' ευθείας χρήση της διεύθυνσης Bluetooth της συσκευής χωρίς να έχει προηγηθεί inquiry.
- Connection: Στην κατάσταση σύνδεσης τα ρολόγια των slave συγχρονίζονται με αυτό του master και όλες οι συσκευές του piconet ακολουθούν την ίδια ακολουθία εναλλαγής συχνοτήτων. Είναι η κατάσταση κατά την οποία δημιουργούνται συνδέσεις και μεταφέρονται δεδομένα. Με σκοπό την εξοικονόμηση ενέργειας, η συσκευή μπορεί να μεταβεί σε μία από τις παρακάτω καταστάσεις.
- Connection – Hold Mode: Η συσκευή μπαίνει σε αυτήν την κατάσταση ύστερα από εντολή του master ώστε αυτός να επανακτήσει εύρος ζώνης. Στην κατάσταση αυτή ο slave διατηρεί την διεύθυνση του στο piconet και τις σύγχρονες συνδέσεις του (SCO) αλλά δεν μπορεί να χρησιμοποιήσει τις ασύγχρονες (ACL).
- Connection – Sniff Mode: Στην κατάσταση αυτή ο slave παύει να ακούει συνεχόμενα τις χρονοσχισμές που ο master αποστέλλει πακέτα και ακούει περιοδικά. Έτσι έχει διαστήματα

κατά τα οποία εξοικονομείται ενέργεια χωρίς όμως να έχει απώλεια σύνδεσης.

- Connection – Park Mode: Ένας master μπορεί να υποστηρίξει ως και 255 μη ενεργά μέλη τα οποία λαμβάνουν διεύθυνση μη ενεργού μέλους (parked member address). Τα μη ενεργά μέλη μπορούν να ενεργοποιηθούν με πακέτο broadcast από τον master αφού διατηρούν τον συγχρονισμό τους με το piconet. Κατά την ενεργοποίηση ενός μέλους, κάποιο άλλο που δεν έχει δεδομένα προς αποστολή ή λήψη, ανταλλάσσει την διεύθυνση του με αυτή του μη ενεργού μέλους.

3.9 Διαχειριστής Ζεύξεων (LM)

Ο διαχειριστής ζεύξεων αναλαμβάνει τις λεπτομέρειες υλοποίησης των συνδέσεων στο piconet προσφέροντας απλά εργαλεία στα παραπάνω στρώματα του πρωτοκόλλου. Το επίπεδο αυτό αναλαμβάνει τον συντονισμό των λειτουργιών και την διαχείριση των ζεύξεων. Το επίπεδο αυτό υλοποιείται στην μονάδα Bluetooth της συσκευής και επικοινωνεί με το επίπεδο L2CAP της συσκευής (host) μέσω του Host Controller Interface (HCI). Τα πακέτα του διαχειριστή ζεύξεων έχουν προτεραιότητα έναντι των πακέτων δεδομένων και ακολουθούνται από μια διακοπή κατά την οποία αναμένεται απάντηση. Το επίπεδο αυτό παρέχει τα εργαλεία για την εγκατάσταση, τον έλεγχο και τον τερματισμό συνδέσεων.

- Πακέτα διαχείρισης ζεύξεων

Για την επικοινωνία των διαχειριστών ζεύξεων ορίζονται ειδικά μηνύματα από το πρωτόκολλο διαχείρισης ζεύξεων τα οποία αποστέλλονται ως ξεχωριστά πακέτα. Το πρώτο ψηφίο του μηνύματος δηλώνει αν ο αποστολέας είναι master ή slave με 0 ή 1 αντίστοιχα (Single Bit Transaction Identifier – TID). Ο κώδικας λειτουργίας (Operation Code – OpCode) που ακολουθεί αποτελείται από 7 bit και χαρακτηρίζει τον τύπο του μηνύματος. Στη συνέχεια αποστέλλονται οι παράμετροι του μηνύματος.

Για την εγκατάσταση μιας σύνδεσης ο διαχειριστής ζεύξεων του slave απαντάει σε αίτηση του διαχειριστή ζεύξεων του master. Στην συνέχεια, μπορεί να ακολουθούν διάφορα πακέτα που ορίζουν λεπτομέρειες της σύνδεσης. Ο διαχειριστής του master τέλος πρέπει να αποστείλει πακέτο που να δηλώνει την ολοκλήρωση της σύνδεσης και ο server να απαντήσει αποδεχόμενος την σύνδεση. Όταν ολοκληρωθεί η εγκατάσταση ασύγχρονης σύνδεσης μπορεί να δημιουργηθεί και σύγχρονη σύνδεση με αντίστοιχα πακέτα μεταξύ των διαχειριστών ζεύξεων.

Για την απεγκατάσταση της ζεύξης και την απελευθέρωση των πόρων, η συσκευή, που θέλει να αποχωρήσει από την σύνδεση, αποστέλλει πακέτο, που περιλαμβάνει την αιτία αποσύνδεσης (τερματισμός από τον χρήστη, ανεπάρκεια πόρων ή απενεργοποίηση συσκευής).

3.10 Διεπαφή Ελεγκτή Host (HCI)

Ο αρχικός στόχος των κατασκευαστών του Bluetooth ήταν να αντικαταστήσουν τα καλώδια που χρησιμοποιούνται για σειριακές επικοινωνίες με ασύρματες ζεύξεις. Φτάνοντας σε αυτό το επίπεδο έχει επιτευχθεί αυτό ο στόχος. Στον ελεγκτή ορίζεται η επικοινωνία μεταξύ των κατώτερων επιπέδων του Bluetooth και των επιπέδων που ο χρήστης μπορεί να χρησιμοποιήσει μέσω προγραμμάτων οδήγησης και εφαρμογών. Στον ελεγκτή ορίζονται 3 μορφές μεταφοράς δεδομένων : USB, UART και RS232. Λόγω της τυποποίησης του επιπέδου, προγράμματα οδήγησης διαφόρων κατασκευαστών ταιριάζουν με διάφορες μονάδες Bluetooth.

Ο ελεγκτής επικοινωνεί με 3 τύπους πακέτων. Παρέχει πακέτα εντολών για τον έλεγχο της μονάδας Bluetooth πακέτα γεγονότων για την πληροφόρηση του host για τις αλλαγές στη μονάδα και μεταβιβάζει πακέτα δεδομένων.

Τα πακέτα εντολών αποτελούνται από τρία πεδία : τον κώδικα λειτουργίας (OpCode), το

πεδίο αρίθμησης των παραμέτρων και τις παραμέτρους. Ο κώδικας λειτουργίας αποτελείται από το πεδίο ομάδας (OpCode Group Field – OGF) ενός byte και το πεδίο εντολής (OpCode Command Field – OCF), επίσης ενός byte. Το επόμενο byte δηλώνει το μήκος σε bytes των παραμέτρων και τέλος ακολουθούν οι παράμετροι με συγκεκριμένο αριθμό bytes.

Σε περίπτωση άμεσης εκτέλεσης της εντολής, η μονάδα Bluetooth απαντάει με ένα γεγονός ολοκλήρωσης εντολής (Command Complete Event). Σε περίπτωση που γίνεται να ολοκληρωθεί απ' ευθείας η εντολή, η μονάδα απαντάει με γεγονός κατάστασης εντολής (Command Status Event) και επιστρέφει ένα γεγονός ολοκλήρωσης κατά την ολοκλήρωση της εντολής. Οι εντολές Set_Event_Mask και Set_Event_Filter ελέγχουν ποια γεγονότα μπορούν να περάσουν. Η εντολή Inquiry_Result_Filter επιστρέφει στον host τα γεγονότα που περνάνε από το φίλτρο. Η εντολή Connection_Setup_Filter ορίζει ένα φίλτρο για τις συνδέσεις που η μονάδα μπορεί να αποδεχτεί.

Αντίστοιχη δομή έχουν και τα πακέτα γεγονότων. Το πρώτο byte είναι ο κώδικας γεγονότος. Το δεύτερο είναι το μήκος των παραμέτρων σε byte και στη συνέχεια ακολουθούν οι παράμετροι.

Η δομή με την οποία περνάνε μέσα από το επίπεδο διεπαφής του ελεγκτή τα δεδομένα, απαιτεί τα 12 πρώτα bit να ελέγχουν την σύνδεση και να ορίζουν τον τύπο σύνδεσης (ACL, SCO), όπου διακινούνται τα πακέτα.

Για ασύγχρονο τύπο σύνδεσης ακολουθούν δύο σημαίες μήκους 2 bit η καθεμία και 2 byte με το μήκος των δεδομένων σε byte. Στη συνέχεια ακολουθούν τα δεδομένα. Η πρώτη από τις σημαίες (Packet Boundary – PB) δηλώνει αν τα δεδομένα αποτελούν την αρχή ενός πακέτου υψηλού επιπέδου (L2CAP) ή αν είναι η συνέχεια ενός πακέτου. Η δεύτερη σημαία (Broadcast) ορίζει αν τα δεδομένα αποστέλλονται από σημείο προς σημείο ή αν είναι προς εκπομπή. Επίσης, ξεχωρίζει την εκπομπή προς ενεργούς slaves και προς όλους (ενεργούς και μη).

Στις σύγχρονες συνδέσεις πάλι τα πρώτα 12 bit αφορούν την σύνδεση. Ακολουθούν τέσσερα bit δεσμευμένα (για μελλοντική χρήση) ένα byte με το μήκος των δεδομένων σε bytes. Τέλος ακολουθούν τα δεδομένα.

3.11 Πρωτόκολλο Ελέγχου Λογικών Ζεύξεων και Προσαρμογής (Logical Link Control and Adaptation Protocol – L2CAP)

Το L2CAP διαβιβάζει τα πακέτα δεδομένων είτε στο HCI, είτε απ' ευθείας στον διαχειριστή ζεύξεων. Μόνο δεδομένα μεταφέρονται μέσω του L2CAP, ο ήχος δεν περνάει από αυτό το επίπεδο. Το πρωτόκολλο αυτό επιτρέπει την χρήση ίδιων ζεύξεων σε διαφορετικές εφαρμογές οργανώνοντας την πολυπλεξία των δεδομένων. Σε περίπτωση που τα δεδομένα είναι μεγαλύτερα από τα πακέτα Bluetooth που μπορούν να αποσταλούν, το επίπεδο αυτό αναλαμβάνει τον τεμαχισμό στην πλευρά του αποστολέα και την ανασυγκρότησή τους στην πλευρά του δέκτη. Το πρωτόκολλο αναλαμβάνει να μοιράσει τους πόρους των κατώτερων στρωμάτων στα ανώτερα επίπεδα. Με τον τρόπο αυτό διαχειρίζεται την ποιότητα υπηρεσιών των ανώτερων επιπέδων.

- Πολυπλεξία με χρήση καναλιών

Το L2CAP διαφορετικών συσκευών σε ένα δίκτυο επικοινωνούν μεταξύ τους μέσω καναλιών. Τα κανάλια προσδιορίζονται από μια 16- ψήφια ταυτότητα καναλιού (Channel Identifier – CID). Η επικοινωνία γίνεται πάντα μέσα από τον master, ο οποίος προωθεί τα πακέτα μέσα από τα κατάλληλα κανάλια, εφόσον δεν προορίζονται γι' αυτόν. Τα κανάλια μπορεί να είναι είτε σύνδεσης, είτε χωρίς σύνδεση.

Τα κανάλια με σύνδεση χρειάζονται αρχικοποίηση για να χρησιμοποιηθούν και επιλέγονται όταν υπάρχει ανάγκη για πλήρη - αμφίδρομη επικοινωνία (full-duplex). Τα κανάλια χωρίς σύνδεση

χρησιμοποιούνται για ήμι – αμφίδρομη επικοινωνία (half-duplex), όπως η εκπομπή σε πολλές συσκευές.

Η μηδενική ταυτότητα καναλιού δεν δίνεται σε καμία συσκευή, ενώ η ταυτότητα 1h δίνεται στο κανάλι σήμανσης. Η ταυτότητα 2h δίνεται σε κανάλια χωρίς σύνδεση, ενώ τα υπόλοιπα κανάλια παίρνουν δυναμικά τιμές μεταξύ 40h και FFFFh. Η περιοχή μεταξύ 3h και 3Fh παραμένει δεσμευμένη.

Τα πακέτα L2CAP αποτελούνται από το πεδίο μήκους μεγέθους 2 byte, την ταυτότητα καναλιού μεγέθους 2 byte επίσης και τα δεδομένα που μπορούν να φτάσουν μέχρι 65535 byte.

- Κανάλι σήμανσης

Το κανάλι αυτό χρησιμοποιείται για τα σήματα ελέγχου μεταξύ επιπέδων L2CAP που αφορούν τον χειρισμό, δημιουργία και κατάργηση συνδέσεων. Τα πακέτα που είναι υπεύθυνα γι' αυτές τις λειτουργίες αποτελούνται από 4 πεδία. Το πρώτο byte είναι ο κωδικός λειτουργίας. Το δεύτερο είναι ένα πεδίο ταυτότητας. Ακολουθεί το μήκος σε bytes μεγέθους 2 bytes και τα δεδομένα. Κάθε υλοποίηση L2CAP επιπέδου οφείλει να υποστηρίζει τουλάχιστον 48 bytes δεδομένων. Σε περίπτωση που λάβει πακέτο με μεγαλύτερο μήκος από αυτό που μπορεί να διαχειριστεί (Maximum Transmission Unit – MTU), το απορρίπτει και απαντάει με ένα ειδικό πακέτο.

Το πεδίο ταυτότητας χρησιμοποιείται για να μπορεί το σήμα να συσχετιστεί με την απάντησή του, η οποία περιέχει την ίδια ταυτότητα. Μέσα σε ένα πακέτο μπορούν να αποσταλούν πολλές εντολές, η μια μετά την άλλη.

- Σύνδεση

Για την δημιουργία σύνδεσης αποστέλλεται μια αίτηση στο επίπεδο L2CAP. Αν δεν υπάρχει έτοιμη σύνδεση το L2CAP αποστέλλει αίτηση σύνδεσης είτε στον ελεγκτή διεπαφής (HCI), είτε κατευθείαν στον διαχειριστή συνδέσεων. Όταν δεχτεί απάντηση και ολοκληρωθεί η δημιουργία σύνδεσης σε κατώτερα επίπεδα, απαντάει στο ανώτερο επίπεδο για την επιτυχία σύνδεσης.

Οι αιτήσεις και εντολές L2CAP επιπέδου (κανάλι 1h) αποστέλλονται μέσω ασύγχρονων ζεύξεων, λόγω της απαίτησης λήψεως των μηνυμάτων χωρίς λάθη.

Το μήνυμα αίτησης σύνδεσης του L2CAP επιπέδου περιλαμβάνει εκτός από τα γνωστά πεδία και την τιμή υπηρεσιών πολυπλέκτη πρωτοκόλλου (Protocol Service Multiplexer – PSM), καθώς και την ταυτότητα του καναλιού της συσκευής αποστολής αίτησης σύνδεσης. Το πεδίο PSM προσδιορίζεται σε υψηλότερο επίπεδο. Ενδεικτικά η τιμή 0x0001 αφορά SDP, η τιμή 0x0003 RFCOMM και η 0x0005 TCS-BIN.

4. Το Λειτουργικό Σύστημα Android

Το Android είναι λειτουργικό σύστημα ανοιχτού κώδικα, για φορητές συσκευές όπως "έξυπνα" τηλέφωνα, tablet και netbook. Είναι βασισμένο στον πυρήνα Linux και ανεπτυγμένο από την Google, με σκοπό να χρησιμοποιηθεί σε φορητές συσκευές. Ο σχεδιασμός του είναι κυρίως για οθόνες αφής, προσφέροντας αναγνώριση από πλήθος κινήσεων όπως απλό άγγιγμα, παρατεταμένο άγγιγμα, σύρσιμο αλλά και κινήσεις πολλών σημείων ταυτόχρονα, όπως για παράδειγμα δύο δάχτυλα που χρησιμοποιούνται για να μεγεθύνουν ή να μικρύνουν μια εικόνα. Το Android αποτελεί το πιο διαδεδομένο λειτουργικό σύστημα για φορητές συσκευές, αλλά και το πιο διαδεδομένο λειτουργικό σύστημα γενικώς, καθώς η χρήση του δεν περιορίζεται σε κινητά τηλέφωνα, αλλά χρησιμοποιείται σε τηλεοράσεις, παιχνιδομηχανές, ψηφιακές κάμερες, μέχρι και ρολόγια.

4.1 Στοίβα Λογισμικού

Στο κατώτερο επίπεδο της στοίβας βρίσκεται ο πυρήνας Android. Ο πυρήνας αποτελεί μια παραλλαγή του πυρήνα Linux προσαρμοσμένο στις ανάγκες φορητών συσκευών με περιορισμένους διαθέσιμους πόρους, για τις οποίες προορίζεται το Android. Ο πυρήνας ελέγχει τη συσκευή μέσω των προγραμμάτων οδήγησης και αποτελεί τον "μεσάζοντα" ανάμεσα στο λογισμικό και τη συσκευή. Η διαχείριση των πόρων της κάθε συσκευής γίνεται από τον πυρήνα, ο οποίος δέχεται αιτήσεις από τις διάφορες εφαρμογές όταν αυτές χρειάζονται να χρησιμοποιήσουν κάποιο μέρος του hardware της συσκευής.

Πάνω από τον πυρήνα υπάρχει το επίπεδο γηγενών βιβλιοθηκών (Native Libraries). Οι βιβλιοθήκες δίνουν την δυνατότητα στη συσκευή να χειριστεί διάφορους τύπους δεδομένων. Ακολουθούν μερικές από τις πιο σημαντικές βιβλιοθήκες:

- **Διαχειριστής Επιφάνειας (Surface Manager):** η βιβλιοθήκη αυτή χρησιμοποιείται για την σύνθεση αντικειμένων στην οθόνη και χρησιμοποιεί έναν ενδιάμεσο καταχωρητή, πριν εμφανιστεί η συντεθειμένη εικόνα στην οθόνη. Αυτό επιτρέπει στις διάφορες εφαρμογές να λειτουργούν σαν να ζωγραφίζουν κατευθείαν στην οθόνη, ενώ στην πραγματικότητα τα σχέδια αποτυπώνονται στον ενδιάμεσο καταχωρητή, συνθέτονται με άλλα και τέλος αποτυπώνονται στην οθόνη, χωρίς ο δημιουργός της εφαρμογής να χρειάζεται να γνωρίζει αυτή την σύνθετη λειτουργία. Επίσης προσφέρει και κάποιες λειτουργίες όπως παράθυρα με μερική διαφάνεια.
- **Πλαίσιο Πολυμέσων (Media Framework):** η βιβλιοθήκη πολυμέσων προσφέρει αποκωδικοποιητές – κωδικοποιητές (CODECS – Coders / Decoders) διαφόρων μορφών πολυμέσων, επιτρέποντας στα ανώτερα επίπεδα της στοίβας λογισμικού να καταγράψουν και να αναπαράγουν διάφορους τύπους δεδομένων πολυμέσων.
- **SQLite:** υλοποιεί μια μηχανή βάσης δεδομένων που βασίζεται στην SQL, αλλά δεν χρειάζεται κάποιον εξυπηρετητή (Server). Αυτό κάνει την SQLite λιγότερο απαιτητική σε πόρους, άρα καταλληλότερη για ενσωματωμένες συσκευές. Η SQLite γράφει απευθείας σε αρχεία βάσης δεδομένων. Η μορφή των αρχείων είναι συμβατή με όλες τις πλατφόρμες χρησιμοποιούν SQLite ανεξάρτητα αν είναι συστήματα 32 ή 64 μπιτ, big-endian ή little-endian αρχιτεκτονικής. Η βιβλιοθήκη καταλαμβάνει μικρό χώρο στον σκληρό δίσκο σε σχέση με αντίστοιχες. Για τους παραπάνω λόγους αποτελεί πολύ δημοφιλή επιλογή.
- **WebKit:** αποτελεί την μηχανή που αποδίδει περιεχόμενα μορφής HTML στον φυλλομετρητή (Browser). Περιλαμβάνει αρκετά εργαλεία για τους φυλλομετρητές και αποτελεί αρκετά δημοφιλή επιλογή, αφού χρησιμοποιείται από τον φυλλομετρητή Chrome της Google αλλά και από τον Safari της Apple.

- **OpenGL:** η βιβλιοθήκη αυτή αποδίδει δισδιάστατα και τρισδιάστατα γραφικά στην οθόνη. Έχει την δυνατότητα συνεργασίας με μονάδα επεξεργασίας γραφικών (GPU – Graphics Processing Unit) επιτυγχάνοντας επιτάχυνση απόδοσης υλικού (Hardware-Accelerated Rendering).

Στο ίδιο επίπεδο με τις βιβλιοθήκες βρίσκεται και το Android Runtime, το οποίο αποτελείται από την εικονική μηχανή Dalvik και βιβλιοθήκες πυρήνα Java. Η εικονική μηχανή Dalvik είναι μια μορφή εικονικής μηχανής Java (JVM – Java Virtual Machine), βελτιστοποιημένη για συσκευές με μικρή μνήμη και επεξεργαστική ισχύ. Σε αντίθεση με την εικονική μηχανή Java, δεν τρέχει αρχεία .class, αλλά .dex αρχεία, τα οποία δημιουργούνται από τα .class αρχεία κατά τη μεταγλώττιση του πηγαίου κώδικα και αποδίδει καλύτερα σε περιβάλλοντα με χαμηλή μνήμη. Επιτρέπει την δημιουργία και την εκτέλεση πολλών εικονικών μηχανών ταυτόχρονα, προσδίδοντας έτσι στις εφαρμογές, ανεξαρτησία, ασφάλεια, καθώς και την δυνατότητα να τρέχουν παράλληλα. Ταυτόχρονα επιτρέπει στο λειτουργικό σύστημα να διαχειριστεί κατάλληλα την μνήμη της συσκευής.

Η εικονική μηχανή Dalvik είναι δομημένη με αρχιτεκτονική βασισμένη σε καταχωριστές, που απαιτεί λιγότερες και πιο απλές εντολές μηχανής από την εναλλακτική της στοίβας, που χρησιμοποιούν οι εικονικές μηχανές Java. Τα προγράμματα γράφονται σε Java με την διεπαφή προγραμματισμού εφαρμογών του Android (API – Application Pogramming Interface), στη συνέχεια μεταγλωττίζονται σε java bytecode και αυτό με τη σειρά του, μετατρέπεται σε εντολές Dalvik που είναι απαραίτητες για την εκτέλεση. Το εργαλείο που μετατρέπει τα .java αρχεία σε .dex ονομάζεται dx και αναλαμβάνει να συνθέσει πολλαπλές κλάσεις σε ένα μόνο αρχείο, ενώνοντας μεταβλητές που εμφανίζονται πολλαπλές φορές, σε μια γλιτώνοντας χώρο. Τα εκτελέσιμα αρχεία Dalvik ενδέχεται να μετασχηματιστούν κι άλλο κατά την εγκατάσταση τους στην συσκευή, επιτρέποντας να γίνει βελτιστοποίηση ανάλογα με τους πόρους της συσκευής.

Η Dalvik από την έκδοση Android 2.2 και μετά, έχει just-in-time (JIT) μεταγλωττιστή, ο οποίος μεταγλωττίζει το πρόγραμμα κατά την εκτέλεση. Ο δυναμικός αυτός τρόπος μεταγλώττισης, αξιοποιεί καλύτερα τους διαθέσιμους πόρους, αφού το πρόγραμμα μετατρέπεται σε γλώσσα μηχανής ανάλογα με τους πόρους που είναι διαθέσιμοι κατά την στιγμή της εκτέλεσης. Αυτό σημαίνει ότι μια εφαρμογή μπορεί να μεταγλωττιστεί σε διαφορετικές γλώσσες μηχανής με την ίδια λειτουργία, ανάλογα με το ποιες άλλες εφαρμογές εκτελούνται δεσμεύοντας μνήμη και χρόνο στον επεξεργαστή. Αυτή η τεχνική συνδυάζει τις παραδοσιακές προσεγγίσεις της μεταγλώττισης πριν την εκτέλεση (AOT – Ahead Of Time compilation) και τη μετάφραση. Το JIT έχει τα πλεονεκτήματα της ταχύτητας του μεταγλωττισμένου κώδικα και την ευελιξία που προσφέρει η χρήση του μεταφραστή, αλλά πέρα από την επιβάρυνση του μεταφραστή έχει και την επιβάρυνση του μεταγλωττιστή, καθώς το πρόγραμμα μεταγλωττίζεται την ώρα της εκτέλεσης.

Οι βιβλιοθήκες πυρήνα Java είναι διαφορετικές από τις Java SE και Java ME αλλά προσφέρουν τις περισσότερες από τις λειτουργίες που προσφέρουν και οι Java SE. Ένα μέρος των βιβλιοθηκών αυτών υλοποιεί βασικές λειτουργίες, όπως την διαχείριση συμβολοακολουθιών, λειτουργίες δικτύων και την διαχείριση αρχείων, με τρόπο που είναι οικείος σε προγραμματιστές Java, παρόλο που οι βιβλιοθήκες είναι κατασκευασμένες για να τρέχουν με την εικονική μηχανή Dalvik. Αυτές συμπληρώνονται από τις βιβλιοθήκες Android, οι οποίες είναι εξειδικευμένες για ανάπτυξη εφαρμογών Android. Αυτές προσφέρουν εργαλεία για την κατασκευή της διεπαφής με τον χρήστη, την δημιουργία γραφικών, την πρόσβαση σε βάσεις δεδομένων κ.α.. Μερικές από τις διαθέσιμες βιβλιοθήκες είναι :

- android.app : προσφέρει εργαλεία για το μοντέλο της εφαρμογής και είναι η βάση για όλες τις εφαρμογές Android

- android.content : εξυπηρετεί στην πρόσβαση των δεδομένων μια εφαρμογής αλλά και την επικοινωνία μεταξύ διαφορετικών εφαρμογών, όπως και αντικειμένων της ίδιας εφαρμογής
- android.database : χρησιμεύει στη πρόσβαση και επεξεργασία δεδομένων από τη βάση δεδομένων και συμπεριλαμβάνει κλάσεις διαχείρισης της SQLite
- android.graphics : εργαλεία σχεδιασμού δισδιάστατων σχεδίων χαμηλού επιπέδου
- android.hardware : αποτελεί την βιβλιοθήκη που προσφέρει τα εργαλεία για την χρήση υλικού της συσκευής, όπως κλινόμετρο, επιταχυνόμετρο και αισθητήρα φωτός που μπορεί να έχει μια συσκευή
- android.opengl : αποτελεί την διεπαφή με γραφικά OpenGL που προσφέρουν και τρισδιάστατη απεικόνιση
- android.os : δίνει στις εφαρμογές πρόσβαση σε βασικές λειτουργίες του λειτουργικού συστήματος όπως μηνύματα, ρυθμίσεις και επικοινωνία ανάμεσα σε διεργασίες
- android.media : βιβλιοθήκες για την αναπαραγωγή βίντεο και ήχου
- android.net : προσφέρει τα εργαλεία για την πρόσβαση και χρήση δικτύων
- android.provider : δίνει πρόσβαση σε βάσεις δεδομένων της συσκευής όπως οι επαφές, το ημερολόγιο, οι υπενθυμίσεις και άλλα
- android.text : χρησιμοποιείται για να διαχειριστεί και να απεικονίσει κείμενο
- android.util : βιβλιοθήκη με κλάσεις που χρησιμεύουν σε διαδικασίες όπως μετατροπές μεταξύ διαφορετικών τύπων δεδομένων (π.χ. αριθμούς σε χαρακτήρες), διαχείριση αρχείων XML, διαχείριση ημερομηνίας και ώρας και άλλα
- android.view : περιλαμβάνει τα βασικά δομικά χαρακτηριστικά της διεπαφής με τον χρήστη για μια εφαρμογή
- android.widget : αποτελεί μια πλούσια συλλογή από έτοιμα αντικείμενα διεπαφής με τον χρήστη όπως κουμπιά, επιγραφές και λίστες διαφόρων τύπων
- android.webkit : βιβλιοθήκη που δίνει την δυνατότητα ενσωμάτωσης λειτουργιών φυλλομετρητή σε μια εφαρμογή.

Οι παραπάνω βιβλιοθήκες, δημιουργημένες σε Java, είναι άμεσα διαθέσιμες στον προγραμματιστή της εφαρμογής. Παρόλα αυτά οι βιβλιοθήκες αυτές δεν είναι αυτές που κάνουν την ουσιαστική λειτουργία. Οι βιβλιοθήκες Java στο επίπεδο Runtime, χρησιμοποιούν τις γηγενείς βιβλιοθήκες (Native Libraries), που είναι δημιουργημένες σε C/C++, καλώντας μεθόδους που υλοποιούνται σε αυτές. Οι βιβλιοθήκες αυτές είναι διαθέσιμες μέσω του Android Native Development Kit (NDK).

Πάνω από το επίπεδο των βιβλιοθηκών και Android Runtime βρίσκεται το επίπεδο του πλαισίου των εφαρμογών (Application Framework). Το πλαίσιο αυτό αποτελείται από ένα σετ υπηρεσιών και είναι το περιβάλλον στο οποίο εκτελούνται και διαχειρίζονται οι εφαρμογές. Στο πλαίσιο αυτό οφείλεται η υλοποίηση της δημιουργίας εφαρμογών με επαναχρησιμοποιήσιμα και αντικαταστήσιμα στοιχεία. Με την ίδια λογική, ο δημιουργός μιας εφαρμογής μπορεί να δημοσιοποιήσει τις δυνατότητες της, μαζί με δεδομένα από την εφαρμογή, ώστε να μπορεί να χρησιμοποιηθεί από άλλες εφαρμογές. Μερικές από τις βασικές υπηρεσίες που υπάρχουν στο πλαίσιο εφαρμογών είναι:

- Διαχειριστής δραστηριότητας (Activity Manager) : ελέγχει κύκλο ζωής των εφαρμογών

καθώς και την στοίβα των δραστηριοτήτων

- Πάροχοι περιεχομένου (Content Providers) : επιτρέπουν την επικοινωνία μεταξύ εφαρμογών, δίνοντας την δυνατότητα σε μια εφαρμογή να δημοσιοποιήσουν και να μοιραστούν τα δεδομένα της
- Διαχειριστής πόρων (Resource Manager) : προσφέρει πρόσβαση σε πόρους όπως ρυθμίσεις χρώματος, κάποιες συμβολοακολουθίες και σχέδια διεπαφής χρήστη
- Διαχειριστής ειδοποιήσεων (Notification Manager) : δίνει την δυνατότητα στις εφαρμογές να προβάλλουν ειδοποιήσεις
- Σύστημα προβολής (View System) : μια επεκτάσιμη συλλογή από υπηρεσίες που χρησιμοποιούνται για την δημιουργία διεπαφής χρήστη
- Διαχειριστής πακέτων (Package Manager) : είναι το σύστημα που παρέχει πληροφορίες σε μια εφαρμογή για άλλες εγκατεστημένες εφαρμογές
- Διαχειριστής τηλεφωνίας (Telephony Manager) : προσφέρει πληροφορίες σχετικά με τις υπηρεσίες τηλεφωνίας που είναι διαθέσιμες στη συσκευή
- Διαχειριστής τοποθεσίας (Location Manager) : προσφέρει πρόσβαση σε υπηρεσίες τοποθεσίας, επιτρέποντας στην εφαρμογή να ενημερώνεται σχετικά με αλλαγές στην τοποθεσία της συσκευής

Στο ανώτερο επίπεδο της στοίβας λογισμικού βρίσκονται οι εφαρμογές. Σε αυτές εμπεριέχονται εφαρμογές που έρχονται προεγκατεστημένες με το λειτουργικό και δεν μπορούν να απεγκατασταθούν, καθώς και οι εφαρμογές που έχει επιλέξει και εγκαταστήσει ο χρήστης. Για την διανομή και την εγκατάσταση, μια εφαρμογή συμπτύσσεται σε αρχείο .apk. Στο αρχείο περιλαμβάνονται, πέρα από τον κώδικα σε μεταγλωττισμένο σε .dex μορφή, οι πόροι (resources) που αποτελούνται κυρίως από μεταβλητές και συμβολοακολουθίες που αποθηκεύονται σε ξεχωριστό αρχείο από αυτό του κώδικα για την εύκολη αλλαγή τους, χωρίς να χρειαστεί ο προγραμματιστής να επέμβει στον κώδικα. Διάφορα αρχεία ή φωτογραφίες που μπορεί να χρειάζεται η εφαρμογή, κατά της εκτέλεση της, τοποθετούνται σε έναν φάκελο που ονομάζεται assets, και συμπεριλαμβάνεται και αυτός στο συμπιεσμένο αρχείο.

Πέρα από τα παραπάνω στο συμπιεσμένο αρχείο συμπεριλαμβάνονται πιστοποιητικά καθώς και το αρχείο μανιφέστο (Manifest File). Το αρχείο αυτό έχει πολύ συγκεκριμένη δομή και πρέπει να υπάρχει οπωσδήποτε για να εκτελεστεί μια εφαρμογή. Προσφέρει σημαντικές πληροφορίες για την εφαρμογή προς το σύστημα μεταξύ των οποίων :

- Ονοματίζει το πακέτο Java της εφαρμογής με μοναδικό όνομα το οποίο χρησιμεύει για την ταυτοποίηση της εφαρμογής.
- Περιγράφει τα διάφορα μέρη της εφαρμογής όπως δραστηριότητες, υπηρεσίες, δέκτες εκπομπής (Broadcast Receivers) και παρόχους δεδομένων. Επίσης, δηλώνει από ποιες κλάσεις υλοποιούνται τα παραπάνω μέρη και δημοσιοποιεί τις δυνατότητες των κλάσεων αυτών. Μέσω αυτών των δηλώσεων, γνωρίζει το λειτουργικό ποια μέρη θα εκτελεστούν και κάτω από ποιες συνθήκες.
- Ορίζει ποιες διεργασίες θα φιλοξενήσουν τα διάφορα μέρη της εφαρμογής.
- Δηλώνει τις άδειες που οφείλει να έχει η εφαρμογή, ώστε να μπορεί να έχει πρόσβαση σε προστατευμένα μέρη των API και να μπορεί να αλληλεπιδράσει με άλλες εφαρμογές.
- Δηλώνει τις άδειες που πρέπει να έχουν άλλες εφαρμογές ώστε να μπορούν να

χρησιμοποιήσουν μέρη της εφαρμογής.

- Περιλαμβάνει μια λίστα με υποκλάσεις της κλάσης Instrumentation που προσφέρουν ανάλυση και πληροφορίες για την εφαρμογή, κατά την διάρκεια της εκτέλεσης της. Η κλάση περιλαμβάνει εργαλεία, που επιτρέπουν την παρακολούθηση της διάδρασης μεταξύ λειτουργικού και εφαρμογής. Αυτές οι δηλώσεις υπάρχουν στο μανιφέστο κατά την ανάπτυξη της εφαρμογής και αφαιρούνται όταν πρόκειται να δημοσιευτεί η εφαρμογή.
- Δηλώνει το ελάχιστο επίπεδο Android API που απαιτείται για την εφαρμογή.
- Δηλώνει τις βιβλιοθήκες με τις οποίες η εφαρμογή πρέπει να συνδεθεί.

Το μανιφέστο είναι ένα αρχείο xml με συγκεκριμένα στοιχεία (Elements). Ο χρήστης δεν μπορεί να δημιουργήσει και να προσθέσει στοιχεία. Από τα διαθέσιμα στοιχεία, το στοιχείο <manifest> και το <application> πρέπει να υπάρχουν μόνο μια φορά. Για τα υπόλοιπα στοιχεία, δεν υπάρχουν περιορισμοί και υποχρεώσεις. Σε αντίθεση με τα συνηθισμένα xml αρχεία, αν κάποιο στοιχείο περιλαμβάνει κάτι, αυτό είναι άλλο στοιχείο, καθώς όλες οι τιμές των στοιχείων καθορίζονται ως χαρακτηριστικά (Attributes) και όχι ως χαρακτήρες μέσα στο στοιχείο. Το στοιχείο <manifest> είναι το root στοιχείο, που σημαίνει ότι όλα τα υπόλοιπα στοιχεία βρίσκονται μέσα σε αυτό. Για στοιχεία του ίδιου επιπέδου η σειρά δεν έχει σημασία εκτός από την περίπτωση του στοιχείου <activity-alias> το οποίο πρέπει να ακολουθείται από το στοιχείο <activity> με το οποίο συνδέεται.

Παρόλο που τα χαρακτηριστικά των στοιχείων είναι προαιρετικά, αυτά είναι που καθορίζουν την λειτουργία του αρχείου. Με εξαίρεση κάποιων από τα χαρακτηριστικά του στοιχείου <manifest>, όλα τα άλλα ξεκινάνε με το πρόθεμα “android:“. Πολλά από τα στοιχεία του αρχείου, αντιστοιχούν σε αντικείμενα στον κώδικα, όπως το στοιχείο <application> αντιστοιχεί στην εφαρμογή. Καθώς κάθε εφαρμογή έχει το δικό της μανιφέστο, το πεδίο <application> που αντιστοιχεί στην εφαρμογή είναι μόνο ένα και δεν μπορεί να λείπει. Αντίστοιχα υπάρχουν τα πεδία <activity>, <service>, <receiver> και <provider> που αντιστοιχούν σε μέρη της εφαρμογής. Για την αντιστοίχιση τους χρησιμοποιείται το χαρακτηριστικό του ονόματος, το οποίο περιλαμβάνει και το όνομα του πακέτου.

4.2 Έκδοση Android – Επίπεδο API (API Level)

- Android 1.0 – API level 1

  Η πρώτη έκδοση του Android, συμπεριλαμβάνει την δυνατότητα του να κατεβάζει ο χρήστης εφαρμογές και αναβαθμίσεις μέσω του Android Market. Έχει φυλλομετρητή ιστοσελίδων (web browser) και υποστήριξη για κάμερα, αν και δεν δίνει την δυνατότητα αλλαγής των παραμέτρων της. Επίσης, δίνει την δυνατότητα πρόσβασης του email υποστηρίζοντας τα πρωτόκολλα POP3, IMAP4 και SMTP. Προσφέρει την δυνατότητα συγχρονισμού με τις βασικότερες υπηρεσίες της Google, όπως το ημερολόγιο, τις επαφές, τους χάρτες αλλά και το gmail. Διαθέτει την δυνατότητα αναπαραγωγής πολυμέσων αλλά και την δυνατότητα κλήσης μέσω φωνητικής εντολής. Πέρα από τις βασικές εφαρμογές του ρολογιού, της αριθμομηχανής, του τηλεφωνητή και των ρυθμίσεων, συμπεριλαμβάνει και εφαρμογή αναπαραγωγής βίντεο Youtube. Επίσης, διαθέτει υποστήριξη για Wi-Fi αλλά και Bluetooth.
- Android 1.1 – API level 2

Η αναβάθμιση αυτή περιλαμβάνει διορθώσεις σε κάποια bugs, καθώς και κάποιες υπηρεσίες όπως την ανάδειξη λεπτομερειών και κριτικών κατά την αναζήτηση επιχειρήσεων στους χάρτες, την δυνατότητα πληκτρολόγησης αριθμών, κατά την διάρκεια τηλεφωνήματος και την δυνατότητα αποθήκευσης συνημμένων στα μηνύματα αρχείων.

- Android 1.5 Cupcake – API level 3

  Η έκδοση αυτή βασίστηκε στον πυρήνα Linux 2.6.27 και έχει την ονομασία Cupcake. Μερικές από τις προσθήκες, συμπεριλαμβάνουν την υποστήριξη εικονικών πληκτρολογίων από τρίτους, με την δυνατότητα χρήσης λεξικών, την υποστήριξη των Widges, που επιτρέπουν την προβολή δεδομένων σε μέρος της οθόνης, καταγραφή και αναπαραγωγή βίντεο σε μορφή MPEG-4 αλλά και 3GP όπως και την υποστήριξη των προφίλ A2DP και AVRCP του Bluetooth, επιτρέποντας την αναπαραγωγή στερεοφωνικού ήχου. Προστέθηκαν επίσης λειτουργίες αντιγραφής-επικόλλησης στον φυλλομετρητή, χρήσης εικόνων στις επαφές και η ανάδειξη ημερομηνίας και ώρας στο μητρώο των κλήσεων. Αλλαγές έγιναν επίσης στον τρόπο μετάβασης από μια εικόνα σε μια άλλη και προσθήκες αυτόματης περιστροφής εικόνας. Το λειτουργικό έδινε την δυνατότητα στον χρήστη να ανεβάσει βίντεο στο Youtube και φωτογραφίες στο Picasa.

- Android 1.6 Donut – API level 4

  Βασισμένη στον πυρήνα Linux 2.6.29, η έκδοση Donut, δίνει την δυνατότητα στον χρήστη να προχωρήσει σε αναζήτηση, είτε με κείμενο, είτε με φωνή, στο διαδίκτυο, για τις επαφές, τους σελιδοδείκτες ή το ιστορικό του φυλλομετρητή. Επίσης οι σχεδιαστές εφαρμογών μπορούν να συμπεριλάβουν το περιεχόμενο των εφαρμογών τους στα αποτελέσματα αναζήτησης. Η έκδοση αυτή έχει την δυνατότητα σύνθεσης φωνής από κείμενο, ενώ προσφέρει ευκολότερη αναζήτηση στην αγορά εφαρμογών. Προσφέρει επίσης υψηλότερη ανάλυση γραφικών και βελτιωμένες υπηρεσίες σχετικά με την κάμερα και τις αποθηκευμένες φωτογραφίες. Διαθέτει επίσης και εμπλουτισμένο πλαίσιο αναγνώρισης χειρονομιών και εργαλεία για την περαιτέρω ανάπτυξή του.

- Android 2.0 Eclair – API level 5

  Η έκδοση Eclair βασίζεται στον ίδιο πυρήνα με την Donut, προσφέροντας πρόσθετες υπηρεσίες. Δίνει την δυνατότητα στον χρήστη να συνδέσει πολλαπλούς λογαριασμούς με την συσκευή και την υποστήριξη του Microsoft Exchange email, με την ταυτόχρονη εμφάνιση πολλαπλών λογαριασμών σε μια εφαρμογή. Προσφέρει υποστήριξη του πρωτοκόλλου Bluetooth 2.1, αλλά και εύκολη κλήση, αποστολή μηνύματος ή email μέσα από τις επαφές. Ο χρήστης σε αυτήν τη έκδοση έχει πλέον τη δυνατότητα αναζήτησης στα μηνύματα, με τα παλαιότερα μηνύματα να σβήνονται αυτόματα όταν ένα προκαθορισμένο όριο ξεπεραστεί. Στην κάμερα προστίθενται νέες λειτουργίες όπως η χρήση φλας, το ψηφιακό ζουμ, η εξισορρόπηση του λευκού και άλλα. Το πληκτρολόγιο προσφέρει μεγαλύτερη ταχύτητα και βελτιωμένο λεξικό, που εμπλουτίζεται κατά τη χρήση του και αναβαθμίστηκε και ο φυλλομετρητής πλέον υποστηρίζει και HTML5.

- Android 2.0.1 Eclair – API level 6

  Η έκδοση αυτή δημιουργήθηκε για να διορθώσει κάποια bugs και συμπεριλαμβάνει κάποιες αλλαγές στο επίπεδο του πλαισίου που εκτελούνται οι εφαρμογές.

- Android 2.1 Eclair – API level 7

  Κυκλοφόρησε τον Ιανουάριο του 2010 με μικρές διορθώσεις αλλά χωρίς ουσιαστικές αλλαγές από την προηγούμενη έκδοση..

- Android 2.2 ~ 2.2.3 Froyo – API level 8

Βασισμένη στον πυρήνα Linux 2.6.32, η έκδοση αυτή, συμπεριλαμβάνει βελτιστοποιήσεις ως προς την ταχύτητα και την διαχείριση της μνήμης. Οι εφαρμογές εκτελούνται πιο γρήγορα με την υλοποίηση της μεταγλώττισης κατά την εκτέλεση (JIT compilation – Just In Time compilation), που επιτρέπει τη δυναμική μεταγλώττιση της εφαρμογής, ανάλογα με τους διαθέσιμους πόρους στη συσκευή, την ώρα της εκτέλεσης. Στον φυλλομετρητή προστέθηκε υποστήριξη για την V8 JavaScript και η συσκευή απέκτησε την υπηρεσία Cloud to Device Messaging (C2DM). Η υποστήριξη για το σύστημα Microsoft Exchange βελτιώθηκε με αναβαθμίσεις σημαντικές ως προς την ασφάλεια, τον συγχρονισμό δεδομένων του ημερολογίου και τη δυνατότητα διαγραφής όλων των δεδομένων από την συσκευή σε περίπτωση απώλειας της, για την προστασία των δεδομένων. Στην έκδοση αυτή βελτιώθηκε και η εφαρμογή που αναλαμβάνει την εκτέλεση των εφαρμογών, χρησιμοποιήθηκαν συντομεύσεις για το τηλέφωνο και τον φυλλομετρητή αλλά προστέθηκαν και οι λειτουργίες USB Tethering και Wi-Fi hotspot που επιτρέπουν την σύνδεση στο δίκτυο κινητής τηλεφωνίας και στο διαδίκτυο άλλων συσκευών ή υπολογιστών μέσω της συσκευής Android. Η εφαρμογή για την σύνδεση στην Αγορά Εφαρμογών αναβαθμίστηκε, προσφέροντας την δυνατότητα αυτόματων ανανεώσεων, το πληκτρολόγιο έγινε πιο φιλικό προς τον χρήστη δίνοντας την δυνατότητα γρήγορης εναλλαγής γλώσσας και στο Bluetooth προστέθηκε υποστήριξη συσκευών ανοιχτής ακρόασης αυτοκινήτου και γραφείου. Στον φυλλομετρητή οι εικόνες με κίνηση της μορφής GIF υποστηρίζονται κανονικά, υποστηρίζονται τα πεδία ιστοσελίδων για το "ανέβασμα" αρχείων και υποστηρίζεται το Adobe Flash. Στις εκδόσεις 2.2.1 και 2.2.2 γίνονται μερικές διορθώσεις σε bugs, ενώ στην έκδοση 2.2.3 προστίθενται δύο τροποποιήσεις σε θέματα ασφάλειας.

- Android 2.3 ~ 2.3.2 Gingerbread – API level 9

  Με ανανεωμένη και απλοποιημένη την διεπαφή με τον χρήστη, η νέα έκδοση που βασίζεται στον πυρήνα Linux 2.6.35, προσφέρει υποστήριξη για μεγαλύτερες οθόνες και τηλεφωνία VOIP μέσω του διαδικτύου. Οι υπηρεσίες αντιγραφής/επικόλλησης και το εικονικό πληκτρολόγιο εμφανίζουν σημαντικές βελτιώσεις δίνοντας την δυνατότητα ταχύτερης πληκτρολόγησης. Προστίθεται η δυνατότητα χρήσης επικοινωνίας κοντινού πεδίου NFC (Near Field Communication) που επιτρέπει την ανάγνωση ειδικών ετικετών (tag). Στην έκδοση αυτή υπάρχουν επίσης και βελτιώσεις σε θέματα αναπαραγωγής ήχου με πρόσθετες δυνατότητες, όπως αντιστάθμιση και ενίσχυση των μπάσων. Υποστηρίζονται πλέον πολλαπλές κάμερες και έτσι οι συσκευές με το λειτουργικό αυτό μπορούν να έχουν μια κάμερα που κοιτάει προς τον χρήστη και μια προς την αντίθετη κατεύθυνση. Προστέθηκε επίσης η δυνατότητα αναπαραγωγής βίντεο κωδικοποίησης WebM/VP8 και ήχου AAC,η υποστήριξη αισθητήρων, όπως γυροσκοπικού και πίεσης και βελτιστοποιήθηκε η διαχείριση ηλεκτρικής κατανάλωσης. Για τους δημιουργούς εφαρμογών και παιχνιδιών, βελτιώθηκαν οι βιβλιοθήκες για τα γραφικά και τους ήχους αλλά και για τον άμεσο προγραμματισμό σε γλώσσες όπως C και C++. Οι εκδόσεις 2.3.1 και 2.3.2 περιλαμβάνουν μικρές διορθώσεις.

- Android 2.3.3 ~ 2.3.7 Gingerbread – API level 10

  Στην έκδοση 2.3.3 υπάρχουν διορθώσεις στην διεπαφή προγραμματισμού εφαρμογών. Στην έκδοση 2.3.4 εισάγεται η δυνατότητα συνομιλίας με βιντεοκλήση μέσω του Google Talk και η δυνατότητα σύνδεσης περιφερειακών συσκευών USB ενώ για την ασφαλή σύνδεση SSL υιοθετείται η προεπιλεγμένη κωδικοποίηση RC4-MD5. Στην επόμενη έκδοση 2.3.5 υπάρχουν διορθώσεις σε προβλήματα του δικτύου για την συσκευή Nexus S και το bluetooth για το Samsung Galaxy S. Επίσης συμπεριλαμβάνονται βελτιώσεις για την απόδοση της μπαταρίας, την χρήση της κάμερας και την εφαρμογή Gmail. Στην 2.3.6 διορθώθηκε ένα σφάλμα στην φωνητική αναζήτηση αλλά κατά την αναβάθμιση προκλήθηκε σφάλμα στις Καναδέζικες Nexous S συσκευές, το οποίο σύντομα διορθώθηκε.

Η έκδοση 2.3.7 συμπεριέλαβε υποστήριξη για το Google Wallet που επιτρέπει συναλλαγές με πολλά διαδικτυακά καταστήματα.

- Android 3.0 Honeycomb – API level 11

  Το Android 3.0 αποτελεί την πρώτη έκδοση που προορίζεται μόνο για ταμπλέτα, με πυρήνα που βασίζεται στον πυρήνα Linux 2.6.36, με αλλαγές στη διεπαφή χρήστη με προσθήκη μπάρας συστήματος και γρήγορης πρόσβασης στις ειδοποιήσεις. Στην έκδοση αυτή έχουν προστεθεί εικονικά κουμπιά πλοήγησης και μπάρα ενεργειών που δίνει πρόσβαση σε εφαρμογές, όπως αυτή της πλοήγησης, με ειδοποιήσεις που εμφανίζονται ανάλογα με το περιεχόμενο εφαρμογών που τρέχουν στο παρασκήνιο. Με τις πρόσφατες εφαρμογές να εμφανίζονται σε μια λίστα που εμφανίζεται με ένα κουμπί στην μπάρα με τα κουμπιά πλοήγησης, ο χρήστης μπορεί πλέον εύκολα να μεταβεί από μια εφαρμογή σε μια άλλη και το επανασχεδιασμένο πληκτρολόγιο κάνει πιο εύκολη και γρήγορη την πληκτρολόγηση. Στον φυλλομετρητή προστέθηκε η χρήση καρτελών και ανώνυμης πλοήγησης, ενώ η πρόσβαση στην κάμερα έγινε πιο γρήγορη και οι εικόνες μπορούν πλέον να παρουσιαστούν σε πλήρη οθόνη. Στης επαφές και το email χρησιμοποιείται απεικόνιση, που χωρίζει την οθόνη στα δύο, προσφέροντας ταυτόχρονη απεικόνιση λίστας στοιχείων και λεπτομέρειες του επιλεγμένου από τη λίστα στοιχείου, διευκολύνοντας έτσι την οργάνωση τους. Από την έκδοση αυτή υποστηρίζονται πλέον πολλαπλοί πυρήνες, αλλά και επιτάχυνση υλικού, ενώ σε επίπεδο λογισμικού όλα τα δεδομένα του χρήστη έχουν την δυνατότητα να κρυπτογραφηθούν και χρησιμοποιείται ο μηχανισμός FUSE (Filesystem in Userspace), που προστατεύει τον κώδικα του πυρήνα από μη εξουσιοδοτημένους χρήστες. Η πρόσβαση των εφαρμογών στον δευτερεύοντα αποθηκευτικό χώρο, όπως κάρτες επέκτασης μνήμης, περιορίστηκε σε συγκεκριμένους ανά εφαρμογή φακέλους, ενώ στην εσωτερική μνήμη επιτρέπεται η πλήρης πρόσβαση με την κατάλληλη άδεια από τον χρήστη.

- Android 3.1 Honeycomb – API level 12

  Στην αναβάθμιση αυτή βελτιώθηκε η διεπαφή χρήστη και υλοποιήθηκε η δυνατότητα σύνδεσης USB OTG (On The Go) προσφέροντας δυνατότητα σύνδεσης αξεσουάρ. Επεκτάθηκε η λίστα πρόσφατων εφαρμογών και δόθηκε η δυνατότητα στον χρήστη να αλλάξει το μέγεθος στα widgets, εφαρμογές που καταλαμβάνουν μέρος της οθόνης και ενημερώνουν τον χρήστη για διάφορα πράγματα όπως ο καιρός ή τα νέα μιας συγκεκριμένης πηγής στο διαδίκτυο. Προστέθηκε υποστήριξη για εξωτερικό πληκτρολόγιο, ποντίκι, αλλά και χειριστηρίων για παιχνίδια. Βελτιώθηκε η χρήση του Wi-Fi δίνοντας στην συσκευή τη δυνατότητα να έχει υψηλή απόδοση ακόμα και όταν η οθόνη είναι κλειστή, ενώ για κάθε σημείο πρόσβασης Wi-Fi που είναι αποθηκευμένος στη συσκευή μπορεί πλέον να αποθηκευτεί και ξεχωριστός εξυπηρετητής proxy.

- Android 3.2 Honeycomb – API level 13

  Στην έκδοση αυτή έγιναν βελτιστοποιήσεις για την καλύτερη χρήση των διαθέσιμων πόρων από το λειτουργικό και αυξήθηκε ο αριθμός των ταμπλετών που υποστηρίζονται. Οι εφαρμογές αποκτούν πρόσβαση σε μεγαλύτερο μέρος της δευτερεύουσας μνήμης για αυξημένες δυνατότητες συγχρονισμού δεδομένων. Για την οθόνη δημιουργήθηκαν επιλογές που επιτρέπουν την βελτιστοποιημένη εμφάνιση εφαρμογών που δεν έχουν σχεδιαστεί για ταμπλέτες, αλλά δίνουν και στους προγραμματιστές εφαρμογών μεγαλύτερο έλεγχο στην εμφάνιση της εφαρμογής ανάλογα με την συσκευή στην οποία αυτή εκτελείται. Στις εκδόσεις 3.2.1 ως 3.2.6 που ακολούθησαν, έγιναν κάποιες διορθώσεις και βελτιώσεις στο Wi-Fi, την αγορά εφαρμογών αλλά και την υποστήριξη του Adobe Flash καθώς και διορθώσεις σε προβλήματα που παρουσιάστηκαν σε συγκεκριμένες συσκευές όπως την χρήση του G4 στην ταμπλέτα Motorola Xoom.

- Android 4.0 ~ 4.0.2 Ice Cream Sandwich – API level 14

  Είναι η τελευταία έκδοση που επίσημα υποστηρίζει το σύστημα αναπαραγωγής πολυμέσων Adobe Flash Player, με τον πυρήνα να βασίζεται στον πυρήνα Linux 3.0.1. Αναπτύχθηκε με τρόπο ώστε να είναι συμβατό με συσκευές με Android 2.3 και πάνω. Το θέμα εμφάνισης "Holo" περιλαμβάνει πολλές αλλαγές και νέες γραμματοσειρές, ενώ υιοθετούνται και τα εικονικά κουμπιά της μπάρας πλοήγησης από τις εκδόσεις 3.x, κάνοντάς τις διαθέσιμες σε κινητά τηλέφωνα. Τα widgets εμφανίζονται πλέον σε νέα καρτέλα, με αντίστοιχο τρόπο που εμφανίζονται τα εικονίδια των εφαρμογών. Ο χρήστης μπορεί πλέον να δημιουργήσει φακέλους με διαισθητικό τρόπο, απλά τραβώντας την μια εφαρμογή πάνω από την άλλη, βοηθώντας έτσι στην εύκολη οργάνωση των εφαρμογών και αρχείων. Η βελτίωση στην χρήση του τηλεφωνητή με την δυνατότητα επιτάχυνσης ή επιβράδυνσης των μηνυμάτων, η δυνατότητα της μεγέθυνσης της εικόνας σε εφαρμογές όπως το ημερολόγιο, καθώς και η ενσωματωμένη δυνατότητα αποθήκευσης της εικόνας, που βλέπει ο χρήστης με την χρήση του κουμπιού εκκίνησης/απενεργοποίησης, κάνουν την έκδοση αυτή περισσότερο φιλική προς τον χρήστη. Επίσης βελτιώθηκε το λεξικό διόρθωσης κειμένου κατά την πληκτρολόγηση και η εγγραφή κειμένου από φωνή, ενώ δόθηκε η δυνατότητα στον χρήστη να έχει πρόσβαση σε εφαρμογές απ' ευθείας από κλειδωμένη οθόνη. Οι λειτουργίες αντιγραφής και επικόλλησης βελτιώθηκαν και έγιναν πιο διαισθητικές. Στις ρυθμίσεις ασφάλειας προστέθηκε η δυνατότητα να ξεκλειδώνει η συσκευή μέσω προγράμματος αναγνώρισης προσώπου. Ο φυλλομετρητής απέκτησε την δυνατότητα συγχρονισμού σελιδοδεικτών με τους σελιδοδείκτες του χρήστη στον φυλλομετρητή Chrome και προστέθηκε η δυνατότητα διακοπής εκτέλεσης εφαρμογών μέσα από την λίστα πρόσφατων εφαρμογών. Στις ρυθμίσεις χρήσης δεδομένων επιτρέπεται πλέον στον χρήστη να ορίσει όρια για την χρήση δεδομένων, έτσι ώστε να ειδοποιηθεί όταν φτάσει η ξεπεράσει το όριο ή να και να διακοπεί η χρήση δεδομένων. Η χρήση της κάμερας βελτιώθηκε με το κλείστρο να ανταποκρίνεται άμεσα και την προσθήκη σύνθετων λειτουργιών, την λήψη φωτογραφιών ανά τακτά χρονικά διαστήματα, τη λήψη πανοραμικών φωτογραφιών, την μεγέθυνση κατά την εγγραφή, την επεξεργασία των αποθηκευμένων εικόνων και την υποστήριξη εικόνων της μορφής WebP. Από πλευράς βίντεο, υπάρχει πλέον η δυνατότητα καταγραφής βίντεο ποιότητας 1080p. Βελτιώθηκε επίσης η οργάνωση των πολυμέσων και οι εφαρμογές κοινωνικής δικτύωσης με την δυνατότητα χρήσης εικόνων υψηλής ανάλυσης. Αναπτύχθηκε η λειτουργία Android Beam, μια μορφή επικοινωνίας κοντινού πεδίου (NFC – Near Field Communication), η χρήση της οποίας επιτρέπει τη γρήγορη μεταφορά δεδομένων όπως οι επαφές, οι σελιδοδείκτες του φυλλομετρητή, οδηγίες πλοήγησης και πολυμέσα, ανάμεσα σε συσκευές που βρίσκονται σε μικρή απόσταση. Προστέθηκε υποστήριξη για την επιτάχυνση γραφικών της διεπαφής χρήστη από υλικούς πόρους της συσκευής, ελευθερώνοντας έτσι πόρους του επεξεργαστή. Σε θέματα δικτύων, οι συσκευές με Android 4 και πάνω έχουν την δυνατότητα δημιουργίας απ' ευθείας δικτύου Wi-Fi με άλλες συσκευές (Wi-Fi Direct) και αναπτύχθηκε η δυνατότητα για την δημιουργία εικονικού ιδιωτικού δικτύου Android VPN (Virtual Private Network). Οι επόμενες δύο εκδόσεις πρόσφεραν ουσιαστικά μικρές διορθώσεις.

- Android 4.0.3 ~ 4.0.4 Ice Cream Sandwich – API level 15

  Με τις επόμενες δύο εκδόσεις διορθώθηκαν αρκετά προβληματάκια και έγιναν βελτιστοποιήσεις ως προς την απόδοση και την κατανάλωση μπαταρίας στην αποθήκευση βάσεων δεδομένων, τον ορθογραφικό έλεγχο, την κάμερα και την λειτουργία του Bluetooth. Για τους προγραμματιστές εκδόθηκε νέο API με νέες δυνατότητες σε θέματα κοινωνικής δικτύωσης. Ανανεώθηκε η εφαρμογή της κάμερας προσφέροντας καινούριες λειτουργίες σταθεροποίησης βίντεο και QVGA ανάλυση. Βελτιώσεις έγιναν επίσης στην αναγνώριση αριθμών και την περιστροφή της οθόνης κατά την περιστροφή της συσκευής.

- Android 4.1 Jelly Bean – API level 16

  Η νέα έκδοση βασισμένη στον πυρήνα Linux 3.0.31, προσφέρει πολύ βελτιωμένη ρευστότητα την διεπαφή χρήστη, χρησιμοποιώντας βελτιωμένες τεχνικές για την άμεση απόκριση σε γεγονότα αφής στην οθόνη και αυξημένη ταχύτητα ανανέωσης της οθόνης στα 60fps. Για τα γραφικά χρησιμοποιήθηκε η τριπλάσια ενδιάμεση μνήμη αποθήκευσης δεδομένων (buffer). Ο χρήστης μπορεί πλέον να εγκαταστήσει πρόσθετα πληκτρολόγια και υποστηρίζονται περισσότερες γλώσσες. Οι ειδοποιήσεις έγιναν επεκτάσιμες και ο χρήστης μπορεί να ενεργοποιήσει ή να απενεργοποιήσει τις ειδοποιήσεις ανά εφαρμογή. Τα widgets προσαρμόζονται αυτόματα σε μέγεθος για να επιτρέψουν σε καινούρια αντικείμενα επιλογής του χρήστη να χωρέσουν στην οθόνη. Υποστηρίζεται το Bluetooth στο σύστημα επικοινωνίας κοντινού πεδίου Android Beam, ο ήχος γίνεται πολυκάναλος, με προεπιλεγμένη κωδικοποίηση / αποκωδικοποίηση AAC 5.1 και η κάμερα διαχειρίζεται από νέα βελτιωμένη εφαρμογή. Υποστηρίζεται επίσης USB Audio που επιτρέπει την σύνδεση εξωτερικής συσκευής μετατροπής του ήχου από ψηφιακό σε αναλογικό (DAC – Digital to Analog Converter) και υλοποιείται αδιάκοπη αναπαραγωγή ήχου (Gapless Playback). Στις αναβαθμίσεις 4.1.1 και 4.1.2 διορθώθηκαν λάθη στον κώδικα σχετικά με την οριοθέτηση της οθόνης και γενικά την απόδοση σε συγκεκριμένες συσκευές όπως το Nexus 7.

- Android 4.2 Jelly Bean – API level 17

  Η έκδοση είναι βασισμένη στον πυρήνα Linux 3.4.0, που περιλαμβάνει και το SELinux (Security – Enhansed Linux) για μεγαλύτερη ασφάλεια, ενώ εμφανίζει βελτιώσεις στο κλείδωμα της οθόνης με τη δυνατότητα να μεταβεί ο χρήστης στην χρήση της κάμερας χωρίς να χρειαστεί να ξεκλειδώσει τη συσκευή. Προστίθενται οι γρήγορες ρυθμίσεις σε ένα πεδίο ειδοποίησης ελέγχου ενέργειας (Notification Power Controls) που οδηγούν στην εύκολη εναλλαγή μεταξύ βελτιστοποίησης ως προς την επίδοση και βελτιστοποίησης ως προς την κατανάλωση. Στις ταμπλέτες υποστηρίζεται η χρήση πολλών χρηστών με τη δημιουργία πολλαπλών λογαριασμών. Επανασχεδιάστηκε η στοίβα πρωτοκόλλου του Bluetooth, βασιζόμενη στην BlueDroid ανοιχτού λογισμικού της Broadcom, που επιτρέπει στην σύνδεση πολλαπλών οθονών μέσω της πλατφόρμας Miracast. Βελτιώσεις έγιναν και στην δυνατότητα χρήσης εικονικού δικτύου VPN, αλλά και την χρήση της επικοινωνίας κοντινού πεδίου NFC. Δόθηκε έμφαση στην πρόσβαση με λειτουργίες όπως μεγέθυνση της οθόνης, ανεξάρτητα της εφαρμογής, με τριπλό άγγιγμα της οθόνης, ομιλία της συσκευής και βελτιωμένης ικανότητας αναγνώρισης χειρονομιών. Στην λειτουργία του ρολογιού προστέθηκε η λειτουργία χρονομέτρου, αντίστροφης μέτρησης και παγκόσμιο ρολόι. Χρησιμοποιείται η ίδια διεπαφή, για όλες τις συσκευές, που χρησιμοποιείται στην έκδοση 4.1 για τηλέφωνα, αλλά αυξήθηκε ο αριθμός των ειδοποιήσεων και δόθηκε η δυνατότητα στον χρήστη να αντιδράσει στις ειδοποιήσεις μέσα από την μπάρα ειδοποιήσεων χωρίς να είναι απαραίτητη η απευθείας εκτέλεση της εφαρμογής από την οποία προήλθε η ειδοποίηση. Βελτιστοποιήθηκε η επιβεβαίωση για τα SMS και υποστηρίζονται τα ομαδικά μηνύματα. Στην έκδοση 4.2.1 διορθώθηκε ένα σφάλμα στην εφαρμογή των επαφών και προστέθηκε χρήση χειριστηρίων παιχνιδιών μέσω Bluetooth. Στην 4.2.2 διορθώθηκαν σφάλματα στη αναπαραγωγή ήχου μέσω Bluetooth και την απόδοση, ενώ προστέθηκε η δυνατότητα ενεργοποίησης / απενεργοποίησης του Bluetooth και του Wi-Fi. Προστέθηκαν επίσης ειδοποιήσεις για τα αρχεία που κατεβαίνουν από το Internet με το χρόνο που χρειάζονται, νέοι ήχοι για την φόρτιση και την χαμηλή μπαταρία και νέα εφαρμογή για τις φωτογραφίες.

- Android 4.3 Jelly Bean – API level 18

  Η έκδοση 4.3 υποστηρίζει Bluetooth χαμηλής κατανάλωσης και το προφίλ τηλεχειρισμού AVRCP 1.3 (Audio / Video Remote Control Protocol). Στα γραφικά επιτεύχθηκε

υποστήριξη για το OpenGL ES 3.0, που δίνει αυξημένες δυνατότητες στα γραφικά των παιχνιδιών. Έγιναν βελτιώσεις στην διαχείριση του αποθηκευτικού χώρου και δόθηκε περιορισμένη πρόσβαση σε νέους χρήστες. Η διεπαφή χρήσης της κάμερας επανασχεδιάστηκε και έγιναν βελτιώσεις στην πληκτρολόγηση αριθμών με αυτόματη συμπλήρωση αριθμών. Προστέθηκε το App Ops, ένα σύστημα ελέγχου άδειας εφαρμογών και η υποστήριξη οθονών ανάλυσης 4K pixels. Έγιναν πολλές διορθώσεις σε σφάλματα και βελτιώσεις ως προς την απόδοση, αλλά και την εύρεση τοποθεσίας με την βοήθεια του Wi-Fi. Βελτιώσεις έγιναν και για τους προγραμματιστές με αυξημένες δυνατότητες καταγραφής σφαλμάτων και εργαλεία ανάλυσης χρήσης των πόρων. Προστέθηκαν 5 γλώσσες και υποστηρίζονται γλώσσες με καταγραφή από δεξιά προς τα αριστερά. Επίσης βελτιώθηκε η διαχείριση ψηφιακών δικαιωμάτων. Σύντομα βγήκε η αναβάθμιση 4.3.1 με μικρές διορθώσεις.

- Android 4.4 KitKat – API level 19

  Με ανανεωμένη εμφάνιση, η έκδοση 4.4 επιτρέπει στις εφαρμογές να εμφανίζονται ημι-διάφανες στην μπάρα κατάστασης και πλοήγησης, τα στοιχεία της διεπαφής χρήστη είναι πλέον άσπρα αντί για γαλάζια και στο ρολόι, στην μπάρα κατάστασης, φαίνονται απλώς τα νούμερα με λιτό τρόπο. Οι εφαρμογές επίσης έχουν την δυνατότητα να χρησιμοποιήσουν μια κατάσταση κατά την οποία οι μπάρες κατάστασης και πλοήγησης είναι κρυμμένες, αλλά ο χρήστης διατηρεί την δυνατότητα να αλληλεπιδράσει με τη συσκευή. Έγιναν βελτιώσεις για συσκευές με μικρή μνήμη και προστέθηκε υποστήριξη στο API για συσκευές με μικρή μνήμη, κάτω από 512MB. Υλοποιήθηκε η λειτουργία ασύρματης εκτύπωσης και προστέθηκε η προσομοίωση εξυπηρετητή καρτών για την μορφή επικοινωνίας NFC, επιτρέποντας την αντικατάσταση "έξυπνων" καρτών (NFC Smart Cards). Τα στοιχεία εμφάνισης δεδομένων από το Διαδίκτυο βασίζονται στη μηχανή Chromium στην οποία είναι βασισμένος και ο φυλλομετρητής, δίνοντας έτσι περισσότερες επιλογές στους σχεδιαστές εφαρμογών. Επεκτάθηκε η δυνατότητα στις εφαρμογές να "ακούνε" ειδοποιήσεις και εκδόθηκαν βιβλιοθήκες για τη διαχείριση γραπτών μηνυμάτων. Το πλαίσιο για τις μεταβάσεις της διεπαφής χρήστη, επανασχεδιάστηκε ώστε οι μεταβάσεις να είναι πιο ομαλές και φυσικές. Προστέθηκε ένα API για τη διαχείριση του αποθηκευτικού χώρου, που προσφέρει αδιάκοπη πρόσβαση σε δεδομένα από διαφορετικές πηγές, επιτρέποντας την πρόσβαση και σε αρχεία που γίνονται προσβάσιμα από υπηρεσίες αποθήκευσης δεδομένων στο διαδίκτυο. Από τις ρυθμίσεις μπορεί να επιλεγεί η προεπιλεγμένη εφαρμογή για την εκκίνηση άλλων εφαρμογών (Launcher Application) και για μηνύματα κειμένου. Υποστηρίζεται μια τεχνική κατά την οποία μέρος της επεξεργασίας της μουσικής γίνεται από το υλικό της συσκευής, στην μονάδα επεξεργασίας ψηφιακού σήματος, μειώνοντας την κατανάλωση. Για τους σχεδιαστές εφαρμογών, υπάρχει η δυνατότητα καταγραφής της εικόνας ενώ υποστηρίζεται και η ύπαρξη πομπού υπερύθρων (IR). Επίσης υπάρχει και ένα πειραματικό περιβάλλον εκτέλεσης εφαρμογών, το Android Runtime (ART), που δημιουργήθηκε με την προοπτική να αντικαταστήσει την εικονική μηχανή Dalvik, αλλά δεν είναι ενεργοποιημένο ως προεπιλογή. Στο Bluetootth προστέθηκε το προφίλ πρόσβασης μηνυμάτων (MAP - Message Access Profile) ενώ η πρόσβαση στα στατιστικά της μπαταρίας, από εφαρμογές τρίτων, καταργήθηκε. Οι ενδείξεις για το Wi-Fi και τα δεδομένα κινητής τηλεφωνίας, μετακινήθηκαν στις γρήγορες ρυθμίσεις. Σύντομα εκδόθηκαν οι αναβαθμίσεις 4.4.1 και 4.4.2 με βελτιώσεις στην αυτόματη εστίαση της κάμερας, την εξισορρόπηση του λευκού αλλά και αρκετές διορθώσεις σφαλμάτων. Βελτιώθηκε η υποστήριξη του Android Runtime και αφαιρέθηκε το σύστημα ελέγχου πρόσβασης App Ops που εμφανίστηκε στην έκδοση 4.3, με την ταυτόχρονη διόρθωση σφαλμάτων σε θέματα ασφάλειας. Περίπου 6 μήνες μετά εκδόθηκαν οι αναβαθμίσεις 4.4.3 και 4.4.4 με κυριότερες αλλαγές την αλλαγή της εφαρμογής κλήσεων, τη βελτίωση της εμφάνισης δεδομένων του

διαδικτύου με καλύτερη υποστήριξη HTML5 και μερικές διορθώσεις σε σφάλματα.

- Android 4.4w KitKat - API level 20

  Η έκδοση 4.4w είναι σαν την προηγούμενη με την προσθήκη επέκτασης, για τους προγραμματιστές, για αξεσουάρ που φοριούνται και επικοινωνούν με την συσκευή.

- Android 5.0 Lollipop – API level 21

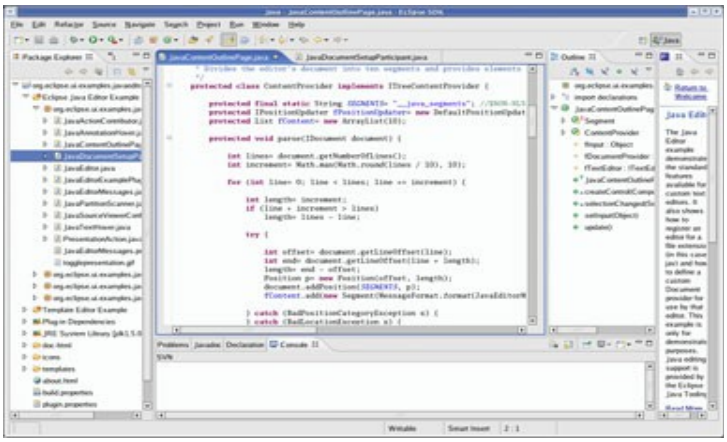
  Στο Lollipop η διεπαφή χρήστη επανασχεδιάστηκε σε γλώσσα σχεδιασμού με πιο γρήγορη απόκριση γνωστή ως Material Design. Συμπεριλαμβάνει βελτιώσεις στις ειδοποιήσεις οι οποίες είναι προσβάσιμες από κλειδωμένη οθόνη και εμφανίζονται μέσα σε άλλες εφαρμογές ως μηνύματα στο πάνω μέρος της οθόνης. Το περιβάλλον Android Runtime (ART) αντικατέστησε επίσημα τη μηχανή Dalvik για βελτιστοποιημένη απόδοση των εφαρμογών με την προσθήκη βελτιστοποιήσεων ως προς την χρήση της μπαταρίας. Το περιβάλλον ART προσφέρει μεταγλώττιση πριν την εκτέλεση (AOT – Ahead Of Time compilation) σε αντίθεση με τη μηχανή Dalvik που μεταγλωττίζει κατά την εκτέλεση. Η νέα έκδοση του λειτουργικού συστήματος υποστηρίζει επεξεργαστές των 64 bit και γραφικά OpenGL ES 3.1 με μονάδες επεξεργασίας γραφικών που το επιτρέπουν. Το μενού με τις πρόσφατες εφαρμογές εμφανίζει και τις καρτέλες κάθε εφαρμογής με όριο καρτελών ανά εφαρμογή που ρυθμίζεται από τον χρήστη. Στην οθόνη, όταν είναι κλειδωμένη η συσκευή, η μπάρα ειδοποιήσεων και οι γρήγορες ρυθμίσεις έχουν ανανεωμένη εμφάνιση ενώ στις ρυθμίσεις προστέθηκε η δυνατότητα αναζήτησης για πιο γρήγορη πρόσβαση. Στην κλειδωμένη οθόνη εμφανίζονται συντομεύσεις για εφαρμογές και τις ρυθμίσεις ειδοποιήσεων. Η προτεραιότητα των ειδοποιήσεων μπορεί να ρυθμιστεί από τον χρήστη. Επιτρέπεται η χρήση της συσκευής από φιλοξενούμενο χρήστη (Guest User), αλλά και η δημιουργία πολλαπλών λογαριασμών χρήστη. Μαζί με την υποστήριξη εξόδου ήχου από συσκευές USB, υποστηρίζεται και η είσοδος ήχου. Οι εφαρμογές τρίτων σχεδιαστών, αποκτούν πάλι το δικαίωμα χρήσης, για ανάγνωση και εγγραφή, οποιουδήποτε μέρους της εξωτερικής μνήμης, όπως κάρτες μνήμης. Η λίστα με τις πρόσφατες εφαρμογές διατηρείται και μετά από επανεκκίνηση της συσκευής. Στην έκδοση αυτή έχει προστεθεί η υποστήριξη για 15 ακόμα γλώσσες. Διευκολύνεται η αλλαγή συσκευής για τον χρήστη, καθώς μέσω NFC και Bluetooth επικοινωνίας, δίνεται η δυνατότητα για την μεταφορά λογαριασμών, ρυθμίσεων, επαφών και άλλων δεδομένων του χρήστη. Στο λειτουργικό έχει προστεθεί η χρήση φακού, στις συσκευές που είναι διαθέσιμος. Η έκδοση 5.0.1 συμπεριλαμβάνει κάποιες διορθώσεις για την αναπαραγωγή βίντεο και την διαχείριση λανθασμένων κωδικών. Τέλος η έκδοση 5.0.2 συμπεριλαμβάνει κάποιες διορθώσεις για ένα σφάλμα που απέτρεπε την εκκαθάριση μνήμης κατά την φόρτιση και αλλαγές για τον τρόπο που οι ειδοποιήσεις ανταγωνίζονται για τους πόρους της συσκευής.

5. Το Περιβάλλον Προγραμματισμού Eclipse

Η πλατφόρμα Eclipse είναι ένα περιβάλλον προγραμματισμού και ανάπτυξης εφαρμογών. Σχεδιασμένη από την αρχή σε ένα μοντέλο που επιτρέπει την ανάπτυξή της και την επέκτασή της ανάλογα με τις ανάγκες του προγραμματιστή. Το μοντέλο χαρακτηρίζεται ως plug-in model και επιτρέπει την γρήγορη ανάπτυξη εργαλείων. Η πλατφόρμα είναι σχεδιασμένη ώστε να λειτουργεί σε πολλά λειτουργικά συστήματα, ενώ τα εργαλεία της, που μπορεί κάποιος να σχεδιάσει, είναι ανεξάρτητα από το λειτουργικό σύστημα του υπολογιστή στον οποίο δημιουργήθηκαν. Έτσι τα εργαλεία μεταφέρονται εύκολα από ένα σύστημα σε ένα άλλο χωρίς να υπάρχει ανάγκη για αλλαγές.

Στον πυρήνα της πλατφόρμας, βρίσκεται ένα σύστημα που ανακαλύπτει, φορτώνει και χρησιμοποιεί με δυναμικό τρόπο τα εργαλεία. Η πλατφόρμα αναλαμβάνει για το κάθε εργαλείο, να διαχειριστεί το μέρος του κώδικα που πρέπει να εκτελεστεί και προσφέρει στον χρήστη ένα απλό περιβάλλον πλοήγησης, μέσα από το οποίο μπορεί να χρησιμοποιήσει τα εργαλεία. Κάθε εργαλείο εκτελεί ένα συγκεκριμένο αριθμό λειτουργιών, όπως αποσφαλμάτωση, αυτόματη διόρθωση του κώδικα, προσθήκη βιβλιοθηκών, εξομοίωση εκτέλεσης κώδικα και μεταγλώττιση κώδικα. Με τη δομή της πλατφόρμας και την εύκολη και γρήγορη δυνατότητα δημιουργίας εργαλείων, υπάρχει μεγάλο πλήθος εργαλείων.

Η αρχιτεκτονική της Eclipse ορίζει τα “σημεία επέκτασης” (extension points) των εργαλείων πάνω στην πλατφόρμα, με τρόπο ώστε να μπορεί να αναπτυχθεί μεγάλος αριθμός εργαλείων χωρίς να επιδρούν το ένα στο άλλο. Η ίδια η πλατφόρμα έχει δημιουργηθεί από επίπεδα εργαλείων που προσφέρουν νέα “σημεία επέκτασης” και συνδέονται στα σημεία επέκτασης των κατώτερων επιπέδων. Κατά την εγκατάσταση η Eclipse προσφέρει συγκεκριμένο αριθμό εργαλείων, αλλά η δομή της επιτρέπει τους προγραμματιστές να επεκτείνουν την πλατφόρμα με νέες λειτουργίες. Το στοιχεία των εργαλείων (αρχεία και άλλα δεδομένα) διαχειρίζονται από ένα κοινό σύστημα διαχείρισης πόρων, προσφέροντας έναν κοινό τρόπο για την χρήση των εργαλείων και ενσωματωμένη στην πλατφόρμα διαχείρισή τους. Έτσι η πλατφόρμα διαχειρίζεται τις πολυπλοκότητες που προκύπτουν από το περιβάλλον στο οποίο εκτελείται, που εξαρτώνται και από το λειτουργικό σύστημα, επιτρέποντας στον προγραμματιστή που αναπτύσσει εργαλεία, να επικεντρωθεί στις συγκεκριμένες λειτουργίες που θέλει να αναπτύξει.



Η Eclipse είναι δομημένη από υποσυστήματα, καθένα εκ των οποίων αποτελείται από ένα ή παραπάνω εργαλεία. Ακολουθούν μερικά από τα κύρια υποσυστήματα της πλατφόρμας.

- **Υποσύστημα του χρόνου εκτέλεσης (Platform Runtime) :** Ορίζει τα σημεία επέκτασης και το μοντέλο των εργαλείων. Ανακαλύπτει δυναμικά τα εργαλεία και διατηρεί πληροφορίες γι' αυτά και τα σημεία επέκτασής τους. Ένα εργαλείο εκτελείται όταν ο χρήστης καλέσει κάποια από τις λειτουργίες του. Το υποσύστημα αυτό είναι υλοποιημένο χρησιμοποιώντας το πλαίσιο OSGi (Open Service Gateway initiative). Κάθε επέκταση πρέπει να ορίζεται μέσα από ένα αρχείο μανιφέστο OSGi με τίτλο MANIFEST.MF και ένα μανιφέστο τύπου xml με τίτλο plugin.xml . Το υποσύστημα λειτουργεί ώστε να μην επιβαρύνει τη μνήμη και τον επεξεργαστή του συστήματος, με εγκατεστημένες επεκτάσεις, παρά μόνο όταν χρησιμοποιούνται. Οι επεκτάσεις ενεργοποιούνται ανάλογα με τις λειτουργίες που καλεί ο χρήστης.

- **Διαχειριστής πόρων (Resource Management / Workspace) :** Ορίζει την διεπαφή προγραμματισμού εφαρμογών (API – Application Programming Interface), τη δημιουργία και τη διαχείριση πόρων, όπως μια εργασία / πρότζεκτ, αρχεία και φακέλους, που δημιουργούνται από τα εργαλεία και διατηρούνται στο σύστημα αρχείων.

- **Πάγκος εργασίας, διεπαφή χρήστη (Workbench UI) :** Υλοποιεί το περιβάλλον χρήσης της πλατφόρμας και ορίζει τα σημεία επέκτασης της διεπαφής χρήστη, για την προσθήκη στοιχείων όπως μενού, παραθύρων και καρτελών. Προσφέρει πρόσθετα εργαλεία για την κατασκευή διεπαφής χρήστη (Jface και SWT). Η επέκταση Jface UI προσφέρει δομές, υψηλού επιπέδου προγραμματισμού, για την υποστήριξη διαλόγων (dialogs), οδηγών (wizards), ρυθμίσεις χρήστη και τη διαχείριση των γραφικών στοιχείων (widget). Η επέκταση SWT (Standard Widget Toolkit) αποτελείται από εργαλεία, χαμηλού επιπέδου προγραμματισμού, ανεξάρτητα από το λειτουργικό σύστημα του υπολογιστή στον οποίο τρέχει η πλατφόρμα. Τα εργαλεία αυτά υποστηρίζουν την ενσωμάτωση και τη φορητότητα των επεκτάσεων. Οι υπηρεσίες διεπαφής χρήστη, είναι δομημένες ώστε ένα υποσύνολο εργαλείων διεπαφής χρήστη να μπορούν να χρησιμοποιηθούν για την ανάπτυξη εφαρμογών ανεξάρτητα από το υποσύστημα διαχείρισης πόρων. Πρόσθετες επεκτάσεις διεπαφής χρήστη ορίζουν πλαίσια που είναι πολύ χρήσιμα για την ανάπτυξη διεπαφών χρήστη. Ο πάγκος εργασίας, ως διεπαφή χρήστη έχει υλοποιηθεί χρησιμοποιώντας τα εργαλεία αυτά. Η χρήση πλαισίων διευκολύνει την δημιουργία επεκτάσεων και προσφέρει ομοιότητα στην εμφάνιση διεπαφών χρήστη, κάνοντας το περιβάλλον ανάπτυξης εφαρμογών οικείο, ανεξάρτητα με τις επεκτάσεις που χρησιμοποιούνται.

- **Υποσύστημα Βοήθειας (Help System) :** Ορίζει τα σημεία επέκτασης για τα εργαλεία που προσφέρουν βοήθεια και κείμενα για την χρήση της πλατφόρμας και των εργαλείων της.

- **Υποστήριξη ομάδας (Team Support) :** Ορίζει ένα μοντέλο ομαδικού προγραμματισμού για την διαχείριση και την ανάπτυξη εκδόσεων για τα αρχεία και τις εφαρμογές που αναπτύσσονται μέσα από την πλατφόρμα. Το πακέτο Eclipse SDK συμπεριλαμβάνει την επέκταση CVS (Concurrent Versioning System), που αναλαμβάνει να διατηρεί ενημερωμένα τα μέλη της ομάδας προγραμματισμού με την τελευταία έκδοση της εφαρμογής.

- **Υποστήριξη αποσφαλμάτωσης (Debug Support) :** Αποτελεί ένα μοντέλο αποσφαλμάτωσης, ανεξάρτητο από την γλώσσα προγραμματισμού, με κλάσεις και βιβλιοθήκες για την ανάπτυξη διεπαφής χρήστη για αποσφαλμάτωση και εκτέλεση κώδικα.

- **Λοιπές λειτουργικότητες (Other Utilities) :** Περιλαμβάνει εργαλεία με λειτουργίες όπως η αναζήτηση, σύγκριση πόρων, δυναμική αναβάθμιση της πλατφόρμα μέσω ίντερνετ και άλλα.

Το πακέτο ανάπτυξης λογισμικού Eclipse SDK (Software Development Kit), πέρα από την βασική πλατφόρμα περιλαμβάνει και δύο βασικά πακέτα εργαλείων. Το ένα αποτελείται από τα εργαλεία ανάπτυξης Java JDT (Java Development Tools), που υλοποιεί ένα περιβάλλον ανάπτυξης εφαρμογών σε Java, ενώ το δεύτερο, υλοποιεί το περιβάλλον ανάπτυξης εργαλείων (PDE – Plug-in Developer Environment). Τα εργαλεία αυτά επιτρέπουν την άμεση χρήση του πακέτου Eclipse SDK, αλλά αποτελούν και ένα τέλειο παράδειγμα για τη δημιουργία εργαλείων επέκτασης της πλατφόρμας.

Παρόλο που η Eclipse είναι γνωστή για την ανάπτυξη εφαρμογών σε Java, με τα διαθέσιμα εργαλεία, μπορεί να χρησιμοποιηθεί για πλήθος λειτουργιών, με υποστήριξη αρκετών γλωσσών προγραμματισμού, μεταξύ των οποίων C και C++, αλλά και για ανάπτυξη και επεξεργασία αρχείων XML, αρχείων κειμένου τύπου Word, αρχείων HTML αλλά και JSP (Java Server Pages).

5.1 Ανάπτυξη Λογισμικού για Android - Android SDK

Το Android SDK αποτελείται από ένα πακέτο εργαλείων για την δημιουργία εφαρμογών για συσκευές Android. Στα εργαλεία συμπεριλαμβάνονται βιβλιοθήκες, εργαλεία αποσφαλμάτωσης, προσομοιωτή, βιβλιογραφία και δείγματα κώδικα μαζί με οδηγίες. Παρόλο που η επίσημα υποστηριζόμενη πλατφόρμα για την ανάπτυξη εφαρμογών σε Android, είναι η Eclipse με την επέκταση εργαλείων ADT (Android Development Tools plug-in) που χρησιμοποιεί το Android SDK, υπάρχουν αρκετά ενσωματωμένα περιβάλλοντα ανάπτυξης εφαρμογών, ακόμα και για λειτουργικό σύστημα Android, επιτρέποντας την ανάπτυξη εφαρμογών στην ίδια τη συσκευή. Καθώς δημοσιεύονται καινούργιες εκδόσεις Android, ενημερώνονται και αναβαθμίζονται και τα εργαλεία για την ανάπτυξη εφαρμογών, ώστε να μπορούν να εκμεταλλευτούν τις καινούργιες λειτουργίες του συστήματος, αλλά και να προσφέρουν νέες λειτουργίες σε παλιότερα συστήματα με βιβλιοθήκες συμβατότητας για προηγούμενες εκδόσεις. Τα εργαλεία επιτρέπουν στον χρήστη να τα αναβαθμίσει μέσω ίντερνετ, αλλά και να κατεβάσει παλιότερες εκδόσεις και βιβλιοθήκες συμβατότητας. Τα εργαλεία αναλαμβάνουν να οργανώσουν την εφαρμογή σε πακέτο .apk που περιλαμβάνει αρχεία μορφής .dex που είναι τα εκτελέσιμα αρχεία για τη μηχανή Dalvik.

Το πακέτο SDK περιλαμβάνει ένα εργαλείο που ονομάζεται γέφυρα αποσφαλμάτωσης Android (ADB – Android Debug Bridge), που αποτελείται από έναν εξυπηρετητή (server) και ένα πελάτη (client) που επικοινωνούν μεταξύ τους. Το ADB χρησιμοποιείται κυρίως από τη γραμμή εντολών, ενώ υπάρχουν γραφικές διεπαφές χρήστη. Προσφέρει λειτουργίες όπως η μεταφορά ενός αρχείου από τον υπολογιστή στη συσκευή μέσω της εντολής push και τη μεταφορά από τη συσκευή στον υπολογιστή μέσω της εντολής pull. Η εντολή shell δίνει στον χειριστή γραμμή εντολών στην συσκευή Android, επιτρέποντας την εκτέλεση εντολών και εφαρμογών. Μέσω της εντολής install, μπορεί κανείς να εγκαταστήσει μια εφαρμογή από το .apk πακέτο της, ενώ η εντολή logcat δίνει στον χειριστή καταγραφές αποσφαλμάτωσης (debug logs) σε πραγματικό χρόνο.

Το εργαλείο fastboot που περιλαμβάνεται στο πακέτο, υλοποιεί ένα διαγνωστικό πρωτόκολλο, που κυρίως χρησιμοποιείται για μεταβολές στο σύστημα αποθήκευσης δεδομένων

τύπου flash, μέσω σύνδεσης USB. Για τη χρήση του εργαλείου, απαιτείται η συσκευή να εκκινήσει με ένα τρόπο που αρχικοποιεί μόνο τις πιο βασικές λειτουργίες της συσκευής. Η εκκίνηση αυτή ονομάζεται Host loader mode. Μετά την εκκίνηση της συσκευής και ενεργοποίηση του πρωτοκόλλου, ο χειριστής μπορεί μέσα από τη γραμμή εντολών να εκτελέσει εντολές που διαμορφώνουν τον αποθηκευτικό χώρο της συσκευής. Οι εντολές αυτές πρέπει να χρησιμοποιούνται με μεγάλη προσοχή καθώς έχουν την δυνατότητα να σβήσουν μέρη της μνήμης της συσκευής. Πιο συγκεκριμένα η εντολή flash, αντιγράφει μια δυαδική εικόνα (binary image) σε ένα τμήμα (partition) της μνήμης. Η εντολή αυτή μπορεί να χρησιμοποιηθεί για να επαναφέρει τη συσκευή σε προηγούμενη αποθηκευμένη κατάσταση. Με την εντολή erase, σβήνει ένα ολόκληρο τμήμα της μνήμης, ενώ με την εντολή format διαμορφώνεται το τμήμα της μνήμης, χάνοντας όλα τα δεδομένα που είχε αποθηκευμένα.

- **Επέκταση της Eclipse με το ADT Plug-in :** Το πακέτο εργαλείων επέκτασης ADT (Android Development Kit) χρειάζεται για την ενσωμάτωση των εργαλείων προγραμματισμού Android εφαρμογών, στην πλατφόρμα και περιβάλλον προγραμματισμού Eclipse. Προσφέρει γραφική διεπαφή για τα περισσότερα από τα εργαλεία του Android SDK, καθώς και μια σειρά από εργαλεία για γρήγορη ανάπτυξη εφαρμογών.

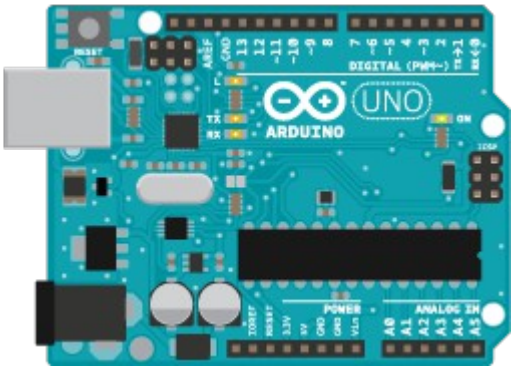
Από τα εργαλεία που ενσωματώνονται, το traceview είναι αυτό που βοηθά στη παρακολούθηση της εκτέλεσης μια εφαρμογής. Το εργαλείο Hierarchy Viewer, προβάλλει την ιεραρχία των γραφικών στοιχείων της εφαρμογής, δίνοντας έναν άμεσο εποπτικό έλεγχο στον προγραμματιστή. Για τη λεπτομερή εξέταση της διεπαφής χρήστη, χρησιμοποιείται το Pixel Perfect. Το DDMS προσφέρει λειτουργίες για την αποσφαλμάτωση, όπως την αποθήκευση της εικόνας, την εμφάνιση πληροφοριών για το κάθε νήμα (thread) και την εμφάνιση καταγραφής logcat. Οι περισσότερες από τις λειτουργίες του εργαλείου adb, ενσωματώνονται στην Eclipse μέσω του ADT Plug-in, ωστόσο υπάρχουν λειτουργίες του εργαλείου που μπορούν μόνο από τη γραμμή εντολών να εκτελεστούν. Για τη βελτιστοποίηση του κώδικα και τη συρρίκνωση του .apk πακέτου, φροντίζει κατά την δημιουργία του το εργαλείο ProGuard, εφόσον είναι ενεργοποιημένο.

Μέρος της επέκτασης αποτελεί και το εργαλείο διαχείρισης Android SDK Manager, που διαχειρίζεται το διάφορα εργαλεία και εκδόσεις του λειτουργικού συστήματος για τα οποία υποστηρίζεται η ανάπτυξη εφαρμογών. Μέσα από το εργαλείο διαχείρισης, ο προγραμματιστής μπορεί να διαλέξει τα εργαλεία που θέλει να κατεβάσει και να εγκαταστήσει, καθώς και τις βιβλιοθήκες για την υποστήριξη λειτουργιών στις εφαρμογές που δημιουργεί. Κάθε καινούρια έκδοση Android SDK γίνεται διαθέσιμη μέσω του εργαλείου διαχείρισης και η εγκατάστασή του είναι πολύ εύκολη.

Ένα ακόμα πολύ χρήσιμο εργαλείο, της επέκτασης, για την ανάπτυξη εφαρμογών είναι ο διαχειριστής εικονικών συσκευών Android, AVD (Android Virtual Device Manager). Μέσα από τον διαχειριστή, ο χρήστης μπορεί να συνθέσει μια εικονική συσκευή, με τα χαρακτηριστικά της συσκευής για την οποία θέλει να αναπτύξει εφαρμογές. Σε αυτήν, μπορεί στη συνέχεια να φορτώσει τις εφαρμογές, για να τις ελέγξει και να προχωρήσει σε αποσφαλμάτωση, χωρίς να χρειαστεί να χρησιμοποιήσει πραγματική συσκευή. Το εργαλείο προσφέρει την δυνατότητα στους προγραμματιστές να ελέγξουν την συμπεριφορά σε πολλές συσκευές με διαφορετικές οθόνες και λοιπά χαρακτηριστικά, ανέξοδα.

6. Arduino – Πλατφόρμα & Περιβάλλον Προγραμματισμού

Το Arduino είναι μια ηλεκτρονική πλατφόρμα που αποτελείται από hardware, μια μικρή ηλεκτρονική πλακέτα αλλά και software. Είναι σχεδιασμένο έτσι ώστε να μπορεί να αποτελέσει το ηλεκτρονικό μέρος για μεγάλο πλήθος δημιουργιών που αλληλεπιδρούν με τον φυσικό κόσμο, καθώς διαθέτει αρκετές εισόδους που μπορούν να "διαβάσουν" αισθητήρες και κουμπιά, αλλά και εξόδους που μπορούν να επικοινωνήσουν με άλλα κυκλώματα ή να ελέγξουν άλλες συσκευές. Το software χρησιμοποιείται για τον προγραμματισμό της πλακέτας σε γλώσσα προγραμματισμού που βασίζεται στην Processing, η οποία έχει βασιστεί στην C. Το hardware είναι μια ηλεκτρονική πλακέτα με βασικά χαρακτηριστικά εισόδους/εξόδους γενικού σκοπού (GPIO), μικρό μέγεθος και ένα μικροεπεξεργαστή της οικογένειας AVR ATMEGA της ATMEL. Ενώ στο software υπάρχει μία επιλογή, υπάρχουν πολλές πλακέτες που προσφέρουν διαφορετικές δυνατότητες. Η βασική πλακέτα, που είναι και η πιο κοινή, το Arduino UNO, έχει θύρα USB που συνδέεται σε ένα ολοκληρωμένο FTDI για τη μετατροπή του USB σήματος σε σειριακό, κεραμικό ταλαντωτή 16MHz και ένα Atmega328. Ο μικροεπεξεργαστής αυτός προσφέρει 14 ψηφιακές εισόδους/εξόδους, 6 από τις οποίες μπορούν να χρησιμοποιηθούν ως έξοδοι PWM, 6 αναλογικές εισόδους και σειριακή UART επικοινωνία. Άλλες πλακέτες προσφέρουν περισσότερες δυνατότητες όπως σύνδεση Ethernet, δεύτερη θύρα σειριακής επικοινωνίας και USB host interface. Άλλες προσφέρουν ακόμα λιγότερες, γεγονός που τις κάνει πιο πολύπλοκες στη χρήση, αλλά πιο μικρές και οικονομικές.



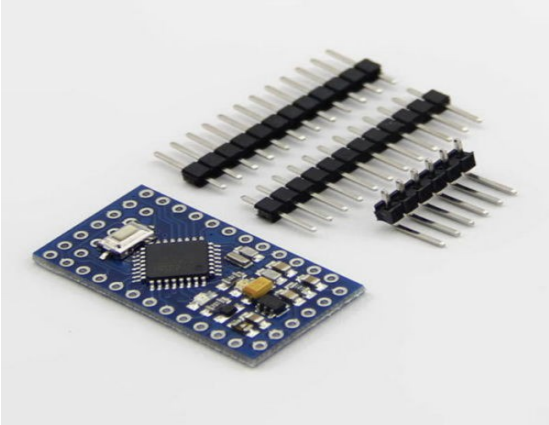
Το Arduino αποτελεί πολύ δημοφιλή επιλογή ανάμεσα σε ερασιτέχνες αφού είναι πολύ οικονομική λύση. Η πλακέτα μπορεί είτε να αγοραστεί έτοιμη, είτε να δημιουργηθεί από τα σχέδια που διατίθενται ελεύθερα. Αυτό δίνει και τη δυνατότητα στους προχωρημένους χρήστες να δημιουργήσουν δικές τους παραλλαγές και να προσαρμόσουν την πλατφόρμα στην δικιά τους εφαρμογή. Επίσης, δίνει την δυνατότητα σε κατασκευαστές να δημιουργήσουν πρόσθετες πλακέτες, που με ευκολία αλληλεπιδρούν με το Arduino και επεκτείνουν τις δυνατότητες της πλατφόρμας. Έτσι έχουν δημιουργηθεί πολλές πλακέτες επέκτασης της πλατφόρμας που χαρακτηρίζονται ως "Arduino compatible shield". Πολλές από αυτές δεν χρειάζονται ούτε καν καλώδια για να συνδεθούν με το Arduino καθώς διαθέτουν ακίδες που ταιριάζουν άμεσα στις υποδοχές του Arduino. Οι κατασκευαστές των πρόσθετων αυτών πλακετών συνήθως τις συνοδεύουν από βιβλιοθήκες "ανοιχτού κώδικα", όπως ταιριάζει στη φιλοσοφία με την οποία είναι φτιαγμένη η πλατφόρμα. Μαζί με τις βιβλιοθήκες προσφέρονται παραδείγματα, έτοιμες εφαρμογές και οδηγίες. Συνήθως, καινούριες εκδόσεις του Arduino IDE προσφέρουν περισσότερες βιβλιοθήκες και παραδείγματα, κάνοντας έτσι ακόμα πιο εύκολη την χρήση των περισσότερων πλακετών επέκτασης.

Ο μικροεπεξεργαστής AVR που χρησιμοποιείται στην πλατφόρμα χρειάζεται να είναι

“φορτωμένος” με ένα μικρό πρόγραμμα που ονομάζεται bootloader, το οποίο λειτουργεί έτσι ώστε να βοηθάει να φορτωθεί το πρόγραμμα από το Arduino IDE μέσω της σειριακής θύρας χωρίς την χρήση τις ISP (In System Programming) θύρας, που χρειάζεται περισσότερες συνδέσεις. Αυτό απλοποιεί την πλατφόρμα, τον προγραμματισμό αυτής, αλλά δεσμεύει λίγη από τη μνήμη της συσκευής. Για τους προχωρημένους χρήστες, είναι δυνατό να σβηστεί αυτό το πρόγραμμα από την συσκευή, ελευθερώνοντας λίγη μνήμη ακόμα, αλλά για τον προγραμματισμό της χρειάζεται ένας ISP προγραμματιστής. Ένα από τα παραδείγματα που διατίθενται με το πρόγραμμα, μπορεί να μετατρέψει το ίδιο το Arduino σε ISP προγραμματιστή, με σκοπό να προγραμματίσει άλλες Arduino συσκευές, αλλά όχι μόνο. Μετατρέποντας το Arduino σε ISP προγραμματιστή μπορεί κανείς να προγραμματίσει πλατφόρμες που βασίζονται στην οικογένεια μικροεπεξεργαστών AVR, καθώς η ιδιότητα προγραμματισμού μέσω ISP αποτελεί χαρακτηριστικό των μικροεπεξεργαστών AVR και όχι του Arduino. Τέλος μπορεί κανείς να προγραμματίσει AVR μικροεπεξεργαστές με το bootloader της πλατφόρμας, ώστε να μπορεί στη συνέχεια να χρησιμοποιήσει την σειριακή θύρα για τον προγραμματισμό τους.

- **Arduino Pro Mini**

Το Arduino Pro Mini, είναι το πιο μικρό Arduino, αφού πέρα από το μικροεπεξεργαστή της σειράς Atmega, στο οποίο είναι βασισμένο, διαθέτει μόνο ένα κρυσταλλικό ταλαντωτή, ένα διακόπτη επανεκκίνησης (reset) κι έναν regulator. Το Pro Mini ήταν αρχικά σχεδιασμένο με το Atmega128, αλλά στη συνέχεια έγινε διαθέσιμο και με το Atmega328, προσφέροντας μεγαλύτερη χωρητικότητα. Επίσης, υπάρχουν 2 επιλογές στη συχνότητα λειτουργίας του Pro Mini. Η πρώτη είναι στα 8MHz και τροφοδοτείται με 3.3V, ενώ η δεύτερη στα 16MHz και χρειάζεται 5V τροφοδοσίας. Η πλακέτα είναι σχεδιασμένη ώστε να έχει όσο γίνεται μικρότερες διαστάσεις αλλά και το μικρότερο κόστος, ώστε να μπορεί να αποτελέσει εύκολα μέρος μιας κατασκευής. Αυτό έχει ως αποτέλεσμα να είναι πιο πολύπλοκη η χρήση του, καθώς για τον προγραμματισμό του χρειάζεται μια πρόσθετη συσκευή που αναλαμβάνει την επικοινωνία μεταξύ του υπολογιστή και της συσκευής. Για την χρήση της συσκευής, απαιτείται η χρήση κολλητηριού αφού η συσκευή δεν διαθέτει υποδοχές ή “ποδαράκια” για την σύνδεση της αλλά υπάρχει η δυνατότητα να τα προσθέσει ο χρήστης προσθέτοντας ευελιξία αλλά και πολυπλοκότητα στην πλατφόρμα.



Για τον προγραμματισμό της πλακέτας υπάρχουν δύο επιλογές. Η πιο πολύπλοκη και χρονοβόρα είναι η χρήση ενός προγραμματιστή ISP (In System Programming) που δίνει την δυνατότητα προγραμματισμού του ATmega με 6 αγωγούς. Η δεύτερη επιλογή, που είναι και πιο εύκολη, απαιτεί την χρήση ενός μετατροπέα από USB σε σειριακή επικοινωνία σε στάθμη TTL.

Η μνήμη της πλατφόρμας είναι 16KB μνήμη τύπου Flash για την έκδοση με το ATmega128 και 32KB με το ATmega328 εκ των οποίων τα 2KB χρησιμοποιούνται από το bootloader, που

τρέχει κάθε φορά που γίνεται επανεκκίνηση στην συσκευή, πριν από οποιοδήποτε άλλο κώδικα. Το ATmega128 προσφέρει 1KB SRAM και 512KB EEPROM ενώ το ATmega328 προσφέρει τις διπλάσιες. Η EEPROM μπορεί να χρησιμοποιηθεί για αποθήκευση δεδομένων που διατηρούνται όταν αφαιρεθεί η τροφοδοσία.

Ο μικροεπεξεργαστής είναι αρχιτεκτονικής RISK των 8 bit, διαθέτει 32 γενικού σκοπού καταχωρητές (8 bit ο καθένας) και σετ 131 εντολών. Διαθέτει επίσης δύο μετρητές-χρονιστές των 8 bit και έναν των 16 που μπορούν να χρησιμοποιηθούν ανεξάρτητα και δίνουν τη δυνατότητα σύγκρισης με κάποιον αριθμό, ενημερώνοντας τον κατάλληλο καταχωρητή-σημαία (Flag Register) και προκαλώντας την προγραμματισμένη διακοπή στο πρόγραμμα. Αυτό δίνει την δυνατότητα συγχρονισμένων λειτουργιών με υψηλή ακρίβεια, ενώ το υπόλοιπο πρόγραμμα μπορεί να εκτελείται κανονικά, μέχρι να έρθει η ώρα να εκτελεστεί η λειτουργία που απαιτεί συγχρονισμό.

Οι δυνατότητες του μικροεπεξεργαστή επεκτείνονται σε 6 κανάλια (εξόδου) διαμόρφωσης εύρους παλμών (PWM – Pulse Width Modulation) καθώς και 8 μετατροπείς αναλογικού σε ψηφιακού σήματος (ADC – Analogue to Digital Converter) των 10 bit. Τα κανάλια διαμόρφωσης εύρους παλμών, δίνουν την δυνατότητα ελέγχου συσκευών, όπως την ταχύτητα ενός κινητήρα ή την φωτεινότητα μιας διόδου φωτοεκπομπής (LED). Τα ADC κανάλια δίνουν την δυνατότητα να μετατρέπει την αναλογική τιμή κάποιου αισθητήρα σε ψηφιακή, όπως για παράδειγμα ενός αισθητήρα θερμοκρασίας ή πίεσης και στη συνέχεια να επεξεργάζεται αυτό το σήμα. Αυτές οι λειτουργίες δίνουν εύκολα υλοποιήσιμες και γρήγορες λύσεις για την αλληλεπίδραση της πλατφόρμας με τον φυσικό κόσμο και το περιβάλλον.

Η αλληλεπίδραση της πλατφόρμας επεκτείνεται και από τις δυνατότητες επικοινωνίας που διαθέτει. Μια από τις δυνατότητες επικοινωνίας που διαθέτει είναι η σειριακή περιφερειακή διεπαφή SPI (Serial Peripheral Interface). Η SPI είναι μια μορφή σύγχρονης σειριακής επικοινωνίας, πολύ διαδεδομένης, που απαιτεί έναν “αφέντη”, ενώ οι περιφερειακές συσκευές λειτουργούν ως “σκλάβοι”. Χρησιμοποιείται σε μικρές αποστάσεις και είναι αμφίδρομη. Συχνά την συναντούμε σε αισθητήρες και κάρτες μνήμης που αποτελούν τους “σκλάβους”, απαντώντας στα αιτήματα του “αφέντη” μικροεπεξεργαστή. Στην ίδια διεπαφή, μπορούν να συνδεθούν πολλές συσκευές ως “σκλάβοι” αλλά ο “αφέντης” πρέπει πάντα να είναι ένας. Για την SPI απαιτούνται 4 αγωγοί εκ των οποίων ο ένας είναι το ρολόι SCLK (Serial Clock), αφού η επικοινωνία είναι σύγχρονη, που οδηγείται από τον “αφέντη”. Οι άλλοι τρεις αποτελούνται από την έξοδο του αφέντη προς τους “σκλάβους” MOSI (Master Output Slave Input), την είσοδο του “αφέντη” από τους “σκλάβους” MISO (Master Input Slave Output) και τον επιλογέα “σκλάβου” SS (Slave Select). Όταν η επαφή SS μιας συσκευής “σκλάβου” βρεθεί σε χαμηλό δυναμικό, αυτή ενεργοποιείται και είναι σε θέση να δεχτεί μηνύματα από τον αφέντη, αλλιώς η έξοδος του “σκλάβου” εμφανίζει υψηλή εμπέδηση (λογική τριών καταστάσεων, tri-state). Αυτό επιτρέπει σε πολλές συσκευές “σκλάβους” να συνδεθούν με έναν αφέντη παράλληλα, δηλαδή οι επαφές SCLK, MOSI και MISO να είναι κοινές. Μέσω του επιλογέα SS, ο αφέντης διαλέγει με ποιόν “σκλάβο” θα επικοινωνήσει.

Η οικογένεια επεξεργαστών ATmega, διαθέτει την δυνατότητα επικοινωνίας με το πρωτόκολλο I^2C (Inter-Integrated Circuit). Το πρωτόκολλο αυτό δημιουργήθηκε από την Philips Semiconductors, που στη συνέχεια μετονομάστηκε σε NXP Semiconductors. Το ISP επιτρέπει την επικοινωνία πολλών συσκευών μέσω δύο μόνο αγωγών, στους οποίους συνδέονται με κύκλωμα ανοικτού συλλέκτη. Οι δύο αγωγοί συνδέονται μέσω αντιστάσεων στην τάση τροφοδοσίας, συνήθως 5V ή 3.3V κι έτσι βρίσκονται σε ψηλό δυναμικό, μέχρι κάποιος κόμβος να το τραβήξει χαμηλά μέσω του συλλέκτη με τον οποίο συνδέεται στον αγωγό. Ο ένας αγωγός χρησιμοποιείται για τον συγχρονισμό και λέγεται SCL (Serial Clock Line), ενώ η γραμμή σειριακών δεδομένων SDL (Serial Data Line) χρησιμοποιείται για την μεταφορά δεδομένων από και προς τον “αφέντη”. Ο “αφέντης” ελέγχει το σήμα συγχρονισμού και απευθύνεται στους σκλάβους μέσω της διεύθυνσης του καθενός που αποτελείται από 7 bit. Στο δίκτυο επικοινωνίας SPI μπορούν να συνυπάρχουν πολλοί κόμβοι “αφέντες” και υπάρχει η δυνατότητα οι “αφέντες” και οι “σκλάβοι” να αλλάξουν

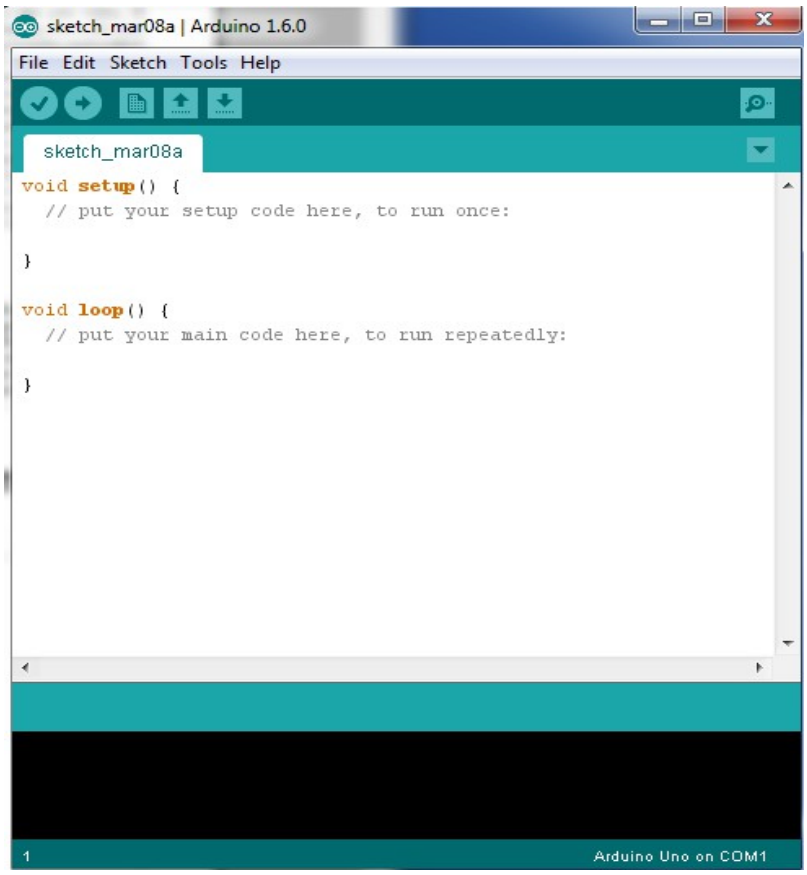
ρόλους. Για κάθε κόμβο οι πιθανές ενεργές καταστάσεις είναι τέσσερις : α) "αφέντης" σε κατάσταση αποστολής δεδομένων, β) "αφέντης" σε κατάσταση λήψης δεδομένων, γ) "σκλάβος" σε κατάσταση αποστολής ή δ) "σκλάβος" σε κατάσταση λήψης. Όταν ένας κόμβος "αφέντης" θέλει να ξεκινήσει μια επικοινωνία, στέλνει ένα bit έναρξης ακολουθούμενο από τα 7 bit διεύθυνσης του "σκλάβου" παραλήπτη και τέλος ένα bit που χαρακτηρίζει εάν θέλει να στείλει δεδομένα στον σκλάβο, εγγραφή (write – 0 bit) ή αν θέλει να διαβάσει (read – 1 bit). Αν υπάρχει "σκλάβος" με διεύθυνση αυτή που ορίζεται από το μήνυμα του "αφέντη", απαντάει με ένα bit αποδοχής (ACK bit). Ο "αφέντης" περνάει σε κατάσταση λήψης ή αποστολής ανάλογα με το μήνυμα που έχει στείλει και ο "σκλάβος" στην συμπληρωματική αυτής ώστε να προχωρήσει η επικοινωνία με το περισσότερο σημαντικό bit πρώτο. Το bit έναρξης (Start bit) αποτελείται από μεταβολή υψηλής σε χαμηλή στάθμη της γραμμής δεδομένων όταν η γραμμή συγχρονισμού βρίσκεται σε υψηλή στάθμη. Το bit λήξης (Stop bit) υλοποιείται από μετάβαση από χαμηλή σε υψηλή στάθμη στην γραμμή μεταφοράς δεδομένων την ώρα που η γραμμή συγχρονισμού βρίσκεται σε υψηλή στάθμη. Όλες οι άλλες μεταβολές στην γραμμή δεδομένων συμβαίνουν σε χρονικές στιγμές που η γραμμή συγχρονισμού βρίσκεται σε χαμηλή στάθμη.

Ο μικροεπεξεργαστής ενσωματώνει και μια μονάδα USART (Universal Synchronous/Asynchronous Receiver/Transmitter), η οποία χρησιμοποιείται για μια από τις πιο διαδεδομένες μορφές επικοινωνίας. Ο πομπός δέχεται τα δεδομένα σε Byte και στέλνει κάθε bit ξεχωριστά με σειριακό τρόπο. Ο δέκτης που λαμβάνει τα bit, τα ομαδοποιεί πάλι ανά Byte. Για τον λόγο αυτό, κάθε μονάδα USART χρησιμοποιεί έναν καταχωρητή ολίσθησης, που είναι βασική μονάδα για την εναλλαγή μεταφοράς δεδομένων από σειριακή σε παράλληλη και αντίστροφα. Στην επικοινωνία μέσω της μονάδας USART δεν υπάρχει η σχέση αφέντη-σκλάβου και οι δύο συσκευές που είναι συνδεδεμένες μπορούν να στείλουν και να λάβουν δεδομένα ταυτόχρονα. Η μεταφορά δεδομένων γίνεται σε έναν αγωγό ανά κατεύθυνση ο οποίος συνδέει το πομπό μίας διεπαφής με τον δέκτη μιας άλλης. Για να συνδεθούν δύο συσκευές αμφίδρομα λοιπόν, χρειάζονται τουλάχιστον 3 αγωγοί με τον τρίτο να είναι η γείωση. Η μορφή αυτής της επικοινωνίας για διάφορους λόγους και αποστάσεις ικανοποιώντας διαφορετικές απαιτήσεις. Ανάλογα με τις απαιτήσεις αυτές, χρησιμοποιείται και διαφορετικό πρωτόκολλο που ορίζει την σηματοδοσία. Μερικά από τα πιο γνωστά παραδείγματα είναι το RS-232, RS-422 και RS-485. Στην περίπτωση που η έξοδος του μικροεπεξεργαστή ATmega χρησιμοποιείται κατευθείαν η σηματοδοσία χαρακτηρίζεται ως TTL (Transistor – Transistor Logic), με το χαμηλό δυναμικό να κυμαίνεται μεταξύ 0 και 0,8 V , ενώ το υψηλό δυναμικό από 2 ως 5 V, όπως συμβαίνει και στο Arduino.

Η USART δίνει αρκετές δυνατότητες στον χρήστη, όπως τον αριθμό bit ανά πλαίσιο μετάδοσης, την δυνατότητα χρήσης bit ισοτιμίας και την χρήση ενός ή δύο bit λήξης. Η αποστολή κάθε πλαισίου ξεκινάει με το bit έναρξης (Start Bit) και τελειώνει με ένα ή δύο bit λήξης (Stop Bits) ανάλογα με την επιλογή του χρήστη. Μετά τον χαρακτήρα έναρξης ακολουθούν τα bit δεδομένων, πάντα με το λιγότερο σημαντικό bit πρώτο. Η διεπαφή μπορεί να προγραμματιστεί ώστε να στέλνει και να λαμβάνει από 5 ως 9 bit δεδομένων ανά πλαίσιο. Στη συνέχεια αποστέλλεται ένα bit ισοτιμίας, αν έχει επιλεγεί από τον χρήστη να χρησιμοποιηθεί και τα bit ανά πλαίσιο είναι λιγότερα από 9. Το bit ισοτιμίας χρησιμοποιείται για την ανίχνευση λαθών και είναι είτε άρτιας ισοτιμίας είτε περιττής, όπως ορίζεται από τον χρήστη. Σε περίπτωση που δεν αποστέλλονται χαρακτήρες, η γραμμή επικοινωνίας διατηρείται σε υψηλό δυναμικό. Το bit έναρξης είναι ένα λογικό 0 (χαμηλή στάθμη) και το/τα bit λήξης λογικό 1 (υψηλή στάθμη). Έτσι σε κάθε αποστολή ενός χαρακτήρα (Byte) υπάρχουν τουλάχιστον δύο αλλαγές στάθμης (από υψηλή σε χαμηλή στάθμη, bit έναρξης και από χαμηλή σε υψηλή, bit λήξης), δίνοντας την δυνατότητα στην διεπαφή του δέκτη να αναγνωρίσει κάποια ζημιά στον αγωγό επικοινωνίας σε περίπτωση που αυτός διατηρηθεί σε χαμηλό δυναμικό για διάρκεια μεγαλύτερη από αυτή που χρειάζεται για την αποστολή ενός χαρακτήρα.

- **Περιβάλλον Προγραμματισμού**

Το Arduino IDE διατίθεται δωρεάν αφού είναι πρόγραμμα "ανοιχτού κώδικα", γεγονός που το κάνει και επεκτάσιμο. Καθώς είναι βασισμένο στο περιβάλλον προγραμματισμού Processing, προσφέρει οικείο περιβάλλον σε όσους το έχουν χρησιμοποιήσει. Είναι κατάλληλο να χρησιμοποιηθεί για εκπαιδευτικούς σκοπούς, αφού είναι εύκολο στη χρήση και την εγκατάσταση, αλλά και διαθέσιμο για Windows, Linux και Macintosh OSX. Διαθέτει μεγάλο πλήθος βιβλιοθηκών με έτοιμες συναρτήσεις, σαφείς οδηγίες για την χρήση αυτών και μεγάλη κοινότητα που υποστηρίζει την πλατφόρμα με παραδείγματα, οδηγίες και συμβουλές.



Μέρος Β : Υλοποίηση

7. Υλοποίηση

7.1 Σχεδιασμός Α' Φάση

Στην προσπάθεια υλοποίησης μιας συσκευής για την παραγωγή σήματος IR, η πρώτη και ίσως πιο απλή ιδέα είναι η χρήση μιας μονάδας μετατροπής επικοινωνίας Bluetooth σε σειριακή UART. Στη μονάδα, τα δεδομένα που λαμβάνονται μέσω Bluetooth, προωθούνται στη σειριακή έξοδο Tx. Από εκεί τροφοδοτούν τον πομπό IR αφού ενισχυθούν κατάλληλα. Σε αυτήν την απλή μορφή, η συσκευή που μεσολαβεί ανάμεσα στο έξυπνο κινητό και τη συσκευή που έχει ο χρήστης σκοπό να χειριστεί, δεν απαιτεί κάποιο είδος προγραμματισμού, πέρα από τον καθορισμό των χαρακτηριστικών της σειριακής θύρας, όπως η ταχύτητα επικοινωνίας, ο αριθμός των ψηφίων, περιττή ή άρτια ισοτιμία και ο αριθμός των ψηφίων λήξης.

Όλα τα παραπάνω χαρακτηριστικά, μπορούν να επιλεχθούν μέσω Bluetooth. Έτσι το μεγαλύτερο μέρος της υλοποίησης, αφορά στην δημιουργία της εφαρμογής. Ανάλογα με το πρωτόκολλο IR που πρέπει να χρησιμοποιηθεί, η εφαρμογή επιλέγει τις κατάλληλες ρυθμίσεις. Στη συνέχεια, ανάλογα με το κουμπί που πατάει ο χρήστης στέλνει μέσω Bluetooth την κατάλληλη ακολουθία ψηφίων, που θα παράξουν το απαιτούμενο IR σήμα.

Στην σειριακή UART επικοινωνία, τα δεδομένα στέλνονται σε πακέτα των 5, 6, 7 ή 8 ψηφίων με ένα ψηφίο έναρξης υψηλής στάθμης, που αντιστοιχεί στο λογικό μηδέν καθώς υπάρχει αρνητική αντιστοίχιση (λογικό ένα / χαμηλή στάθμη , λογικό μηδέν / υψηλή στάθμη). Στο πακέτο, μετά τα δεδομένα μπορεί προαιρετικά να αποσταλεί ένα ψηφίο ισοτιμίας, για έλεγχο σφαλμάτων. Τέλος ακολουθούν 1 ή 2 ψηφία λήξης χαμηλής στάθμης (λογικό ένα). Παρόλο που κάποιες συσκευές μπορούν να υποστηρίξουν και άλλους ρυθμούς μετάδοσης, οι παρακάτω ρυθμοί είναι στις περισσότερες συσκευές διαθέσιμοι και συνηθίζονται να χρησιμοποιούνται:

A/A	Baud Rate
1	300
2	600
3	1200
4	2400
5	4800
6	9600
7	14400
8	19200
9	28800
10	38400
11	57600
12	115200

Από τα παραπάνω προκύπτει ότι αν θέλουμε στην έξοδο της θύρας επικοινωνίας UART, να παράγουμε σήμα κοντά στα 38Khz, που όπως είδαμε είναι η συχνότητα για τα περισσότερα πρωτόκολλα επικοινωνίας, δεν έχουμε παρά να επιλέξουμε ταχύτητα επικοινωνίας 115200 ψηφία ανά δευτερόλεπτο με 7 ψηφία ωφέλιμο φορτίο, χωρίς ψηφίο ισοτιμίας, ένα ψηφίο λήξης και να στείλουμε την ακολουθία 1101101. Με τα ψηφία έναρξης, αυτή θα γίνει 0-1101101-1. Το σήμα που

προκύπτει στην έξοδο Tx, είναι περιοδικό με περίοδο 3 ψηφίων. Καθώς το μηδέν αντιστοιχεί σε υψηλό δυναμικό και το 1 σε χαμηλό, το σήμα έχει υψηλό δυναμικό στο 1/3 της περιόδου, άρα duty cycle 33%. Η συχνότητα που προκύπτει είναι f=115200/3 Hz = 38400Hz = 38.4 Khz

Σε κάθε πακέτο περιλαμβάνονται 3 περίοδοι του φέροντος κύματος και διαρκεί περίπου 78,125 μs. Έτσι ανάλογα με τη διάρκεια t που θέλουμε να πετύχουμε, πρέπει να στείλουμε την ακολουθία t/78,125μs φορές. Για να δημιουργήσουμε παλμό 500 μs του φέροντος, στέλνουμε την ακολουθία 5 φορές.

Για την απουσία του φέροντος, υπάρχουν δύο τρόποι. Ο πρώτος και πιο απλός είναι η διακοπή αποστολής για συγκεκριμένο χρονικό διάστημα. Ο δεύτερος τρόπος είναι η αποστολή δεδομένων που είναι είτε μηδέν, είτε ένα. Και οι δύο παραπάνω τρόποι, όπως θα δούμε παρακάτω, προκαλούν λάθη στην αποστολή και οδηγούν σε εναλλακτικό σχεδιασμό.

Ξεκινώντας την κατασκευή, προχώρησα στην προσπάθεια τηλεχειρισμού τηλεόρασης από τον υπολογιστή. Στην αρχή έκανα δοκιμές με μηδενικά για ωφέλιμο φορτίο για τις στιγμές που χρειαζόμουν απουσία παλμού, αλλά λόγω της αποτυχίας, προχώρησα σε δοκιμές με χρονικές καθυστερήσεις. Στη δεύτερη περίπτωση υπήρχαν αρκετές επιτυχίες, αλλά γενικά έγινε εμφανές ότι χρειαζόμουν έναν τρόπο να “διαβάζω” το σήμα IR που εκπέμπεται.

7.2 Αναγνώστης – Ελεγκτής

Για τη λήψη σήματος IR χρησιμοποίησα ένα δέκτη TSOP48xx συνδεδεμένο σε μια είσοδο ενός Arduino. Καθώς ο δέκτης εμφανίζει χαμηλό δυναμικό όταν λαμβάνει σήμα της φέρουσας συχνότητας και υψηλό όταν δεν λαμβάνει, το “διάβασμα” του λαμβανόμενου σήματος, απλοποιείται στην χρονομέτρηση της εξόδου του δέκτη TSOP48xx κατά τα διαστήματα που έχει υψηλή και χαμηλή στάθμη. Οι χρόνοι που προκύπτουν αποστέλλονται στον υπολογιστή μέσω της σειριακής θύρας. Για να δει κανείς τα αποτελέσματα χρειάζεται να ανοίξει ένα παράθυρο παρακολούθησης της σειριακής θύρας. Τέτοιο εργαλείο περιλαμβάνεται και στο Arduino IDE παρόλο που δεν είναι απαραίτητο να χρησιμοποιηθεί αυτό.

Από την επεξεργασία των αποτελεσμάτων του αναγνώστη, προέκυψε ότι για την απουσία φέρουσας συχνότητας δεν μπορούν να χρησιμοποιηθούν 0 ή 1 ως ωφέλιμο φορτίο σειριακών πακέτων. Πιο συγκεκριμένα, η χρήση του λογικού μηδέν οδηγεί σε ένα πακέτο της μορφής 0-0000000-1 μαζί με τα ψηφία έναρξης και λήξης. Το πακέτο που προκύπτει, όταν όλα τα δεδομένα είναι το λογικό 1, είναι της μορφή 0-1111111-1. Καθώς και στα δύο σήματα υπάρχει μια αλλαγή μέσα στο πακέτο, υπάρχει τουλάχιστον μια περίοδος του φέροντος σε κάθε πακέτο, δηλαδή μία περίοδος κάθε 78,125μs. Αυτό δε θα δημιουργούσε πρόβλημα αν θα μπορούσε να απορριφθεί από τον δέκτη, αλλά από το φύλλο δεδομένων του δέκτη TSOP48xx προκύπτει ότι το σήμα δεν μπορεί να απορριφθεί γιατί πρέπει να μεσολαβήσουν τουλάχιστον 10 κύκλοι της φέρουσας συχνότητας. Έτσι το σήμα που προκύπτει εμφανίζει υψηλό δυναμικό στην έξοδο του δέκτη, μέχρι τη διακοπή αποστολής χαρακτήρων.

Από τα παραπάνω έγινε εμφανές ότι έπρεπε να χρησιμοποιηθεί η διακοπή αποστολής για συγκεκριμένα χρονικά διαστήματα. Προχωρώντας σε μια υλοποίηση με διακοπές και με τη βοήθεια του αναγνώστη, επιτεύχθηκε το ζητούμενο αποτέλεσμα. Μετά από το “διάβασμα” του σήματος με τον αναγνώστη, με την αποστολή των κατάλληλων χαρακτήρων και τις κατάλληλες διακοπές, επιτεύχθηκε ο έλεγχος μιας τηλεόρασης. Σειρά είχαν οι δοκιμές μέσω Bluetooth.

7.3 Bluetooth σε Σειριακή Επικοινωνία.

Έχοντας στη διάθεσή μου ένα BlueSmirf, μια συσκευή μόντεμ Bluetooth σε σειριακή

επικοινωνία, η ιδέα της υλοποίησης ήταν να γίνει η αποστολή των κατάλληλων χαρακτήρων μέσω Bluetooth. Η συσκευή BlueSmirf, στέλνει τα δεδομένα που λαμβάνει από την διεπαφή Bluetooth στον εκπομπό Tx της σειριακής θύρας, ενώ τα δεδομένα που λαμβάνει από τη σειριακή θύρα μέσω της επαφής Rx, τα προωθεί μέσω Bluetooth.

Για τη συνέχεια των δοκιμών, χωρίς πολλές αλλαγές στον αναγνώστη, τους χαρακτήρες προς αποστολή αλλά και τα διαστήματα ανάμεσά τους, ένωσα το κύκλωμα του εκπομπού IR στη σειριακή έξοδο του BlueSmirf και έστειλα τα δεδομένα μέσω Bluetooth. Προς μεγάλη μου έκπληξη, τα αποτελέσματα ήταν πολύ ασταθή, άλλοτε εκμηδενίζοντας τις απαιτούμενες διακοπές ανάμεσα στους παλμούς κι άλλοτε επεκτείνοντάς τες.

Η αιτία για την ασταθή αυτή λειτουργία, είναι η ομαδοποίηση των δεδομένων και αποστολή τους μέσω Bluetooth σε πακέτα κατά πολύ μεγαλύτερα των πακέτων της σειριακής θύρας. Αυτό σημαίνει ότι καθώς εισέρχονται στη διεπαφή Bluetooth τα πακέτα που προορίζονται για αποστολή, αποθηκεύονται σε έναν καταχωρητή. Κατά την αποθήκευσή τους στον καταχωρητή, οι μικρές διακοπές που υπάρχουν κατά την παραγωγή τους, εξαλείφονται, αφού τα μικρά πακέτα των 9 ψηφίων (μαζί με τα ψηφία έναρξης και λήξης) τοποθετούνται το ένα δίπλα στο άλλο σε ένα μεγαλύτερο πακέτο. Το μεγαλύτερο πακέτο λαμβάνεται από τη συσκευή BlueSmirf και τα μικρά πακέτα αποστέλλονται συνεχόμενα στη σειριακή έξοδο. Όταν κάποιο μικρό πακέτο της σειριακής επικοινωνίας, ξεκινάει ένα καινούριο πακέτο Bluetooth, αυτό αποκτά μεγαλύτερη καθυστέρηση από το προηγούμενό του.

Από τα παραπάνω προκύπτει ότι παρόλο που η υλοποίηση τηλεχειριστηρίου μέσω σειριακής θύρας υπολογιστή, μπορεί να υλοποιηθεί με αποστολή χαρακτήρων και κατάλληλες διακοπές, το ίδιο δεν μπορεί να γίνει με τη χρήση διεπαφής Bluetooth. Για τη λύση του προβλήματος, χρησιμοποιήθηκε ένας μικροεπεξεργαστής της οικογένειας AVR.

7.4 Επανασχεδιασμός

Μετά από τα παραπάνω προβλήματα που προέκυψαν, η συσκευή επανασχεδιάστηκε ώστε να μπορεί να παράγει τα απαραίτητα σήματα IR. Για τον λόγο αυτό χρησιμοποιήθηκε ένας μικροεπεξεργαστής.

Ο μικροεπεξεργαστής, για να μπορέσει να δημιουργήσει τα κατάλληλα σήματα, πρέπει να γνωρίζει το πρωτόκολλο με το οποίο θα επικοινωνήσει με τον δέκτη υπερύθρων. Ταυτόχρονα, ο πομπός θα πρέπει να μπορεί να χρησιμοποιεί όλα τα διαθέσιμα πρωτόκολλα, αλλά και να μπορεί να αναβαθμιστεί με τα καινούρια. Στην αρχή η καινούργια συσκευή σχεδιάστηκε με τα πρωτόκολλα αποθηκευμένα στην μνήμη της, αλλά ο σχεδιασμός αυτός απαιτούσε αναβάθμιση της συσκευής κάθε φορά που θέλει ο χρήστης να προσθέσει ένα πρωτόκολλο IR. Ταυτόχρονα απαιτείται και η αναβάθμιση της εφαρμογής.

Για την αποφυγή της ανάγκης του προγραμματισμού της συσκευής κάθε φορά που θέλουμε να προσθέσουμε ένα πρωτόκολλο, η λύση που δόθηκε είναι το πρωτόκολλο να λαμβάνεται δυναμικά από την συσκευή.

Ο μικροεπεξεργαστής συνδέεται μέσω UART με το BlueSmirf, τη συσκευή που λαμβάνει τις εντολές μέσω Bluetooth και τις προωθεί μέσω της σειριακής θύρας στον AVR μικροεπεξεργαστή. Ο μικροεπεξεργαστής με τη σειρά του, λαμβάνει τις εντολές και δημιουργεί τα κατάλληλα IR σήματα για τον τηλεχειρισμό της συσκευής που θέλουμε να χειριστούμε.

Για τη δυναμική χρήση πρωτοκόλλων υπερύθρων, καταγράφηκαν τα χαρακτηριστικά που ορίζονται σε ένα πρωτόκολλο υπερύθρων, όπως η συχνότητα του φέροντος, ο παλμός έναρξης και λήξης, η αντιστοίχιση των παλμών IR με το λογικό 0 και 1, ο χαρακτήρας επανάληψης και η χρονική διάρκεια μεταξύ δύο διαδοχικών εντολών. Όλα αυτά τα χαρακτηριστικά αποστέλλονται

μέσω Bluetooth και αποθηκεύονται στην συσκευή κατά την επιλογή της τελικής συσκευής που θέλουμε να χειριστούμε. Όταν ο χρήστης πατάει στη συνέχεια ένα κουμπί, η εντολή που αντιστοιχεί στο κουμπί στέλνεται στη συσκευή, η οποία έχει ήδη στη μνήμη της τα χαρακτηριστικά του πρωτοκόλλου που θα χρησιμοποιήσει. Με αυτόν τον τρόπο, για την προσθήκη πρωτοκόλλων, απαιτείται μόνο η αναβάθμιση της εφαρμογής, η οποία ενημερώνεται με το καινούργιο σετ χαρακτηριστικών για το νέο πρωτόκολλο.

8. Συσκευή Μετατροπής Bluetooth σε IR

Στη συνέχεια περιγράφεται η τελική συσκευή, όπως προέκυψε από τον παραπάνω σχεδιασμό. Αναλύεται το firmware που εκτελείται στη συσκευή και το κύκλωμα του εκπομπού IR.

8.1 Μονάδα Bluetooth με Σειριακή Επικοινωνία – BlueSmirf Silver

Το BlueSmirf Silver είναι μια μονάδα βασισμένη στο RN-42 Module. Προσφέρει σειριακή επικοινωνία σε ταχύτητες από 2400 έως 115200 bps. Δέχεται τροφοδοσία από 3.3V έως 6V, διαθέτει ενσωματωμένη κεραία και έχει μικρές διαστάσεις.

Η μονάδα RN-42 είναι συμβατή με συστήματα Bluetooth 2.1, 2.0, 1.2 και 1.1 και έχει χαμηλή κατανάλωση, της τάξης των 3mA όταν είναι συνδεδεμένη με άλλες συσκευές και 30mA όταν εκπέμπει. Εκτός από την UART διεπαφή, διαθέτει και διεπαφή USB. Διαθέτει τη δυνατότητα χρήσης των προφίλ GAP, SDP, RFCOMM και L2CAP. Υποστηρίζει το πρωτόκολλο Bluetooth EDR (Enhanced Data Rate), επιτρέποντας μετάδοση δεδομένων με ταχύτητα μέχρι 3 Mbps και απόσταση μέχρι 20 μέτρα. Ο εκπομπός κατηγοριοποιείται ως κλάσης 2 με εκπομπή στα 4 dBm και ευαισθησία δέκτη -80 dBm. Η συσκευή μπορεί να χρησιμοποιηθεί είτε ως master είτε ως slave.

Για την υλοποίηση, το προφίλ που χρησιμοποιείται είναι το RFCOMM και η ταχύτητα επικοινωνίας στην UART είναι 115200 bps. Τα χαρακτηριστικά αυτά ορίζονται στη συσκευή μέσω της UART θύρας με τη βοήθεια των εντολών AT. Από το εγχειρίδιο χρήσης της συσκευής προκύπτει ότι με τους χαρακτήρες “$$$”, αν αυτοί αποσταλούν σε 60 δευτερόλεπτα από την έναρξη λειτουργίας της συσκευής, η συσκευή μπαίνει σε κατάσταση αποδοχής εντολών και επιστρέφει τους χαρακτήρες “CMD”. Σε αυτή την κατάσταση η συσκευή δέχεται εντολές για αλλαγές ρυθμίσεων. Αν η εντολή γίνει αποδεκτή, επιστρέφονται οι χαρακτήρες “AOK” ενώ αν δεν γίνει δεκτή “ERR”. Σε περίπτωση που η εντολή δεν αναγνωρίζεται, επιστρέφεται ο χαρακτήρας “?”. Έτσι η εικόνα για τη ρύθμιση της συσκευής με τις απαιτούμενες ρυθμίσεις όπως φαίνονται από ένα τερματικό, είναι :

```
$$$         // εισαγωγή σε κατάσταση αποδοχής εντολών
CMD         // απάντηση : Επιτυχής είσοδος σε command mode
SL,N        // χωρίς χρήση ισοτιμίας
AOK         // απάντηση αποδοχής
SU,11       // ορισμός ταχύτητας επικοινωνίας UART στα 115200 bps
AOK         // απάντηση αποδοχής
---         // έξοδος από την κατάσταση εισαγωγής εντολών
```

Οι παραπάνω ρυθμίσεις αποθηκεύονται στη συσκευή και χρειάζεται να εισαχθούν μόνο μια φορά στη συσκευή. Μετά από αυτό το στάδιο η συσκευή είναι έτοιμη να συνδεθεί με το Arduino, με το οποίο θα επικοινωνεί με 115200 bps, 8 bit και χωρίς ισοτιμία.

8.2 Προγραμματισμός Arduino για την Παραγωγή Σήματος IR

Παρακάτω αναλύεται ο κώδικας που χρησιμοποιήθηκε στο Arduino για την παραγωγή του σήματος IR ανάλογα με τις εντολές που λαμβάνει το Arduino στη σειριακή θύρα. Στο Arduino

υπάρχουν πάντα δύο συναρτήσεις, η πρώτη “void setup”, που εκτελείται μια φορά στην αρχή της λειτουργίας της πλατφόρμας και η δεύτερη “void loop”, που εκτελείται συνέχεια με κυκλικό τρόπο, μετά την εκτέλεση της πρώτης.

Στην αρχή του προγράμματος δηλώνονται κάποιες σταθερές και ορίζονται πίνακες που θα χρησιμοποιηθούν.

```
#define IrLedPin 3
#define delayFactor 0
volatile int startPulseOn,startPulseOff,stopPulseOf,stopPulseOn,stopPulseOff;
volatile int ZeroOf,ZeroOn,ZeroOff,OneOf,OneOn,OneOff;
volatile int RepeatPulseOn2,RepeatPulseOn,RepeatPulseOff;
volatile int numberOfBits,breakDelayMilisecs, RepeatBreakMilisecs;
char startPulse[9],Zero[13],One[13],stopPulse[13],commandBits[3],breakDelay[4], repeatPulse[16];
char command[50];
```

Για την αρχικοποίηση της πλατφόρμας η συνάρτηση που χρησιμοποιήθηκε είναι η παρακάτω :

```
void setup(void){
  delay(1000);                    // καθυστέρηση 1000 ms
  Serial.begin(115200);           // αρχικοποίηση της σειριακής θύρας με ταχύτητα 115200 bps
  Serial.flush();                 // εκκαθάριση του buffer της σειριακής θύρας
  pinMode(IrLedPin,OUTPUT);       // ορισμός του pin 3 ως έξοδος
}
```

Η παραπάνω συνάρτηση ξεκινάει με μια καθυστέρηση, ώστε να μπορέσει να αρχικοποιηθεί η μονάδα Bluesmirf. Στη συνέχεια αρχικοποιεί τη σειριακή θύρα με τη ταχύτητα επικοινωνίας που έχουμε διαλέξει και σβήνοντας τον ενδιάμεσο καταχωρητή της μονάδας επικοινωνίας UART. Τέλος ορίζει το pin 3 (IrLedPin = 3), ως έξοδο, κάτι που προαπαιτείται για να μπορέσει το υπόλοιπο πρόγραμμα να “γράψει” σε αυτό το άκρο.

Η συνάρτηση void loop που ακολουθεί, ξεκινάει ορίζοντας μια μεταβλητή χαρακτήρα με όνομα inChar, για την ενδιάμεση αποθήκευση χαρακτήρων από τη σειριακή θύρα. Η συνάρτηση ελέγχει διαρκώς αν υπάρχουν διαθέσιμα δεδομένα στη σειριακή θύρα και αν δεν υπάρχουν, τελειώνει και ξεκινάει από την αρχή για να ελέγξει κατά πόσο υπάρχουν διαθέσιμοι χαρακτήρες στη σειριακή θύρα. Σε περίπτωση που υπάρχει διαθέσιμος χαρακτήρας, ελέγχει κατά πόσο αυτός είναι ένας από τους 4 αποδεκτούς. Συγκεκριμένα, αν ο χαρακτήρας που έχει λάβει είναι ο “o”, το πρόγραμμα προχωράει στην αρχικοποίηση – μηδενισμό των εσωτερικών του παραμέτρων και στην εγκατάσταση νέου πρωτοκόλλου IR, αποθηκεύοντας στις παραμέτρους τα κατάλληλα στοιχεία που

έχει λάβει από τη σειριακή θύρα. Σε περίπτωση που ο χαρακτήρας που έχει λάβει είναι ο “w”, το πρόγραμμα αναλαμβάνει να “τυπώσει” στη σειριακή θύρα τα στοιχεία του πρωτοκόλλου IR που έχει αποθηκευμένα. Αυτή η λειτουργία, χρησιμοποιήθηκε για τον έλεγχο της συσκευής και την επιδιόρθωση προβλημάτων στον κώδικα. Ο τρίτος αποδεκτός χαρακτήρας είναι ο “c”, με τον οποίο η συσκευή μπαίνει σε λειτουργία λήψης εντολής, την οποία στη συνέχεια στέλνει μέσω IR, βάση του ήδη αποθηκευμένου, στις τοπικές μεταβλητές, πρωτοκόλλου. Ο τελευταίος χαρακτήρας είναι ο “r”, που αναλαμβάνει να στείλει τον ειδικό χαρακτήρα επανάληψης, όπως αυτός ορίζεται από το πρωτόκολλο. Σε περίπτωση που το πρωτόκολλο δεν ορίζει χαρακτήρα επανάληψης, αποστέλλεται πάλι η τελευταία εντολή.

```
void loop(void){
 char inChar = '0';
 if(Serial.available()){                    // έλεγχος για διαθέσιμα δεδομένα στην UART
  inChar = Serial.read();                   // ο διαθέσιμος χαρακτήρας αποθηκεύεται τοπικά
 }

 switch(inChar){
  case 'o':                                 // αν ο χαρακτήρας είναι “o” εκτελούνται τα παρακάτω
   {
    Serial.flush();                         // εκκαθάριση του buffer της UART
    InitializeParams();                     // αρχικοποίηση-μηδενισμός των παραμέτρων
    InstallProtocol();                      // εγκατάσταση πρωτοκόλλου
    CharToInt();                            // μετατροπή των λαμβανομένων χαρακτήρων
    break;
   }
  case 'w':                                 // αν ο χαρακτήρας είναι “w” εκτελούνται τα παρακάτω
   {
    PrintProtocol();                        // εκτύπωση πρωτοκόλλου
    break;
   }
  case 'c':                                 // αν ο χαρακτήρας είναι “c” εκτελούνται τα παρακάτω
   {
    Serial.flush();                         // εκκαθάριση του buffer της UART
    Serial.readBytesUntil('c',command, 50);         // ανάγνωση / αποθήκευση της εντολής
    SendCommand();                          // αποστολή μέσω IR της εντολής
    break;
   }
  case 'r':                                 // αν ο χαρακτήρας είναι “r” εκτελούνται τα παρακάτω
```

```
    {
     SendRepeatPulse();                    // αποστολή του χαρακτήρα επανάληψης
     break;
    }
  }
}
```

Η συνάρτηση InitializeParams(), μηδενίζει τους πίνακες στους οποίους αποθηκεύεται το πρωτόκολλο επικοινωνίας και η εντολή IR. Με αυτόν τον τρόπο, δεδομένα που δεν υπάρχουν σε κάποιο πρωτόκολλο, όταν αυτό εγκατασταθεί, θα είναι μηδενικά και δε θα έχουν την τιμή που υπήρχε πριν στους πίνακες από κάποιο άλλο πρωτόκολλο. Όταν η συσκευή κλείσει, οι τιμές των πινάκων χάνονται και όταν ανοίξει πάλι είναι μηδενικοί. Έτσι η συνάρτηση αυτή έχει νόημα για πολλαπλές εγκαταστάσεις πρωτοκόλλων στον ίδιο κύκλο λειτουργίας. Η εντολή memset, με την οποία αρχικοποιούνται οι πίνακες, θέτει τα στοιχεία του πίνακα που της βάζουμε ως πρώτο όρισμα, με τον χαρακτήρα που χρησιμοποιούμε ως δεύτερο όρισμα. Το τρίτο όρισμα είναι το μήκος του πίνακα.

```
void InitializeParams(){

  memset(command,0,sizeof(command));              // μηδενισμός (NULL) του πίνακα εντολής
  memset(startPulse,'0',sizeof(startPulse));      // μηδενισμός του χαρακτήρα έναρξης
  memset(Zero,'0',sizeof(Zero));                  // μηδενισμός του πίνακα του λογικού 0
  memset(One,'0',sizeof(One));                    // μηδενισμός του πίνακα του λογικού 1
  memset(stopPulse,'0',sizeof(stopPulse));        // μηδενισμός του πίνακα του χαρακτήρα λήξης
  memset(commandBits,'0',sizeof(commandBits));    // μηδενισμός του πίνακα μεγέθους εντολής
  memset(breakDelay,'0',sizeof(breakDelay));      // μηδενισμός του πίνακα καθυστέρησης
  memset(repeatPulse,'0',sizeof(repeatPulse));    // μηδενισμός του χαρακτήρα επανάληψης
}
```

Η συνάρτηση εγκατάστασης πρωτοκόλλου, InstallProtocol, λαμβάνει τους χαρακτήρες που αφορούν τα στοιχεία του πρωτοκόλλου και τους αποθηκεύει στους κατάλληλους πίνακες. Για να ξεχωρίζουν τα στοιχεία που αντιστοιχούν σε κάθε χαρακτηριστικό του πρωτοκόλλου έχουν οριστεί χαρακτήρες που περιβάλλουν τα δεδομένα που αντιστοιχούν στο κάθε χαρακτηριστικό. Πιο συγκεκριμένα τα δεδομένα που αντιστοιχούν στον χαρακτήρα επανάληψης, περιβάλλονται από τον χαρακτήρα “e”. Τα δεδομένα που αντιστοιχούν στον παλμό έναρξης περιβάλλονται από τον χαρακτήρα “s”, ενώ στον παλμό λήξης περιβάλλονται από το “p”. Τα δεδομένα που αντιστοιχούν στο λογικό μηδέν περιβάλλονται από τον χαρακτήρα “)” και τα δεδομένα για το λογικό 1 περιβάλλονται από τον χαρακτήρα “!”. Ο αριθμός των bit ανά εντολή ξεκινάει και τελειώνει με τον χαρακτήρα “n” και η καθυστέρηση μεταξύ δύο διαδοχικών εντολών με τον χαρακτήρα “t”.

Ο πίνακας του χαρακτήρα έναρξης αποτελείται από 8 ψηφία, εκ των οποίων τα 4 πρώτα είναι η διάρκεια αποστολής παλμού στη συχνότητα του φέροντος και τα επόμενα 4 είναι η παύση

του παλμού έναρξης. Ο παλμός λήξης έχει 12 ψηφία με τα πρώτα 4 να είναι διάρκεια παύσης, τα επόμενα να είναι διάρκεια παλμού και τα τελευταία 4 να είναι διάρκεια παύσης σε μικροδευτερόλεπτα. Το λογικό μηδέν και ένα είναι πίνακες αντίστοιχοι με τον πίνακα του παλμού λήξης. Ο αριθμός των bit αποτελείται από 2 ψηφία ενώ η καθυστέρηση από 3 ψηφία που αντιστοιχούν σε χιλιοστά του δευτερολέπτου. Τέλος ο χαρακτήρας επανάληψης αποτελείται από 15 ψηφία, με τα πρώτα 4 να αντιστοιχούν σε διάρκεια παλμού , τα επόμενα 4 σε παύση, τα επόμενα 4 σε διάρκεια παλμού σε μικροδευτερόλεπτα και τα τελευταία 3 σε παύση μεταξύ διαδοχικών χαρακτήρων επανάληψης σε χιλιοστά του δευτερολέπτου. Έτσι για παράδειγμα για την εγκατάσταση του πρωτοκόλλου NEC που χρησιμοποιεί η LG, στέλνουμε του παρακάτω χαρακτήρες :

“os90004500s)000005600560)!000005601690!n33nt040te900022500560055eo”

```
int InstallProtocol(){

 char schar;
 while(schar !='o'){
  if(Serial.available()){
   schar = Serial.read();

   switch(schar){
    case 'e':                                   // αποθήκευση χαρακτήρων επανάληψης
    {
     Serial.readBytesUntil('e',repeatPulse,16);
     break;
    }
   case 's':                                    // αποθήκευση χαρακτήρων παλμού έναρξης
    {
     Serial.readBytesUntil('s',startPulse,9);
     break;
    }
   case ')':                                    // αποθήκευση χαρακτήρων λογικού μηδέν
    {
     Serial.readBytesUntil(')',Zero,13);
     break;
    }
   case '!':                                    // αποθήκευση χαρακτήρων λογικού ένα
    {
     Serial.readBytesUntil('!',One,13);
```

```
        break;
      }
    case 'p':                                   // αποθήκευση χαρακτήρων παλμού λήξης
      {
        Serial.readBytesUntil('p',stopPulse,13);
        break;
      }
    case 'n':                                   // αποθήκευση χαρακτήρων αριθμού bit
      {
        Serial.readBytesUntil('n',commandBits,3);
        break;
      }
    case 't':                                   // αποθήκευση χαρακτήρων καθυστέρησης
      {
        Serial.readBytesUntil('t',breakDelay,4);
        break;
      }
    }
   }
  }
}
```

Μετά την εκτέλεση της συνάρτησης εγκατάστασης πρωτοκόλλου, τρέχει η συνάρτηση CharToInt. Η συνάρτηση αυτή μετατρέπει τα δεδομένα του πρωτοκόλλου που έχουν ληφθεί και αποθηκευτεί στους πίνακες, σε ακέραιους αριθμούς και τους αποθηκεύει στις κατάλληλες μεταβλητές. Στη συνέχεια διορθώνει τους αριθμούς που αντιστοιχούν σε παλμό του φέροντος με μια σταθερά καθυστέρησης και αντικαθιστά όσους είναι μηδενικοί με τη μονάδα γιατί το μηδέν δε μπορεί να χρησιμοποιηθεί σε ρουτίνες χρονικής καθυστέρησης, που χρησιμοποιούνται παρακάτω.

Τα δεδομένα εισέρχονται ως χαρακτήρες. Βάση του πίνακα ASCII οι χαρακτήρες των αριθμών ξεκινάνε από το νούμερο 48 και ανεβαίνουν. Για τον λόγο αυτό, για να μετατραπεί η ακολουθία χαρακτήρων “0560” στον αριθμό 560, θα πρέπει ο πρώτος χαρακτήρας να πολλαπλασιαστεί με το 1000 αφού του έχει αφαιρεθεί το 48, ο δεύτερος με το 100 και ο τρίτος με το δέκα. Στο συγκεκριμένο παράδειγμα ο ASCII χαρακτήρας είναι το 48, ο χαρακτήρας “5” το 53 και ο χαρακτήρας “6” το 54. Έτσι προκύπτει :

1000*(48-48) = 0

100*(53-48) = 500

10*(54-48) = 60

1*(48-48) = 0

το άθροισμα των οποίων δίνει 560.

```
void CharToInt(void){
                startPulseOn = (1000*(int(startPulse[0])-48))+(100*(int(startPulse[1])-48))
+(10*(int(startPulse[2])-48))+(int(startPulse[3])-48);
                startPulseOff = (1000*(int(startPulse[4])-48))+(100*(int(startPulse[5])-48))
+(10*(int(startPulse[6])-48))+(int(startPulse[7])-48);
  ZeroOf = (1000*(int(Zero[0])-48))+(100*(int(Zero[1])-48))+(10*(int(Zero[2])-48))+(int(Zero[3])-
48);
  ZeroOn = (1000*(int(Zero[4])-48))+(100*(int(Zero[5])-48))+(10*(int(Zero[6])-48))+(int(Zero[7])-
48);
        ZeroOff = (1000*(int(Zero[8])-48))+(100*(int(Zero[9])-48))+(10*(int(Zero[10])-48))+
(int(Zero[11])-48);
    OneOf = (1000*(int(One[0])-48))+(100*(int(One[1])-48))+(10*(int(One[2])-48))+(int(One[3])-
48);
    OneOn = (1000*(int(One[4])-48))+(100*(int(One[5])-48))+(10*(int(One[6])-48))+(int(One[7])-
48);
  OneOff = (1000*(int(One[8])-48))+(100*(int(One[9])-48))+(10*(int(One[10])-48))+(int(One[11])-
48);
  stopPulseOf = (1000*(int(stopPulse[0])-48))+(100*(int(stopPulse[1])-48))+(10*(int(stopPulse[2])-
48))+(int(stopPulse[3])-48);
                stopPulseOn = (1000*(int(stopPulse[4])-48))+(100*(int(stopPulse[5])-48))
+(10*(int(stopPulse[6])-48))+(int(stopPulse[7])-48);
                stopPulseOff = (1000*(int(stopPulse[8])-48))+(100*(int(stopPulse[9])-48))
+(10*(int(stopPulse[10])-48))+(int(stopPulse[11])-48);
  numberOfBits = (10*(int(commandBits[0])-48))+(int(commandBits[1])-48);
          breakDelayMilisecs = (100*(int(breakDelay[0])-48))+(10*(int(breakDelay[1])-48))+
(int(breakDelay[2])-48);
            RepeatPulseOn = (1000*(int(repeatPulse[0])-48))+(100*(int(repeatPulse[1])-48))
+(10*(int(repeatPulse[2])-48))+(int(repeatPulse[3])-48);
            RepeatPulseOff = (1000*(int(repeatPulse[4])-48))+(100*(int(repeatPulse[5])-48))
+(10*(int(repeatPulse[6])-48))+(int(repeatPulse[7])-48);
           RepeatPulseOn2 = (1000*(int(repeatPulse[8])-48))+(100*(int(repeatPulse[9])-48))
+(10*(int(repeatPulse[10])-48))+(int(repeatPulse[11])-48);
        RepeatBreakMilisecs = (100*(int(repeatPulse[12])-48))+(10*(int(repeatPulse[13])-48))+
(int(repeatPulse[14])-48);


  startPulseOn = normalize(startPulseOn);
  startPulseOff = max(startPulseOff,1);
  ZeroOf = max(ZeroOf,1);
```

```
  ZeroOn = normalize(ZeroOn);
  ZeroOff = max(ZeroOff,1);
  OneOf = max(OneOf,1);
  OneOn = normalize(OneOn);
  OneOff = max(OneOff,1);
  stopPulseOf = max(stopPulseOf,1);
  stopPulseOn = normalize(stopPulseOn);
  stopPulseOff = max(stopPulseOff,1);
  RepeatPulseOn2 = normalize(RepeatPulseOn2);
  RepeatPulseOn = normalize(RepeatPulseOn);
  RepeatPulseOff = max(RepeatPulseOff,1);
}

int normalize(int n){
  n = max(n-delayFactor,1);
  return n;
}
```

Για λόγους ελέγχου της συσκευής κατά την υλοποίηση, δημιουργήθηκε η συνάρτηση PrintProtocol. Η συνάρτηση αυτή στέλνει στη σειριακή θύρα τα στοιχεία του πρωτοκόλλου όπως τα έχει αποθηκευμένα στη μνήμη. Επίσης στέλνει και το bit (δυαδικό σύστημα) της εντολής.

```
void PrintProtocol(){
  Serial.println();
  Serial.println("Protocol");
  Serial.print("Start pulse on time: ");
  Serial.println(startPulseOn);
  Serial.print("Start pulse off time: ");
  Serial.println(startPulseOff);
  Serial.print("Zero pulse off time: ");
  Serial.println(ZeroOf);
  Serial.print("Zero pulse on time: ");
  Serial.println(ZeroOn);
  Serial.print("Zero pulse off time: ");
  Serial.println(ZeroOff);
  Serial.print("One pulse off time: ");
```

```
  Serial.println(OneOf);
  Serial.print("One pulse on time: ");
  Serial.println(OneOn);
  Serial.print("One pulse off time: ");
  Serial.println(OneOff);
  Serial.print("Stop pulse off time: ");
  Serial.println(stopPulseOf);
  Serial.print("Stop pulse on time: ");
  Serial.println(stopPulseOn);
  Serial.print("Stop pulse off time: ");
  Serial.println(stopPulseOff);
  Serial.print("Number of bits: ");
  Serial.println(numberOfBits);
  Serial.print("Break time in milisecs: ");
  Serial.println(breakDelayMilisecs);
  Serial.print("Repeat pulse on time: ");
  Serial.println(RepeatPulseOn);
  Serial.print("Repeat pulse off time: ");
  Serial.println(RepeatPulseOff);
  Serial.print("Reapeat puse on2 time: ");
  Serial.println(RepeatPulseOn2);
  Serial.print("Reapeat break time: ");
  Serial.println(RepeatBreakMilisecs);
  Serial.print("Command :  ");
  for(int i=0;i<40;i++){
    Serial.print(command[i]);
  }
  Serial.println();
}
```

Όταν ληφθεί ο χαρακτήρας “c”, οι χαρακτήρες που ακολουθούν αποθηκεύονται στον πίνακα της εντολής και στη συνέχεια καλείται η συνάρτηση SendCommand. Η συνάρτηση αυτή αποστολής εντολής IR αναλαμβάνει να δημιουργήσει το κατάλληλο σήμα IR, βάσει των αποθηκευμένων στοιχείων του πρωτοκόλλου.

```
void SendCommand(void){
```

```
  PulseIr(startPulseOn);
  delayMicroseconds(startPulseOff);                    // αποστολή παλμού έναρξης
  for(int i =0;i<numberOfBits;i++){
   switch(command[i]){                                 // έλεγχος του επόμενου bit
   case '0':                                           // αν το bit είναι 0, αποστολή παλμού 0
     delayMicroseconds(ZeroOf);
     PulseIr(ZeroOn);
     delayMicroseconds(ZeroOff);
     break;
   case '1':                                           // αν το bit είναι 1, αποστολή παλμού 1
     delayMicroseconds(OneOf);
     PulseIr(OneOn);
     delayMicroseconds(OneOff);
     break;

    }
  }
  delayMicroseconds(stopPulseOf);
  PulseIr(stopPulseOn);
  delayMicroseconds(stopPulseOff);                     // αποστολή παλμού λήξης
  delay(breakDelayMilisecs);
  Serial.println("Command ok");                        // εκτύπωση στη σειριακή θύρα, debugging
}
```

Η παραπάνω συνάρτηση, χρησιμοποιεί για την αποστολή παλμού με τη συχνότητα φέροντος την συνάρτηση PulseIr. Η συνάρτηση αυτή παίρνει ως όρισμα τη διάρκεια του παλμού σε μικροδευτερόλεπτα. Για κύμα φέροντος 38 KHz, η περίοδος που είναι 1/38000 = 0,0000263 s = 26,3 us, μπορεί να χωριστεί σε 13,15 us με το σήμα σε ψηλή στάθμη και 13,15 us με το σήμα σε χαμηλή στάθμη. Η συνάρτηση PulseIr, αναλαμβάνει να κάνει αυτή την εναλλαγή για τη χρονική διάρκεια του παλμού, σύμφωνα με το πρωτόκολλο. Επειδή οι εντολές καταναλώνουν λίγο χρόνο μέχρι να εκτελεστούν, αντί για 13 us χρησιμοποιούνται 10 us καθυστέρησης.

```
void PulseIr(long microsecs){
  cli();
  while(microsecs>0){
    digitalWrite(IrLedPin,HIGH);
```

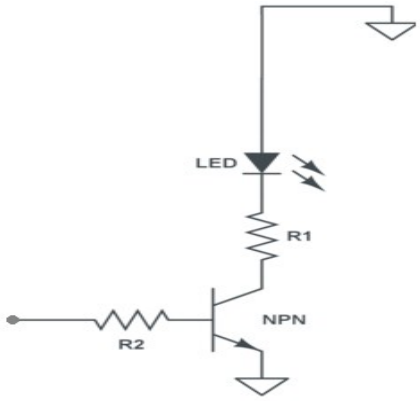
```
    delayMicroseconds(10);
    digitalWrite(IrLedPin,LOW);
    delayMicroseconds(10);
    microsecs -= 33;
  }
  sei();
}
```

Όταν η συσκευή λάβει τον χαρακτήρα επανάληψης “r”, εκτελείται η συνάρτηση SendRepeatPulse. Η συνάρτηση αυτή αναλαμβάνει να στείλει την τελευταία εντολή κάνοντας χρήση της συνάρτησης SendCommand, εκτός αν το πρωτόκολλο ορίζει παλμό επανάληψης. Αν ορίζεται παλμός επανάληψης, η συνάρτηση αναλαμβάνει να στείλει αυτόν.

```
void SendRepeatPulse(){
  Serial.flush();
  if (RepeatPulseOn > 1 || RepeatPulseOn2 > 1) {
    PulseIr(RepeatPulseOn);
    delayMicroseconds(RepeatPulseOff);
    PulseIr(RepeatPulseOn2);
    delay(RepeatBreakMilisecs);
  } else {
    SendCommand();
  }
}
```

8.3 Πομπός IR

Για την εκπομπή σήματος IR, χρησιμοποιήθηκε μια δίοδος φωτοεκπομπής (LED) IR. Πιο συγκεκριμένα το L53F3C, που εμφανίζει πτώση τάσης 1,2V κατά την ορθή πόλωση. Το κύκλωμα οδήγησης της διόδου, είναι ένα απλό τρανζίστορ σε συνδεσμολογία κοινού εκπομπού. Το τρανζίστορ που χρησιμοποιήθηκε είναι το PN2222A. Η συνδεσμολογία φαίνεται στο παρακάτω σχήμα :



Οι αντιστάσεις που χρησιμοποιήθηκαν, υπολογίστηκαν ώστε το ρεύμα της βάσης, Ib να είναι περίπου 6mA και η δίοδος να διαρρέεται από ρεύμα έντασης Ic, γύρω στα 60mA. Έτσι έχουμε:

- R_1 = (Vcc – Vled)/Ic = (5 – 1,2)/0,060 = 63,33 Ohm και
- R_2 = (Vcc – 0,7)/Ib = (5 – 0,7)/0,006 = 716,66 Ohm

Για πρακτικούς λόγους, χρησιμοποιήθηκαν οι παρακάτω αντιστάσεις :

- R_1 = 68 Ohm
- R_2 = 680 Ohm

και τελικά τα ρεύματα βάσης και εκπομπού υπολογίζονται :

- Ib = 6,3mA
- Ic = 55,9mA

Με την τιμή αυτή του ρεύματος συλλέκτη Ic, η εμβέλεια της συσκευής είναι κοντά στα 4 με 5 μέτρα, που για τους σκοπούς της εργασίας, κρίθηκε κατάλληλη.

9. Συμπεράσματα – Δυνατότητες για Επέκταση

Η συσκευή που κατασκευάστηκε για την εργασία, σε συνδυασμό με την εφαρμογή για κινητό Android, δίνει την δυνατότητα τηλεχειρισμού μιας συσκευής που διαθέτει δέκτη υπερύθρων, μέσα από μία συσκευή που ο χρήστης ήδη διαθέτει και είναι εξοικειωμένος, το κινητό του τηλέφωνο. Η χρήση του κινητού τηλεφώνου με αυτόν τον τρόπο, μπορεί να είναι πιο πολύπλοκη από την απλή χρήση του τηλεκοντρόλ που είναι κατασκευασμένο γι' αυτή και μόνο τη λειτουργία, παρόλα αυτά, ο χρήστης μπορεί να απλοποιήσει την καθημερινότητα του, καταργώντας πολλά τηλεχειριστήρια και αντικαθιστώντας τα με μία μικρή συσκευή, τοποθετημένη διακριτικά στον χώρο, και μία εφαρμογή στο κινητό που ήδη διαθέτει. Με αυτόν τον τρόπο, ο χειρισμός πολλών συσκευών, γίνεται από ένα σημείο, εύκολα και άμεσα.

Η συσκευή έχει ορισμένους περιορισμούς, που δε θεωρήθηκαν σημαντικοί για τους σκοπούς της άσκησης, αλλά μέσα από τους οποίους προκύπτει η δυνατότητα επέκτασης της στο μέλλον. Ένας βασικός περιορισμός είναι η εμβέλεια του bluetooth, για την επικοινωνία με το κινητό, αλλά και η εμβέλεια του σήματος υπερύθρων που παράγει. Συνεπώς, η συσκευή πρέπει να τοποθετηθεί σε σημείο που να μην απέχει πολύ από τον χρήστη αλλά και από τις συσκευές τις οποίες χειρίζεται, όπως η τηλεόραση ή το στερεοφωνικό. Αυτό, σε λίγες μόνο περιπτώσεις αποτελεί πρακτικό περιορισμό, αφού ο χειριστής βρίσκεται στον χώρο όπου βρίσκεται η τηλεόραση ή το στερεοφωνικό και η απαιτούμενη εμβέλεια είναι λίγα μέτρα.

Ένας ακόμα περιορισμός είναι η αυτονομία της συσκευής. Για τη λειτουργία της συσκευής, χρησιμοποιήθηκε μια κοινή μπαταρία 9 V. Με τη χρήση της μπαταρίας η συσκευή γίνεται φορητή, ανεξάρτητη και εύκολη στη χρήση και τοποθέτηση. Ταυτόχρονα όμως την περιορίζει, καθώς κάποια στιγμή η μπαταρία θα εξαντληθεί και θα πρέπει να αντικατασταθεί.

Τέλος, για να χειριστεί κανείς κάποια συγκεκριμένη συσκευή, θα πρέπει να υπάρχει στα δεδομένα της εφαρμογής το αρχείο που αντιστοιχεί σε αυτήν. Δεδομένου ότι η εφαρμογή είναι εύκολα επεκτάσιμη, μπορεί με μία απλή ενημέρωση να συμπεριληφθεί η νέα συσκευή.

Από τα παραπάνω προκύπτουν οι παρακάτω δύο δυνατότητες για επέκταση της παρούσας εργασίας :

α) Η χρήση επαναφορτιζόμενης μπαταρίας λιθίου, με την οποία θα μπορεί η συσκευή να είναι φορητή και αυτόνομη, αλλά ταυτόχρονα να μπορεί να χρησιμοποιηθεί και με τροφοδοσία από το ηλεκτρικό δίκτυο. Η συσκευή θα μπορεί ενδεχομένως να ειδοποιεί τον χρήστη μέσω κάποιων διόδων φωτοεκπομπής (LED) αλλά και μέσω της εφαρμογής, όταν χρειάζεται επαναφόρτιση.

β) Η ενσωμάτωση ενός αναγνώστη IR στην συσκευή, όπως αυτός που χρησιμοποιείται στην παρούσα άσκηση. Με την χρήση του αναγνώστη, θα μπορεί ο χρήστης να “διαβάσει” κάποιο τηλεχειριστήριο που δεν υπάρχει στην εφαρμογή και να το αντιγράψει, επεκτείνοντας τις δυνατότητες της.

Μέρος Γ' : Βιβλιογραφία

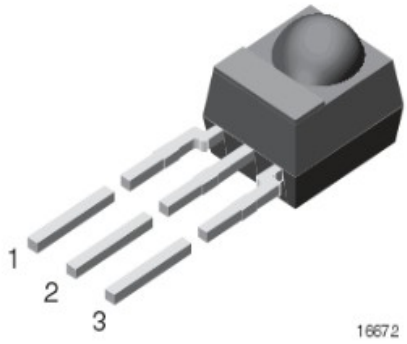
1. Αρχές Ηλεκτρονικών Υλικών και Διατάξεων, S.O. Kasap, Εκδόσεις Παπασωτηρίου
2. Μικροηλεκτρονικά Κυκλώματα, Sedra / Smith, Εκδόσεις Παπασωτηρίου
3. Οπτοηλεκτρονική Θεωρία – Πειράματα – Εφαρμογές, Αλεξανδρής Ν. Αλέξανδρος, Εκδόσεις Τζιόλα
4. Ψηφιακή Σχεδίαση, M. Morris Mano, Εκδόσεις Παπασωτηρίου
5. Προγραμματίζοντας τον Μικροελεγκτή AVR, Dhananjay V. Gardre, Εκδόσεις Τζιόλα
6. Γενικά Ηλεκτρονικά, Εμμανουήλ Γ. Τσαγάκη, Ίδρυμα Ευγενίδου
7. Εισαγωγή στην Java, Γιώργος Λιακέας, Εκδόσεις Κλειδάριθμος
8. Android Application Development All-In-One For Dummies, Barry Burd, Εκδόσεις John Wiley & Sons, Inc
9. http://www.sbprojects.com/knowledge/ir/
10. http://www.sbprojects.com/knowledge/ir/nec.php
11. http://www.sbprojects.com/knowledge/ir/itt.php
12. http://www.sbprojects.com/knowledge/ir/jvc.php
13. http://www.sbprojects.com/knowledge/ir/xsat.php
14. http://www.sbprojects.com/knowledge/ir/nrc17.php
15. http://www.sbprojects.com/knowledge/ir/sharp.php
16. http://www.sbprojects.com/knowledge/ir/sirc.php
17. http://www.sbprojects.com/knowledge/ir/rc5.php
18. http://www.sbprojects.com/knowledge/ir/rc6.php
19. https://www.bluetooth.org/en-us
20. https://www.bluetooth.org/en-us/specification
21. https://www.bluetooth.org/en-us/specification/adopted-specifications
22. http://en.wikipedia.org/wiki/Bluetooth
23. http://en.wikipedia.org/wiki/Bluetooth_stack
24. http://en.wikipedia.org/wiki/Remote_control
25. http://en.wikipedia.org/wiki/Android_%28operating_system%29
26. http://en.wikipedia.org/wiki/Arduino
27. https://mahalelabs.wordpress.com/2013/03/18/android-software-stack/
28. http://www.arduino.cc/
29. http://arduino.cc/en/Reference/HomePage
30. http://developer.android.com/index.html
31. http://developer.android.com/reference/packages.html
32. https://eclipse.org/
33. https://eclipse.org/users/

Παράρτημα Α'

Ο Αναγνώστης IR – Κύκλωμα και Κώδικας

1. **Δέκτης IR, TSOP4838**

Ο δέκτης TSOP4838, που χρησιμοποιήθηκε για την άσκηση, έχει 3 άκρα. Το ένα είναι για την τροφοδοσία του, το άλλο για την σύνδεση του με τη γη (μηδενικό δυναμικό) και το τρίτο είναι η έξοδος του. Ο δέκτης δέχεται τροφοδοσία από 2,5V μέχρι 5V, άρα μπορεί να τροφοδοτηθεί απευθείας από το Arduino. Στην έξοδο του εμφανίζεται υψηλό δυναμικό κατά την απουσία σήματος IR, ενώ όταν υπάρχει σήμα, η έξοδος πέφτει σε μηδενικό δυναμικό.



MECHNICAL DATA

Pinning for TSOP44.., TSOP48..:

1 = OUT, 2 = GND, 3 = V_S

Pinning for TSOP22.., TSOP24..:

1 = OUT, 2 = V_S, 3 = GND

Ο κάθε παλμός για να αναγνωριστεί, πρέπει να περιλαμβάνει τουλάχιστον 10 περιόδους της φέρουσας συχνότητας και η απόσταση μεταξύ παλμών πρέπει να είναι μεγαλύτερη από 12 περιόδους της φέρουσας συχνότητας.

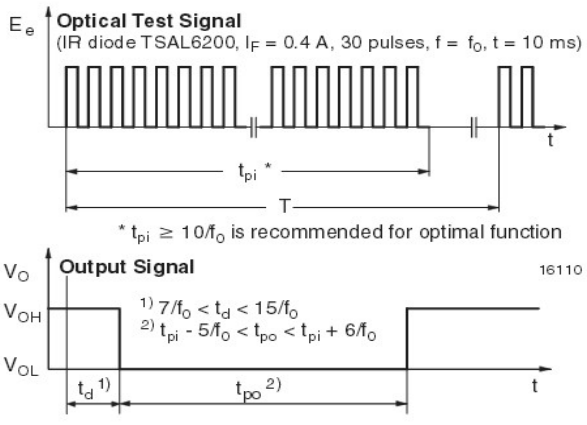
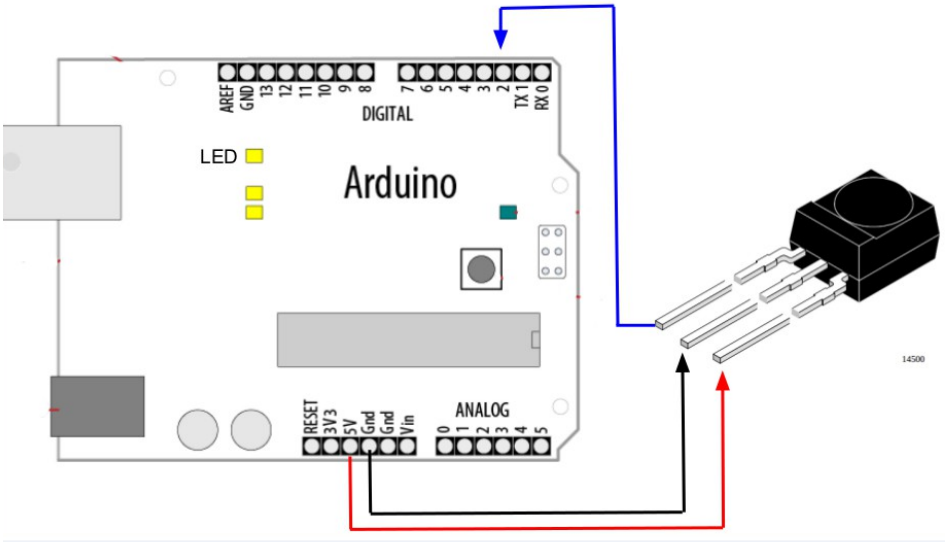


Fig. 1 - Output Active Low

2. Arduino και Σύνδεση Δέκτη IR

Το Arduino συνδέεται με τον δέκτη σύμφωνα με το παρακάτω σχήμα :



3. Κώδικας

Ο παρακάτω κώδικας διαβάζει την είσοδο του Arduino στην οποία είναι συνδεδεμένος ο δέκτης, και στέλνει στη σειριακή θύρα τα αποτελέσματα.

```
#define Irpin_PIN PIND
#define Irpin     2
#define MAXPULSE  65000
#define RESOLUTION  20

uint16_t pulses[100][2];
uint8_t currentpulse = 0;

void setup(void){
  Serial.begin(9600);
  Serial.println("Ready to decode IR!!");
}

void loop(void){
  uint16_t highpulse, lowpulse;
  highpulse = lowpulse = 0;

  while (Irpin_PIN & _BV(Irpin)){
    highpulse++;
    delayMicroseconds(RESOLUTION);

    if ((highpulse >= MAXPULSE) && (currentpulse !=0)){
      printpulses();
```

```
      currentpulse = 0;
      return;
    }
  }

  pulses[currentpulse][0] = highpulse;

  while (!(Irpin_PIN & _BV(Irpin))){
    lowpulse++;
    delayMicroseconds(RESOLUTION);
    if ((lowpulse >= MAXPULSE) && (currentpulse !=0)){
      printpulses();
      currentpulse = 0;
      return;
    }
  }
  pulses[currentpulse
  ][1] = lowpulse;

  currentpulse++;
}

void printpulses(void){
  Serial.println("\n\r\n\rReceived: \n\rOFF \tON");
  for (uint8_t i = 0; i<currentpulse;i++){
    Serial.print(pulses[i][0] * RESOLUTION, DEC);
    Serial.print(" usec, ");
    Serial.print(pulses[i][1] * RESOLUTION, DEC);
    Serial.println(" usec");
  }
}
```

Παράρτημα Β'

Προγραμματισμός Arduino Pro Mini για την Παραγωγή IR Σήματος

1. **Κώδικας**

```
#define IrLedPin 3
#define delayFactor 0
volatile int startPulseOn,startPulseOff,stopPulseOf,stopPulseOn,stopPulseOff;
volatile int ZeroOf,ZeroOn,ZeroOff,OneOf,OneOn,OneOff;
volatile int RepeatPulseOn2,RepeatPulseOn,RepeatPulseOff;
volatile int numberOfBits,breakDelayMilisecs, RepeatBreakMilisecs;
char startPulse[9],Zero[13],One[13],stopPulse[13],commandBits[3],breakDelay[4], repeatPulse[16];
char command[50];


void setup(void){   // Η συνάρτηση αυτή εκτελείται μια φορά κατά την εκκίνηση της συσκευής
  delay(1000);
  Serial.begin(115200);
  Serial.flush();
  pinMode(IrLedPin,OUTPUT);
}

void loop(void){  // Η συνάρτηση αυτή εκτελείται συνεχώς
  char inChar = '0';
  if(Serial.available()){
    inChar = Serial.read();
  }

  switch(inChar){
    case 'o':
      {
       Serial.flush();
       InitializeParams();              //  Αρχικοποίηση των στοιχείων πρωτοκόλλου
       InstallProtocol();               // Αποθήκευση των στοιχείων πρωτοκόλλου
       CharToInt();                     // Μετατροπή των στοιχείων πρωτοκόλλου στις κατάλληλες μορφές
       break;
      }
    case 'w':
      {
       PrintProtocol();                 // Εκτυπώνει το πρωτόκολλο στη σειριακή θύρα
       break;
      }
    case 'c':
      {
       Serial.flush();
       Serial.readBytesUntil('c',command, 50);          // Αποθήκευση εντολής
       SendCommand();                                   // Αποστολή εντολής
       break;
      }
    case 'r':
      {
       SendRepeatPulse();                               // Αποστολή χαρακτήρα επανάληψης
       break;
      }
  }
}
```

```
void CharToInt(void){
  startPulseOn = (1000*(int(startPulse[0])-48))+(100*(int(startPulse[1])-48))+(10*(int(startPulse[2])-48))+
(int(startPulse[3])-48);
  startPulseOff = (1000*(int(startPulse[4])-48))+(100*(int(startPulse[5])-48))+(10*(int(startPulse[6])-48))+
(int(startPulse[7])-48);
  ZeroOf = (1000*(int(Zero[0])-48))+(100*(int(Zero[1])-48))+(10*(int(Zero[2])-48))+(int(Zero[3])-48);
  ZeroOn = (1000*(int(Zero[4])-48))+(100*(int(Zero[5])-48))+(10*(int(Zero[6])-48))+(int(Zero[7])-48);
  ZeroOff = (1000*(int(Zero[8])-48))+(100*(int(Zero[9])-48))+(10*(int(Zero[10])-48))+(int(Zero[11])-48);
  OneOf = (1000*(int(One[0])-48))+(100*(int(One[1])-48))+(10*(int(One[2])-48))+(int(One[3])-48);
  OneOn = (1000*(int(One[4])-48))+(100*(int(One[5])-48))+(10*(int(One[6])-48))+(int(One[7])-48);
  OneOff = (1000*(int(One[8])-48))+(100*(int(One[9])-48))+(10*(int(One[10])-48))+(int(One[11])-48);
  stopPulseOf = (1000*(int(stopPulse[0])-48))+(100*(int(stopPulse[1])-48))+(10*(int(stopPulse[2])-48))+
(int(stopPulse[3])-48);
  stopPulseOn = (1000*(int(stopPulse[4])-48))+(100*(int(stopPulse[5])-48))+(10*(int(stopPulse[6])-48))+
(int(stopPulse[7])-48);
  stopPulseOff = (1000*(int(stopPulse[8])-48))+(100*(int(stopPulse[9])-48))+(10*(int(stopPulse[10])-48))+
(int(stopPulse[11])-48);
  numberOfBits = (10*(int(commandBits[0])-48))+(int(commandBits[1])-48);
  breakDelayMilisecs = (100*(int(breakDelay[0])-48))+(10*(int(breakDelay[1])-48))+(int(breakDelay[2])-48);
  RepeatPulseOn = (1000*(int(repeatPulse[0])-48))+(100*(int(repeatPulse[1])-48))+(10*(int(repeatPulse[2])-48))+
(int(repeatPulse[3])-48);
  RepeatPulseOff = (1000*(int(repeatPulse[4])-48))+(100*(int(repeatPulse[5])-48))+(10*(int(repeatPulse[6])-48))+
(int(repeatPulse[7])-48);
  RepeatPulseOn2 = (1000*(int(repeatPulse[8])-48))+(100*(int(repeatPulse[9])-48))+(10*(int(repeatPulse[10])-48))+
(int(repeatPulse[11])-48);
  RepeatBreakMilisecs = (100*(int(repeatPulse[12])-48))+(10*(int(repeatPulse[13])-48))+(int(repeatPulse[14])-48);

  startPulseOn = normalize(startPulseOn);
  startPulseOff = max(startPulseOff,1);
  ZeroOf = max(ZeroOf,1);
  ZeroOn = normalize(ZeroOn);
  ZeroOff = max(ZeroOff,1);
  OneOf = max(OneOf,1);
  OneOn = normalize(OneOn);
  OneOff = max(OneOff,1);
  stopPulseOf = max(stopPulseOf,1);
  stopPulseOn = normalize(stopPulseOn);
  stopPulseOff = max(stopPulseOff,1);
  RepeatPulseOn2 = normalize(RepeatPulseOn2);
  RepeatPulseOn = normalize(RepeatPulseOn);
  RepeatPulseOff = max(RepeatPulseOff,1);
}

int normalize(int n){
  n = max(n-delayFactor,1);
  return n;
}

void InitializeParams(){

  memset(command,0,sizeof(command));
  memset(startPulse,'0',sizeof(startPulse));
  memset(Zero,'0',sizeof(Zero));
  memset(One,'0',sizeof(One));
  memset(stopPulse,'0',sizeof(stopPulse));
  memset(commandBits,'0',sizeof(commandBits));
  memset(breakDelay,'0',sizeof(breakDelay));
  memset(repeatPulse,'0',sizeof(repeatPulse));
}
```

```
void PrintProtocol(){
  Serial.println();
  Serial.println("Protocol");
  Serial.print("Start pulse on time: ");
  Serial.println(startPulseOn);
  Serial.print("Start pulse off time: ");
  Serial.println(startPulseOff);
  Serial.print("Zero pulse off time: ");
  Serial.println(ZeroOf);
  Serial.print("Zero pulse on time: ");
  Serial.println(ZeroOn);
  Serial.print("Zero pulse off time: ");
  Serial.println(ZeroOff);
  Serial.print("One pulse off time: ");
  Serial.println(OneOf);
  Serial.print("One pulse on time: ");
  Serial.println(OneOn);
  Serial.print("One pulse off time: ");
  Serial.println(OneOff);
  Serial.print("Stop pulse off time: ");
  Serial.println(stopPulseOf);
  Serial.print("Stop pulse on time: ");
  Serial.println(stopPulseOn);
  Serial.print("Stop pulse off time: ");
  Serial.println(stopPulseOff);
  Serial.print("Number of bits: ");
  Serial.println(numberOfBits);
  Serial.print("Break time in milisecs: ");
  Serial.println(breakDelayMilisecs);
  Serial.print("Repeat pulse on time: ");
  Serial.println(RepeatPulseOn);
  Serial.print("Repeat pulse off time: ");
  Serial.println(RepeatPulseOff);
  Serial.print("Reapeat puse on2 time: ");
  Serial.println(RepeatPulseOn2);
  Serial.print("Reapeat break time: ");
  Serial.println(RepeatBreakMilisecs);
  Serial.print("Command :  ");
  for(int i=0;i<40;i++){
    Serial.print(command[i]);
  }
  Serial.println();
}


int InstallProtocol(){

  char schar;
  while(schar !='o'){
    if(Serial.available()){
      schar = Serial.read();

      switch(schar){
        case 'e':
        {
          Serial.readBytesUntil('e',repeatPulse,16);
          break;
        }
      case 's':
        {
```

```
          Serial.readBytesUntil('s',startPulse,9);
          break;
        }
      case ')':
        {
          Serial.readBytesUntil(')',Zero,13);
          break;
        }
      case '!':
        {
          Serial.readBytesUntil('!',One,13);
          break;
        }
      case 'p':
        {
          Serial.readBytesUntil('p',stopPulse,13);
          break;
        }
      case 'n':
        {
          Serial.readBytesUntil('n',commandBits,3);
          break;
        }
      case 't':
        {
          Serial.readBytesUntil('t',breakDelay,4);
          break;
        }
      }
    }
  }
}

void SendCommand(void){
  PulseIr(startPulseOn);
  delayMicroseconds(startPulseOff);
  for(int i =0;i<numberOfBits;i++){
    switch(command[i]){
    case '0':
      delayMicroseconds(ZeroOf);
      PulseIr(ZeroOn);
      delayMicroseconds(ZeroOff);
      break;
    case '1':
      delayMicroseconds(OneOf);
      PulseIr(OneOn);
      delayMicroseconds(OneOff);
      break;

    }
  }
  delayMicroseconds(stopPulseOf);
  PulseIr(stopPulseOn);
  delayMicroseconds(stopPulseOff);
  delay(breakDelayMilisecs);
  Serial.println("Command ok");
}
```

```
void PulseIr(long microsecs){
  cli();
  while(microsecs>0){
    digitalWrite(IrLedPin,HIGH);
    delayMicroseconds(10);
    digitalWrite(IrLedPin,LOW);
    delayMicroseconds(10);
    microsecs -= 33;
  }
  sei();
}

void SendRepeatPulse(){
  Serial.flush();
  if (RepeatPulseOn > 1 || RepeatPulseOn2 > 1) {
    PulseIr(RepeatPulseOn);
    delayMicroseconds(RepeatPulseOff);
    PulseIr(RepeatPulseOn2);
    delay(RepeatBreakMilisecs);
  } else {
    SendCommand();
  }
}
```

Παράρτημα Γ'

Η Κατασκευή της Συσκευής

1. **Κουτί**

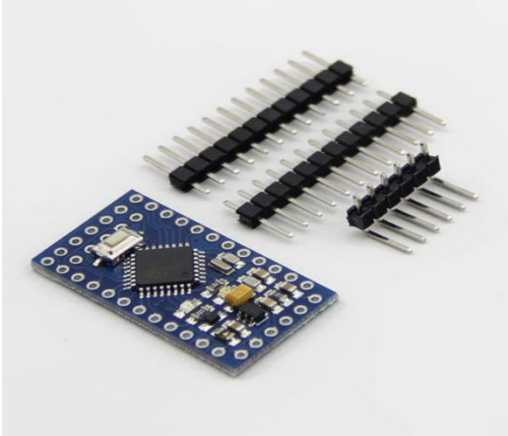
Η συσκευή τοποθετήθηκε στο κουτί που φαίνεται στην παρακάτω φωτογραφία, διαστάσεων 7x5x2.8 cm :

2. **BlueSmirf**

Στην παρακάτω φωτογραφία φαίνεται η συσκευή BlueSmirf Silver που χρησιμοποιήθηκε για την σύνδεση Bluetooth με το κινητό. Το BlueSmirf έχει σειριακή θύρα που συνδέθηκε απευθείας με το Arduino Pro Mini. Οι διαστάσεις του BlueSmirf είναι 45x16.6x3.9mm.

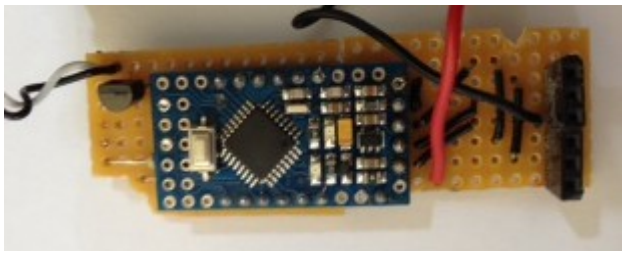
3. **Arduino Pro Mini**

Με διαστάσεις 33x18x6 mm, το Arduino Pro Mini που απεικονίζεται παρακάτω, μπορεί να δεχτεί τροφοδοσία από 5V μέχρι 12V και έρχεται με το Arduino Bootloader φορτωμένο, ώστε να μπορεί να προγραμματιστεί από τη σειριακή θύρα.



4. **Κατασκευή**

Για την τελική κατασκευή, το Arduino Pro Mini, κολλήθηκε πάνω σε μια διάτρητη πλακέτα στην οποία τοποθετήθηκε υποδοχή για το BlueSmirf. Ο αρνητικός πόλος της μπαταρίας συνδέθηκε απευθείας στην πλακέτα, ενώ ο θετικός συνδέθηκε μέσω ενός διακόπτη που καταλήγει στο εξωτερικό του κουτιού. Για τη δίοδο φωτοεκπομπής (led), δημιουργήθηκε μια τρύπα 5 mm. Τέλος, για να στερεωθούν όλα τα εξαρτήματα, χρησιμοποιήθηκε αφρώδης ταινία που κόπηκε στις κατάλληλες διαστάσεις καθώς και μια διαχωριστική πλαστική επιφάνεια που προστατεύει και τις επαφές της πλακέτας.



Το Arduino Pro Mini Πάνω στη διάτρητη πλακέτα

Το BlueSmirf τοποθετημένο στην πλακέτα

Η συσκευή τοποθετημένη στο κάτω μέρος του κουτιού

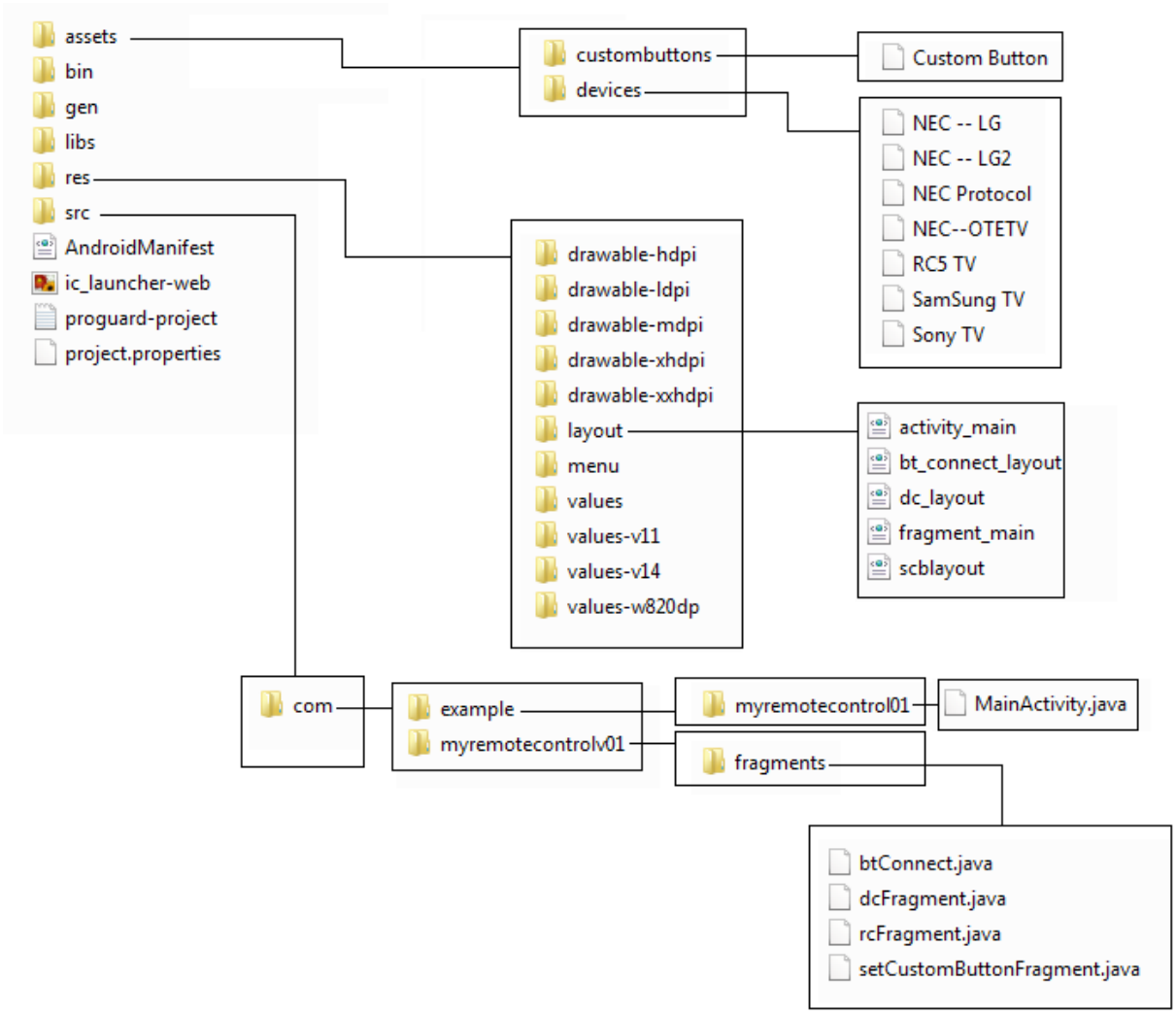
Η κατασκευή ολοκληρωμένη.

Παράρτημα Δ'

Η Android Εφαρμογή

1. **Δομή app**

Στην παρακάτω εικόνα φαίνεται η οργάνωση των φακέλων και των αρχείων της εφαρμογής. Πολλά από τα αρχεία και τους φακέλους δημιουργούνται αυτόματα από το περιβάλλον προγραμματισμού, ωστόσο αυτά που φαίνονται στην παρακάτω φωτογραφία, είναι αυτά που είτε δημιουργήθηκαν, είτε μεταβλήθηκαν με κάποιο τρόπο.



2. **Δομή αρχείων πρωτοκόλλων IR**

Κάθε αρχείο που βρίσκεται στον φάκελο Assets, σύμφωνα με την παραπάνω δομή, αντιστοιχεί σε ένα πρωτόκολλο IR. Το όνομα του αρχείου αποτελεί και το όνομα που βλέπει ο χρήστης της εφαρμογής όταν θέλει να επιλέξει πρωτόκολλο. Τα στοιχεία που υπάρχουν μέσα στο αρχείο, αντιστοιχούν με τα δεδομένα που θα στείλει η εφαρμογή, μέσω Bluetooth στη συσκευή. Τα στοιχεία αυτά πρέπει να βρίσκονται στο αρχείο με την παρακάτω δομή, ώστε να αναγνωριστούν σωστά από την εφαρμογή :

1. χαρακτήρες που αντιστοιχούν στο πρωτόκολλο
2. άνοιγμα συσκευής

3. αύξηση έντασης
4. μείωση έντασης
5. αριθμός 0
6. αριθμός 1
7. αριθμός 2
8. αριθμός 3
9. αριθμός 4
10. αριθμός 5
11. αριθμός 6
12. αριθμός 7
13. αριθμός 8
14. αριθμός 9
15. κλείσιμο της συσκευής (συνήθως ίδια εντολή με άνοιγμα)
16. αύξηση καναλιού
17. μείωση καναλιού
18. μενού
19. σίγαση ήχου

Τα δεδομένα ξεχωρίζουν από τα προηγούμενα γιατί βρίσκονται σε διαφορετική σειρά. Συνολικά το αρχείο πρέπει να αποτελείται από 19 σειρές, ακόμα και αν μια λειτουργία δεν υποστηρίζεται στο πρωτόκολλο, οπότε στη σειρά που αντιστοιχεί σε αυτήν τοποθετείται ο χαρακτήρας NULL.

3. **Τα βασικότερα αρχεία**

3.1 **Αρχεία Πρωτοκόλλων**

Τα παρακάτω αρχεία βρίσκονται στον φάκελο *assets/devices/,* και αποτελούν τα αρχεία των διαθέσιμων πρωτοκόλλων :

3.2 **Αρχεία Layout (xml)**

Τα παρακάτω xml αρχεία, βρίσκονται στον φάκελο *res/layout/*, και δημιουργούν τις αντίστοιχες διεπαφές χρήστη.

3.2.1 **activity_main.xml**

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:id="@+id/container"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   tools:context="com.example.myremotecontrol01.MainActivity"
   tools:ignore="MergeRootFrame">
```

```
</FrameLayout>
```

3.2.2 **bt_connect_layout.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" >

</LinearLayout>
```

3.2.3 **dc_layout.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" >
</LinearLayout>
```

3.2.4 **fragment_main.xml**

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:baselineAligned="false"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal"
    android:weightSum="3"
    android:layout_marginTop="20dp" >

    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:layout_weight="1"
```

```
android:orientation="vertical" >

<Button
    android:id="@+id/mute"
    style="?android:attr/buttonStyleSmall"
    android:layout_width="fill_parent"
    android:layout_height="38dp"
    android:layout_gravity="center"
    android:layout_marginLeft="15dp"
    android:layout_marginRight="15dp"
    android:text="Mute" />

<Button
    android:id="@+id/button1"
    style="?android:attr/buttonStyleSmall"
    android:layout_width="fill_parent"
    android:layout_height="38dp"
    android:layout_gravity="center"
    android:layout_marginLeft="15dp"
    android:layout_marginRight="15dp"
    android:text="1" />

<Button
    android:id="@+id/button4"
    style="?android:attr/buttonStyleSmall"
    android:layout_width="fill_parent"
    android:layout_height="38dp"
    android:layout_gravity="center"
    android:layout_marginLeft="15dp"
    android:layout_marginRight="15dp"
    android:text="4" />

<Button
    android:id="@+id/button7"
    style="?android:attr/buttonStyleSmall"
    android:layout_width="fill_parent"
    android:layout_height="38dp"
    android:layout_gravity="center"
    android:layout_marginLeft="15dp"
    android:layout_marginRight="15dp"
    android:text="7" />

<Button
    android:id="@+id/buttonCustom"
    style="?android:attr/buttonStyleSmall"
    android:layout_width="fill_parent"
    android:layout_height="38dp"
    android:layout_gravity="center"
    android:layout_marginLeft="15dp"
    android:layout_marginRight="15dp"
```

```
        android:text="Custom" />

    <Button
        android:id="@+id/vol_down"
        style="?android:attr/buttonStyleSmall"
        android:layout_width="fill_parent"
        android:layout_height="38dp"
        android:layout_gravity="center"
        android:layout_marginLeft="15dp"
        android:layout_marginRight="15dp"
        android:text="Vol -"
        android:layout_marginTop="58dp"/>
</LinearLayout>

<LinearLayout
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:layout_weight="1"
    android:orientation="vertical" >

    <Button
        android:id="@+id/button2"
        style="?android:attr/buttonStyleSmall"
        android:layout_width="fill_parent"
        android:layout_height="38dp"
        android:layout_gravity="center"
        android:layout_marginTop="38dp"
        android:layout_marginLeft="15dp"
        android:layout_marginRight="15dp"
        android:text="2" />

    <Button
        android:id="@+id/button5"
        style="?android:attr/buttonStyleSmall"
        android:layout_width="fill_parent"
        android:layout_height="38dp"
        android:layout_gravity="center"
        android:layout_marginLeft="15dp"
        android:layout_marginRight="15dp"
        android:text="5" />

    <Button
        android:id="@+id/button8"
        style="?android:attr/buttonStyleSmall"
        android:layout_width="fill_parent"
        android:layout_height="38dp"
        android:layout_gravity="center"
        android:layout_marginLeft="15dp"
        android:layout_marginRight="15dp"
        android:text="8" />
```

```
    <Button
        android:id="@+id/button0"
        style="?android:attr/buttonStyleSmall"
        android:layout_width="fill_parent"
        android:layout_height="38dp"
        android:layout_gravity="center"
        android:layout_marginLeft="15dp"
        android:layout_marginRight="15dp"
        android:text="0" />

    <Button
        android:id="@+id/ch_up"
        style="?android:attr/buttonStyleSmall"
        android:layout_width="fill_parent"
        android:layout_height="38dp"
        android:layout_gravity="center"
        android:layout_marginLeft="15dp"
        android:layout_marginRight="15dp"
        android:text="Ch +"
        android:layout_marginTop="20dp" />

    <Button
        android:id="@+id/menu"
        style="?android:attr/buttonStyleSmall"
        android:layout_width="fill_parent"
        android:layout_height="38dp"
        android:layout_gravity="center"
        android:layout_marginLeft="15dp"
        android:layout_marginRight="15dp"
        android:text="Menu" />

    <Button
        android:id="@+id/ch_down"
        style="?android:attr/buttonStyleSmall"
        android:layout_width="fill_parent"
        android:layout_height="38dp"
        android:layout_gravity="center"
        android:layout_marginLeft="15dp"
        android:layout_marginRight="15dp"
        android:text="Ch -" />
</LinearLayout>

<LinearLayout
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:layout_weight="1"
    android:orientation="vertical" >

    <Button
```

```
        android:id="@+id/power"
        style="?android:attr/buttonStyleSmall"
        android:layout_width="fill_parent"
        android:layout_height="38dp"
        android:layout_gravity="center"
        android:layout_marginLeft="15dp"
        android:layout_marginRight="15dp"
        android:text="Power" />

    <Button
        android:id="@+id/button3"
        style="?android:attr/buttonStyleSmall"
        android:layout_width="fill_parent"
        android:layout_height="38dp"
        android:layout_gravity="center"
        android:layout_marginLeft="15dp"
        android:layout_marginRight="15dp"
        android:text="3" />

    <Button
        android:id="@+id/button6"
        style="?android:attr/buttonStyleSmall"
        android:layout_width="fill_parent"
        android:layout_height="38dp"
        android:layout_gravity="center"
        android:layout_marginLeft="15dp"
        android:layout_marginRight="15dp"
        android:text="6" />

    <Button
        android:id="@+id/button9"
        style="?android:attr/buttonStyleSmall"
        android:layout_width="fill_parent"
        android:layout_height="38dp"
        android:layout_gravity="center"
        android:layout_marginLeft="15dp"
        android:layout_marginRight="15dp"
        android:text="9" />

    <Button
        android:id="@+id/vol_up"
        style="?android:attr/buttonStyleSmall"
        android:layout_width="fill_parent"
        android:layout_height="38dp"
        android:layout_gravity="center"
        android:layout_marginLeft="15dp"
        android:layout_marginRight="15dp"
        android:text="Vol +"
        android:layout_marginTop="96dp" />
</LinearLayout>
```

```
</LinearLayout>
```

3.2.5 **scblayout.xml**

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:android1="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:layout_marginTop="20dp"
   android:baselineAligned="false"
   android:orientation="horizontal"
   android:weightSum="1" >

   <LinearLayout
      android1:layout_width="wrap_content"
      android1:layout_height="match_parent"
      android1:layout_weight="1"
      android1:weightSum="10"
      android1:orientation="vertical" >

      <TextView
         android1:id="@+id/textView1"
         android1:layout_width="match_parent"
         android1:layout_height="wrap_content"
         android1:text="Text" />


      <LinearLayout
         android1:layout_width="match_parent"
         android1:layout_height="wrap_content"
         android1:layout_weight="8" >

         <LinearLayout
            android1:layout_width="wrap_content"
            android1:layout_height="match_parent"
            android1:orientation="vertical"
            android1:layout_weight="1" >

            <ListView
               android1:id="@+id/availableDeviceList"
               android1:layout_width="match_parent"
               android1:layout_height="match_parent"
               android1:layout_weight="1" >
            </ListView>
```

```
            <ListView
               android1:id="@+id/operationList"
               android1:layout_width="match_parent"
               android1:layout_height="match_parent"
                                    android1:layout_weight="1" >
            </ListView>
         </LinearLayout>

         <ListView
            android1:id="@+id/CBList"
            android1:layout_width="match_parent"
            android1:layout_height="match_parent"
            android1:layout_weight="1" >
         </ListView>

      </LinearLayout>

      <LinearLayout
         android1:layout_width="match_parent"
         android1:layout_height="wrap_content"
         android1:layout_weight="1" >

         <Button
            android1:id="@+id/addOperation"
            android1:layout_width="wrap_content"
            android1:layout_height="wrap_content"
            android1:text="  ADD  " />

         <Button
            android1:id="@+id/doneSCB"
            android1:layout_width="wrap_content"
            android1:layout_height="wrap_content"
            android1:text=" DONE  " />

         <Button
            android1:id="@+id/resetButton"
            android1:layout_width="wrap_content"
            android1:layout_height="wrap_content"
            android1:text="RESET" />

      </LinearLayout>

   </LinearLayout>

</LinearLayout>
```

3.3 **Αρχεία Java**

Τα παρακάτω αρχεία, είναι αυτά που έχουν τον κώδικα Java της εφαρμογής. Το αρχείο *MainActivity.java*, βρίσκεται στον φάκελο *src/com/example/myremotecontrol01/*, ενώ τα υπόλοιπα στον φάκελο *src/com/myremotecontrolv01/fragments*.

3.3.1 **MainActivity.java**

/* Όλες οι λειτουργίες της εφαρμογής εκτελούνται μέσα από ένα activity, το MainAactivity.java. Το activity αυτό έχει την υποχρέωση να δημιουργήσει τις διεπαφές χρήστη, καθώς ο χρήστης επιλέγει λειτουργίες από το μενού, να συνδεθεί με την Bluetooth συσκευή που επιλέγει ο δέκτης και να στείλει μέσω αυτής τους χαρακτήρες που πρέπει, ανάλογα με τα πλήκτρα που πατιούνται στη διεπαφή του rcFragment. Τα fragments επικοινωνούν με την MainActivity μέσω διεπαφών (public) interface. Οι διεπαφές αυτές δηλώνονται στην κύρια κλάση με τη χρήση του όρου implements.*/

```
package com.example.myremotecontrol01;

import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.util.Set;
import java.util.UUID;

import com.myremotecontrolv01.fragments.*;
import com.myremotecontrolv01.fragments.dcFragment.DCTalkToActivity;
import com.myremotecontrolv01.fragments.btConnect.TalkToActivity;
import com.myremotecontrolv01.fragments.rcFragment.sendOverBluetooth;
import com.myremotecontrolv01.fragments.setCustomButtonFragment.scbTalkToActivity;

import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.bluetooth.BluetoothSocket;
import android.content.Intent;
import android.content.res.Configuration;
import android.os.Bundle;
import android.app.Fragment;
import android.app.FragmentManager;
import android.app.FragmentTransaction;
import android.support.v7.app.ActionBarActivity;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.ListView;
import android.widget.Toast;


public class MainActivity extends ActionBarActivity
                    implements TalkToActivity, DCTalkToActivity, sendOverBluetooth,
scbTalkToActivity{
```

```
        private static final int REQUEST_ENABLE_BT = 1; //must be greater than zero
        private static final int t1 = 100;
        private static final int t2 = 80;
        public  boolean bluetoothConnected = false;
        public  boolean deviceChosen = false;
        public  boolean          startWithBtDC = false; // true to start with connecting to bluetooth
device
        public  boolean buttonDown = false;
        BluetoothAdapter myBlueAdapter = BluetoothAdapter.getDefaultAdapter();
        protected static final UUID MY_UUID = UUID.fromString("00001101-0000-1000-8000-
00805F9B34FB");
        public String  btDevAddress;
        public String tvDevice = null;
        public OutputStream myOutputStream;
        public BluetoothDevice  connectedDevice = null;
        public BluetoothDevice myDevice;
        public BluetoothSocket mySocket;
```

```
        /* Η παρακάτω συνάρτηση, εκτελείται κατά τη δημιουργία της MainActivity, ανανεώνει τη
διεπαφή χρήστη και ελέγχει αν είναι ενεργοποιημένο το Bluetooth. Σε περίπτωση που δεν είναι
καλεί μία λειτουργία του συστήματος για την ενεργοποίηση του. */

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

      updateUI(3);

        if (!myBlueAdapter.isEnabled()){
            Intent enableBtIntent =new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
            startActivityForResult(enableBtIntent, REQUEST_ENABLE_BT);
        }

    }


        /*Η παρακάτω συνάρτηση ανανεώνει τη διεπαφή χρήστη, χρησιμοποιώντας την κλάση
FragmentManager, μέσω της οποίας αντικαθιστά την υπάρχουσα διεπαφή, με αυτή που πρέπει να
δημιουργηθεί, ανάλογα με τις επιλογές του χρήστη. Για να φορτώσει την κατάλληλη διεπαφή,
καλώντας το αντίστοιχο fragment, η συνάρτηση δέχεται ως όρισμα έναν ακέραιο από το 1 ως το 4.
*/

    public void updateUI(int i){

        Configuration config = getResources().getConfiguration();
        FragmentManager fragmentManager = getFragmentManager();
        FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();
```

```
        // added at line 48
        if ((!bluetoothConnected&&startWithBtDC)||i == 1){
            btConnect bt_connect = new btConnect();
            fragmentTransaction.replace(android.R.id.content, bt_connect);
            fragmentTransaction.commit();
        }else if((!deviceChosen&&startWithBtDC)||i == 2){
            dcFragment dc_fragment = new dcFragment();
            fragmentTransaction.replace(android.R.id.content, dc_fragment);
            fragmentTransaction.commit();
        }else if((deviceChosen&&bluetoothConnected&&startWithBtDC)||i==3){
            rcFragment my_rc_fragment = new rcFragment();
            Bundle args = new Bundle();
            if(tvDevice == null){
                    try{
                            tvDevice = getAssets().list("devices")[0];
                            //Toast.makeText(this, "Default TV : "+tvDevice,
Toast.LENGTH_LONG).show();
                    }catch (Exception e){}
            }
            args.putString("tvDevice", tvDevice);
            my_rc_fragment.setArguments(args);
            fragmentTransaction.replace(android.R.id.content, my_rc_fragment);
            fragmentTransaction.commit();

        }else if (i==4){
            setCustomButtonFragment my_setCustomButtonFragment = new
setCustomButtonFragment();
            fragmentTransaction.replace(android.R.id.content, my_setCustomButtonFragment);
            fragmentTransaction.addToBackStack(null);
            fragmentTransaction.commit();
        }



    }
```

/*Η επόμενη συνάρτηση, καλείται από το btConnect.java μέσω της διεπαφής TalkToActivity. Κατά την επιλογή της συσκευής Bluetooth από τον χρήστη, φροντίζει να ελέγξει αν είναι ήδη συνδεδεμένη η συσκευή. Αν η συσκευή είναι συνδεδεμένη με άλλη ή αν δεν είναι συνδεδεμένη, θα προσπαθήσει να συνδεθεί με τη συσκευή που επέλεξε ο χρήστης και στο τέλος θα καλέσει τη συνάρτηση updateUI που είδαμε παραπάνω με όρισμα 3, ώστε να φορτώσει το rcFragment. */

```
    public void listenFrBt(String btDevice, int i){
        //final BluetoothDevice myDevice;
        //final BluetoothSocket mySocket;

        if (i == 1){
```

```
                //code when function is called from btConnect
                String actionStateChanged = BluetoothAdapter.ACTION_STATE_CHANGED;
                btDevAddress = btDevice;
                if(connectedDevice == (myDevice =
myBlueAdapter.getRemoteDevice(btDevAddress))){
                        //already connected to this device
                        //Toast.makeText(this,"Already
Connected",Toast.LENGTH_SHORT).show();
                }else if(connectedDevice != null){
                        try {
                                mySocket.close();
                                connectedDevice = null;
                        } catch (IOException e) {Toast.makeText(this, "Couldn't close BT
connection", Toast.LENGTH_SHORT).show(); }
                }
                if(connectedDevice == null){
                        BluetoothSocket tmp = null;
                        //final BluetoothSocket mySocket;
                        OutputStream tmpOutStream = null;
                        try{
                                tmp = myDevice.createRfcommSocketToServiceRecord(MY_UUID);
                        } catch (IOException e){}
                        mySocket = tmp;
                        if (mySocket != null){
                                try{
                                        mySocket.connect();
                                        tmpOutStream = mySocket.getOutputStream();

        connectedDevice=myBlueAdapter.getRemoteDevice(btDevAddress);
                                        bluetoothConnected = true;
                                }catch(IOException e){}
                        }
                        myOutputStream = tmpOutStream;
                }
        } else if (i==2){

        }

        updateUI(3);
    }


        /* Η παρακάτω συνάρτηση καλείται από το dcFragment, μέσω της διεπαφής
DCTalkToActivity και αναλαμβάνει να φορτώσει το rcFragment αποστέλλοντας του πληροφορίες
για τη συσκευή που θέλει να χειριστεί ο χρήστης. */


    public void listenFrDc(String tvDev, int i){

        tvDevice = tvDev;
```

```
        if (tvDevice != null){
                deviceChosen = true;
                //Toast.makeText(this,tvDevice,Toast.LENGTH_SHORT).show();
        }
        updateUI(3);
    }

    public void listenFrScb(){

        updateUI(3);
    }


        /* Οι επόμενες δύο συναρτήσεις καλούνται μέσω της διεπαφής sendOverBluetooth, για την
αποστολή των κατάλληλων χαρακτήρων μέσω της επαφής Bluetooth. */


    public void sendString(String command){

        if(myOutputStream!=null){
                try{
                        myOutputStream.write(command.getBytes());
                } catch (IOException e){}
        }else {
                Toast.makeText(this,"You are not connected to a
bluetooth",Toast.LENGTH_LONG).show();
        }
    }

    public void setButtonDown(boolean settingValue){
        buttonDown = settingValue;
        //Toast.makeText(this, "Setting the button dow var", Toast.LENGTH_SHORT).show();
        new Thread(new Runnable(){
                public void run(){
                        try {
                                        Thread.sleep(t1);
                                } catch (InterruptedException e) {
                                        // TODO Auto-generated catch block
                                        e.printStackTrace();
                                }

                        while(buttonDown){
                                try {
                                        Thread.sleep(t2);
                                } catch (InterruptedException e) {
                                        // TODO Auto-generated catch block
                                        e.printStackTrace();
                                }

                                sendString("r");
```

```
                        }
                }
        }).start();

    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {

        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }


        /* Με την επόμενη συνάρτηση, ορίζονται οι λειτουργίες που εκτελούνται όταν ο χρήστης
κάνει μια επιλογή στο μενού.  */


    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();
        if (id == R.id.connect_bluetooth) {
            updateUI(1);
        }
        if (id == R.id.choose_tv) {
            updateUI(2);
        }
        if (id == R.id.set_custom_button){
            updateUI(4);
        }
        return super.onOptionsItemSelected(item);
    }


    private class ConnectThread extends Thread {
        private final BluetoothSocket mySocket;
        private final BluetoothDevice myDevice;

        public ConnectThread(BluetoothDevice device){
                BluetoothSocket tmp = null;

                myDevice = myBlueAdapter.getRemoteDevice(btDevAddress);

                try{
                        tmp = device.createRfcommSocketToServiceRecord(MY_UUID);
                } catch (IOException e){}
```

```
            mySocket = tmp;

        }

        public void run() {
            myBlueAdapter.cancelDiscovery();
            try {
                mySocket.connect();
            } catch (IOException connectException){
                try{
                    mySocket.close();
                } catch (IOException closeException){
                }
            }
        }

}

private class ConnectedThread extends Thread {
        private final BluetoothSocket mySocket;
        private final InputStream myInputStream;
        private final OutputStream myOutputStream;

        public ConnectedThread(BluetoothSocket socket){
            mySocket = socket;
            InputStream tmpIn = null;
            OutputStream tmpOut = null;
            try {
                tmpIn = socket.getInputStream();
                tmpOut = socket.getOutputStream();
            } catch (IOException e){}
            myInputStream = tmpIn;
            myOutputStream = tmpOut;
        }

        public void run(){
            byte[] buffer = new byte[1024];
            int bytes;

            while(true){
                try{
                    bytes=myInputStream.read(buffer);
                }catch (IOException e){
                    break;
                }
            }
        }

        public void write(byte[] bytes){
            try{
```

```
                        myOutputStream.write(bytes);
                } catch (IOException e){}
        }
        public void cancel(){
                try {
                        mySocket.close();
                } catch (IOException e){}
        }
    }

    @Override
    protected void onDestroy(){
        super.onDestroy();
        if(connectedDevice != null){
                        try {
                                mySocket.close();
                                connectedDevice = null;
                        } catch (IOException e) {Toast.makeText(this, "Couldn't close BT
connection", Toast.LENGTH_SHORT).show(); }
                }
    }

}
```

3.3.2 **btConnect.java**

/* Η κλάση btConnect αποτελεί μια επέκταση της κλάσης ListFragment, που υλοποιεί μια λίστα. Υλοποιεί τη διεπαφή TalkToActivity για να καλεί μέσω αυτής τη συνάρτηση listenFrBt της MainActivity. */

```
package com.myremotecontrolv01.fragments;

import java.util.ArrayList;
import java.util.List;
import java.util.Set;
import com.example.myremotecontrol01.R;
import android.app.Activity;
import android.app.Fragment;
import android.app.ListFragment;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.content.Intent;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.ListView;

public class btConnect extends ListFragment{
	TalkToActivity mListener;
	private int selectedItemNumber = -1;
	private String selectedDevice = null;

	public interface TalkToActivity {
		public void listenFrBt(String btDevice, int i);
	}

	List<String> pairedList = new ArrayList<String>();
```

/* Κατά τη δημιουργία της, η κλάση btConnect, ζητάει από το σύστημα τη λίστα με τις συσκευές Bluetooth, που είναι ζευγαρωμένες (paired) με το κινητό. Τη λίστα αυτή την εμφανίζει στον χρήστη για την επιλογή της συσκευής με την οποία θέλει να συνδεθεί. */

```
	@Override
	public void onActivityCreated(Bundle savedInstanceState){
```

```
		super.onActivityCreated(savedInstanceState);
			BluetoothAdapter myBlueAdapter = BluetoothAdapter.getDefaultAdapter();
			ArrayAdapter pairedArrayAdapter =new
ArrayAdapter<String>(getActivity(), android.R.layout.simple_list_item_1); //, list);

			setListAdapter(pairedArrayAdapter);


			Set<BluetoothDevice> pairedDevices = myBlueAdapter.getBondedDevices();
			if (pairedDevices.size() > 0){
				for(BluetoothDevice device : pairedDevices){
					pairedList.add(device.getAddress());
					pairedArrayAdapter.add(device.getName() + "  :  " +
device.getAddress());
				}
			}
			pairedArrayAdapter.add("Discover More Devices");

	}

	@Override
	public void onAttach(Activity activity){
		super.onAttach(activity);
		try {
			mListener = (TalkToActivity) activity;
		} catch (ClassCastException e){
			throw new ClassCastException(activity.toString() + " must implement
TalkToActivity");
		}
	}


	/* Όταν ο χρήστης πατήσει πάνω σε κάποια συσκευή, καλείται η παρακάτω συνάρτηση που
με τη σειρά της καλεί τη συνάρτηση listenFrBt της MainActivity μέσω της διεπαφής
TalkToActivity.   */


	@Override
	public void onListItemClick(ListView l, View v, int position, long id){
		mListener.listenFrBt(pairedList.get(position), 1);
	}

}
```

3.3.3 **dcFragment.java**

/* Η κλάση αυτή, υλοποιεί τη διεπαφή DCTalkToActivity, για να μπορέσει μέσω αυτής να καλέσει τη συνάρτηση listenFrDc. */

```
package com.myremotecontrolv01.fragments;

import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
import java.util.Set;

import com.example.myremotecontrol01.R;

import android.app.Activity;
import android.app.Fragment;
import android.app.ListFragment;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.content.Context;
import android.content.res.AssetManager;
import android.os.Bundle;
import android.os.Environment;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.ListView;

import com.example.myremotecontrolv01.utilities.*;
import com.myremotecontrolv01.fragments.btConnect.TalkToActivity;


public class dcFragment extends ListFragment{

	DCTalkToActivity mListener;

	public interface DCTalkToActivity {
		public void listenFrDc(String btDevice, int i);
	}


	/* Κατά τη δημιουργία της διαβάζει τον φάκελο Assets και επιστρέφει σε λίστα τα ονόματα των αρχείων που αποτελούν και τις συσκευές για τις οποίες είναι διαθέσιμα τα πρωτόκολλα IR.  */
```

```
	@Override
	public void onActivityCreated(Bundle savedInstanceState){
		super.onActivityCreated(savedInstanceState);
			//BluetoothAdapter myBlueAdapter = BluetoothAdapter.getDefaultAdapter();
			ArrayAdapter devicesArrayAdapter =new
ArrayAdapter<String>(getActivity(), android.R.layout.simple_list_item_1);
			setListAdapter(devicesArrayAdapter);
			AssetManager aMan = getActivity().getAssets();

			try {
				String[] listOfDevices = aMan.list("devices");

				if (listOfDevices != null){
					for(int i=0; i < listOfDevices.length; i++){
						devicesArrayAdapter.add(listOfDevices[i]);
					}
				}
			} catch (Exception e) {
				String[] funnyList = {"test1", "test2"};
				devicesArrayAdapter.add(funnyList[0]);
				devicesArrayAdapter.add(funnyList[1]);
				e.printStackTrace();
			}

	}

	@Override
	public void onAttach(Activity activity){
		super.onAttach(activity);
		try {
			mListener = (DCTalkToActivity) activity;
		} catch (ClassCastException e){
			throw new ClassCastException(activity.toString() + " must implement
DCTalkToActivity");
		}
	}


	/* Με την επιλογή μίας από τις συσκευές που εμφανίζονται στη λίστα, καλείται η listenFrDc
με τα κατάλληλα ορίσματα. Τα υπόλοιπα τα αναλαμβάνει η MainActivity.  */


	@Override
	public void onListItemClick(ListView l, View v, int position, long id){
		mListener.listenFrDc(getListView().getItemAtPosition(position).toString(), 2);
			}

}
```

3.3.4 **rcFragment.java**

/* Στην κλάση αυτή υλοποιείται η διεπαφή sendOverBluetooth, μέσω της οποίας καλούνται οι συναρτήσεις sendString και setButtonDown, για την αποστολή στοιχείων μέσω Bluetooth, και την αποστολή του χαρακτήρα επανάληψης όσο είναι πατημένο κάποιο πλήκτρο. */

```
package com.myremotecontrolv01.fragments;

import java.io.BufferedInputStream;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.util.ArrayList;
import java.util.List;
import com.example.myremotecontrol01.R;
import com.myremotecontrolv01.fragments.dcFragment.DCTalkToActivity;
import android.app.Activity;
import android.app.Fragment;
import android.content.Context;
import android.os.Bundle;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.MotionEvent;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.View.OnTouchListener;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.Toast;

public class rcFragment extends Fragment{

	static final String scbFilename = "setCustomButtonFile";

	sendOverBluetooth tmpListener;
	Button pButton;
	Button volUpButton;
	Button volDownButton;
	Button muteButton;
	Button chUpButton;
	Button chDownButton;
	Button menuButton;
	Button ch0Button;
	Button ch1Button;
	Button ch2Button;
	Button ch3Button;
	Button ch4Button;
	Button ch5Button;
```

```
Button ch6Button;
Button ch7Button;
Button ch8Button;
Button ch9Button;
Button customButton;
View V;
String tvDev;
List<String> buttonValues = new ArrayList<String>();
List<String> customButtonValues = new ArrayList<String>();

public interface sendOverBluetooth {
	public void sendString(String value);
	public void setButtonDown(boolean settingValue);
}
```

/* Κατά τη δημιουργία της κλάσης, φορτώνονται στα πλήκτρα οι λειτουργίες που τους αντιστοιχούν, από το αρχείο που υπάρχει στον φάκελο assets και έχει επιλέξει ο χρήστης. Επίσης φορτώνονται οι λειτουργίες του σύνθετου κουμπιού. */

```
@Override
public View onCreateView(LayoutInflater inflater,
		ViewGroup container, Bundle savedInstanceState){
	Bundle args = (Bundle) getArguments();
	tvDev = args.getString("tvDevice");

	V = inflater.inflate(R.layout.fragment_main, container, false);
	//Toast.makeText(V.getContext(),/* tvDev* + */" Printed @ onCreateView",
Toast.LENGTH_SHORT).show();

	pButton = (Button)V.findViewById(R.id.power);
	volUpButton = (Button) V.findViewById(R.id.vol_up);
	volDownButton = (Button) V.findViewById(R.id.vol_down);
	muteButton = (Button) V.findViewById(R.id.mute);
	chUpButton = (Button) V.findViewById(R.id.ch_up);
	chDownButton = (Button) V.findViewById(R.id.ch_down);
	menuButton = (Button) V.findViewById(R.id.menu);
	ch0Button = (Button) V.findViewById(R.id.button0);
	ch1Button = (Button) V.findViewById(R.id.button1);
	ch2Button = (Button) V.findViewById(R.id.button2);
	ch3Button = (Button) V.findViewById(R.id.button3);
	ch4Button = (Button) V.findViewById(R.id.button4);
	ch5Button = (Button) V.findViewById(R.id.button5);
	ch6Button = (Button) V.findViewById(R.id.button6);
	ch7Button = (Button) V.findViewById(R.id.button7);
	ch8Button = (Button) V.findViewById(R.id.button8);
	ch9Button = (Button) V.findViewById(R.id.button9);
	customButton = (Button) V.findViewById(R.id.buttonCustom);
```

```
            loadTvDevice();
            loadCustomButton();
            tmpListener.sendString(buttonValues.get(0));
            setButtons();

            return V;
        }
```

/* Η παρακάτω συνάρτηση φορτώνει στη λίστα buttonValues τα στοιχεία από το επιλεγμένο αρχείο του φακέλου assets. */

```
        public void loadTvDevice(){
            InputStream is;
            InputStreamReader isr;
            BufferedReader buf;


            try {
                    is = getActivity().getAssets().open("devices/"+tvDev);
                    isr = new InputStreamReader(is);
                    buf = new BufferedReader(isr);
                    String buf2;
                    buttonValues.clear();
                    while((buf2 = buf.readLine())!= null){
                            buttonValues.add(buf2);
                    }
            } catch (IOException e){
                    return;
            }
        }
```

/* Στη λίστα customButtonValues, φορτώνονται με την παρακάτω συνάρτηση τα στοιχεία για τις λειτουργίες του σύνθετου πλήκτρου. */

```
        public void loadCustomButton(){

            InputStream is;
            InputStreamReader isr;
            BufferedReader buf;


            try {
                    is = new BufferedInputStream(getActivity().openFileInput(scbFilename));
                    isr = new InputStreamReader(is);
                    buf = new BufferedReader(isr);
                    String buf2;
                    customButtonValues.clear();
```

```
                int i = 0;
                while((buf2 = buf.readLine())!= null){
                        customButtonValues.add(buf2);
                        i++;
                }
                buf.close();
                isr.close();
                is.close();

        } catch (IOException e){
                return;
        }
}
```

/* Με τις δύο παρακάτω συναρτήσεις, αντιστοιχίζονται τα πλήκτρα της διεπαφής χρήστη με τα στοιχεία που πρέπει να αποσταλούν μέσω Bluetooth και βρίσκονται στη λίστα buttonValues. Κάθε κουμπί αντιστοιχίζεται και στη συνάρτηση που καλείται κατά το πάτημα του. */

```
public void setButtons(){
        setOnTouchMethod(pButton, 1);
        setOnTouchMethod(volUpButton, 2);
        setOnTouchMethod(volDownButton, 3);
        setOnTouchMethod(muteButton, 18);
        setOnTouchMethod(chUpButton, 15);
        setOnTouchMethod(chDownButton, 16);
        setOnTouchMethod(menuButton, 17);
        setOnTouchMethod(ch0Button, 4);
        setOnTouchMethod(ch1Button, 5);
        setOnTouchMethod(ch2Button, 6);
        setOnTouchMethod(ch3Button, 7);
        setOnTouchMethod(ch4Button, 8);
        setOnTouchMethod(ch5Button, 9);
        setOnTouchMethod(ch6Button, 10);
        setOnTouchMethod(ch7Button, 11);
        setOnTouchMethod(ch8Button, 12);
        setOnTouchMethod(ch9Button, 13);
        setOnTouchMethodCustomButton(customButton);
}



private void setOnTouchMethod(Button a, final int i){

        a.setOnTouchListener(new OnTouchListener(){

                @Override
                public boolean onTouch(View v, MotionEvent event){
                        int eventAction;
```

```
                        boolean runLoop = false;
                        switch(eventAction = event.getAction()){
                        case MotionEvent.ACTION_DOWN:
                                tmpListener.sendString(buttonValues.get(i));
                                //Toast.makeText(V.getContext(), "On Touch Listener",
Toast.LENGTH_SHORT).show();
                                tmpListener.setButtonDown(true);
                                runLoop = true;
                                break;
                        case MotionEvent.ACTION_UP:
                                eventAction = MotionEvent.ACTION_UP;
                                tmpListener.setButtonDown(false);
                                runLoop = false;
                                //Toast.makeText(V.getContext(), "button UP event",
Toast.LENGTH_SHORT).show();
                                break;
                        }
                        return false;
                }

        });
    }
```

/* Με το πάτημα του σύνθετου κουμπιού, πρέπει να αποσταλούν τα δεδομένα που υπάρχουν στη λίστα customButtonValues. Πιο συγκεκριμένα, στη λίστα υπάρχουν για κάθε λειτουργία, τρεις σειρές με δεδομένα. Στην πρώτη σειρά αναφέρεται το όνομα της συσκευής ή πρωτοκόλλου IR και η λειτουργία της συσκευής που θα κληθεί. Η δεύτερη σειρά έχει αποθηκευμένο τα δεδομένα του πρωτοκόλλου και η τρίτη την εντολή με τη μορφή που πρέπει να αποσταλούν μέσω Bluetooth. Έτσι όταν πατιέται το πλήκτρο αυτό αποστέλλεται κάθε δεύτερη και τρίτη σειρά της λίστας customButtonValues μέσω Bluetooth. Η λειτουργία αυτή ορίζεται στην παρακάτω συνάρτηση. */

```
    private void setOnTouchMethodCustomButton(Button a){

        a.setOnTouchListener(new OnTouchListener(){

                @Override
                public boolean onTouch(View v, MotionEvent event){
                        int eventAction;
                        boolean runLoop = false;
                        switch(eventAction = event.getAction()){
                        case MotionEvent.ACTION_DOWN:
                                for(int i=0 ; i+2 < customButtonValues.size() ; i=i+3){
                                        tmpListener.sendString(customButtonValues.get(i+1));
                                        tmpListener.sendString(customButtonValues.get(i+2));
                                }
                                break;
                        case MotionEvent.ACTION_UP:
                                break;
```

```
                        }
                        return false;
                    }

                });
        }


        @Override
        public void onAttach(Activity activity){
                super.onAttach(activity);
                try {
                        tmpListener = (sendOverBluetooth) activity;
                } catch (ClassCastException e){
                        throw new ClassCastException(activity.toString() + " must implement
SendOverBluetooth");
                }
        }

}
```

3.3.5 **setCustomButtonFragment.java**

/* Στην κλάση αυτή υλοποιείται η διεπαφή scbTalkToActivity μέσω της οποίας καλείται η συνάρτηση listenFrScb. Η κλάση αναλαμβάνει να δημιουργήσει την διεπαφή για την παραμετροποίηση του customButton και να αποθηκεύσει τα δεδομένα του κουμπιού στο κατάλληλο αρχείο. */

```
package com.myremotecontrolv01.fragments;

import java.io.BufferedInputStream;
import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.util.ArrayList;
import java.util.List;
import android.app.Activity;
import android.app.Fragment;
import android.content.Context;
import android.content.res.AssetManager;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.ListView;
import android.widget.TextView;

import com.example.myremotecontrol01.R;
import com.myremotecontrolv01.fragments.dcFragment.DCTalkToActivity;


public class setCustomButtonFragment extends Fragment{

	public interface scbTalkToActivity {
		public void listenFrScb();
	}

	scbTalkToActivity scbListener;

	View SCB;
	List<String> availableDeviceList = new ArrayList<String>();
	List<String> CBFileContents = new ArrayList<String>();
```

```
	//String CBList[];
	List<String> CBList = new ArrayList<String>();
	String[] operationList = {"Power","Vol +","Vol
-","0","1","2","3","4","5","6","7","8","9",null,"Channel +","Channel -","Menu","Mute"};
	String chosenDevice, chosenOperation;
	int operationLine = 0;
	static final String scbFilename = "setCustomButtonFile";
	String tvDev;
```

/* Η διεπαφή χρήστη της κλάσης αυτής έχει μία γραμμή κειμένου, που εμφανίζεται η επιλογή πρωτοκόλλου και λειτουργίας, τρεις λίστες και τρία κουμπιά. Μια από τις λίστες εμφανίζει τα διαθέσιμα πρωτόκολλα IR. Η δεύτερη εμφανίζει τις διαθέσιμες λειτουργίες ανά πρωτόκολλο ενώ τέλος η τρίτη λίστα εμφανίζει τις λειτουργίες που εκτελούνται με το πάτημα του σύνθετου πλήκτρου. Το πλήκτρο Add, αναλαμβάνει να προσθέσει την επιλεγμένη λειτουργία στη λίστα λειτουργιών του σύνθετου πλήκτρου. Το Reset αναλαμβάνει να σβήσει τη λίστα λειτουργιών του σύνθετου πλήκτρου και το Done επικοινωνεί με τη MainActivity για να κλείσει αυτή η διεπαφή χρήστη και να φορτωθεί η διεπαφή χρήστη rcFragment. */

```
	@Override
	public void onCreate(Bundle savedInstanceState){
		super.onCreate(savedInstanceState);


	}


	@Override
	public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
savedInstanceState){

		SCB = inflater.inflate(R.layout.scblayout, container, false);
		ArrayAdapter<String> availableDevicesAdapter =new
ArrayAdapter<String>(getActivity(), android.R.layout.simple_list_item_1, availableDeviceList);
		final ListView lv = (ListView) SCB.findViewById(R.id.availableDeviceList);
		lv.setAdapter(availableDevicesAdapter);

		ArrayAdapter<String> operationAdapter = new ArrayAdapter<String>(getActivity(),
android.R.layout.simple_list_item_1, operationList);
		final ListView operationListView = (ListView)
SCB.findViewById(R.id.operationList);
		operationListView.setAdapter(operationAdapter);

		ArrayAdapter<String> CBFileAdapter = new ArrayAdapter<String>(getActivity(),
android.R.layout.simple_list_item_1, CBList);
		final ListView CBContentsLV = (ListView) SCB.findViewById(R.id.CBList);
		CBContentsLV.setAdapter(CBFileAdapter);
```

```
		AssetManager assetMan = getActivity().getAssets();
		final TextView textView1 = (TextView) SCB.findViewById(R.id.textView1);
		textView1.setText("Device : " + chosenDevice + " Operation : " + chosenOperation);
		final Button add = (Button)SCB.findViewById(R.id.addOperation);
		final Button done = (Button)SCB.findViewById(R.id.doneSCB);
		final Button reset = (Button)SCB.findViewById(R.id.resetButton);



		lv.setOnItemClickListener(new OnItemClickListener(){
			@Override
			public void onItemClick(AdapterView<?> parent, View view, int position,
long id) {
				chosenDevice = (String)lv.getItemAtPosition(position);
				textView1.setText("Device : " + chosenDevice + " Operation : " +
chosenOperation);
			}

		});

		operationListView.setOnItemClickListener(new OnItemClickListener(){
			@Override
			public void onItemClick(AdapterView<?> parent, View view, int position,
long id) {
				operationLine = position +1;
				chosenOperation =
(String)operationListView.getItemAtPosition(position);
				textView1.setText("Device : " + chosenDevice + " Operation : " +
chosenOperation);
			}

		});

		add.setOnClickListener(new OnClickListener(){

			@Override
			public void onClick(View v) {

				String text = chosenDevice + " -- " + chosenOperation + "\n";
				String protocol;
				String operation;

				protocol = assetFileLine(0)+"\n";
				operation = assetFileLine(operationLine)+"\n";

				try {
					FileOutputStream scbOutputStream =
getActivity().openFileOutput(scbFilename, Context.MODE_APPEND);
					scbOutputStream.write(text.getBytes());
```

```
                        scbOutputStream.write(protocol.getBytes());
                        scbOutputStream.write(operation.getBytes());
                        scbOutputStream.close();
                    } catch (FileNotFoundException e) {
                        e.printStackTrace();
                    } catch (IOException e) {
                        e.printStackTrace();
                    }

                    updateCBList();

                }

            });

            done.setOnClickListener(new OnClickListener(){

                @Override
                public void onClick(View v) {
                    scbListener.listenFrScb(); //updateUI

                }

            });

            reset.setOnClickListener(new OnClickListener(){

                @Override
                public void onClick(View arg0) {

                    try {
                        FileOutputStream scbOutputStream =
getActivity().openFileOutput(scbFilename, 0);
                        scbOutputStream.flush();
                        scbOutputStream.close();
                    } catch (FileNotFoundException e) {
                        e.printStackTrace();
                    } catch (IOException e) {
                        e.printStackTrace();
                    }

                    updateCBList();
                }

            });

            try {
                String[] listOfDevices = assetMan.list("devices");

                if (listOfDevices != null){
```

```
                    for(int i=0; i < listOfDevices.length; i++){
                        availableDeviceList.add(listOfDevices[i]);
                    }
                }
            } catch (Exception e) {
                e.printStackTrace();
            }

            updateCBList();

            return SCB;
        }
```

/* Με την παρακάτω συνάρτηση ανανεώνεται η λίστα εμφάνισης λειτουργιών σύνθετου πλήκτρου. Η συνάρτηση καλείται κάθε φορά που υπάρχει αλλαγή στις λειτουργίες σύνθετου πλήκτρου, ώστε η αλλαγή αυτή να αποτυπώνεται και στη διεπαφή χρήστη. */

```
        public void updateCBList(){
            InputStream is;
            InputStreamReader isr;
            BufferedReader buf;


            try {
                is = new BufferedInputStream(getActivity().openFileInput(scbFilename));
                isr = new InputStreamReader(is);
                buf = new BufferedReader(isr);
                String buf2;
                CBFileContents.clear();
                CBList.clear();
                int i=0;
                while((buf2 = buf.readLine())!= null){
                    CBFileContents.add(buf2);
                    if(i % 3 == 0){
                        CBList.add(buf2);
                    }
                    i++;
                }
                buf.close();
                isr.close();
                is.close();

            } catch (IOException e){
                return;
            }

            ArrayAdapter<String> CBFileAdapter = new ArrayAdapter<String>(getActivity(),
```

```
android.R.layout.simple_list_item_1, CBList);
                final ListView CBContentsLV = (ListView) SCB.findViewById(R.id.CBList);
                CBContentsLV.setAdapter(CBFileAdapter);

        }


        /* Κατά την επιλογή του πλήκτρου Add, συλλέγονται τα στοιχεία από το αρχείο που
αντιστοιχεί στο επιλεγμένο πρωτόκολλο και αποθηκεύονται στη λίστα list. Για την επιλογή
συγκεκριμένης γραμμής από το αρχείο του φακέλου assets, χρησιμοποιείται η παρακάτω
συνάρτηση.   */


        public String assetFileLine(int i){
                InputStream is;
                InputStreamReader isr;
                BufferedReader buf;
                List<String> list = new ArrayList<String>();

                try {
                        is = getActivity().getAssets().open("devices/"+chosenDevice);
                        isr = new InputStreamReader(is);
                        buf = new BufferedReader(isr);
                        String buf2;
                        list.clear();
                        while((buf2 = buf.readLine())!= null){
                                list.add(buf2);
                        }
                } catch (IOException e){
                        return null;
                }
                return list.get(i);
        }

        @Override
        public void onAttach(Activity activity){
                super.onAttach(activity);
                try {
                        scbListener = (scbTalkToActivity) activity;
                } catch (ClassCastException e){
                        throw new ClassCastException(activity.toString() + " must implement
DCTalkToActivity");
                }
        }
}
```

Παράρτημα Ε'

Περιεχόμενα του CD

Περιεχόμενα του CD

Στο παρακάτω σχήμα εμφανίζονται τα περιεχόμενα του CD. Ο φάκελος *MyRemotecontrol01*, έχει τη δομή που παρουσιάζεται στο πρώτο μέρος του παραρτήματος Δ'.