Τεχνολογικό Εκπαιδευτικό Ίδρυμα Κρήτης

Σχολή Τεχνολογικών Εφαρμογών
Τμήμα Μηχανικών Πληροφορικής

Πτυχιακή Εργασία

Ανάπτυξη εφαρμογής τηλεχειριζόμενου οχήματος, με πλατφόρμα Arduino

Τομέας: Επικοινωνιών και Πολυμέσων

Πετράκης Ζαχαρίας 2242

Επιβλέπων Καθηγητής : Βλησίδης Ανδρέας

Επιτροπή Αξιολόγησης : Παναγιωτάκης Σπύρος, Στρατάκης Δημήτρης

Ημερομηνία Παρουσίασης : 27/11/2014

Ευχαριστίες

Η παρούσα πτυχιακή εργασία πραγματοποιήθηκε από τον φοιτητή Πετράκη Ζαχαρία του τμήματος Μηχανικών Πληροφορικής στο ΤΕΙ Κρήτης κατά το ακαδημαϊκό έτος 2013-2014.

Θα ήθελα να ευχαριστήσω θερμά τον επιβλέποντα καθηγητή μου, κύριο Βλησίδη Ανδρέα, που μου έδωσε την δυνατότητα να ασχοληθώ με ένα τόσο ενδιαφέρον θέμα και για την υποστήριξη, την καθοδήγηση και τι πολύτιμες συμβουλές του καθ' όλη την διάρκεια διεκπεραίωσης της πτυχιακής μου εργασίας. Επίσης θα ήθελα να ευχαριστήσω όλους του καθηγητές του τμήματος μηχανικών πληροφορικής για τις πολύτιμες γνώσεις μου έδωσαν όλα αυτά τα χρόνια.

Τέλος, ευχαριστώ τους γονείς μου, τον αδερφό μου, τους συναδέλφους μου και τους φίλους μου που με στήριξαν με κάθε τρόπο σε όλη την διάρκεια των σπουδών μου.

Σύνοψη

Σκοπός της πτυχιακής αυτής είναι η δημιουργία ενός τηλεχειριζόμενου οχήματος το οποίο θα μπορεί να εκτελεί της βασικές λειτουργίες κίνησης δηλαδή εμπρός, αριστερά, δεξιά και πίσω. Ο έλεγχος από τον χρήστη γίνεται ασύρματα με Bluetooth μέσω εφαρμογής Android η οποία έχει εγκατασταθεί σε μια Tablet συσκευή.

Το όχημα μας αποτελείται από το σασί, μια ειδική πλατφόρμα για Arduino με τέσσερα DC μοτέρ, το Arduino Uno R3, ένα Motor shield το οποίο χρησιμεύει για να δώσουμε κίνηση στα μοτέρ, το Bluetooth module απαραίτητο για την επικοινωνία, ένα σερβοκινητήρα με δυνατότητα περιστροφής 90 μοίρες αριστερά και 90 δεξιά, μία οθόνη LCD και τέλος έχει τοποθετηθεί ένας αισθητήρας απόστασης επάνω στο σερβοκινητήρα που όπως θα εξηγήσουμε παρακάτω θα μας βοηθήσει να αποφύγουμε τυχόν σύγκρουση με κάποιο σταθερό εμπόδιο.

Ο προγραμματισμός του Arduino έχει γίνει χρησιμοποιώντας το open-source λογισμικό Arduino IDE της εταιρίας και γράφοντας σε γλώσσα C/C++ ενώ για την Android εφαρμογή έχουμε χρησιμοποιήσει το πρόγραμμα Eclipse της Android γράφοντας στη γλώσσα Java.

Abstract

The purpose of this thesis is to provide a remotely controlled vehicle which will be able to perform some basic steering functions like moving forward, left, right and rear. It is controlled by the user wirelessly, using the Bluetooth protocol, through an android application which is installed on a Tablet device.

It consists of a chassis, a specific platform for Arduino with four DC motors, an Arduino Uno R3, a Motor Shield which serves to give movement to the motors, the Bluetooth module necessary for communication, a servo motor rotatably ninety degrees left and ninety degrees right. Finally a distance sensor is placed over the servo motor and will help us to avoid any conflict with a rigid obstacle.

Arduino is programmed using the open source Arduino IDE software in C/C++ language and the Android application to the Android Eclipse software in Java.

Πίνακας Περιεχομένων

Πίνακας Εικόνων

Πίνακας Πινάκων

1 Εισαγωγή

Στην παρούσα πτυχιακή εργασία θα ασχοληθούμε με την υπολογιστική πλατφόρμα Arduino Uno R3 και θα μελετήσουμε την εφαρμογή της σε σύστημα ασύρματου ελέγχου, τηλεχειριζόμενο όχημα, όπου η επικοινωνία γίνεται μέσω της τεχνολογίας Bluetooth. Θα αναφέρουμε κάποια βασικά ιστορικά στοιχεία που αφορούν την πλατφόρμα αυτή. Επίσης θα εξηγήσουμε αναλυτικά τι είναι το Arduino, γιατί το χρησιμοποιούμε και πως, αναλυτική περιγραφή του Hardware αλλά και του Software που χρησιμοποιούμε για να το προγραμματίσουμε. Ακόμα, θα γίνει αναφορά σε όλο τον απαραίτητο εξοπλισμό, πέραν του Arduino, που χρησιμοποιήθηκε στο τηλεχειριζόμενο όχημα.

Ένα σπουδαίο κεφάλαιο της εργασίας μας αφορά το λειτουργικό σύστημα Android και την ανάπτυξη εφαρμογών για το συγκεκριμένο λειτουργικό σύστημα, οπότε θα δούμε αρχικά τι είναι το Android, τα χαρακτηριστικά του, τα πλεονεκτήματα του, την αρχιτεκτονική του κ.α. Στη συνέχεια, θα δούμε τα πρώτα βήματα που πρέπει να ακολουθήσουμε για να δημιουργήσουμε μια εφαρμογή. Τέλος, θα αναφερθούμε αναλυτικά στην εφαρμογή που δημιουργήσαμε εμείς για την ασύρματη επικοινωνία και έλεγχο του Arduino.

Ένα τηλεκατευθυνόμενο όχημα μπορεί να έχει αρκετές επιστημονικές εφαρμογές όπως χρήση σε επικίνδυνο για τον άνθρωπο περιβάλλον, την εξερεύνηση του διαστήματος, σε στρατιωτικές επιχειρήσεις αλλά και στην απλούστερη περίπτωση ως χόμπι.

1.1 Το Κίνητρο για την Διεξαγωγή της Εργασίας

Κίνητρο αποτέλεσε το μάθημα των βιομηχανικών αυτοματισμών, κατά το ακαδημαϊκό έτος 2012-2013, με διδάσκων τον Κ. Ανδρέα Βλησίδη στο οποίο μου δόθηκε η ευκαιρία να ασχοληθώ για πρώτη φορά με την υπολογιστική πλατφόρμα Arduino και να μελετήσω τις δυνατότητες της. Είναι πολύ ενδιαφέρον το πλήθος των εφαρμογών που μπορούμε να δημιουργήσουμε προγραμματίζοντας ένα Arduino, κάποια ωραία παραδείγματα είναι το σύστημα ελέγχου κλιματισμού κατοικίας, το χρονόμετρο, το Led Matrix Control, τα τηλεκατευθυνόμενα οχήματα, το ρολόι από Led κ.α. Επίσης ήθελα να ασχοληθώ με το λειτουργικό σύστημα Android, καθώς είναι αρκετά δημοφιλές στις μέρες μας, και το οποίο παρέχει την δυνατότητα δημιουργίας πολύ ωραίων εφαρμογών με αρκετές δυνατότητες.

1.2 Ο Σκοπός και οι Στόχοι της Εργασίας

Ο στόχος της εργασίας αυτής είναι η εξοικείωση με το Arduino και την γλώσσα C/C++ στην οποία το προγραμματίζουμε, καθώς και η απόκτηση εμπειρίας στην κατασκευή εφαρμογών Android χρησιμοποιώντας την γλώσσα Java. Ο σκοπός της εργασίας είναι η δημιουργία ενός τηλεχειριζόμενου οχήματος το οποίο θα λειτουργεί ασύρματα και θα μπορεί να εκτελεί τις βασικές λειτουργίες κίνησης, όλα αυτά θα τα δούμε στα κεφάλαια που ακολουθούν. Θα ήθελα η εργασία αυτή να αποτελέσει ένα καλό οδηγό για κάποιον που θα ήθελε να δημιουργήσει από την αρχή ένα τηλεκατευθυνόμενο όχημα που θα ελέγχεται ασύρματα από κάποια Android εφαρμογή.

1.3 Η Δομή της Εργασίας

Στο δεύτερο κεφάλαιο παρουσιάζεται η ασύρματη τεχνολογία Bluetooth. Θα γίνει μία μικρή ιστορική αναδρομή και στην συνέχεια θα μιλήσουμε για την αρχιτεκτονική του πρωτοκόλλου, τον τρόπο λειτουργίας του κτλ. Δεδομένου ότι χρησιμοποιούμε το συγκεκριμένο πρωτόκολλο για την επικοινωνία του Arduino με την Android εφαρμογή μας, το κεφάλαιο αυτό θα μας βοηθήσει αρκετά να μάθουμε βασικά στοιχεία για τον τρόπο λειτουργίας της τεχνολογίας αυτής.

Στο τρίτο κεφάλαιο γίνεται μία αρκετά καλή εισαγωγή στους μικροελεγκτές. Ασχολούμαστε με τις κατηγορίες που διακρίνουν τους διάφορους τύπους μικροελεγκτών. Στη συνέχεια επικεντρωνόμαστε περισσότερο στους μικροελεγκτές AVR, σε ότι αφορά το Hardware τους, μιας και οι συγκεκριμένοι μικροελεγκτές χρησιμοποιούνται στις πλατφόρμες Arduino. Το κεφάλαιο αυτό είναι αρκετά ενδιαφέρον διότι μας βοηθά να κατανοήσουμε βασικά στοιχεία για τους μικροελεγκτές, από την αρχιτεκτονική τους μέχρι και τον τρόπο λειτουργίας τους.

Στο τέταρτο κεφάλαιο ασχολούμαστε με την υπολογιστική πλατφόρμα Arduino Uno R3. Το Arduino αποτελεί ένα από τα πιο βασικά στοιχεία της εφαρμογής μας. Ξεκινούμε με μία μικρή ιστορική αναδρομή και στην συνέχεια μιλούμε λεπτομερώς για τις προδιαγραφές της πλατφόρμας αυτής, τον τρόπο λειτουργίας της και τον προγραμματισμό της. Τελειώνοντας το κεφάλαιο αυτό θα έχουμε μία πολύ καλή γνώση σχετικά με την συγκεκριμένη πλατφόρμα, πράγμα που θα βοηθήσει στην κατανόηση του τρόπου λειτουργίας του τηλεχειριζόμενου οχήματος.

Στο πέμπτό κεφάλαιο γίνεται μία περιγραφή όλου του εξοπλισμού που χρησιμοποιούμε στο τηλεχειριζόμενο όχημα, την σωστή συνδεσμολογία του, τα χαρακτηριστικά του καθενός, την χρησιμότητα του κ.α. Πρόκειται για ένα αρκετά ενδιαφέρον κεφάλαιο καθώς βλέπουμε τον τρόπο με τον οποίο το όχημα μας αποκτά μορφή. Στο τέλος παρουσιάζεται ένα σχηματικό διάγραμμα με όλη την συνδεσμολογία του τηλεχειριζόμενου οχήματος.

Στο έκτο κεφάλαιο προγραμματίζουμε το Arduino. Παρουσιάζεται όλος ο απαραίτητος κώδικας για την σωστή λειτουργία του τηλεχειριζόμενου οχήματος με όλες τις αναγκαίες διευκρινήσεις έτσι ώστε να γίνει όσο το δυνατόν πιο κατανοητός.

Στο έβδομο κεφάλαιο παρουσιάζουμε το λειτουργικό σύστημα Android. Ξεκινάμε με μία ιστορική αναδρομή και στην συνέχεια αναφέρουμε τις διάφορες εκδόσεις του λειτουργικού συστήματος, μέχρι και σήμερα. Έπειτα αναλύεται η αρχιτεκτονική του Android, πράγμα σημαντικό στην κατανόηση της λειτουργίας του, και τα συστατικά μίας εφαρμογής. Τέλος θα δούμε πως μπορούμε να δημιουργήσουμε ένα Project, να τρέξουμε μία εφαρμογή σε εικονική ή και πραγματική συσκευή κ.α.

Στο όγδοο κεφάλαιο πλέον, δημιουργούμε την εφαρμογή μας στο λειτουργικό σύστημα Android, για τον ασύρματο έλεγχο του τηλεχειριζόμενου οχήματος. Όλος ο κώδικας της εφαρμογής μας αναλύεται όσο το δυνατόν καλύτερα με έμφαση στα πιο σημαντικά κομμάτια του.

Στο ένατο και τελευταίο κεφάλαιο παρουσιάζεται όλη η βιβλιογραφία που χρησιμοποιήθηκε για την σύνταξη της παρούσας εργασίας.

2 Η Ασύρματη Τεχνολογία Bluetooth

Το Bluetooth αποτελεί ένα τυποποιημένο πρωτόκολλο για την αποστολή και λήψη δεδομένων στη μη αδειοδοτημένη ραδιοσυχνότητα ISM (Industrial, Scientific and Medical radio bands) των 2.4 GHz. Πρόκειται για ένα ασφαλές πρωτόκολλο, τέλειο για χαμηλών αποστάσεων, χαμηλής κατανάλωσης και χαμηλού κόστους ασύρματες μεταδόσεις, μεταξύ ηλεκτρονικών συσκευών. Μπορεί να αντικαταστήσει πολύ καλά διασυνδέσεις σειριακής επικοινωνίας. Στις μέρες μας οι ασύρματες συσκευές κυριαρχούν και το Bluetooth αποτελεί ένα μεγάλο μέρος της ασύρματης επανάστασης.

Εικόνα 2-1: Το λογότυπο του Bluetooth

2.1 Ιστορική Αναδρομή

Το 1994 η εταιρία κινητής τηλεφωνίας Ericson έδειξε ενδιαφέρον για την σύνδεση των κινητών της τηλεφώνων με άλλες συσκευές, π.χ PDA, χωρίς την χρήση καλωδίων. Μαζί με τις εταιρίες IBM, Intel, Nokia και Toshiba σχημάτισε μία ομάδα ειδικών ενδιαφερόντων ή αλλιώς SIG (Special Interest Group) για την ανάπτυξη ενός πρότυπου ασύρματης διασύνδεσης υπολογιστικών και επικοινωνιακών συσκευών και βοηθημάτων με την χρήση ραδιοκυμάτων πομποδεκτών μικρής εμβέλειας, χαμηλής ισχύος, και χαμηλού κόστους. Το έργο ονομάστηκε Bluetooth από τον Harald Blatand, ένα βασιλιά των Βίκινγκς που ενοποίησε τη Δανία και τη Νορβηγία, επίσης χωρίς να χρησιμοποιεί καλώδια.

Εικόνα 2-2: Συσκευή Bluetooth της εταιρίας Bluefruit

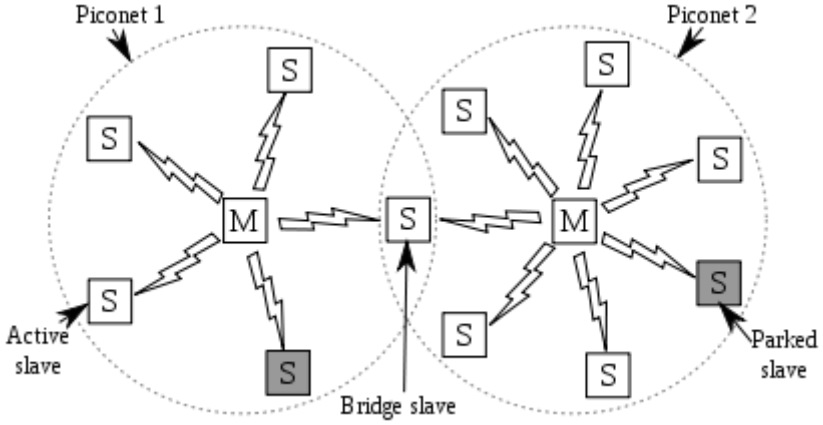
2.2 Η Αρχιτεκτονική του Bluetooth

Η βασική μονάδα ενός συστήματος Bluetooth είναι ένα μικροσκοπικό δίκτυο, το οποίο ονομάζουμε piconet, το οποίο αποτελείτε από ένα κόμβο κυρίου (master) και μέχρι και επτά ενεργούς κόμβου υπηρέτη (slaves) μέσα σε μία απόσταση δέκα μέτρων (Εικόνα 2-3). Πολλά μικροσκοπικά δίκτυα μπορούν να συνυπάρχουν στον ίδιο χώρο, ενώ μπορούν να είναι και συνδεδεμένα μέσω ενός κόμβου γέφυρας (Bridge Slave). Ένα διασυνδεδεμένο σύνολο μικροσκοπικών δικτυών ονομάζεται διάσπαρτο δίκτυο (scatternet) .

Εκτός από τους επτά ενεργούς slaves στο piconet, μπορούν να υπάρχουν μέχρι και 255 σταθμευμένοι (parked) κόμβοι στο δίκτυο. Οι κόμβοι αυτοί είναι συσκευές τις οποίες ο master έχει φέρει σε κατάσταση χαμηλής ισχύος, έτσι ώστε να μειώση την κατανάλωση των μπαταριών τους. Στην σταθμευμένη κατάσταση, η συσκευή δεν μπορεί να κάνει τίποτα άλλο από το να αποκρίνεται σε ένα σήμα ενεργοποίησης ή σε ένα σήμα φάρο από τον master. Υπάρχουν και οι ενδιάμεσε καταστάσεις ισχύος, η δέσμευση (hold) και η ανίχνευση (sniff).

Ο λόγος για την σχεδίαση master και slave είναι ότι οι σχεδιαστές ήθελαν να διευκολύνουν την υλοποίηση ολοκληρωμένων κυκλωμάτων Bluetooth με μικρό κόστος.

Όταν δύο Bluetooth συσκευές συνδέονται μεταξύ τους, αυτό το ονομάζουμε pairing. Η δομή και η παγκόσμια αποδοχή της τεχνολογίας Bluetooth σημαίνει πως κάθε συσκευή με δυνατότητα Bluetooth, σχεδόν παντού στον κόσμο, μπορεί να συνδεθεί σε άλλη συσκευή Bluetooth αρκεί να βρίσκονται η μία κοντά στην άλλη.



Εικόνα 2-3: Παράδειγμα δικτύου piconet

2.2.1 Τα Πλεονεκτήματα της Τεχνολογίας Bluetooth

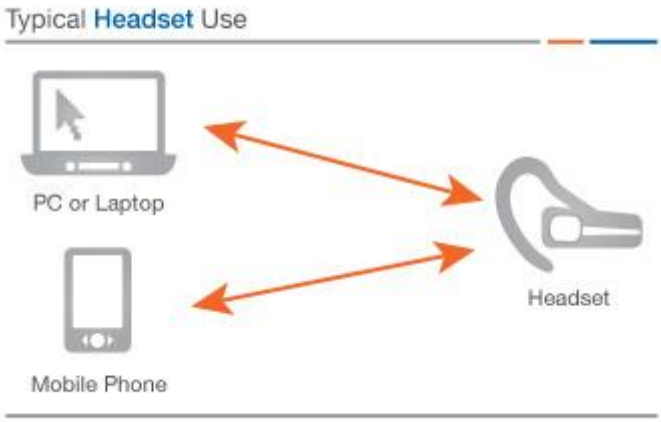
- Είναι ασύρματη και αυτόματη. Δε χρειάζονται καλώδια, μετατροπείς και συνδέσεις. Οι συσκευές αυτοαναγνωρίζονται και επικοινωνούν μεταξύ τους χωρίς τη βοήθεια του χρήστη, παρά μόνο σε ειδικές περιπτώσεις, π.χ για την πληκτρολόγηση κωδικού πρόσβασης.
- Το κόστος κατασκευής του κυκλώματος είναι χαμηλό. Οικονομικοί αναλυτές υπολογίζουν την τιμή του μικροκυκλώματος στα 15-20 ευρώ και μελλοντικά στα 5 ευρώ.
- Η ραδιοσυχνότητα λειτουργίας είναι ελεγχόμενη, αλλά ελεύθερη. Οι ίδιες συσκευές μπορούν να χρησιμοποιηθούν παντού στον πλανήτη (και στο διάστημα) χωρίς προβλήματα.
- Η τεχνολογία ελέγχει τη μετάδοση δεδομένων και φωνής. Οι εφαρμογές μπορούν να διαχειριστούν τα πλεονεκτήματα και των δύο μέσων μετάδοσης. Συνομιλία με handsfree μέσω φορητού υπολογιστή και ηχογράφηση ή εκτύπωση της συνομιλίας.
- Η ζεύξη είναι αμφίδρομη και περνά μέσα από τείχους και συρτάρια, η αναπήδηση συχνότητας (frequency hopping), ελαχιστοποιεί τον κίνδυνο διακοπής ή παρεμβολών στην επικοινωνία.
- Η ταχύτητα μεταφοράς δεδομένων είναι 1 Mb/s (Gross Data Rate).

2.3 Οι Εφαρμογές του Bluetooth

Για να χρησιμοποιήσουμε την ασύρματη τεχνολογία Bluetooth, μία συσκευή θα πρέπει να είναι σε θέση να ερμηνεύσει συγκεκριμένα προφίλ του Bluetooth. Τα προφίλ του Bluetooth, είναι ορισμοί πιθανών εφαρμογών και προσ-

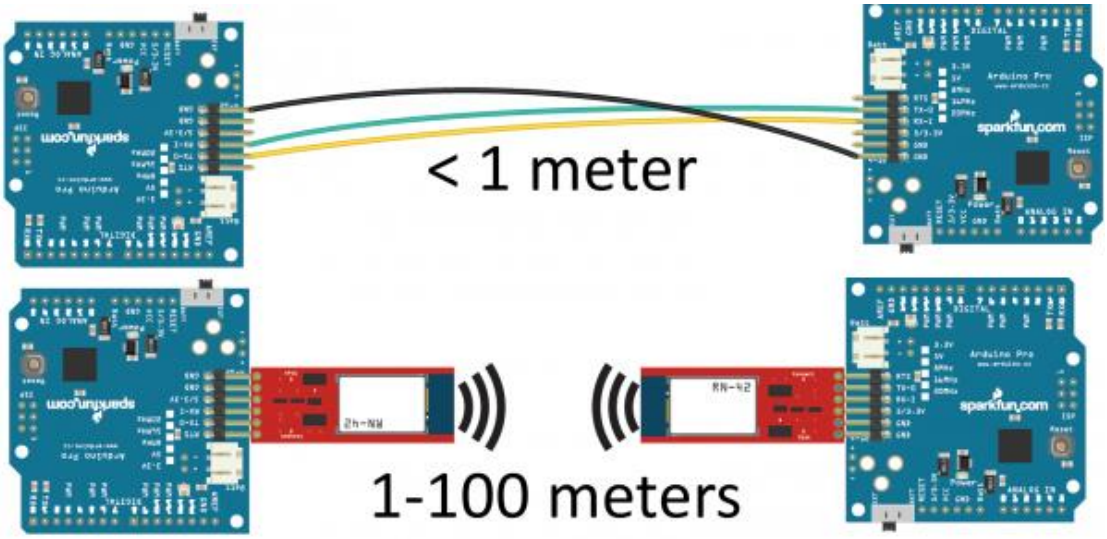
διορίζουν γενικές συμπεριφορές που χρησιμοποιούν οι συσκευές με Bluetooth για να επικοινωνήσουν με άλλες Bluetooth συσκευές. Υπάρχει ένα ευρύ φάσμα των προφίλ του Bluetooth που περιγράφουν πολλούς διαφορετικούς τύπους εφαρμογών ή περιπτώσεις χρήσης των συσκευών. Κάποια από τα κυριότερα προφίλ είναι τα εξής:

- HSP (Headset Profile): Προφίλ που περιγράφει την επικοινωνία ακουστικών Bluetooth με μία Bluetooth συσκευή (Εικόνα 2-4).



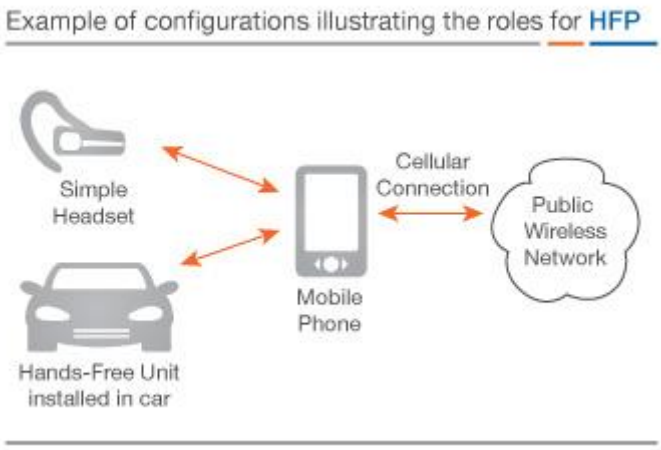
Εικόνα 2-4: Το προφίλ HSP

- SPP (Service Port Profile): Χρησιμοποιείται για την προσομοίωση σειριακής σύνδεσης με τη χρήση του επιπέδου RFCOMM (Εικόνα 2-5).



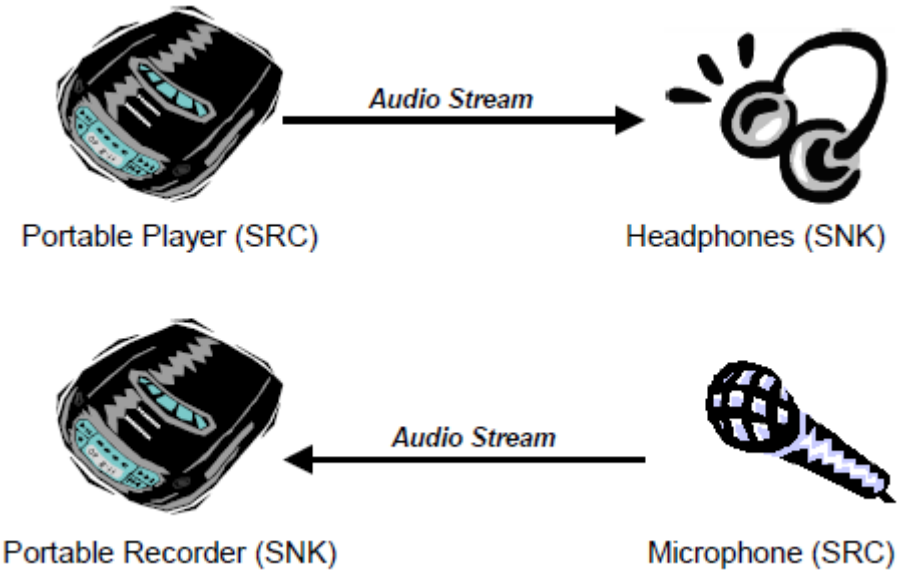
Εικόνα 2-5: Το προφίλ SPP

- HFP (Hands Free Profile): Παρόμοιο προφίλ με το HSP εκφράζει τον τρόπο που μία συσκευή μπορεί να χρησιμοποιηθεί για να λαμβάνει κλήσεις π.χ μονάδα Bluetooth εγκατεστημένη στο αυτοκίνητο μας (Εικόνα 2-6).



Εικόνα 2-6: Το προφίλ HFP

- A2DP (Advanced Audio Distribution Profile): Το προφίλ αυτό περιγράφει την μετάδοση stereo ήχου ανάμεσα σε μία Bluetooth συσκευή και ένα ηχοσύστημα (Εικόνα 2-7).

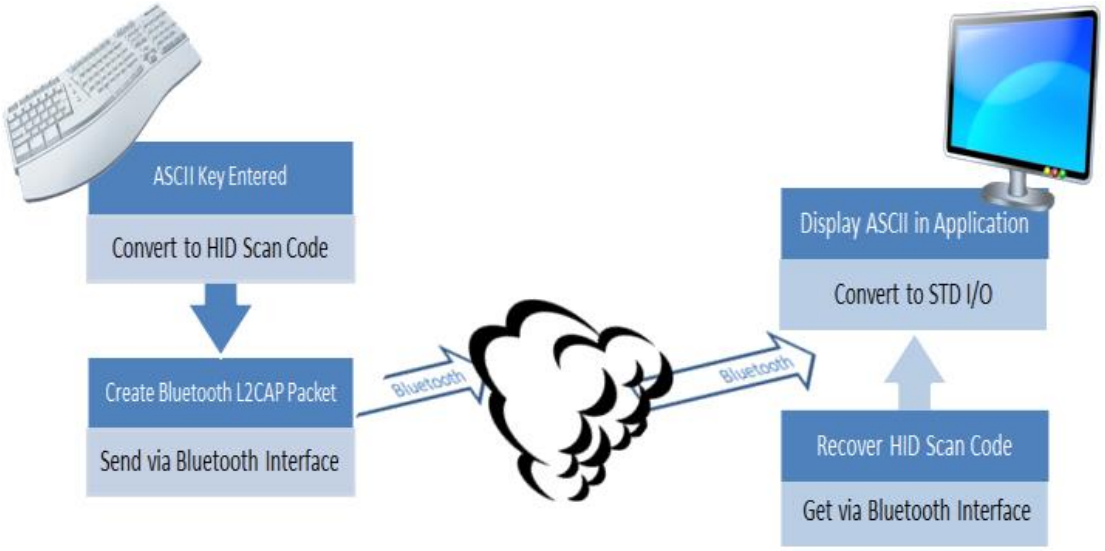


Εικόνα 2-7: Το προφίλ A2DP

- PBAP (Phone Book Access Profile): Το προφίλ αυτό καθορίζει τις διαδικασίες και τα πρωτόκολλα για την ανταλλαγή αντικειμένων τηλεφωνικού καταλόγου μεταξύ συσκευών (Εικόνα 2-8).

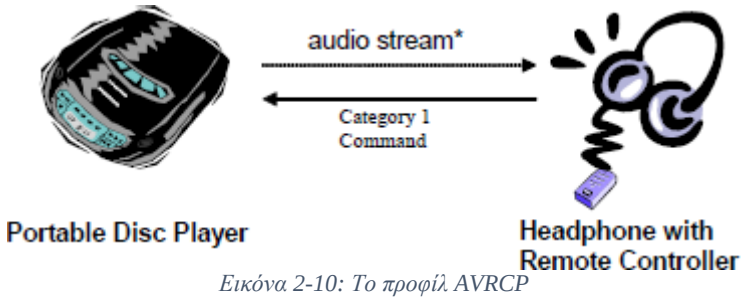
Εικόνα 2-8: Το προφίλ PBAP

- HID (Human Interface Device Profile): Το προφίλ αυτό ορίζει τα πρωτόκολλα, τις διαδικασίες και τα χαρακτηριστικά που πρέπει να χρησιμοποιούνται από Bluetooth πληκτρολόγια, ποντίκια συσκευές παιχνιδιών και συσκευές παρακολούθησης (Εικόνα 2-9).



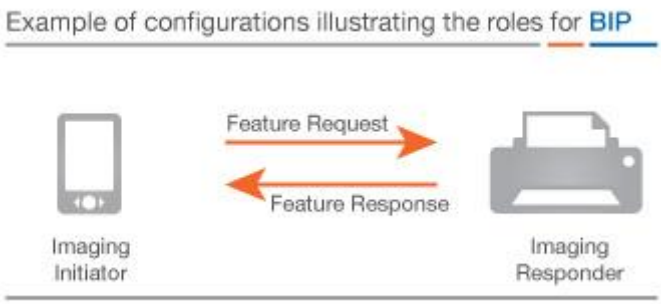
Εικόνα 2-9: Το προφίλ HID

- AVRCP (Audio/Video Remote Control Profile): Το προφίλ αυτό επιτρέπει σε ένα smart phone να λειτουργεί ως τηλεχειριστήριο για υπολογιστές με υποστήριξη Bluetooth και για εξοπλισμό Home Theater (Εικόνα 2-10).



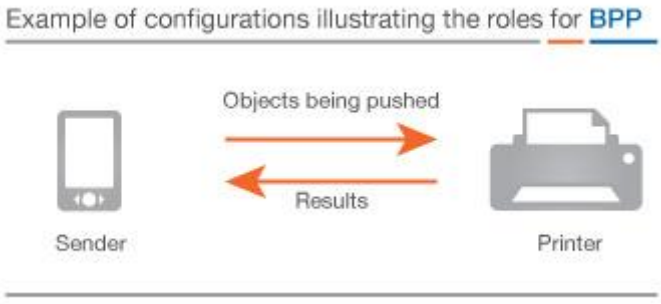
Εικόνα 2-10: Το προφίλ AVRCP

- BIP (Basic Imaging Profile): Το προφίλ αυτό υποστηρίζει την μετάδοση εικόνων μεταξύ συσκευών Bluetooth (Εικόνα 2-11).



Εικόνα 2-11: Το προφίλ BIP

- BPP (Basic Printing Profile): Το προφίλ αυτό υποστηρίζει την αποστολή των δεδομένων στον εκτυπωτή χωρίς να απαιτείται κάποιο πρόγραμμα οδήγησης στην συσκευή μετάδοσης (Εικόνα 2-12).



Εικόνα 2-12: Το προφίλ BPP

2.4 Η Λειτουργία του Bluetooth

Όπως αναφέραμε νωρίτερα το πρωτόκολλο bluetooth λειτουργεί στα 2.4 GHz, στην ίδια μη αδειοδοτημένη συχνότητα όπου πρωτόκολλα όπως το Zig Bee και Wi-Fi συνυπάρχουν όλα μαζί. Κάθε συσκευή Bluetooth διαθέτει μία μοναδική διεύθυνση 48-bit, συντομογραφία της οποίας είναι το BD_ADDR. Αυτή συνήθως θα παρουσιάζεται με τη μορφή μίας δεκαεξαδικής τιμής με δώδεκα ψηφία. Το πιο σημαντικό μισό (24-bit) της διεύθυνσης αποτελεί μοναδικό αναγνωριστικό του οργανισμού (Organization Unique Identifier, OUI), ο οποίος προσδιορίζει τον κατασκευαστή. Τα επόμενα 24-bit αποτελούν το πιο μοναδικό μέρος της διεύθυνσης. Η διεύθυνση αυτή θα πρέπει να είναι εμφανής στις περισσότερες Bluetooth συσκευές. Για παράδειγμα στη Bluetooth συσκευή RN-42 που βλέπουμε παρακάτω (εικόνα 2-13) η διεύθυνση MAC είναι η «00066644F3AA». Το «000666» τμήμα της διεύθυνσης είναι το αναγνωριστικό του οργανισμού της Roving Networks, ο κατασκευαστής της μονάδας. Κάθε μονάδα RN θα μοιραστεί αυτά τα 24-bit. Το «44F3AA» τμήμα της διεύθυνσης αποτελεί το μοναδικό αναγνωριστικό της συσκευής.

Εικόνα 2-13: Η MAC διεύθυνση του RN-42

Στις Bluetooth συσκευές δίνονται συνήθως φιλικά προς τον χρήστη ονόματα. Αυτά παρουσιάζονται στον χρήστη, στη θέση της διεύθυνσης, ώστε να μπορεί να αναγνωριστεί η επιθυμητή συσκευή. Οι κανόνες για τα ονόματα των συσκευών δεν είναι τόσο αυστηρά. Μπορούν να φτάσουν μέχρι και τα 248 Byte σε μέγεθος, και δύο συσκευές μπορούν να μοιραστούν το ίδιο όνομα. Κάποιες φορές, τα μοναδικά ψηφία της διεύθυνσης της συσκευής μπορεί να περιλαμβάνονται στο όνομα ώστε να διευκολύνεται η αναγνώριση των συσκευών.

2.4.1 Η Διαδικασία της Σύνδεσης

Η δημιουργία μίας Bluetooth σύνδεσης μεταξύ δύο συσκευών είναι μία διαδικασία πολλών σταδίων η οποία περιλαμβάνει τρείς προοδευτικές καταστάσεις. Η πρώτη αποτελεί την ανίχνευση (inquiry). Σε αυτή την κατάσταση, εάν οι δύο συσκευές Bluetooth δεν γνωρίζουν απολύτως τίποτα η μία για την άλλη, η μία θα πρέπει να τρέξει την διαδικασία της ανίχνευσης για να ανακαλύψει την άλλη. Η μία συσκευή θα στείλει ένα αίτημα ανίχνευσης και κάθε συσκευή που ακούει σε τέτοια αιτήματα θα ανταποκριθεί με την διεύθυνση της και πιθανά με το όνομα της και άλλες πληροφορίες. Η δεύτερη είναι ο συγχρονισμός. Σε αυτή την κατάσταση εκτελείται η διαδικασία σχηματισμού της σύνδεσης μεταξύ των δύο συσκευών. Πριν ξεκινήσει αυτή η σύνδεση κάθε συσκευή θα πρέπει να γνωρίζει την διεύθυνση της άλλης (αυτής που βρέθηκε κατά την ανίχνευση). Στην τελευταία κατάσταση, αυτήν της σύνδεσης, μεταβαίνει μία

συσκευή που έχει ολοκληρώσει την διαδικασία του συγχρονισμού. Ενώ η συσκευή είναι συνδεδεμένη, μπορεί είτε να συμμετέχει ενεργά είτε μπορεί να τεθεί σε κατάσταση χαμηλής ισχύος ύπνου. Υπάρχουν τέσσερις λειτουργίες της συσκευής όντας συνδεδεμένη: Active, Sniff, Hold και Park Mode. Το Active Mode αποτελεί την κανονική λειτουργεία σύνδεσης, όπου η συσκευή ενεργά στέλνει και λαμβάνει δεδομένα. Το Sniff Mode αποτελεί λειτουργία εξοικονόμησης ενέργειας, η συσκευή είναι λιγότερο ενεργή, θα κοιμηθεί και θα ακούει για μεταδόσεις σε ένα διάστημα π.χ 100 ms. Το Hold Mode αποτελεί μία λειτουργία προσωρινής εξοικονόμησης ενέργειας όπου η συσκευή κοιμάται για μία ορισμένη χρονική περίοδο και στην συνέχεια επιστρέφει στην κανονική λειτουργία όταν η περίοδος αυτή έχει περάσει. Μία Master συσκευή μπορεί να πει σε μία Slave συσκευή να μπει σε Hold Mode. Η λειτουργία Park Mode αποτελεί την βαθύτερη κατάσταση ύπνου. Μία Master συσκευή μπορεί να πει σε μία Slave συσκευή να μπει σε Pak Mode και αυτή η Slave συσκευή θα μείνει ανενεργή μέχρι η Master συσκευή να δώσει εντολή να ξυπνήσει και πάλι.

2.4.2 Η Λειτουργία της Αντιστοίχησης

Οι συσκευές που έχουν σχηματίσει δεσμό μεταξύ τους, αυτόματα δημιουργούν σύνδεση οποτεδήποτε βρίσκονται αρκετά κοντά. Για παράδειγμα, το Bluetooth ακουστικό του κινητού μας τηλεφώνου, συνδέεται αυτόματα με την συσκευή μας όταν βρίσκονται η μία κοντά στην άλλη. Δεν απαιτείται κάποια δράση από τον χρήστη. Οι δεσμοί δημιουργούνται μέσα από μία διαδικασία που ονομάζεται αντιστοίχηση (Pairing). Όταν οι συσκευές αντιστοιχίζονται, μοιράζονται τις διευθύνσεις τους, τα ονόματα τους, και τα προφίλ τους και συνήθως τα αποθηκεύουν στην μνήμη. Επίσης μοιράζονται ένα κοινό μυστικό κλειδί, το οποίο τους επιτρέπει να συνδέονται κάθε φορά που βρίσκονται κοντά στο μέλλον.

Η αντιστοίχηση (Pairing), απαιτεί μία διαδικασία ελέγχου ταυτότητας, όπου ο χρήστης πρέπει να επικυρώσει την σύνδεση μεταξύ των συσκευών. Η ροή της διαδικασία επαλήθευσης της ταυτότητας ποικίλει και συνήθως εξαρτάται από τις δυνατότητες της διεπαφής των συσκευών. Κάποιες φορές είναι αρκετά απλό, όπου με το πάτημα ενός κουμπιού γίνεται η αντιστοίχηση π.χ στα ακουστικά, σε άλλες πάλι περιπτώσεις απαιτείται η πληκτρολόγηση ενός κωδικού PIN ο οποίος κυμαίνεται σε μήκος και πολυπλοκότητα από τέσσερις αριθμούς έως μία αλφαριθμητική σειρά δεκαέξι χαρακτήρων.

2.4.3 Κλάσεις Εμβέλειας Ισχύος

Η εμβέλεια ισχύος του σήματος μίας συσκευής Bluetooth χωρίζεται σε τρεις κλάσεις (πίνακας 2-1).

Πίνακας 2-1: Εμβέλεια ισχύος ανά κλάση

Αριθμός κλάσης	Μέγιστη ισχύς εξόδου(dBm)	Μέγιστη ισχύς εξόδου (mW)	Μέγιστη εμβέλεια
Κλάση 1	20 dBm	100 mW	100 m
Κλάση 2	4 dBm	2.5 mW	10 m
Κλάση 3	0 dBm	1 mW	10 cm

2.5 Οι Διάφορες Εκδόσεις του Bluetooth

Το Bluetooth εξελίσσεται συνεχώς από τότε που σχεδιάστηκε το 1994. Η πιο τελευταία έκδοση, το Bluetooth v4.0, μόλις αρχίζει να κερδίζει έδαφος στην βιομηχανία των ηλεκτρονικών ειδών ευρεία κατανάλωσης, ωστόσο μερικές από τις προηγούμενες εκδόσεις εξακολουθούν να χρησιμοποιούνται ευρέως. Στον παρακάτω πίνακα θα δούμε τα χαρακτηριστικά που υποστηρίζονται από κάθε έκδοση (Πίνακας 2-2).

Πίνακας 2-2: Οι εκδόσεις του Bluetooth

Χαρακτηριστικά	Bluetooth 1.1	Bluetooth 1.2	Bluetooth 2.0	Bluetooth 2.1+ EDR	Bluetooth 3.0	Bluetooth 4.0
Φωνητική κλήση	Ναι	Ναι	Ναι	Ναι	Ναι	Ναι
Σίγαση κλήσης	Ναι	Ναι	Ναι	Ναι	Ναι	Ναι
Επανάκληση τελευταίου αριθμού	Ναι	Ναι	Ναι	Ναι	Ναι	Ναι

Υψηλές ταχύτητες μετάδοσης	Όχι	Όχι	Ναι	Ναι	Ναι	Ναι
Χαμηλή κατανάλωση ενέργειας	Όχι	Όχι	Ναι	Ναι	Ναι	Ναι
Βελτιωμένο Pairing	Όχι	Όχι	Όχι	Ναι	Ναι	Ναι
Καλύτερη ασφάλεια	Όχι	Ναι	Ναι	Ναι	Ναι	Ναι
Bluetooth Low Energy	Όχι	Όχι	Όχι	Όχι	Όχι	Ναι
Υποστήριξη NFC	Όχι	Όχι	Ναι	Ναι	Ναι	Ναι

2.6 Σύγκριση του Bluetooth με άλλα Πρωτόκολλα

Το Bluetooth σαφέστατα δεν είναι το μόνο ασύρματο πρωτόκολλο που υπάρχει. Στον παρακάτω πίνακα θα δούμε κάποια συγκριτικά χαρακτηριστικά σε σχέση με άλλα ασύρματα πρωτόκολλα (Πίνακας 2-3).

Πίνακας 2-3: Το Bluetooth και άλλα πρωτόκολλα

Όνομα	**Bluetooth Classic**	**Bluetooth 4.0 Low Energy (BLE)**	**ZigBee**	**WiFi**
Πρότυπα IEEE	802.15.1	802.15.1	802.15.4	802.11 (a, b, g, n)
Συχνότητα (GHz)	2.4	2.4	0.868, 0.915, 2.4	2.4 and 5
Μέγιστος ρυθμός μετάδοσης (Mbps)	1-3	1	0.250	11 (b), 54 (g), 600 (n)
Throughput (Mbps)	0.7-2.1	0.27	0.2	7 (b), 25 (g), 150 (n)
Μέγιστη (σε εξωτερικούς χώρους) εμβέλεια (Meters)	10 (class 2), 100 (class 1)	50	10-100	100-250
Κατανάλωση ενέργειας	Μεσαία	Πολύ χαμηλή	Πολύ χαμηλή	Υψηλή
Διάρκεια μπαταρίας	Μέρες	Μήνες έως Χρόνια	Μήνες έως Χρόνια	Ώρες
Μέγεθος δικτύου	7	Απροσδιόριστο	64,000+	255

3 Εισαγωγή στους Μικροελεγκτές

Στο κεφάλαιο αυτό θα αναλύσουμε την έννοια του ενσωματωμένου συστήματος. Θα εξηγήσουμε την διαφορά μεταξύ ενός μικροεπεξεργαστή μη ενσωματωμένου συστήματος και ενός μικροελεγκτή, και στην συνέχεια θα δούμε βασικές κατηγορίες μικροελεγκτών. Τέλος, θα ασχοληθούμε με το Hardware του μικροελεγκτή AVR, ο οποίος και χρησιμοποιείται σε πλακέτες Arduino.

3.1 Ενσωματωμένο Υπολογιστικό Σύστημα

Ένας απλός ορισμός ενός ενσωματωμένου υπολογιστικού συστήματος, είναι οποιαδήποτε συσκευή περιλαμβάνει ένα προγραμματιζόμενο υπολογιστή, ο οποίος όμως δεν αποτελεί υπολογιστή γενικού σκοπού. Συνεπώς, ο ηλεκτρονικός μας υπολογιστής δεν αποτελεί ενσωματωμένο υπολογιστικό σύστημα. Αντίθετα, το PDA, το MP3 player, τα κινητά τηλέφωνα, οι ψηφιακές κάμερες, οι εκτυπωτές και τα GPS αποτελούν παραδείγματα ενός ενσωματωμένου υπολογιστικού συστήματος.

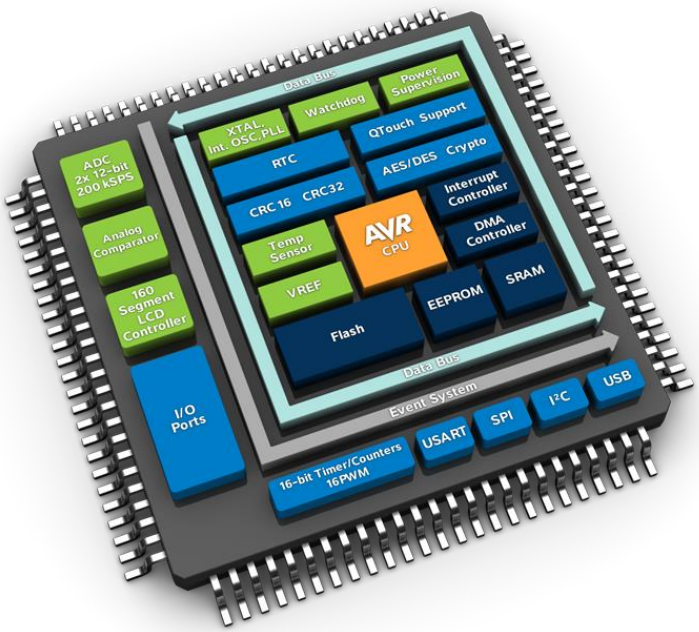
Η σχεδίαση ενσωματωμένων υπολογιστικών συστημάτων, τα οποία βασίζονται συχνά σε μικροελεγκτές, είναι μία σημαντική δεξιότητα για την ανάπτυξη πολλών τύπων προϊόντων, βασικό και αναπόσπαστο κομμάτι των οποίων είναι η χρήση του μικροεπεξεργαστή. Οι σχεδιαστές λοιπόν πρέπει να έχουν την ικανότητα να αναγνωρίζουν που μπορούν να χρησιμοποιηθούν οι μικροεπεξεργαστές, να σχεδιάζουν την πλατφόρμα του υλικού με τις απαραίτητες συσκευές εισόδου και εξόδου, οι οποίες μπορούν να υποστηρίξουν τις απαιτούμενες εργασίες και να υλοποιούν λογισμικό το οποίο εκτελεί την απαιτούμενη επεξεργασία.

Το βασικό χαρακτηριστικό των ενσωματωμένων υπολογιστικών συστημάτων είναι η αφοσίωση τους στην διεκπεραίωση μίας συγκεκριμένης εργασίας. Εφόσον λοιπόν απευθύνονται σε συγκεκριμένες εργασίες, οι σχεδιαστές μπορούν να τα βελτιώσουν ώστε να μειωθεί το κόστος και το μέγεθος του προϊόντος, αυξάνοντας ταυτόχρονα την αξιοπιστία και την απόδοση τους.

Εικόνα 3-1: Ενσωματωμένο σύστημα από την εταιρία Altera

3.2 Τι είναι ένας Μικροελεγκτής

Οι μικροελεγκτές συχνά ορίζονται ως ένας μίνι πλήρης υπολογιστής σε ένα ενιαίο τσιπ, και αυτό είναι πραγματικά αλήθεια (Εικόνα 3-2). Στον πυρήνα τους, οι μικροελεγκτές έχουν έναν μικροεπεξεργαστή, που είναι παρόμοιος με την CPU του υπολογιστή μας, μία σταθερή ποσότητα μνήμης RAM και ROM και άλλα περιφερειακά όλα ενσωματωμένα σε ένα τσιπ. Σήμερα, αρκετοί κατασκευαστές παράγουν μικροελεγκτές με ένα ευρύ φάσμα δυνατοτήτων και είναι διαθέσιμα σε διάφορες εκδόσεις. Κάποιοι από τους κατασκευαστές είναι η ATMEL, η Microchip, η Freescale, η Philips, η Motorola κ.α.



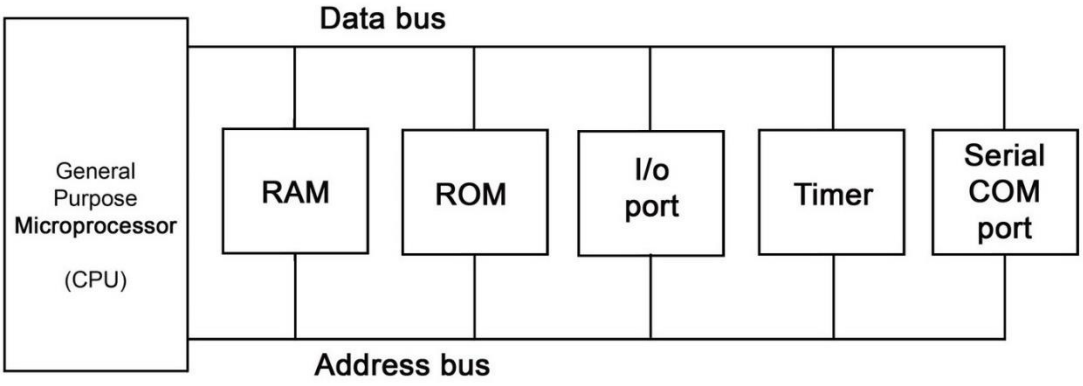
Εικόνα 3-2: Ο μικροελεγκτής Atmel AVR της σειράς XMEGA, όλα είναι οργανωμένα σε ένα και μόνο τσιπ

Οι μικροελεγκτές είναι σχεδιασμένοι για να εκτελούν συγκεκριμένες λειτουργίες. Λέγοντας συγκεκριμένες, εννοούμε εφαρμογές όπου η σχέση μεταξύ της εισόδου και της εξόδου είναι προκαθορισμένη. Αναλόγως την είσοδο, ακολουθεί μία συγκεκριμένη επεξεργασία και έτσι παράγεται η έξοδος μας. Εφόσον λοιπόν απευθύνεται σε συγκεκριμένες εφαρμογές, οι απαιτήσεις σε μνήμη RAM, ROM και περιφερειακές εισόδους εξόδους είναι περιορισμένες, πράγμα που καθιστά εύκολη την ενσωμάτωση αυτών σε ένα μόνο τσιπ. Αυτό με την σειρά του μειώνει τόσο το κόστος όσο και το μέγεθος του μικροελεγκτή.

3.3 Τι είναι ο Μικροεπεξεργαστής

Με τον όρο μικροεπεξεργαστή, εννοούμε ενός γενικού σκοπού μικροεπεξεργαστή όπως η σειρά της Intel Sandy Bridge που περιλαμβάνει τους επεξεργαστές Celeron, Pentium, Core i3, Core i5, Core i7. Αυτοί οι μικροεπεξεργαστές δεν περιέχουν ενσωματωμένη μνήμη RAM ή ROM ούτε εισόδους ή εξόδους για σύνδεση περιφερειακών συσκευών. Η προσθήκη αυτών πρέπει να γίνει με εξωτερική παρέμβαση από τον χρήστη.

Οι μικροεπεξεργαστές βρίσκουν χρήση σε γενικού σκοπού εφαρμογές όπως στην ανάπτυξη λογισμικού , παιχνιδιών, ιστοσελίδων στην επεξεργασία εικόνας και ήχου στην δημιουργία εγγράφων κ.α. Έχουν υψηλές απαιτήσεις σε μνήμη RAM ,ROM και περιφερειακές εισόδους και εξόδους .



Εικόνα 3-3: Γενικό διάγραμμα ενός συστήματος με μικροεπεξεργαστή

3.4 Διαφορές του Μικροελεγκτή από τον Μικροεπεξεργαστή

Στους σύγχρονους μικροεπεξεργαστές για μη ενσωματωμένα συστήματα (πχ τους μικροεπεξεργαστές των προσωπικών υπολογιστών), δίνεται έμφαση στην υπολογιστική ισχύ. Η ευελιξία ανάπτυξης διαφορετικών εφαρμογών είναι μεγάλη, καθώς η λειτουργικότητα του τελικού συστήματος καθορίζεται από τα εξωτερικά περιφερειακά τα οποία διασυνδέονται με την κεντρική μονάδα (μικροεπεξεργαστή), η οποία δεν είναι εξειδικευμένη. Αντίθετα, στους μικροεπεξεργαστές για ενσωματωμένα συστήματα (μικροελεγκτές), οι οποίοι έχουν μικρότερες ή και μηδαμινές δυνατότητες συνεργασίας με εξωτερικά περιφερειακά, αυτού του είδους, η ευελιξία είναι περιορισμένη, καθώς και η υπολογιστική ισχύς. Οι μικροελεγκτές δίνουν έμφαση στο μικρό αριθμό ολοκληρωμένων κυκλωμάτων που απαιτείται για τη λειτουργία μιας συσκευής, το χαμηλό κόστος και την εξειδίκευση.

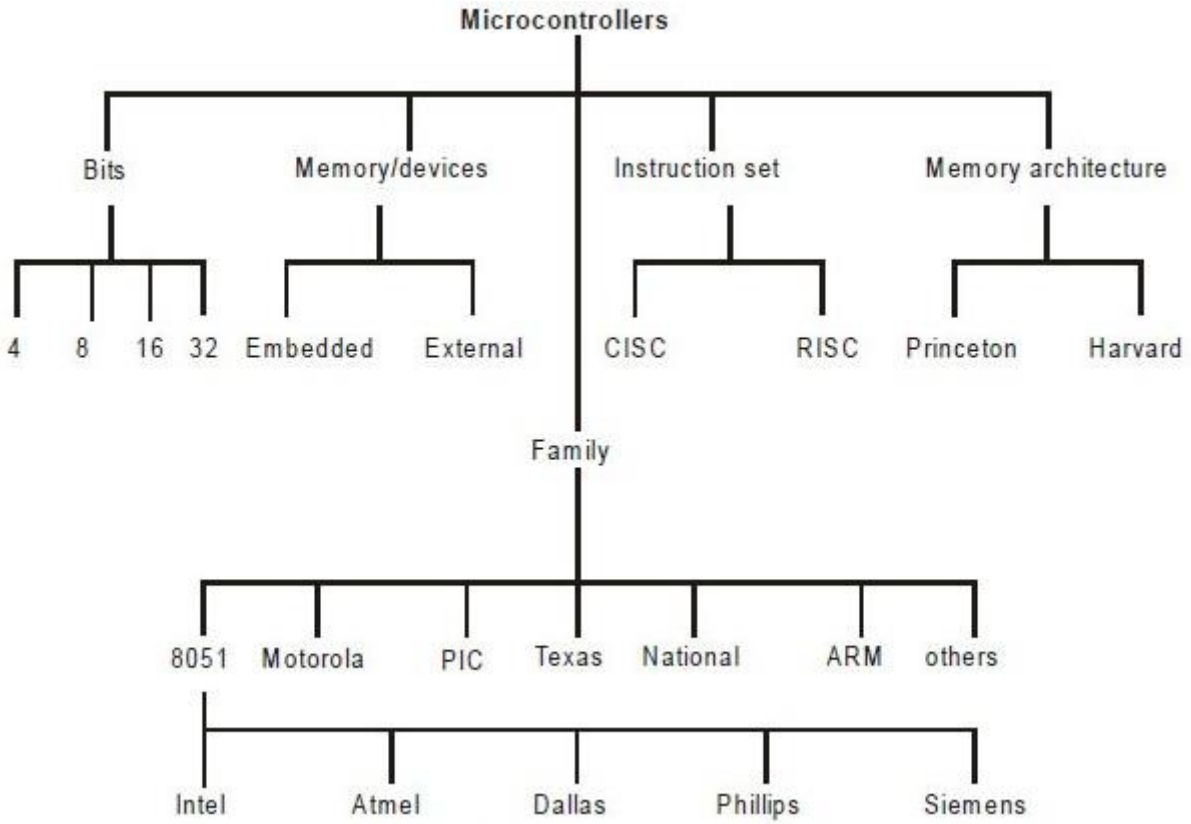
Αναλυτικά, τα πλεονεκτήματα των μικροελεγκτών είναι:

- Αυτονομία, μέσω της ενσωμάτωσης σύνθετων περιφερειακών υποσυστημάτων όπως μνήμες και θύρες επικοινωνίας. Έτσι πολλοί μικροελεγκτές δεν χρειάζονται κανένα άλλο ολοκληρωμένο κύκλωμα για να λειτουργήσουν.
- Η ενσωμάτωση περιφερειακών σημαίνει ευκολότερη υλοποίηση εφαρμογών λόγω των απλούστερων διασυνδέσεων. Επίσης, οδηγεί σε χαμηλότερη κατανάλωση ισχύος, μεγιστοποιώντας τη φορητότητα και ελαχιστοποιεί το κόστος της συσκευής στην οποία ενσωματώνεται ο μικροελεγκτής.
- Χαμηλό κόστος.
- Μεγαλύτερη αξιοπιστία, και πάλι λόγω των λιγότερων διασυνδέσεων.
- Μειωμένες εκπομπές ηλεκτρομαγνητικών παρεμβολών και μειωμένη ευαισθησία σε αντίστοιχες παρεμβολές από άλλες ηλεκτρικές και ηλεκτρονικές συσκευές. Το πλεονέκτημα αυτό προκύπτει από το μικρότερο αριθμό και μήκος εξωτερικών διασυνδέσεων καθώς και τις χαμηλότερες ταχύτητες λειτουργίας.
- Περισσότεροι διαθέσιμοι ακροδέκτες για ψηφιακές εισόδους-εξόδους (για δεδομένο μέγεθος ολοκληρωμένου κυκλώματος), λόγω της μη δέσμευσής τους για τη σύνδεση εξωτερικών περιφερειακών.
- Μικρό μέγεθος συνολικού υπολογιστικού συστήματος.

Η βασική αρχιτεκτονική των μικροελεγκτών δεν διαφέρει από αυτή των κοινών μικροεπεξεργαστών, αν και στους πρώτους απαντάται συχνά η αρχιτεκτονική μνήμης τύπου Harvard, η οποία χρησιμοποιεί διαφορετικές αρτηρίες σύνδεσης της μνήμης προγράμματος και της μνήμης δεδομένων (πχ οι σειρές AVR από την Atmel και PIC από την Microchip). Στους κοινούς μικροεπεξεργαστές συνηθίζεται η ενιαία διάταξη μνήμης τύπου Von Neumann (φον Νόιμαν).

3.5 Κατηγορίες Μικροελεγκτών

Οι μικροελεγκτές ταξινομούνται αναλόγως, το εύρος διαύλου, το σετ εντολών, την αρχιτεκτονική μνήμης ,την γλώσσα προγραμματισμού που χρησιμοποιείται κ.α. (Εικόνα 3-4.)



Εικόνα 3-4: Διάφοροι τύποι μικροελεγκτών

3.5.1 Μικροελεγκτές 8, 16 και 32-bit

Όσες περισσότερες γραμμές διευθύνσεων έχει ένας δίαυλος, τόσο περισσότερη μνήμη μπορεί να προσπελάσει άμεσα ο μικροεπεξεργαστής. Αν ο δίαυλος έχει n γραμμές διευθύνσεων, τότε ο μικροεπεξεργαστής μπορεί να τον χρησιμοποιήσει για να απευθύνεται σε 2^n διαφορετικές διευθύνσεις μνήμης. Για να μπορούν να χρησιμοποιηθούν μεγάλες μνήμες, οι δίαυλοι χρειάζεται να έχουν πολλές γραμμές διευθύνσεων. Σύμφωνα με το εύρος του διαύλου διακρίνονται σε μικροελεγκτές των 8, 16 και 32-bit αντίστοιχα.

Μικροελεγκτής 8-bit: Όταν το εύρος του εσωτερικού διαύλου σε ένα μικροελεγκτή είναι 8-bit και η μονάδα αριθμητικής λογικής (ALU) εκτελεί τις αριθμητικές και λογικές πράξεις σε ένα Byte (8 bit), ο μικροελεγκτής ονομάζεται μικροελεγκτής 8-bit. Παραδείγματα 8bit μικροελεγκτών είναι το ATmega328, ο Intel 8031/8051 και η σειρά MC68HC11 της Motorolla. .

Μικροελεγκτής 16-bit: Έχει μεγαλύτερη ακρίβεια και καλύτερη απόδοση σε σύγκριση με τον μικροελεγκτή 8-bit. Όταν το εύρος του διαύλου είναι 16-bit και η μονάδα αριθμητικής λογικής εκτελεί αριθμητικές και λογικές πράξεις μίας λέξης (16-bit) σε μία εντολή, τότε ο μικροελεγκτής ονομάζεται μικροελεγκτής 16-bit. Για παράδειγμα ένας μικροελεγκτής 8-bit μπορεί να προσπελάσει 2^8 θέσεις μνήμης, καταλήγοντας σε συνολικό εύρος 0×00 – 0xFF (0-255) για κάθε κύκλο. Αντίθετα, ένας μικροελεγκτής με μέγεθος 16-bit έχει εύρος 2^{16}, δηλαδή 0×0000 – 0xFFFF (0-65535) ανά κύκλο. Παραδείγματα μικροελεγκτών 16-bit είναι το 8051XA, PIC2X, ο Intel 8096, η σειρά MC68HC12 της Motorolla.

Μικροελεγκτής 32-bit: Χρησιμοποιεί εντολές των 32-bit για να εκτελέσει αριθμητικές και λογικές πράξεις. Παρέχουν πολύ μεγάλη ακρίβεια και απίστευτη απόδοση σε σχέση με τους μικροελεγκτές 16-bit. Χρησιμοποιούνται σε συσκευές αυτόματου ελέγχου, όπως σε εφαρμογές συστημάτων ελέγχου ενός κινητήρα, σε μηχανές γραφείου κ.ά. Επίσης, βρίσκουν εφαρμογή σε ενσωματωμένα συστήματα υπολογιστών, όπως σε κινητά τηλέφωνά, ηχοσυστήματα MP3, αεροδιαστημικά συστήματα κ.α. Κάποια παραδείγματα μικροελεγκτών 32-bit, είναι η σειρά 32-bit AVR UC3 της Atmel, το PIC3x, το Motorola M683xx και η οικογένεια ARM 7, 9 και 11.

3.5.2 Μικροελεγκτές Εσωτερικής και Εξωτερικής Μνήμης

Όταν ένα ενσωματωμένο σύστημα έχει μία μονάδα μικροελεγκτή που έχει όλα τα λειτουργικά τμήματα (συμπεριλαμβανομένου της μνήμης για την αποθήκευση των δεδομένων αλλά και του προγράμματος) διαθέσιμα σε ένα τσιπ, ονομάζεται ενσωματωμένος μικροελεγκτής. Για παράδειγμα, ο 8051 που έχει την μνήμη των δεδομένων και του προγράμματος, εισόδους και εξόδους, σειριακή επικοινωνία, μετρητές και timers και την ICL (Interrupt Control Logic) όλα σε ένα τσιπ αποτελεί παράδειγμα ενσωματωμένου μικροελεγκτή

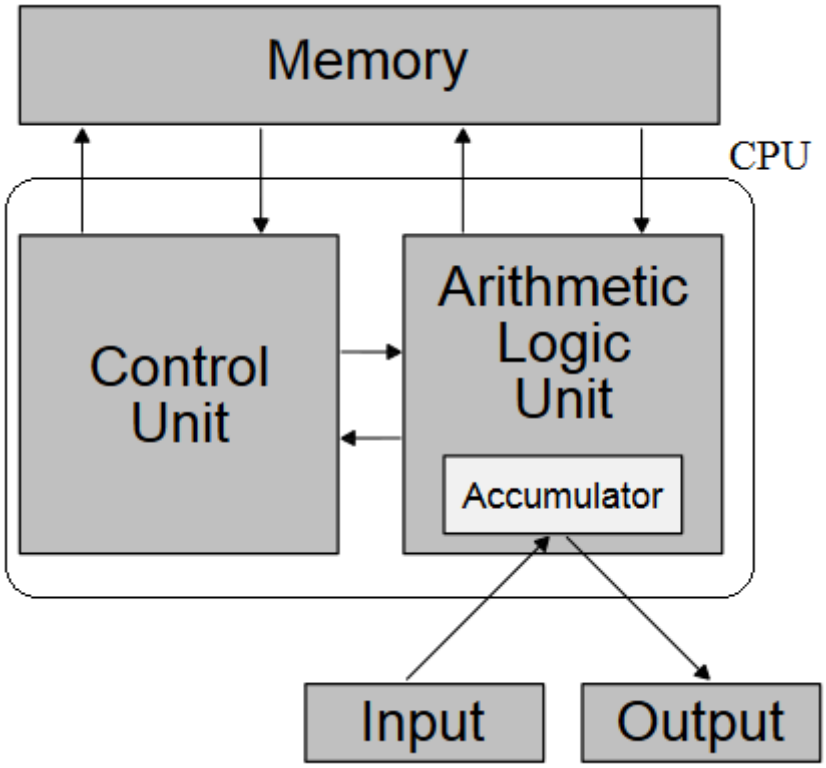
Όταν ένα ενσωματωμένο σύστημα έχει μία μονάδα μικροελεγκτή που δεν έχει όλα τα διαθέσιμα λειτουργικά τμήματα σε ένα chip , ονομάζεται μικροελεγκτής με εξωτερική μνήμη. Στους μικροελεγκτές με εξωτερική μνήμη, το σύνολο ή μέρος των μονάδων μνήμης είναι διασυνδεδεμένα εξωτερικά χρησιμοποιώντας το λεγόμενο «glue Circuit». Για παράδειγμα ο 8031 που δεν έχει ενσωματωμένη μνήμη για την αποθήκευση του προγράμματος, είναι μικροελεγκτής εξωτερικής μνήμης.

3.5.3 Τα Χαρακτηριστικά της Αρχιτεκτονικής ενός Μικροελεγκτή

Υπάρχουν κυρίως δύο κατηγορίες επεξεργαστών, με αρχιτεκτονική Von-Newman (ή Princeton) και αρχιτεκτονική Harvard. Αυτές οι δύο αρχιτεκτονικές διαφέρουν στον τρόπο που γίνεται η αποθήκευση και η προσπέλαση των δεδομένων και του προγράμματος.

3.5.3.1 Η Αρχιτεκτονική Von-Neumann

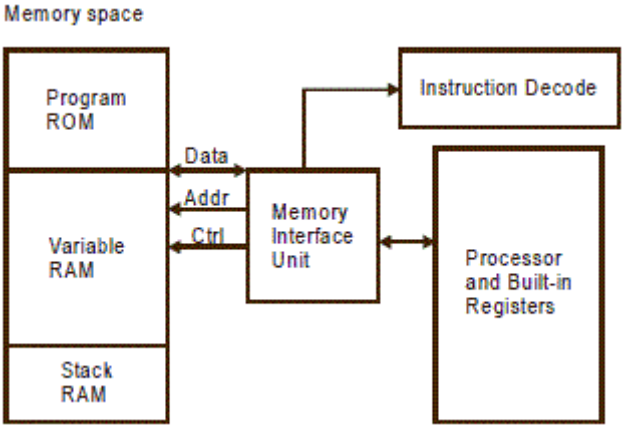
Χρησιμοποιήθηκε στον EDSAC, τον πρώτο υπολογιστή με αποθηκευόμενο πρόγραμμα, και εξακολουθεί να είναι η βάση όλων σχεδόν των ψηφιακών υπολογιστών, ακόμα και σήμερα. Η μηχανή του Von Neumann είχε πέντε βασικά τμήματα: Την μνήμη, την αριθμητική λογική μονάδα, την μονάδα ελέγχου, και τον εξοπλισμό εισόδου και εξόδου. Η μνήμη αποτελούνταν από 4096 λέξεις, και η κάθε λέξη είχε 40-bit με τιμή 0 ή 1. Η κάθε λέξη περιείχε είτε δύο εντολές των 20-bit είτε ένα προσημασμένο ακέραιο των 40-bit. Οι εντολές χρησιμοποιούσαν 8-bit για τον προσδιορισμό του τύπου της εντολής και 12-bit για τον προσδιορισμό μίας από τις 4096 λέξεις της μνήμης.



Εικόνα 3-5: Η Αρχική μηχανή του Von Neumann

Στις αρχιτεκτονικές Von Neumann η μνήμη έχει έναν ενιαίο χώρο φυσικών διευθύνσεων. Δηλαδή τα δεδομένα και οι εντολές των εκτελούμενων προγραμμάτων αποθηκεύονται σε μία και μοναδική μνήμη. Η σειριακή αυτή προσπέλαση των δεδομένων και των εντολών δημιουργεί συμφόρηση και καθιστά τον ελεγκτή πιο αργό. Για παράδειγμα, ένας μικροελεγκτής με δίαυλο διευθύνσεων 16 bit μπορεί να «δει» μέχρι 65536 θέσεις μνήμης με ίδιο μήκος

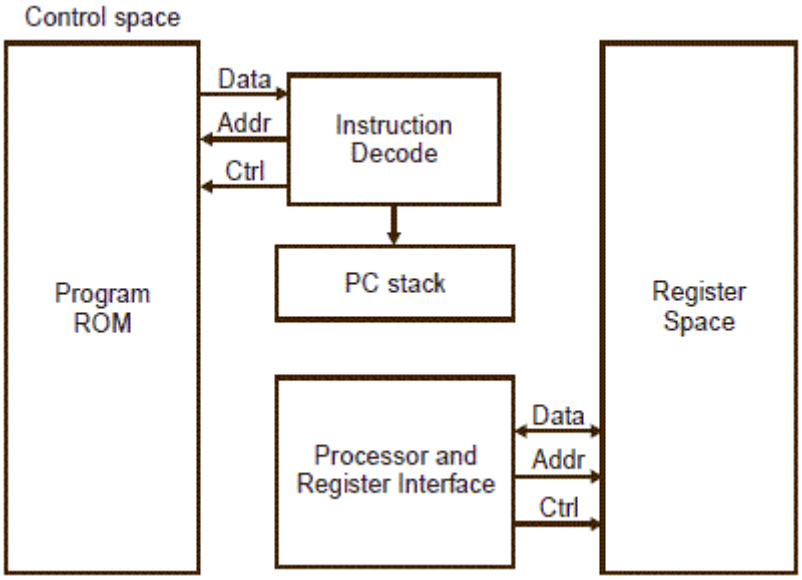
λέξης. Κάποια τμήματα αυτού του χώρου διευθύνσεων μπορούν να χρησιμοποιηθούν από μνήμη RAM, ROM ή και περιφερειακά. Στην εικόνα 3-6 παρατηρούμε τον μοναδικό δίαυλο που χρησιμοποιείται για την λήψη δεδομένων και εντολών.



Εικόνα 3-6: Διάγραμμα της αρχιτεκτονικής Von Neumann

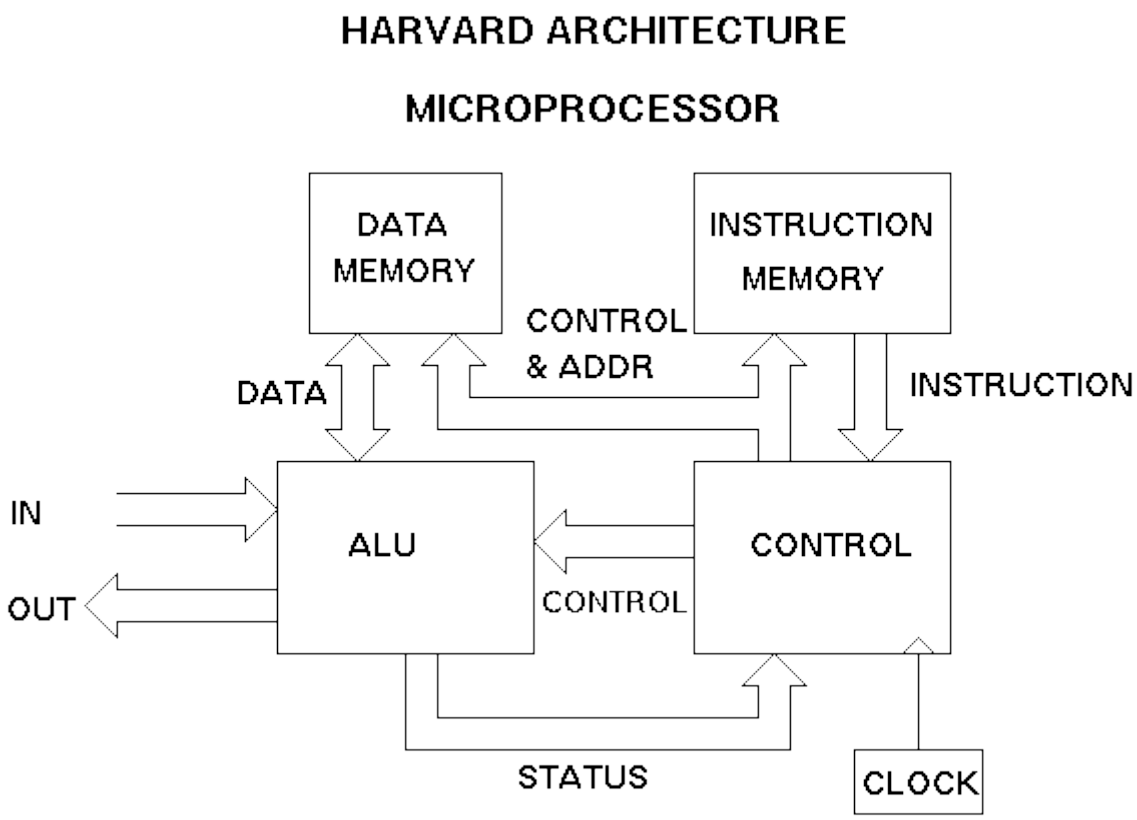
3.5.3.2 Η Αρχιτεκτονική Harvard

Η σημερινή τάση είναι υπερ της λεγόμενης διαμερισμένης κρυφής μνήμης (split cache), όπου οι εντολές βρίσκονται σε μία κρυφή μνήμη και τα δεδομένα σε άλλη. Ο σχεδιασμός αυτός λέγεται και αρχιτεκτονική Harvard. Αυτή η ονομασία παραπέμπει πίσω στον υπολογιστή Mark III του Howard Aiken, που είχε διαφορετικές μνήμες για τις εντολές και για τα δεδομένα. Στην εικόνα 3-7, παρατηρούμε τον ξεχωριστό δίαυλο για τα δεδομένα και για τις εντολές, πράγμα που επιτρέπει την παράλληλη προσπέλαση.



Εικόνα 3-7: Διάγραμμα της αρχιτεκτονικής Harvard

Το κίνητρο που ωθεί τους σχεδιαστές σε αυτή την κατεύθυνση είναι η διαδεδομένη χρήση των μικροεπεξεργαστών με διοχέτευση (pipelined CPU). Η μονάδα προσκόμισης εντολών χρειάζεται να προσπελάζει τις εντολές την ίδια ώρα που η μονάδα προσκόμισης τελεστέων χρειάζεται να προσπελάζει τα δεδομένα. Μία διαμερισμένη κρυφή μνήμη, επιτρέπει τις παράλληλες προσπελάσεις, ενώ μία ενοποιημένη δεν τις επιτρέπει. Επίσης αφού οι εντολές κανονικά δεν τροποποιούνται κατά την εκτέλεση, το περιεχόμενο της κρυφής μνήμης εντολών δεν χρειάζεται ποτέ να ξανά γράφεται πίσω στην κύρια μνήμη. Τέλος, να αναφέρουμε πως το ολοκληρωμένο Atmega328 που χρησιμοποιείται στην πλατφόρμα του Arduino βασίζεται στην αρχιτεκτονική Harvard.



Εικόνα 3-8: Βοηθητικό διάγραμμα της αρχιτεκτονικής Harvard

3.5.3.3 Σειρά οδηγιών CISC και RISC

Η καρδιά ενός AVR μικροελεγκτή είναι ένας επεξεργαστής που ανήκει στην κατηγορία RISC (Reduced Instruction Set Computer). Επεξεργαστές όπως ο 8086 ή ο 6800 καθώς εξελίσσονταν σε πιο σύγχρονες εκδόσεις όπως 80286 ή 68000 αντίστοιχα, στήριζαν την εξέλιξή τους, εν πολλοίς στον εμπλουτισμό του συνόλου εντολών τους με περισσότερες, πιο σύνθετες αλλά και ανομοιόμορφες εντολές. Το αποτέλεσμα αυτής της εξέλιξης ήταν οι χρήστες να μην είναι σε θέση να εκμεταλλευτούν όλες τις δυνατότητες που προσφέρονταν αφού περιορίζονταν στη χρήση μερικών μόνο εύχρηστων και ευκολομνημόνευτων εντολών. Επεξεργαστές της κατηγορίας αυτής ονομάζονται CISC (Complex Instruction Set Computers).

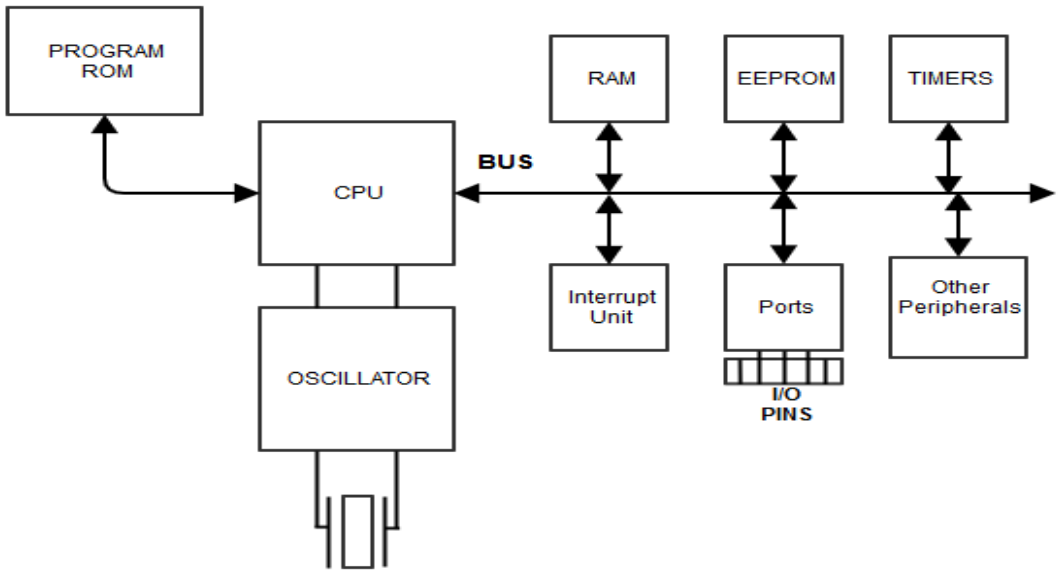
Σε αντίθεση με τους επεξεργαστές της κατηγορίας CISC, οι επεξεργαστές RISC διαθέτουν ένα μικρό και εύχρηστο σύνολο ομοιόμορφων εντολών που όλες μπορούν να εφαρμοστούν σε ένα μεγάλο πλήθος καταχωρητών και όχι για παράδειγμα μόνο στον Accumulator (συσσωρευτή) ή σε ένα ή δύο ειδικούς καταχωρητές. Ο χρόνος εκτέλεσης κάθε εντολής είναι συνήθως ένας κύκλος ρολογιού. Επίσης, το μήκος όλων των RISC εντολών είναι σταθερό πχ, 2 bytes. Έτσι αν για παράδειγμα σε ένα CISC επεξεργαστή μια λειτουργία μπορεί να εκτελεστεί από μία πολύπλοκη εντολή τεσσάρων bytes σε 4 κύκλους ρολογιού, η ίδια λειτουργία σε RISC επεξεργαστή μπορεί να εκτελεστεί από δύο απλούστερες εντολές των δύο bytes που κάθε μία εκτελείται σε έναν ή δύο κύκλους ρολογιού. Υποστηρίζεται ότι τα RISC προγράμματα είναι συνήθως μικρότερα σε μέγεθος και γρηγορότερα εξαιτίας του ότι απαιτούν λιγότερες χρονοβόρες εντολές μετακίνησης και κάθε εντολή εκτελείται σε έναν συνήθως κύκλο ρολογιού. Οι λιγότερες μετακινήσεις οφείλονται στο ότι ο αριθμός των καταχωρητών στους οποίους αποθηκεύονται τα ενδιάμεσα αποτελέσματα πράξεων είναι πολύ μεγαλύτερος από εκείνους των CISC επεξεργαστών.

3.6 Οι Μικροελεγκτές AVR

Η βασική αρχιτεκτονική των AVR επινοήθηκε από δύο μαθητές στο Νορβηγικό Ινστιτούτο Τεχνολογίας τους Alf-Bogen EGIL και Vegard Wollan. Αργότερα η πατέντα για τους AVR μικροελεγκτές αγοράστηκε από την εταιρία ATMEL και η εσωτερική αρχιτεκτονική τους αναπτύχθηκε για πρώτη φορά το 1996. Χρησιμοποιούν την τροποποιημένη αρχιτεκτονική Harvard 8-bit RISC και κάνουν χρήση της ενσωματωμένης μνήμης FLASH για την αποθήκευση του προγράμματος σε αντίθεση με τις OTP ROM (One Time Programmable Read Only Memory), EPROM (Erasable

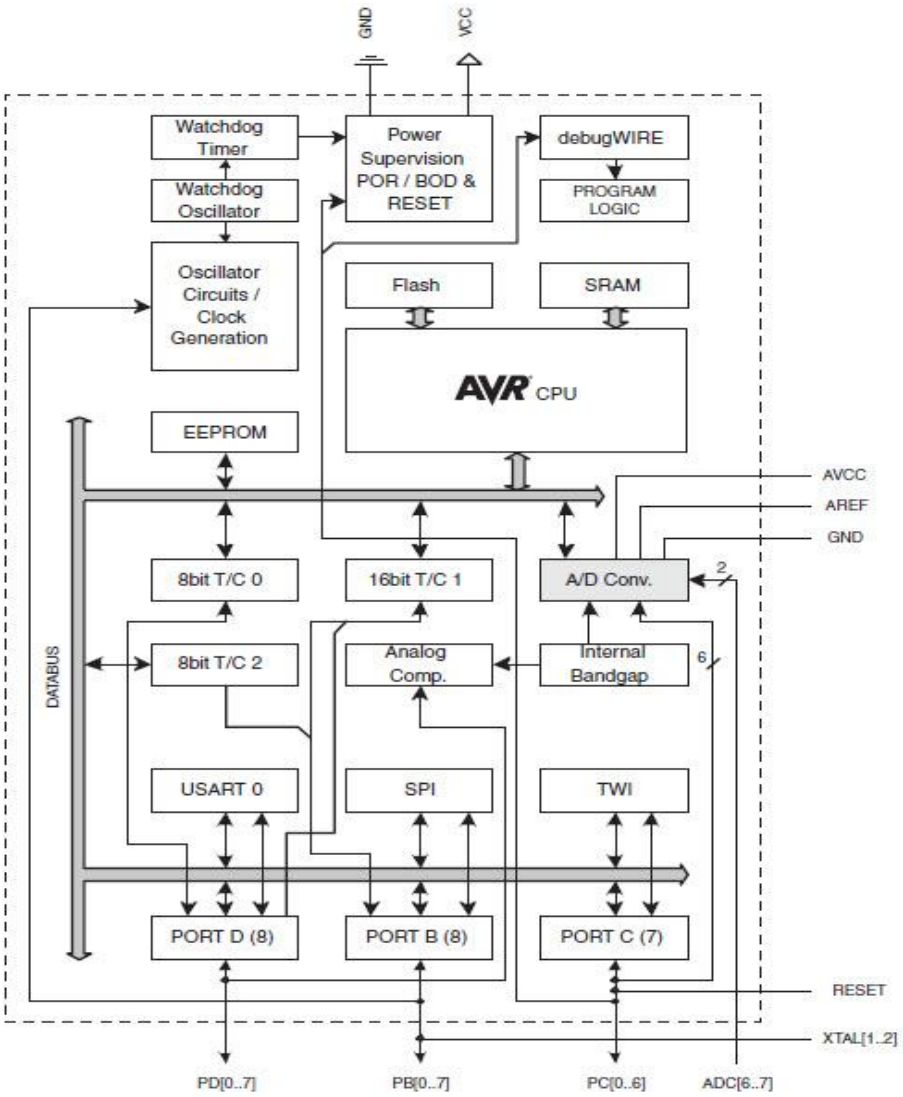
Programmable Read Only Memory) και EEPROM (γράφεται και E^2PROM, Electrically Erasable Programmable Read-Only Memory) που χρησιμοποιούνται από άλλους μικροελεγκτές. Η θυγατρική της ATMEL στην Νορβηγία ιδρύθηκε από τους δύο φοιτητές. Το όνομα AVR, σύμφωνα με την ATMEL, δεν αποτελεί κάτι ιδιαίτερο όσον αφορά την ερμηνεία του, ωστόσο είναι πιθανό να προέρχεται από το Advanced Virtual RISC ή από τα ονόματα των σχεδιαστών, δηλαδή Alf and Vegard RISC. Το όνομα AVR ορίζει όλη την οικογένεια των μικροελεγκτών τύπου 8-bit και 32-bit RISC.

Υπάρχουν διάφορα είδη AVR μικροελεγκτών με διαφορετικές ιδιότητες ο καθένας. Εκτός από τον AVR32, ο οποίος είναι ένας 32-bit μικροεπεξεργαστής, όλοι οι άλλοι είναι 8-bit, πράγμα που σημαίνει ότι η CPU μπορεί να δουλέψει μόνο 8 bits δεδομένων την φορά. Δεδομένα μεγαλύτερα από 8-bit θα σπάσουν σε κομμάτια των 8-bit για να υποβληθούν σε επεξεργασία από την CPU του μικροελεγκτή. Ένα από τα προβλήματα με τους μικροελεγκτές AVR, από την άποψη του λογισμικού, είναι ότι δεν υπάρχει εκατό τις εκατό συμβατότητα μεταξύ τους. Για να τρέξουμε ένα πρόγραμμα γραμμένο για το ATtiny25 στον Atmega64, θα πρέπει να μεταγλωττίσουμε το πρόγραμμα ξανά και, ενδεχομένως, να αλλάξουμε θέση σε κάποιους registers πριν το φορτώσουμε στο Atmega64. Οι AVRs ταξινομούνται σε τέσσερις μεγάλες ομάδες οι οποίες είναι : tinyAVR, megaAVR, XMEGA, Application Specific AVR, FPSLIC, 32-bit AVRs.



Εικόνα 3-9: Απλό διάγραμμα ενός AVR μικροελεγκτή

Ένα πράγμα που πρέπει να παρατηρήσουμε είναι ότι τα τσιπ της σειράς AVR, από το ATtiny15 μέχρι το Atmega328 περιλαμβάνουν το μέγεθος της μνήμης Flash στο όνομα τους. Δηλαδή ο διαθέσιμος χώρος που έχουμε για το πρόγραμμα μας μπορεί να είναι από 1KB έως 32KB. Για παράδειγμα το Arduino Uno R3 διαθέτει τον μικροελεγκτή Atmega 328, ο οποίος έχει 32 KB (0.5 KB χρησιμοποιούνται από τον Bootloader) μνήμη Flash, το Arduino Mega 2560 διαθέτει τον μικροελεγκτή Atmega2560, ο οποίος έχει 256 KB μνήμη flash (8 KB χρησιμοποιούνται από τον Bootloader). Ο περιορισμός αυτός ίσως μας δυσκολέψει σε ιδιαίτερα πολύπλοκες εφαρμογές με πολλές γραμμές κώδικα.

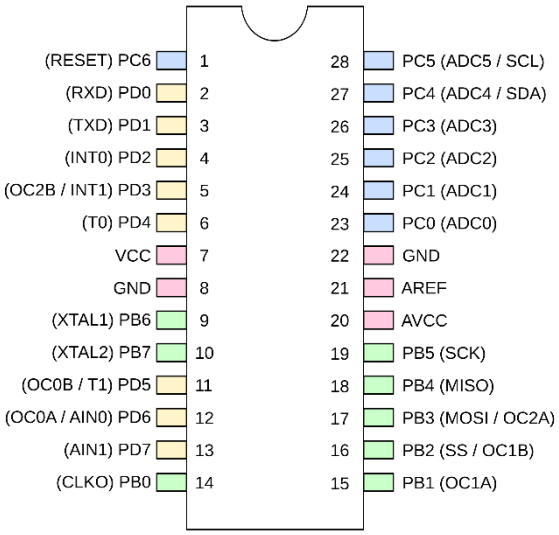


Εικόνα 3-10: Διάγραμμα της αρχιτεκτονικής του Atmega328

3.6.1 Το Hardware ενός AVR μικροελεγκτή

Κάθε pin στον μικροελεγκτή AVR έχει ένα όνομα. Αν για παράδειγμα συνδέσουμε ένα Led στο pin 14, μπορούμε μετά να κάνουμε το pin αυτό HIGH ή LOW, να περάσουμε δηλαδή μία τάση, μιλώντας σε αυτό ως PB0. Τα περισσότερα pin σε ένα AVR έχουν και δευτερεύουσες λειτουργίες, αυτές παρατίθενται ως μνημονικά μέσα σε παρένθεση. Για παράδειγμα, τα RXD και TXD που χρησιμοποιούνται στην σειριακή επικοινωνία, βρίσκονται στα pins PD0 και PD1 αντίστοιχα.

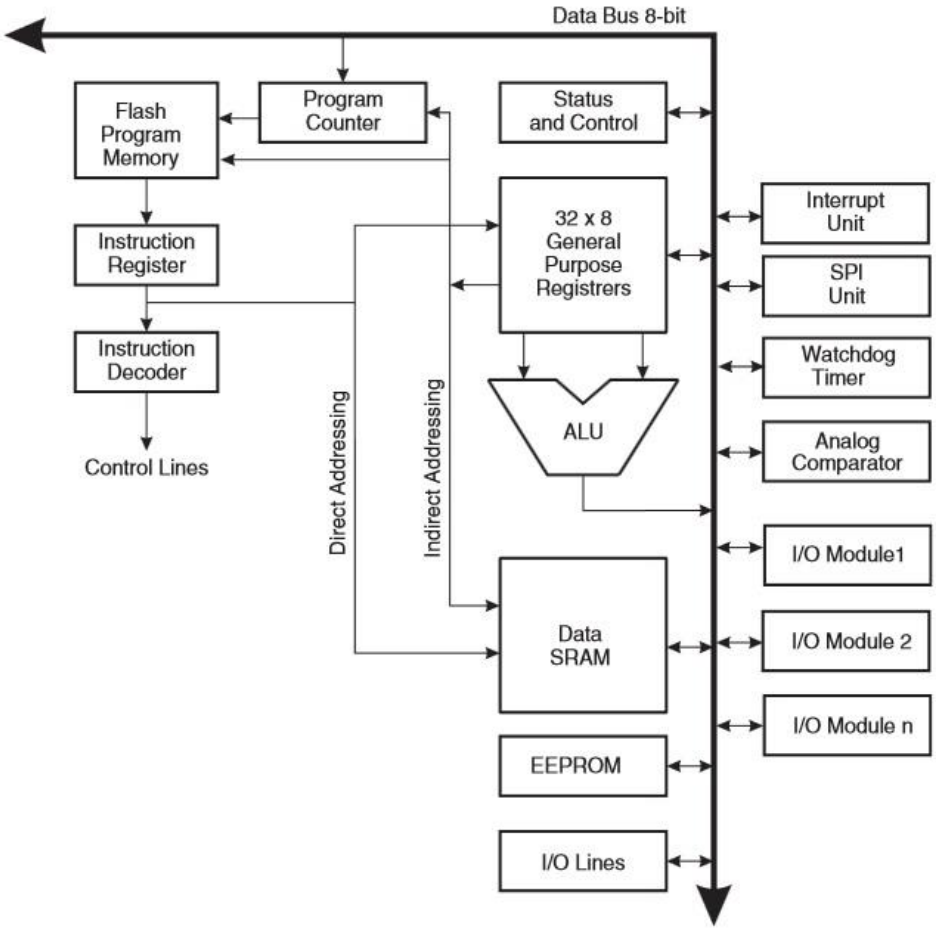
Εσωτερικά, ας πούμε αναλόγως την λειτουργία τους, τα pins είναι χωρισμένα σε ομάδες το πολύ των οκτώ pins. Όπως βλέπουμε στην εικόνα 3-11 κάθε ομάδα είναι χρωματισμένη με διαφορετικό χρώμα. Επειδή λοιπόν κάθε ομάδα περιλαμβάνει το πολύ 8 pin, μπορούμε να αναφερόμαστε σε αυτά χρησιμοποιώντας ένα δυαδικό αριθμό των 8 bit, ώστε να ενεργοποιήσουμε ή όχι την τάση τους.



Εικόνα 3-11: Διάγραμμα των Pins ενός AVR, ATmega48A/PA/88A/PA/168A/PA/328/P

3.6.1.2 Ο Πυρήνας του Μικροελεγκτή AVR

CPU: Η κεντρική μονάδα επεξεργασίας (CPU) ενός AVR είναι παρόμοια με εκείνη του ηλεκτρονικού μας υπολογιστή. Είναι ένα ηλεκτρονικό κύκλωμα που έχει ενσωματωμένες προκαθορισμένες μαθηματικές και λογικές λειτουργίες, και ξέρει που να βρει τη λίστα αυτών των λειτουργιών που απαιτούνται από το πρόγραμμα μας, και που να βρει τα δεδομένα που χρειάζεται για να τις εκτελέσει.



Εικόνα 3-12: Διάγραμμα της αρχιτεκτονικής AVR επεξεργαστή

ΜΝΉΜΗ :Οι μικροελεγκτές AVR έχουν τρείς διαφορετικούς τύπους μνήμης, κάθε ένας με διαφορετική χρήση.

- Μνήμη Flash : Οι εντολές του προγράμματος μας αποθηκεύονται σε μία μη πτητική μνήμη flash. Τα δεδομένα της μνήμης δεν σβήνονται σε περίπτωση απώλειας ρεύματος. Στην πραγματικότητα είναι εγγυημένο ότι θα χάσει μόνο 1 bit ανά εκατομμύριο μετά από εκατό χρόνια σε θερμοκρασία δωματίου. Το μέγεθος της μνήμης του προγράμματος αναφέρεται συνήθως στην ονοματολογία της ίδια της συσκευής (για παράδειγμα ο Atmega328 έχει Flash 32 KB).
- Μνήμη RAM : Χρησιμοποιείται για την προσωρινή αποθήκευση μεταβλητών κατά την εκτέλεση του προγράμματος μας.
- Μνήμη EEPROM : Είναι αργή στην εγγραφή. Όπως και η Flash Memory, η EEPROM μπορεί να διατηρήσει το περιεχόμενό της ακόμη και με απουσία τάσης.

CLOCK: Σε πολλά ψηφιακά κυκλώματα η σειρά με την οποία πραγματοποιούνται τα συμβάντα είναι κρίσιμη. Μερικές φορές κάποιο συμβάν πρέπει να προηγείται κάποιου άλλου, ενώ μερικές φορές δύο συμβάντα πρέπει να πραγματοποιούνται ταυτόχρονα. Για να μπορούν οι σχεδιαστές να πετύχουν τις απαιτούμενες χρονικές σχέσεις, πολλά ψηφιακά κυκλώματα χρησιμοποιούν ρολόγια για την επίτευξη του χρονισμού.

Υπάρχουν τέσσερις τύποι ρολογιών που χρησιμοποιούνται στους μικροελεγκτές, κρύσταλλα (crystals), κεραμικά αντηχεία (ceramic resonators), RC (resistor, capacitor) ταλαντωτές και ταλαντωτές πυριτίου (silicon oscillators). Θα χρησιμοποιήσουμε τον RC ταλαντωτή ως βασικό ρολόι. Τρέχει στα 8 Mhz περίπου. Στη συνέχεια το ρολόι της CPU διαιρείται από το βασικό ρολόι, και τρέχει περίπου στο 1Mhz από προεπιλογή, αλλά μερικές φορές όταν χρειαζόμαστε επιπλέον ταχύτητα, το ανεβάζουμε στα 8Mhz . Μετά από το ρολόι της CPU έρχονται όλα τα άλλα περιφερειακά ρολόγια, τα περισσότερα εκ των οποίων έχουν δικό τους prescaler σε σχέση με την CPU. Υπάρχουν ρολόγια για το υποσύστημα εισόδου εξόδου, για τον ADC (μετατροπέα από αναλογικό σε ψηφιακό), RAM, FLASH και EEPROM.

OUTPUTS: Σχεδόν όλα τα pins σε ένα AVR chip μπορούν χρησιμοποιηθούν ως ψηφιακές έξοδοι, που σημαίνει ότι το pin μπορεί να δεχθεί εντολή, μέσω του προγράμματος, ώστε να εξάγει είτε το επίπεδο της τάσης τροφοδοσία είτε το επίπεδο της τάσης γείωσης.

INPUTS: Ακριβώς όπως όλα τα pin μπορούν να οριστούν ως έξοδοι, μπορούν επίσης να οριστούν και ως ψηφιακοί είσοδοι, όπου ελέγχουν αν η τάση που εφαρμόζεται εξωτερικά στο pin είναι high η low. Πιο συγκεκριμένα εάν η τάση στο pin είναι πάνω από το μισό της τάσης τροφοδοσίας, το chip θέτει ένα bit μίας εσωτερικής μεταβλητής ίσο με 1. Αν η τάση είναι κάτω από το μισό της τάσης τροφοδοσίας, το ίδιο bit θα γίνει ίσο με 0.

3.6.1.3 Περιφερειακά του Μικροελεγκτή AVR

Serial Communications: Ο AVR διαθέτει τρείς σειριακές επικοινωνίες, την USART, η οποία είναι χρήσιμη στην επικοινωνία με τον ηλεκτρονικό μας υπολογιστή, με radio modems, GPS. Τα αρχικά του USART σημαίνουν Universal Synchronous and Asynchronous Receiver and Transmitter, και δεν κάνει χρήση του ρολογιού σε αντίθεση με το UART, Universal Asynchronous Receive and Transmit, το οποίο χρησιμοποιεί το ρολόι. Την SPI, Serial Peripheral Interface, η οποία είναι καλή για γρήγορη επικοινωνία σε μικρές αποστάσεις, όπως μνήμες, ADCs και DACs. Τέλος την I2C η οποία είναι σαν ένα μικρό δίκτυο, και μας επιτρέπει να συνδέσουμε πάνω από 127 διαφορετικούς αισθητήρες στο ίδιο ζεύγος καλωδίων. Συσκευές που διαχειρίζονται μικρή ποσότητα δεδομένων συνήθως χρησιμοποιούν την I2C. Επειδή κάθε μία από τις περιφερειακές συσκευές είναι ξεχωριστή μέσα στον AVR, μπορούμε να χρησιμοποιήσουμε κάθε μία από αυτές ταυτόχρονα.

Analog to digital converter: Ένας αριθμός από αισθητήρια που θα χρησιμοποιήσουμε στα project μας, δεν «μιλούν» την ψηφιακή μητρική γλώσσα του μικροελεγκτή. Αντίθετα θα λέγαμε ότι ο τρόπος επικοινωνίας τους γίνεται με συνεχόμενα αναλογικά σήματα τάσης. Για να διαβάσουμε και να διαχειριστούμε αυτές τις τιμές όπως θα κάναμε με οποιαδήποτε άλλα ψηφιακά δεδομένα, θα χρειαστεί να τις περάσουμε από ένα analog to digital converter, ADC. Υπάρχουν δύο τεχνικές που μπορούν να μας αποφέρουν μεγαλύτερη ακρίβεια και μείωση του θορύβου στον ADC, αυτές είναι η oversampling και exponential smoothing.

Interrupts: Είναι μία συνάρτηση που μπορούμε να γράψουμε στο πρόγραμμα μας, η οποία θα εκτελείτε αυτόματα κάθε φορά που συναντάται η συνθήκη ενός interrupt. Λέγεται ρουτίνα διακοπής (interrupt) επειδή ο επεξεργαστής σταματάει οτιδήποτε έκανε στην κύρια ροή του προγράμματος και τρέχει την κατάλληλη συνάρτηση. Μόλις τελειώσει

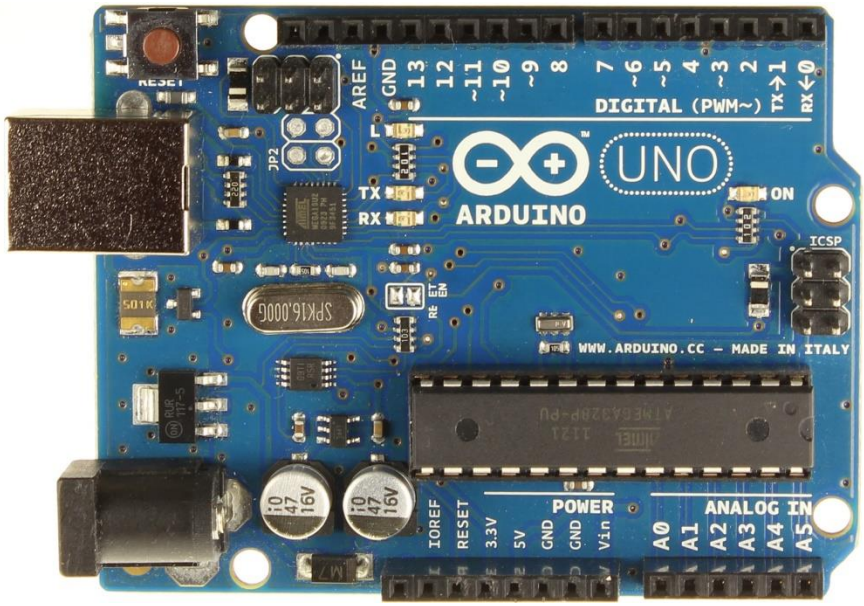
με την ρουτίνα διακοπής, δηλαδή μετά την εκτέλεση της συνάρτησης του interrupt, ο επεξεργαστής ξεκινά και πάλι από το σημείο που είχε σταματήσει.

Υπάρχουν πολλοί τρόποι να προκαλέσουμε ένα interrupt σε ένα AVR μικροελεγκτή. Πατώντας το κουμπί reset, αλλάζοντας μία τιμή εισόδου, σε παλμό του ρολογιού, συγκεκριμένη τιμή στον counter, δεδομένα από και προς την σειριακή θύρα, στο πέρας της μετατροπής analog-to-digital και πολλά άλλα.

Timers/Counters: Οι μικροεπεξεργαστές AVR έχουν ενσωματωμένους hardware μετρητές (counters). Οι μετρητές αυτοί κάνουν το προφανές, μετρούν πόσες φορές ένα pin ή μία εσωτερική πηγή άλλαξε την τάση της. Η πιο απλή εφαρμογή ενός counter, είναι να συνδέσουμε τον εσωτερικό counter με ένα κουμπί. Τώρα μπορούμε να δούμε πόσες φορές πατήθηκε αυτό το κουμπί απλά διαβάζοντας τον καταχωρητή (register) του μετρητή. Οι μετρητές μπορούν επίσης να συνδυαστούν και με ρολόγια, γι' αυτό και συχνά αναφέρονται ως timers/counters. Με ένα ρολόι και ένα timer, μπορείς να μετρήσεις είτε πόσο χρόνο διαρκεί κάποιο γεγονός, ή τη συχνότητα του γεγονότος αυτού. Θα έχουμε επίσης τη δυνατότητα να εξάγουμε σε συγκεκριμένα pins του AVR και να εκτελέσουμε περιοδικά συγκεκριμένες υπορουτίνες.

4 Η Υπολογιστική Πλατφόρμα Arduino

Ένα κεφάλαιο αποκλειστικά αφιερωμένο στο Arduino. Στην εικόνα 4-1 βλέπουμε το Arduino Uno R3, ένα από τα δημοφιλή μέλη της οικογένειας Arduino, το οποίο και χρησιμοποιούμε στην παρούσα εργασία. Οι δυνατότητες του παρουσιάζουν τεράστιο ενδιαφέρον και παρακάτω θα αναλύσουμε ότι αφορά την συγκεκριμένη πλατφόρμα, από το Hardware μέχρι τον τρόπο λειτουργίας της και όχι μόνο.



Εικόνα 4-1: Arduino Uno R3

4.1 Ιστορική Αναδρομή

Το Arduino ξεκίνησε σε ένα μικρό εργοστάσιο στην πόλη της Ιβρέα (Interaction Design Institute Ivrea γνωστό και ως Interaction Ivrea, IDII ή Ivrea), η οποία είναι κωμόπολη της επαρχίας Τορίνο στην περιοχή πεδεμόντιο της βορειοδυτικής Ιταλίας, στην ίδια περιοχή στην οποία στεγαζόταν η εταιρία υπολογιστών Olivetti. Στόχος ήταν να φτιαχτεί μία συσκευή για τον έλεγχο προγραμμάτων διαδραστικών σχεδίων από μαθητές, η οποία θα ήταν πιο φθηνή από τα άλλα πρωτότυπα συστήματα που ήταν διαθέσιμα εκείνη την περίοδο. Οι ιδρυτές Massimo Banzi και David Cueartielles ονόμασαν το σχέδιο Arduino από τον Arduin της Ιβρέα, βασιλιά της Ιταλίας. Το πρόγραμμα Arduino έλαβε τιμητική μνεία στην κατηγορία Digital Communities στο Prix Ars Electronica το 2006.

4.2 Τι είναι το Arduino

Το Arduino είναι μία υπολογιστική πλατφόρμα ανοιχτού κώδικα βασισμένη σε μία απλή μητρική πλακέτα με ενσωματωμένο μικροελεγκτή η οποία μπορεί να προγραμματιστεί με τη γλώσσα Wiring, μια παραλλαγή της C/C++ για μικροελεγκτές αρχιτεκτονικής AVR όπως το ATmega, και υποστηρίζει όλες τις βασικές δομές της C καθώς και μερικά χαρακτηριστικά της C++. Θα λέγαμε ότι είναι ένα εργαλείο για να κατασκευάσουμε ένα υπολογιστικό σύστημα με την έννοια ότι αυτό θα ελέγχει συσκευές του φυσικού κόσμου, σε αντίθεση με τον κοινό μας Ηλεκτρονικό Υπολογιστή.

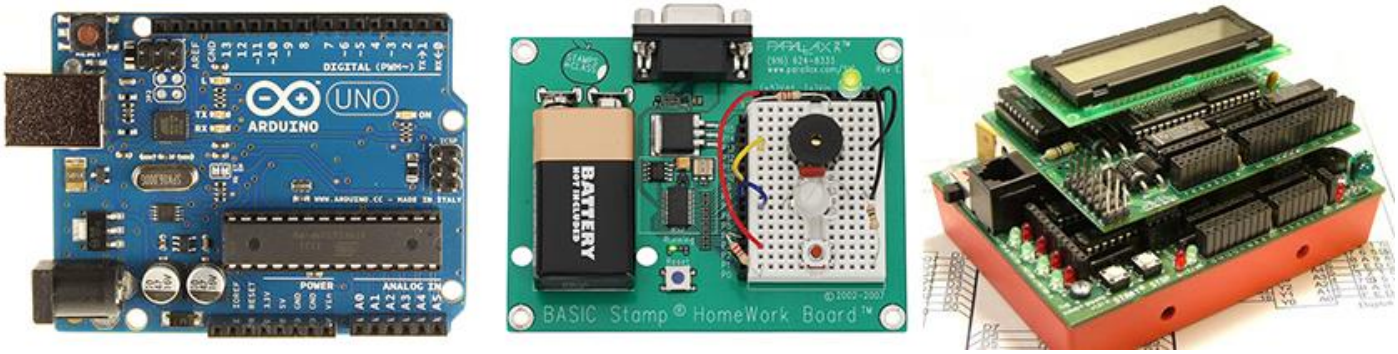
Μπορεί να χρησιμοποιηθεί στην ανάπτυξη διαδραστικών αντικειμένων, μπορεί να πάρει είσοδο από διάφορους διακόπτες ή αισθητήρες και να ελέγξει φώτα, κινητήρες και άλλες φυσικές εξόδους. Τα project μπορεί να είναι είτε stand-alone (αυτόνομα), είτε να επικοινωνούν με λογισμικό που τρέχει στον υπολογιστή μας όπως Flash, Processing, MaxMSP. Οι πλακέτες μπορούν να συναρμολογηθούν στο χέρι ή να αγοραστούν έτοιμες. Το περιβάλλον ανάπτυξης λογισμικού, το οποίο είναι ανοιχτού κώδικα, μπορεί κάποιος να το κατεβάσει δωρεάν από την επίσημη ιστοσελίδα του Arduino http://arduino.cc/en/Main/Software.

4.3 Γιατί το Προτιμούμε

Υπάρχουν αρκετοί μικροελεγκτές διαθέσιμοι στην αγορά για κάποιον που θέλει να ασχοληθεί. Κάποιοι από αυτούς είναι ο Basic Stamp της Parallax, ο BX-24 της NetMedia, το Handyboard του MIT, και πολύ άλλοι οι οποίοι

προσφέρουν παρόμοιες δυνατότητες. Όλα αυτά τα εργαλεία που προαναφέραμε είναι απλά και για τον αρχάριο χρήστη καθώς «κρύβουν» τις δύσκολες λεπτομέρειες της αρχιτεκτονικής και επιτρέπουν τον άμεσο προγραμματισμό του μικροελεγκτή, προσφέροντας τα πάντα σε ένα και μόνο «πακέτο» έτοιμο για χρήση. Το Arduino διαφέρει από τους προηγούμενους γιατί απλοποιεί την διαδικασία να δουλεύει κάποιος με μικροελεγκτές, αλλά κάποια πλεονεκτήματα που προσφέρει σε σχέση με άλλους μικροελεγκτές για χρήση από δασκάλους, μαθητές και άλλους χομπίστες είναι τα παρακάτω:

- Είναι φθηνό. Οι πλακέτες Arduino είναι σχετικά φθηνές σε σχέση με άλλους μικροελεγκτές. Η φθηνότερη εκδοχή του Arduino μπορεί να κατασκευαστεί στο χέρι αν και η έτοιμη πλακέτα δεν θα κοστίσει πάνω από πενήντα ευρώ.
- Τρέχει σε διάφορα Λειτουργικά Συστήματα. Οι μηχανικοί λογισμικού, ανέπτυξαν το περιβάλλον προγραμματισμού του Arduino για Windows, Macintosh OSX και για λειτουργικά συστήματα Linux. Τα περισσότερα συστήματα ανάπτυξης μικροελεγκτών περιορίζονται στα Windows.
- Απλό, ξεκάθαρο προγραμματιστικό περιβάλλον. Το περιβάλλον προγραμματισμού ενός Arduino ενδείκνυται για αρχάριους, αλλά είναι ταυτόχρονα και ευέλικτο και για πιο προχωρημένους χρήστες.
- Ανοιχτού λογισμικού και λογισμικού που επεκτείνεται και παραμετροποιείται. Το Software του Arduino διανέμεται με την μορφή εργαλείων ανοιχτού λογισμικού και είναι διαθέσιμο προς επέκταση για έμπειρους προγραμματιστές. Η γλώσσα προγραμματισμού του μπορεί να επεκταθεί διαμέσου των βιβλιοθηκών της C++ και οι άνθρωποι που θέλουν να ασχοληθούν περισσότερο με τους μικροελεγκτές μπορούν να μεταβούν από το Arduino στην AVR C που είναι για προγραμματισμό των Atmel Μικροελεγκτών και η γλώσσα στην οποία βασίστηκε το λογισμικό του Arduino. Ομοίως μπορεί κάποιος να προσθέσει κώδικα της AVR C στο πρόγραμμα που έχει γράψει για το Arduino του.
- Ανοιχτού Υλικού το οποίο μπορεί να επεκταθεί. Το Arduino βασίζεται στους μικροελεγκτές της Atmel ATMEGA8, ATMEGA168 και πλέον στο ATMEGA328. Τα σχηματικά για τα αναπτυξιακά είναι κάτω από την άδεια της Creative Commons, επιτρέποντας σε έμπειρους σχεδιαστές να κατασκευάσουν το δικό τους αναπτυξιακό, εξελίσσοντας το ήδη υπάρχον χωρίς να έχουν νομικά προβλήματα. Η ακόμη καλύτερα όχι τόσο έμπειροι χρήστες μπορούν να επιδιώξουν την αντιγραφή και κατασκευή της πλακέτας σε ράστερ για να καταλάβουν την λειτουργία ενός Arduino.



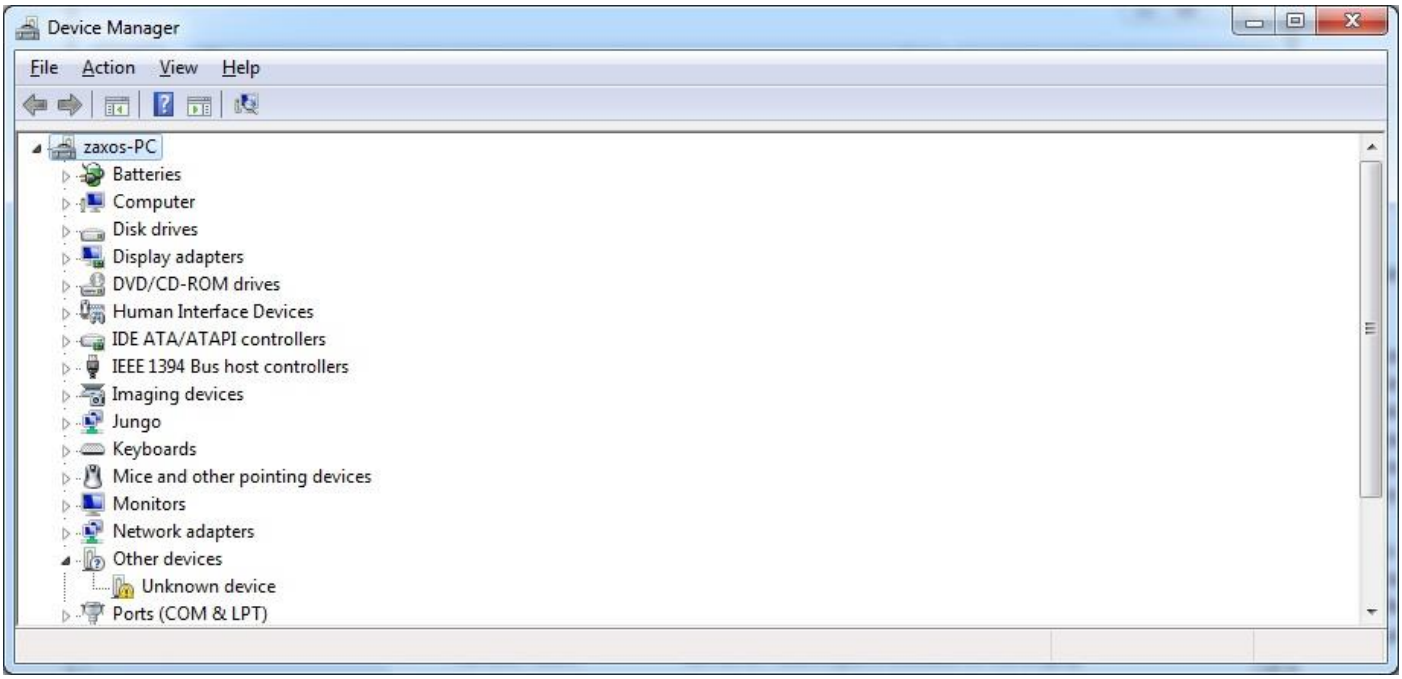
Εικόνα 4-2: Arduino Uno R3, Basic Stamp της Parallax και HandyBoard του Mit

4.4 Εγκατάσταση του Arduino στον Υπολογιστή μας

Θα ξεκινήσουμε κατεβάζοντας την νεότερη έκδοση του προγράμματος ανάπτυξης λογισμικού (Arduino IDE) από την ιστοσελίδα http://arduino.cc/en/Main/Software το οποίο είναι διαθέσιμο για Windows, Mac και Linux, στη συνέχεια χρησιμοποιώντας ένα καλώδιο Usb (A plug σε B plug) θα συνδέσουμε το Arduino στον ηλεκτρονικό μας υπολογιστή. Ένα πράσινο led με την ένδειξη PWR θα ανάψει αμέσως. Σε περίπτωση που πάρουμε το μήνυμα από τα Windows «Found New Hardware» το αγνοούμε και ακυρώνουμε κάθε προσπάθεια που κάνουν τα windows για να εγκαταστήσουν τους Drivers για εμάς.

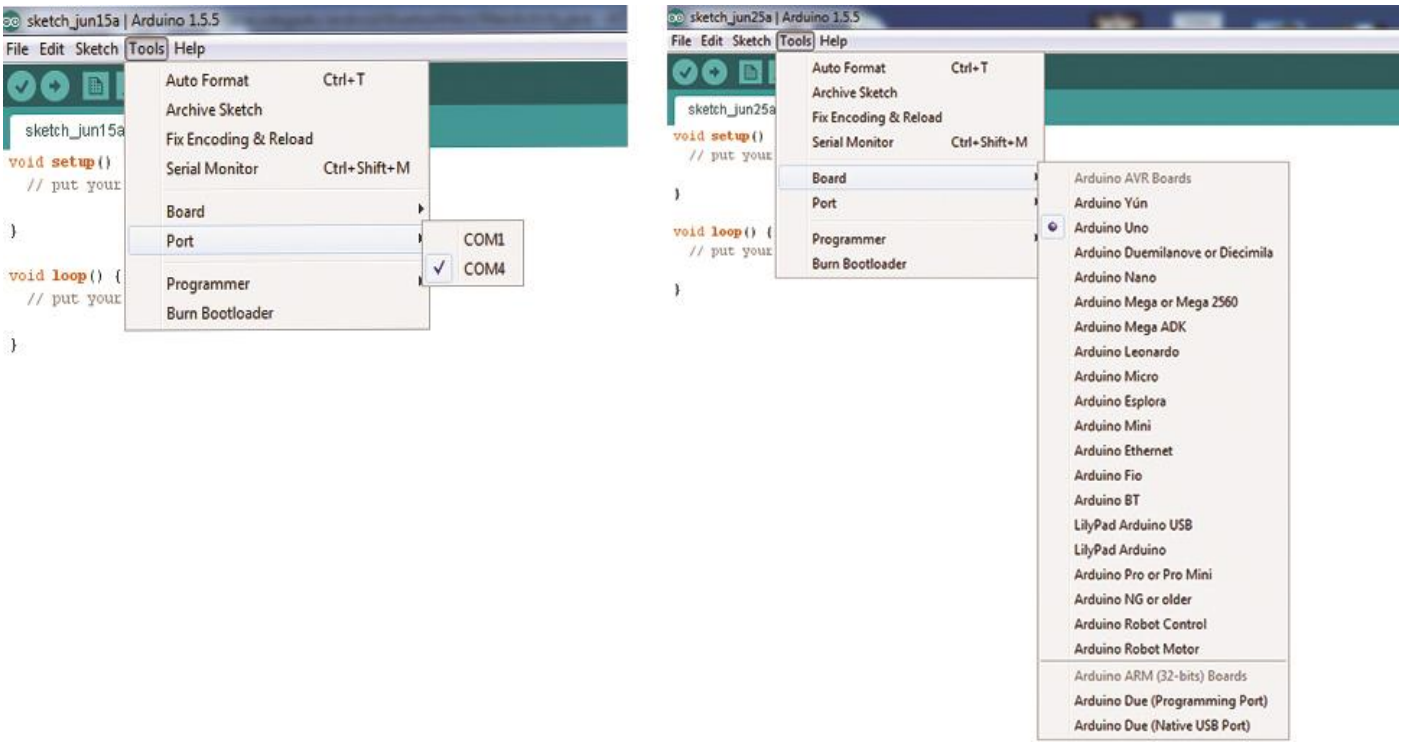
Κάνουμε κλικ στο μενού έναρξη και ανοίγουμε τον πίνακα ελέγχου, από εκεί πηγαίνουμε στο «Σύστημα και ασφάλεια» και ανοίγουμε την «Διαχείριση συσκευών». Εκεί κάτω από τον τομέα «Ports (COM & LPT)» ή πιθανών στον τομέα «Άλλες συσκευές» υπάρχει ένα εικονίδιο «Άγνωστη συσκευή» με ένα μικρό κίτρινο τρίγωνο δίπλα του. Αυτό είναι το Arduino μας (Εικόνα 4-3). Κάνουμε δεξί κλικ στη συσκευή και επιλέγουμε από το μενού «ενημέρωση

προγράμματος οδήγησης». Τέλος από το πλαίσιο που θα ανοίξει επιλέγουμε το «αναζήτηση λογισμικού προγράμματος από τον υπολογιστή μου» και πηγαίνουμε στο φάκελο όπου κατεβάσαμε το λογισμικό μας, εκεί υπάρχει ο υποφάκελος «drivers» σε αυτόν θα βρούμε το αρχείο «arduino.inf» το οποίο και θα επιλέξουμε. Κάνουμε κλικ στο «επόμενο» και μάλλον θα πάρουμε μια προειδοποίηση ασφαλείας, αν ναι, αφήνουμε το λογισμικό να εγκατασταθεί. Μόλις το λογισμικό εγκατασταθεί, θα λάβουμε ένα μήνυμα επιβεβαίωσης. Τώρα είμαστε έτοιμη να τρέξουμε για πρώτη φορά το Arduino IDE.



Εικόνα 4-3: Διαχείριση συσκευών, βρέθηκε άγνωστη συσκευή

Από εκεί πρέπει να επιλέξουμε τον τύπο της πλακέτας που χρησιμοποιούμε και την θύρα που είναι συνδεδεμένο το Arduino, δεν πρέπει να ξεχνάμε ποτέ αυτό το βήμα. Πάμε λοιπόν στο μενού εργαλείων, tools >Board και επιλέγουμε το Arduino Uno (Εικόνα 4-4). Από το ίδιο μενού θα βρούμε την επιλογή «Serial ports». Κατά πάσα πιθανότητα θα υπάρχει μόνο η θύρα COM3 ή κάποια μεγαλύτερη (η COM1 και COM2 είναι συνήθως είδη δεσμευμένες από το σύστημα). Ένας εύκολος τρόπος να βρούμε σε ποια θύρα είναι συνδεδεμένο το Arduino είναι να το αποσυνδέσουμε και να ξανά ανοίξουμε το μενού «serial ports», η θύρα που εξαφανίστηκε είναι εκείνη στην οποία είναι συνδεδεμένο το Arduino. Ξανά συνδέουμε το Arduino και επιλέγουμε την συγκεκριμένη θύρα.



Εικόνα 4-4: Επιλογή πλατφόρμας και θύρας

Τώρα μπορούμε απλά να περάσουμε στο Arduino μας κώδικα κάνοντας κλικ στο κουμπί «Upload». Περιμένοντας μερικά δευτερόλεπτα θα δούμε τις λυχνίες Rx και Tx να αναβοσβήνουν. Αν περάσαμε τον κώδικα επιτυχώς στην πλακέτα θα εμφανιστεί ένα μήνυμα «Done Uploading». Λίγο μετά το ανέβασμα του κώδικα μας, θα δούμε το 13 (L) LED στην πλακέτα μας να αναβοσβήνει πορτοκαλί. Αν ναι, συγχαρητήρια! Το arduino μας είναι έτοιμο και τρέχει τον κώδικα μας.

4.5 Το Arduino IDE

Το περιβάλλον ανάπτυξης λογισμικού (IDE) του Arduino, είναι μία εφαρμογή γραμμένη σε JAVA, που διατίθεται για τις πλατφόρμες Windows, Mac και Linux. Μας βοηθά στη συγγραφή των προγραμμάτων (σκίτσα ή στα αγγλικά sketch, στην ορολογία του Arduino), διαθέτει αρκετά έτοιμα παραδείγματα, μερικές έτοιμες βιβλιοθήκες, για προέκταση της γλώσσας και για τον εύκολο χειρισμό των εξαρτημάτων που είναι συνδεδεμένα με το Arduino μέσα από τον κώδικα, compiler για την μεταγλώττιση των προγραμμάτων μας, ένα Serial monitor, που παρακολουθεί την σειριακή επικοινωνία (Usb), αναλαμβάνει να στείλει αλφαριθμητικά της επιλογής μας στο Arduino μέσω αυτής και είναι ιδιαίτερα χρήσιμο για το debugging (επιδιόρθωση σφαλμάτων) των προγραμμάτων μας. Επίσης, είναι σε θέση να φορτώνει (Upload) το πρόγραμμά μας στο arduino με ένα μόνο κλικ.

Τα σκίτσα αυτά γράφονται στον επεξεργαστή κειμένου που παρέχει το Arduino IDE και αποθηκεύονται με την επέκταση αρχείου «.ino». Επίσης, το IDE του Arduino μας παρέχει τα χαρακτηριστικά γνωρίσματα της αντιγραφής/επικόλλησης και της αναζήτησης/αντικατάστασης κειμένου. Στην περιοχή μηνύματος εμφανίζονται μηνύματα που αφορούν την διαχείριση του Arduino, όπως μηνύματα επιβεβαίωσης μετά από αποθήκευση ή φόρτωση του προγράμματος. Ακόμα, η κονσόλα απεικονίζει μηνύματα κειμένου από το περιβάλλον του Arduino, συμπεριλαμβανομένου και των μηνυμάτων λάθους που υπάρχουν στο πρόγραμμα μετά την μεταγλώττιση. Στην κάτω δεξιά γωνία του παραθύρου εμφανίζεται το όνομα της πλατφόρμας που έχουμε συνδεδεμένη στον υπολογιστή μας (πχ. Arduino UNO στην δική μας περίπτωση) και η θύρα που είναι συνδεδεμένη (πχ.COM3). Τα κουμπιά που υπάρχουν στη γραμμή εργαλείων του Arduino IDE, μας επιτρέπουν να δημιουργήσουμε, να αποθηκεύσουμε και να ανοίξουμε σκίτσα, να ελέγξουμε και να ανεβάσουμε προγράμματα στο arduino, καθώς και να ανοίξουμε το παράθυρο της σειριακής οθόνης (Serial monitor).

Επαλήθευση (Verify)
Ελέγχει τον κώδικα μας για λάθη, (Compile).

Φόρτωση (Upload)
Μεταγλωττίζει (Compile) το πρόγραμμά μας και το φορτώνει στη πλατφόρμα Arduino.

Νέο (New)
Δημιουργεί ένα νέο σκίτσο.

Άνοιγμα (Open)
Παρουσιάζει ένα μενού με όλα τα σκίτσα στο προεπιλεγμένο φάκελο αποθήκευσης (Sketchbook) ώστε να επιλέξουμε αυτό που θέλουμε να ανοίξουμε.

Αποθήκευση (Save)
Αποθηκεύει το σκίτσο μας.

Σειριακή οθόνη (Serial Monitor)
Ανοίγει την σειριακή οθόνη.

Πρόσθετες εντολές θα βρούμε στα πέντε μενού: Αρχείο (File), Επεξεργασία (Edit), Σχέδιο (Sketch), Εργαλεία (Tools), Βοήθεια (Help). Τα μενού αυτά έχουν ευαισθησία πλαισίου, δηλαδή μόνο τα στοιχεία που σχετίζονται με τις εργασίες που πραγματοποιούνται είναι διαθέσιμα.
Επεξεργασία (Edit):

- Αντιγραφή για το Forum, αντιγράφει τον κώδικα από το σκίτσο μας στο πρόχειρο σε κατάλληλη μορφή προς δημοσίευση στο forum του Arduino.
- Αντιγραφή ως HTML, αντιγράφει τον κώδικα από το σκίτσο μας στο πρόχειρο, σε μορφή HTML που είναι κατάλληλη για την ενσωμάτωση σε ιστοσελίδες.

Σχέδιο (Sketch):

- Επαλήθευση / Μεταγλώττιση (Verify / Compile).
  Ελέγχει το σκίτσο μας για τυχόν σφάλματα.
- Εμφάνιση Φακέλου Σχεδίου (Show Sketch Folder). Ανοίγει τον τρέχοντα φάκελο που βρίσκεται το σκίτσο μας.
- Προσθήκη Αρχείου (Add File). Προσθέτει ένα αρχείο στο σκίτσο μας (θα αντιγραφεί από την τρέχουσα θέση του). Το νέο αρχείο εμφανίζεται σε μια νέα καρτέλα στο παράθυρο σκίτσο (sketch). Τα αρχεία μπορούν να αφαιρεθούν από το σκίτσο χρησιμοποιώντας το μενού καρτέλα (Tab).
- Εισαγωγή Βιβλιοθήκης (Import Library). Προσθέτει μία βιβλιοθήκη στο σκίτσο μας περιλαμβάνοντας την δήλωση «#include» κατά την έναρξη του κώδικά μας.

Εργαλεία (Tools):

- Αυτόματη Μορφοποίηση (Auto Format). Μορφοποιεί το κώδικα μας ομοιόμορφα, για παράδειγμα στοιχίζει το περιεχόμενο που βρίσκεται μέσα στα άγκιστρα αρχή και τέλους του κώδικα.
- Αρχειοθέτηση Σχεδίου (Archive Sketch). Αρχειοθετεί ένα αντίγραφο του τρέχοντος σκίτσου σε μορφή «.zip». Το αρχείο βρίσκεται στον ίδιο κατάλογο με το σκίτσο.
- Πλακέτα (Board). Επιλέγουμε την πλατφόρμα που χρησιμοποιούμε.
- Σειριακή Θύρα (Serial Port). Αυτό το μενού περιέχει όλες τις σειριακές συσκευές (πραγματικές ή εικονικές) στον υπολογιστή μας. Ανανεώνεται κάθε φορά που ανοίγουμε το μενού «Εργαλεία».
- Προγραμματιστής (Programmer). Συνήθως δεν χρησιμοποιείται, παρά μόνο όταν θέλουμε να προγραμματίσουμε έναν νέο μικροελεγκτή χρησιμοποιώντας τον Bootloader του Arduino.
- Γράψιμο Bootloader (Burn Bootloader). Η επιλογή αυτή μας επιτρέπει να γράψουμε έναν Bootloader στο μικροελεγκτή σε μια πλακέτα Arduino. Αυτό δεν είναι απαραίτητο, αλλά είναι χρήσιμο στη περίπτωση που έχουμε αγοράσει ένα νέο μικροελεγκτή ATmega, ο οποίος συνήθως δεν έχει Bootloader. Πρέπει να βεβαιωθούμε ότι έχουμε επιλέξει την σωστή πλατφόρμα από το μενού «Πλακέτα» πριν την εγγραφή του Bootloader.

Το περιβάλλον του Arduino χρησιμοποιεί την έννοια του τετραδίου (Sketchbook), δηλαδή έναν προεπιλεγμένο χώρο για την αποθήκευση των προγραμμάτων μας. Τα σκίτσα που βρίσκονται στο φάκελο Sketchbook μπορούμε να τα ανοίξουμε από το μενού Αρχείο >Σχέδια (File > Sketchbook) ή από το κουμπί Άνοιγμα (Open) της γραμμής εργαλείων την πρώτη φορά που θα εκτελέσουμε το λογισμικό Arduino, θα δημιουργήσουμε έναν κατάλογο sketchbook.

Μπορούμε να αλλάξουμε ή να δούμε τη θέση της τοποθεσίας του φακέλου από το παράθυρο διαλόγου που βρίσκετε στο υπομενού «Προτιμήσεις» (File > Preferences).

Το περιβάλλον του Arduino μας επιτρέπει να διαχειριζόμαστε σκίτσα με περισσότερα από ένα αρχεία (καθένα από τα οποία εμφανίζεται σε δική του καρτέλα). Αυτά μπορεί να είναι κανονικά αρχεία κώδικα Arduino (χωρίς επέκταση), C αρχεία (επέκταση «.C»), C++ αρχεία (επέκταση «.CPP») ή αρχεία κεφαλίδας («.H») .

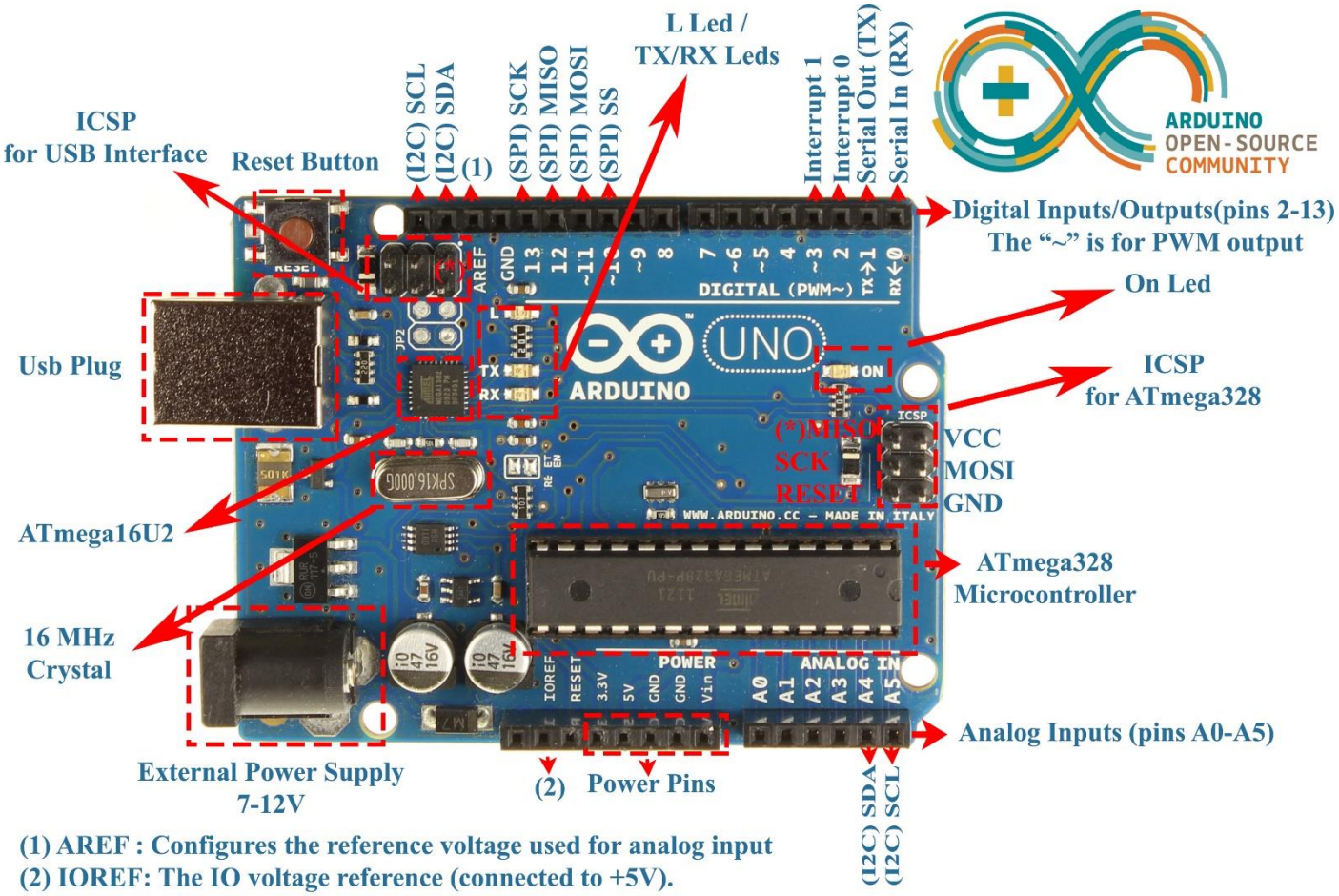
Πριν την φόρτωση του σκίτσου μας, θα πρέπει να επιλέξουμε τα σωστά στοιχεία από το μενού Εργαλεία> Πλακέτα (Tools>Board) και Εργαλεία>Σειριακή Θύρα (Tools>SerialPort). Όταν έχουμε επιλέξει τη σωστή σειριακή θύρα και τη σωστή πλατφόρμα, πατάμε το κουμπί αποστολής από το μενού Αρχείο. Η τρέχουσα πλατφόρμα Arduino θα επαναφερθεί αυτόματα (Reset) και θα αρχίσει η μεταφόρτωση (Upload). Στις περισσότερες πλατφόρμες, θα παρατηρήσουμε τα LED RX και TX να αναβοσβήνουν καθώς φορτώνεται το σκίτσο. Το περιβάλλον Arduino θα εμφανίσει ένα μήνυμα όταν η αποστολή έχει ολοκληρωθεί ή θα εμφανίσει σφάλμα εάν αυτή αποτύχει. Όταν φορτώνουμε ένα σκίτσο, χρησιμοποιούμε τον Bootloader του Arduino, ένα μικρό πρόγραμμα που έχει φορτωθεί στο μικροελεγκτή της πλατφόρμας. Μας επιτρέπει να φορτώσουμε κώδικα χωρίς να χρησιμοποιήσουμε εξωτερικό προγραμματιστή.

Οι βιβλιοθήκες παρέχουν επιπλέον λειτουργίες για την χρήση σε σκίτσα, για παράδειγμα συνεργασία με το υλικό ή διαχείριση δεδομένων. Για να χρησιμοποιήσουμε μια βιβλιοθήκη σε ένα σκίτσο, την επιλέγουμε από το μενού Σχέδιο> Εισαγωγή Βιβλιοθήκης (Sketch>Import Library). Αυτό θα εισάγει μια ή περισσότερες δηλώσεις «#include» στην κορυφή του κώδικά μας και έτσι η βιβλιοθήκη θα μεταγλωττιστεί μαζί με το πρόγραμμά μας. Επειδή οι βιβλιοθήκες φορτώνονται και αυτές μαζί με το σκίτσο μας στη πλατφόρμα, αυξάνουν τη ποσότητα του χώρου που καταλαμβάνει το πρόγραμμα στη μνήμη του Arduino. Εάν λοιπόν το πρόγραμμά μας δεν χρειάζεται μια βιβλιοθήκη απλά την διαγράφουμε. Μερικές βιβλιοθήκες συμπεριλαμβάνονται στο περιβάλλον του Arduino, ενώ άλλες πρέπει να τις κατεβάσουμε.

Η σειριακή οθόνη (Serial Monitor) εμφανίζει σειριακά δεδομένα που αποστέλλονται από το Arduino (μέσω Usb ή serial board). Για να στείλουμε δεδομένα στη πλατφόρμα Arduino απλά εισάγουμε το κείμενο και κάνουμε κλικ στο κουμπί «Αποστολή» (Send) και πατάμε «Enter». Επιλέγουμε το ίδιο baud rate από το μενού που υπάρχει στο Serial monitor, με αυτό που έχουμε δηλώσει στο Serial.begin του κώδικά μας. Σημειώνουμε ότι σε Mac ή Linux, η πλατφόρμα Arduino κάνει επανεκκίνηση, όταν συνδεόμαστε με την σειριακή οθόνη. Μπορούμε ακόμα να «μιλήσουμε» με την πλατφόρμα και με άλλα προγράμματα (κυρίως απεικόνισης), όπως είναι τα Processing, Flash, MaxMSP κ.α.

4.6 Arduino Uno Rev 3 Hardware

Το Arduino Uno R3 είναι μια υπολογιστική πλατφόρμα βασισμένη στο ATmega328 (Εικόνα 4-5). Έχει 14 ψηφιακούς ακροδέκτες εισόδου και εξόδου (από τους οποίους οι 6 μπορούν να χρησιμοποιηθούν ως έξοδοι PWM), έξι αναλογικές εισόδους (μπορούν να λειτουργήσουν και ως General purpose input/output (GPIO) pins (όπως τα pins από 0 - 13) που χρησιμοποιούνται για να διαβάσουν αναλογικές τιμές πχ από μια μπαταριά η ένα ποτενσιόμετρο κτλ., επιστρέφοντας ακέραιες τιμές από 0-1023, ένα κεραμικό κρύσταλλο (Ceramic resonator) στα 16 MHz, μία σύνδεση Usb, βύσμα τροφοδοσίας, ICSP header και ένα κουμπί reset. Επίσης το Uno χρησιμοποιεί το ATmega16U2 προγραμματισμένο να δουλεύει ως μετατροπέας Usb σε Serial σε αντίθεση με προηγούμενες πλακέτες που χρησιμοποιούσαν το FTDI Usb σε Serial driver chip .Στον πίνακα 4-1 θα δούμε εν συντομία τα χαρακτηριστικά του Uno.



Εικόνα 4-5: Αναλυτική παρουσίαση του Hardware του Arduino

Στον παρακάτω πίνακα παρουσιάζονται συγκεντρωτικά τα χαρακτηριστικά του Arduino.

Πίνακας 4-1: Χαρακτηριστικά του Arduino Uno R3

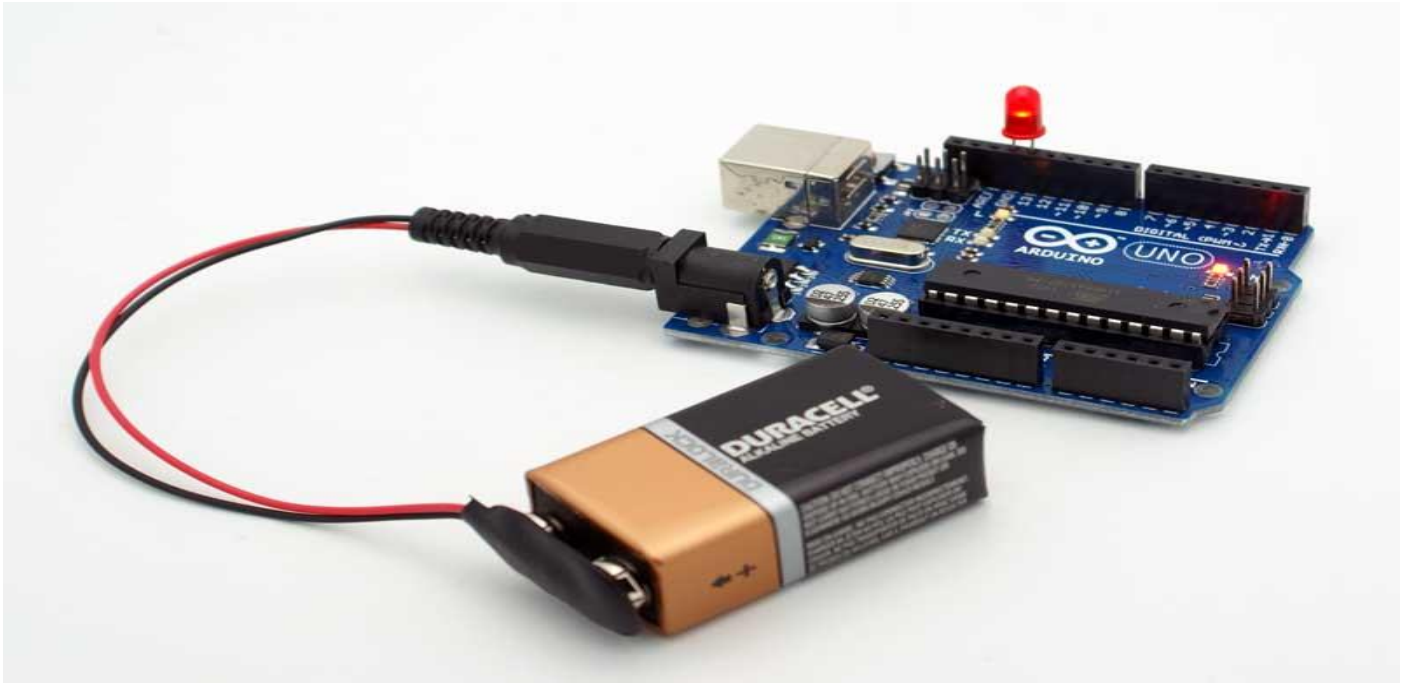
Μικροελεγκτής	**ATmega328**
Τάση λειτουργίας	5V
Συνιστώμενη τάση εισόδου	7-12V
Όρια τάσης εισόδου	6-20V
Ψηφιακοί ακροδέκτες εσόδου/εξόδου	14 (από τις οποίες οι έξι είναι έξοδοι PWM)
Αναλογικοί ακροδέκτες εισόδου	6
DC ρεύμα ανά I/O Pin	40 mA
DC ρεύμα για τον ακροδέκτη 3.3V	50 mA
Μνήμα Flash	32 KB (ATmega328) 0.5 KB χρησιμοποιούνται από τον Bootloader
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Ταχύτητα ρολογιού	16 MHz

4.6.1 Τροφοδοσία

Το Arduino Uno μπορεί να τροφοδοτηθεί είτε μέσω καλωδίου Usb είτε με κάποια εξωτερική πηγή ενέργειας. Η επιλογή της πηγής γίνεται αυτόματα από το Arduino. Η εξωτερική τροφοδοσία μπορεί να προέρχεται είτε από έναν AC σε Dc μετασχηματιστή είτε από μπαταρία. Για να συνδέσουμε το μετασχηματιστή τοποθετούμε το βύσμα στην

υποδοχή που υπάρχει με τον θετικό πόλο στο κέντρο. Από την άλλη η μπαταρία μπορεί να συνδεθεί στις υποδοχές Vin και Gnd όπου τοποθετούνται ο θετικός πόλος και ο αρνητικός (γείωση) αντίστοιχα.

Η πλακέτα μπορεί να λειτουργήσει σε εξωτερική τροφοδοσία από 6 έως 20 Volts (αυτά είναι τα όρια, θα πρέπει να τα αποφεύγουμε). Αν τροφοδοτηθεί με λιγότερα από 7V, τα pin εξόδου 5V δεν θα καταφέρουν να εξάγουν τάση 5V και η πλακέτα ίσως παρουσίαση προβλήματα αστάθειας. Από την άλλη αν δώσουμε πάνω από 12V, υπάρχει η πιθανότητα να υπερθερμανθεί ο σταθεροποιητής τάσης και να καταστραφεί η πλακέτα. Επομένως η ιδανική τάση είναι από 7 έως 12 Volts.



Εικόνα 4-5: Παράδειγμα τροφοδοσίας από μία μπαταρία 9V

Οι ακροδέκτες τροφοδοσία είναι οι εξής:

- Vin. Εδώ συνδέεται συνήθως μια εξωτερική πηγή τροφοδοσίας για το Arduino μας, είναι ακροδέκτης μη σταθεροποιημένης τάσης. Ωστόσο αν τροφοδοτούμε την πλακέτα μέσω του βύσματος τροφοδοσίας (Power jack), μπορούμε να χρησιμοποιήσουμε το Vin ως έξοδο.
- 5V. Ακροδέκτης σταθεροποιημένης τάσης. Η πλακέτα μπορεί να τροφοδοτηθεί είτε μέσω του DC power jack (7-12V), είτε μέσω του καλωδίου Usb (5V), είτε από το Vin (7-12V). Προσοχή, δεν ενδείκνυται να τροφοδοτήσουμε την πλακέτα από τους ακροδέκτες 5V η 3.3V γιατί παρακάμπτετε ο σταθεροποιητής τάσης και είναι πολύ πιθανό να κάνουμε ζημιά στην πλακέτα.
- 3.3V. παράγεται τάση 3.3V από τον ενσωματωμένο σταθεροποιητή με μέγιστο ρεύμα 50mA.
- GND. Ακροδέκτες γείωσης.
- IOREF. Ο συγκεκριμένος ακροδέκτης δίνει την τάση λειτουργίας του μικροελεγκτή και χρησιμοποιείται κυρίως στα Shield ώστε να αναγνωρίζεται από αυτά αν το κύκλωμα δουλεύει στα 3.3 η 5 Volts.

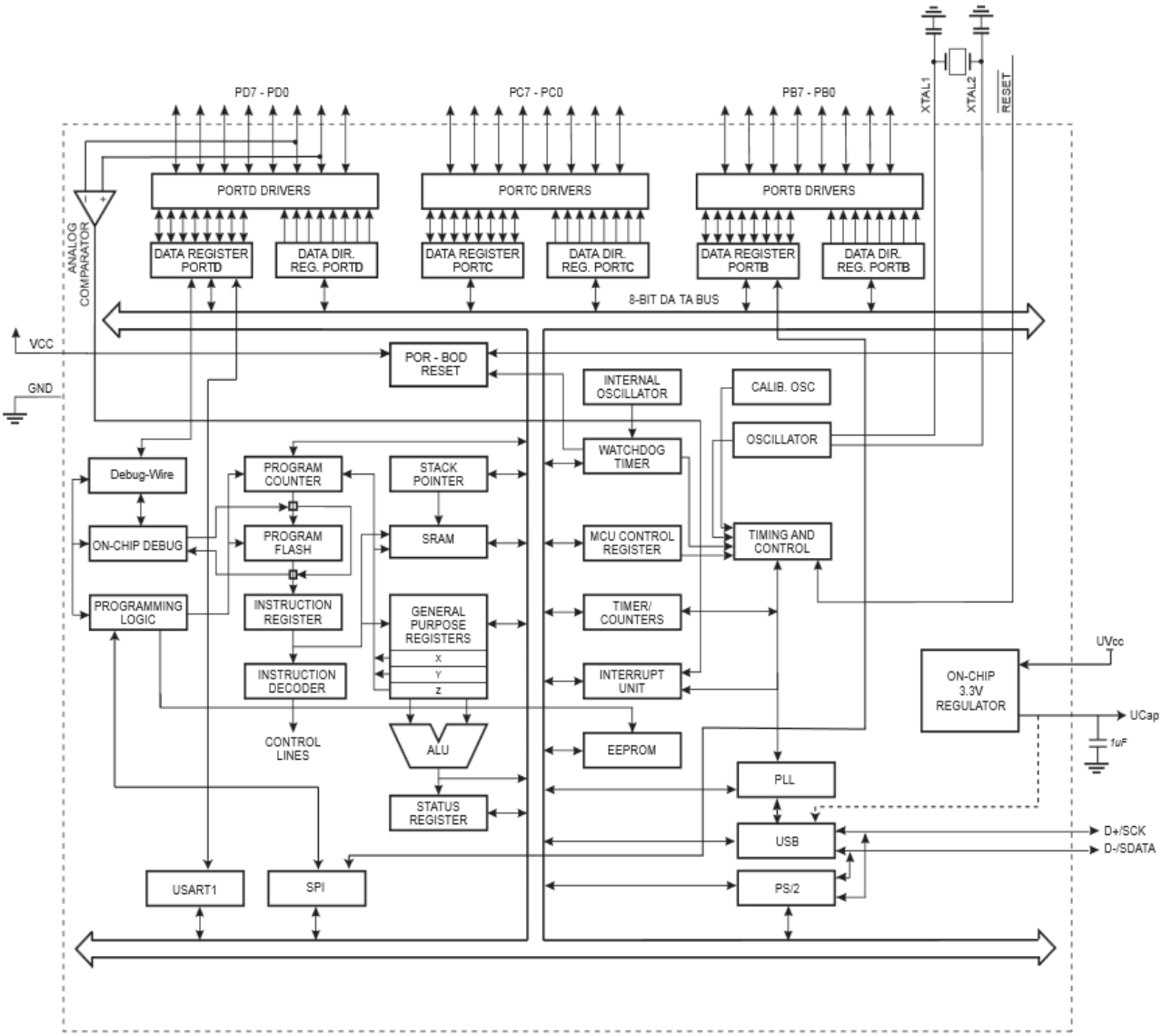
4.6.2 Περιορισμοί Τροφοδοσίας

Εάν μία εξωτερική πηγή τροφοδοσίας είναι συνδεδεμένη στο Vin, το διαθέσιμο ρεύμα από το pin των 5V εξαρτάται από την ενέργεια που καταναλώνεται από τον ενσωματωμένο ρυθμιστή (Regulator) που βρίσκεται στην πλακέτα μας, η οποία είναι το πολύ 1.5W. Χρησιμοποιώντας τον τύπο Watts = Current (I)×Volts, (W = I × V) καταλήγουμε στα παρακάτω συμπεράσματα. Αν τροφοδοτήσουμε το Vin με 7V τότε το διαθέσιμο ρεύμα θα είναι 750mA γιατί 1,5/(7-5) μας κάνει 750, αντίστοιχα στα 9V θα είναι 375 mA και στα 12V θα είναι 214 mA.

Το Atmega328 άλλα και άλλα ολοκληρωμένα καταναλώνουν περίπου 120mA, οπότε ότι έχει απομείνει για τις εξωτερικές συσκευές (LED, αισθητήρια, και λοιπές συσκευές που είναι συνδεδεμένες στο Arduino) είναι ο περιορισμός τις προηγούμενης παραγράφου μείον περίπου 120 mA. Αυτό σημαίνει ότι το διαθέσιμο ρεύμα στα 7V είναι 750-120=630 mA, στα 9V είναι 375-120=255mA και στα 12V είναι 214-120=94 mA.

4.6.3 Μνήμη

Το ATmega328 έχει συνολική μνήμη 32 KB εκ των οποίων τα 0.5 χρησιμοποιούνται από τον Bootloader. Έχει επίσης 2 KB SRAM και 1KB EEPROM η οποία μπορεί να διαβαστεί και να εγγραφεί μέσω της αντίστοιχης βιβλιοθήκης EEPROM library. Στην εικόνα 4-7 μπορούμε να διακρίνουμε τα μπλοκ της SRAM, της program Flash και της EEPROM του Atmega 16U2.



Εικόνα 4-6: ATmega16U2 Block Diagram

4.6.4 Είσοδοι και Έξοδοι

Κάθε ένα απ' τα 14 ψηφιακά pins του Uno μπορούν να χρησιμοποιηθούν ως είσοδοι η έξοδοι κάνοντας χρήση των συναρτήσεων pinMode(), digitalWrite(), και digitalRead(). Λειτουργούν στα 5V και έχουν τη δυνατότητα να πάρουν ή να δώσουν ένταση της τάξεως 40 mA το πολύ το καθένα. Σε κάθε Pin υπάρχει εσωτερικά ένας Pull-up αντιστάτης στα 20-50KΩ. Κάποια από αυτά έχουν ειδικές λειτουργίες :

- Serial: 0 (RX) και 1 (TX). Τα pins αυτά χρησιμοποιούνται για την αποστολή (TX) και την λήψη (RX) δεδομένων μέσω της TTL σειριακής θύρας. Αυτά τα pins είναι συνδεδεμένα και με το ATmega16U2 Usb to TTL Serial chip.
- External Interrupts: 2 και 3. Τα pins αυτά μπορούν να χρησιμοποιηθούν για την χρήση εξωτερικών διακοπών κάνοντας χρήση της συνάρτησης attachInterrupt().
- PWM: pin 3, 5, 6, 9, 10 και 11. Τα pins αυτά μπορούν να χρησιμοποιηθούν ως έξοδοι παλμών (8-bit PWM) με την χρήση της συνάρτησης analogWrite().
- SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). Τα pins αυτά επιτυγχάνουν την επικοινωνία μέσω SPI κάνοντας χρήση της SPI βιβλιοθήκης.
- LED: pin 13. Υπάρχει ένα ενσωματωμένο LED στο pin 13. Όταν υπάρχει λογικό 1, δηλαδή HIGH, το LED ανάβει, ενώ αν υπάρχει λογικό 0, δηλαδή LOW, το LED είναι σβηστό.

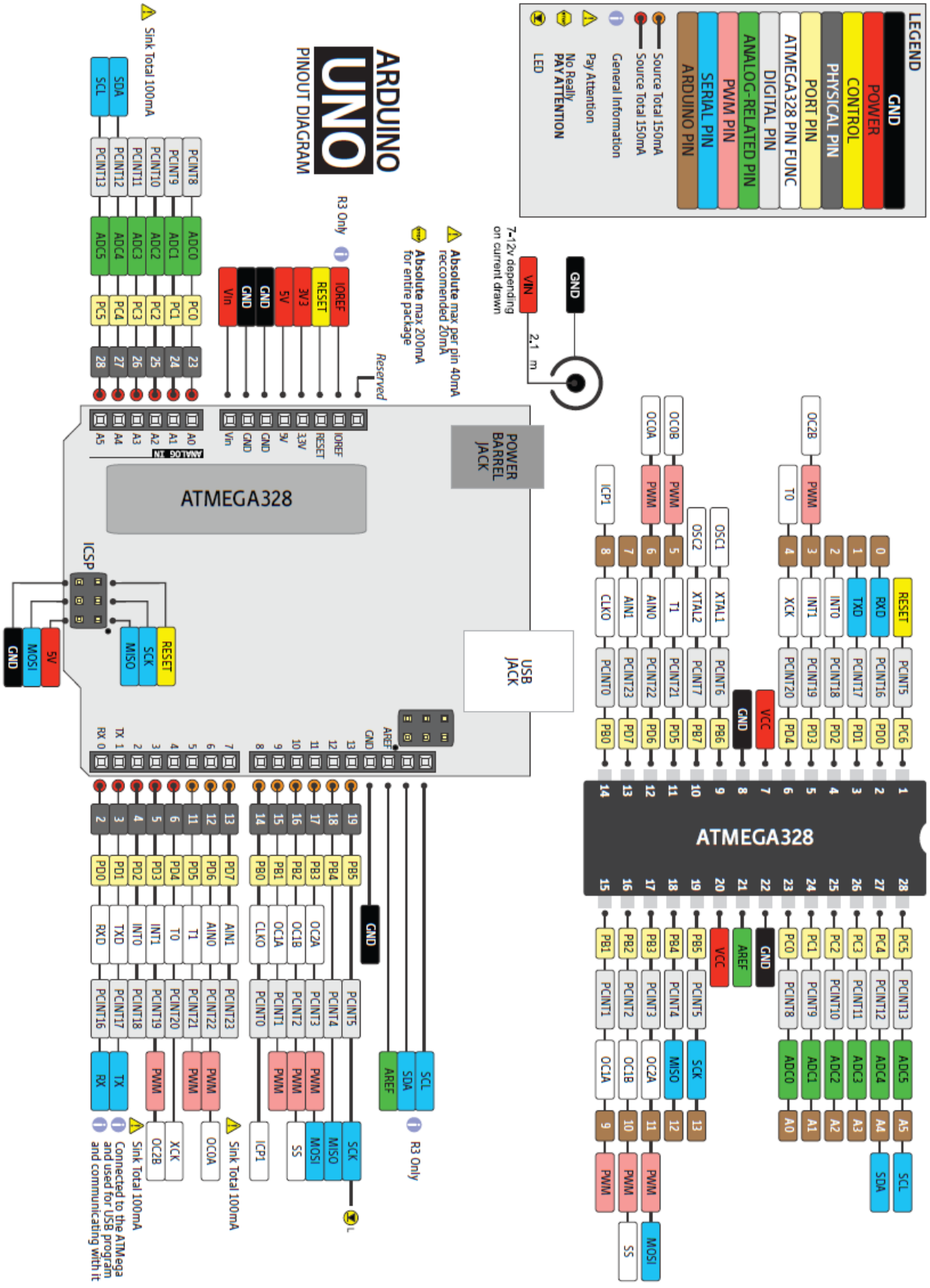
Το Uno έχει επίσης και 6 αναλογικές εισόδους οι οποίες διακρίνονται ως A0 έως A5, κάθε μία από αυτές μπορούν να πάρουν μέχρι 1024 διαφορετικές τιμές (10bits). Από προεπιλογή μπορούν να δώσουν τιμές από 0 (ground) έως 5V, τα όρια αυτά μπορούν να αλλάξουν με την χρήση του AREF pin και της συνάρτησης analogReference(). Κάποια από αυτά έχουν ειδικές λειτουργίες:

- TWI: A4 ή SDA pin και A5 ή SCL pin Χρησιμοποιούνται για την επικοινωνία TWI κάνοντας χρήση της Wire βιβλιοθήκης.

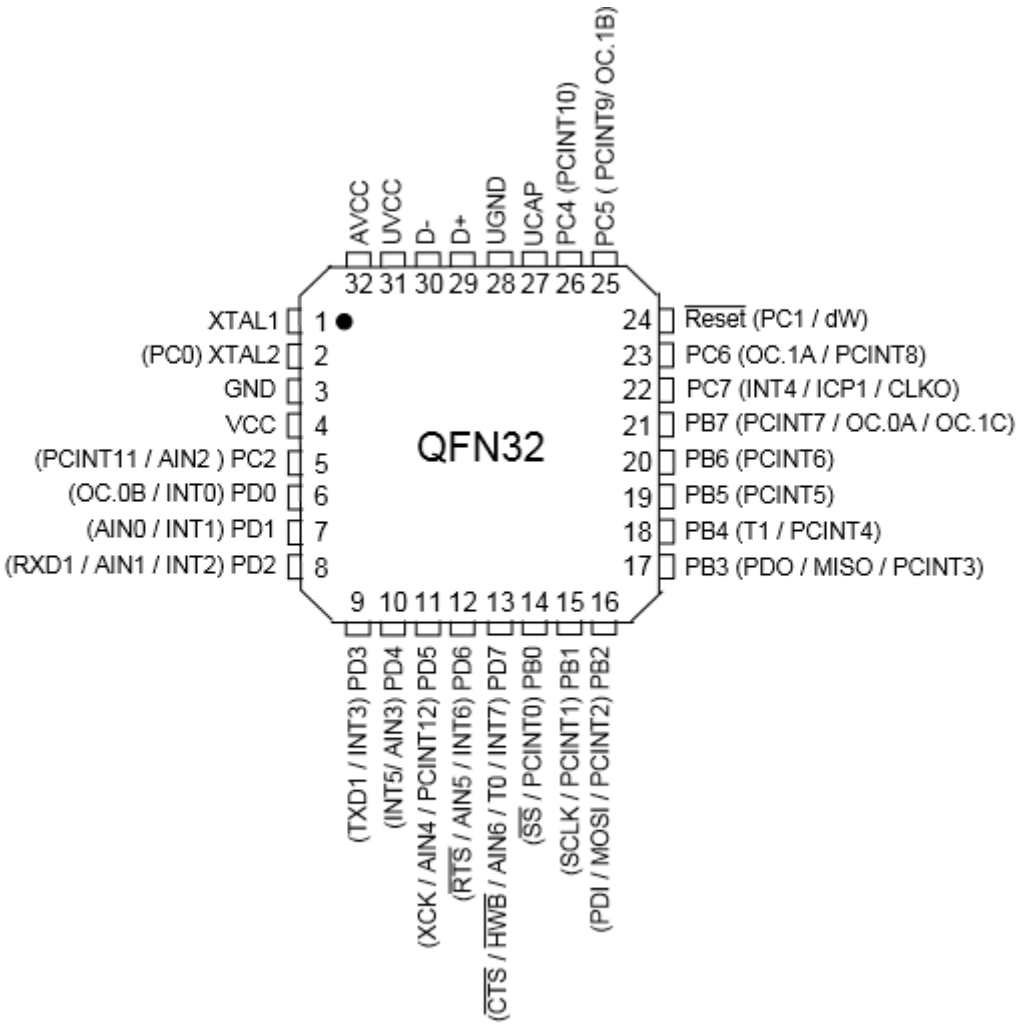
Υπάρχουν και κάποιοι άλλοι ακροδέκτες:

- AREF (Reference voltage for the analog inputs). Χρησιμοποιείται με την συνάρτηση analogReference() και μπορεί να πάρει ως είσοδο τάση αποκλειστικά από 0 έως 5V.
- Reset. Αν τεθεί σε κατάσταση LOW τότε επανεκκινεί τον Μικροελεγκτή. Σε αυτή τη γραμμή τοποθετείται ένας διακόπτης.

Στην εικόνα 4-8 παρουσιάζεται αναλυτικά η αντιστοίχηση όλων των ακροδεκτών του Arduino no R3.



Εικόνα 4-7: Arduino Uno R3 και ATmega 328 Pin Mapping



Εικόνα 4-8: ATmega16U2 Pin Mapping

4.6.5 Επικοινωνία και Προγραμματισμός

Το Arduino Uno έχει την δυνατότητα να επικοινωνεί με έναν Ηλεκτρονικό Υπολογιστή, ένα άλλο Arduino ή άλλους μικροελεγκτές. Το ολοκληρωμένο ATMega 328 παρέχει σειριακή επικοινωνία TTL 5Volt UART, η οποία είναι διαθέσιμη από τους ακροδέκτες 0 (RX) και 1 (TX) του ολοκληρωμένου. Επιπλέον στην αναπτυξιακή πλακέτα του Arduino είναι ενσωματωμένο ένα ολοκληρωμένο το ATmega16U2 το οποίο παρέχει σειριακή επικοινωνία με τον Ηλεκτρονικό Υπολογιστή για προγραμματισμό, μέσω της θύρας Usb.

Όπως αναφέρουμε και παραπάνω ο προγραμματισμός του Arduino γίνεται μέσω του λογισμικού που είναι διαθέσιμο από την Arduino . Το ATmega 328 έχει ενσωματωμένο Bootloader που μας επιτρέπει να φορτώσουμε κώδικα σε αυτόν χωρίς να είναι αναγκαιά η χρήση εξωτερικού προγραμματιστή. Η επικοινωνία του βασίζεται στο πρωτόκολλο STK500. Ωστόσο υπάρχει η δυνατότητα να παρακάμψουμε τον Bootloader και να προγραμματίσουμε τον μικροελεγκτή μας χρησιμοποιώντας το ICSP (In-Circuit Serial Programming) header.

4.6.6 TTL Λογικά Επίπεδα

Με απλά λόγια, ένα λογικό επίπεδο είναι μια συγκεκριμένη τάση ή μια κατάσταση την οποία μπορεί να έχει ένα σήμα (Εικόνα 4-10). Συχνά αναφερόμαστε στις δύο καταστάσεις σε ένα ψηφιακό κύκλωμα και ξέρουμε ότι είναι ON ή OFF. Η εκπροσώπησή τους στο δυαδικό, είναι το δυαδικό 1 για το ON και το δυαδικό 0 για το OFF. Στο Arduino, ονομάζουμε αυτά τα σήματα HIGH και LOW αντίστοιχα. Υπάρχουν διάφορες τεχνολογίες που έχουν εξελιχθεί τα τελευταία 30 χρόνια στο χώρο της ηλεκτρονικής για να καθορίσουν τα επίπεδα της τάσης.

Τα ψηφιακά ηλεκτρονικά βασίζονται στη δυαδική λογική για την αποθήκευση, την επεξεργασία και τη μετάδοση δεδομένων ή πληροφοριών. Η δυαδική λογική αναφέρεται σε μια από τις δύο καταστάσεις ON ή OFF, που

μεταφράζονται στο δυαδικό 1 και το δυαδικό 0. Το δυαδικό 1 αναφέρεται επίσης σε ένα υψηλό σήμα και το δυαδικό 0 σε ένα χαμηλό σήμα. Η ισχύς του σήματος περιγράφεται συνήθως από το επίπεδο της τάσης του. Το πώς ορίζεται ένα λογικό 0 (LOW) ή ένα λογικό 1 (HIGH) οι κατασκευαστές των τσιπ το ορίζουν αυτό στα φύλλα προδιαγραφών τους. Το πιο κοινό πρότυπο είναι το TTL ή Transistor-Transistor Logic.

Όταν δουλεύουμε με ολοκληρωμένα κυκλώματα και μικροεπεξεργαστές, θα αντιμετωπίσουμε πιθανόν ακροδέκτες που είναι active-low και ακροδέκτες που είναι active-high. Με απλά λόγια, αυτό περιγράφει πως έχει ενεργοποιηθεί ο ακροδέκτης. Αν είναι ένας active-low ακροδέκτης, θα πρέπει να τον συνδέσουμε με την γείωση. Ενώ ένας active-high ακροδέκτης συνδέεται σε υψηλή τάση (συνήθως 3.3V/5V). Πολλά ολοκληρωμένα κυκλώματα έχουν ακροδέκτες active-low και active-high μπερδεμένους. Απλά πρέπει να τσεκάρουμε καλά τα ονόματα των ακροδεκτών που έχουν μια γραμμή από πάνω τους. Η γραμμή χρησιμοποιείται για να αντιπροσωπεύσει το NOT. Όταν κάτι είναι σημειωμένο με αυτή την γραμμή, αλλάζει η κατάστασή του και γίνεται αντίθετη. Έτσι αν μια active-high είσοδος έχει γραμμή από πάνω, αλλάζει σε active-low.

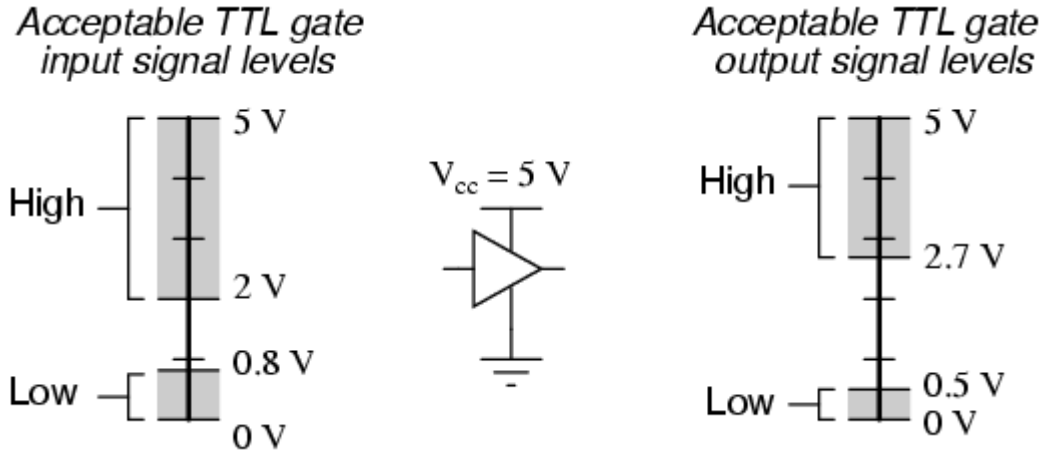
Για οποιαδήποτε λογική οικογένεια, υπάρχει μία σειρά από επίπεδα τάσης που πρέπει να γνωρίζουμε. Αυτά είναι :

- V_{OH}:Ελάχιστο επίπεδο τάσης εξόδου που πρέπει να παρέχει μία συσκευή TTL για ένα HIGH σήμα.
- V_{IH} :Ελάχιστο επίπεδο τάσης εισόδου για να θεωρείται ως HIGH.
- V_{OL}:Μέγιστο επίπεδο τάσης εξόδου που πρέπει να παρέχει μία συσκευή για ένα LOW σήμα.
- V_{IL} :Μέγιστο επίπεδο τάσης εισόδου για να εξακολουθεί να θεωρείται ως LOW.

Θα παρατηρήσουμε ότι το ελάχιστο επίπεδο τάσης HIGH εξόδου (V_{OH}) είναι 2.7V. Αυτό βασικά σημαίνει ότι η τάση εξόδου μίας συσκευής που θέλει να στείλει ένα HIGH σήμα πρέπει να είναι αναγκαστικά το λιγότερο 2.7V. Το ελάχιστο επίπεδο τάσης εισόδου (V_{IH}) είναι 2V, η πιο απλά, οποιαδήπωτε τάση η οποία είναι το λιγότερο 2V θα διαβάζεται ως λογικό 1 (HIGH) από μία TTL συσκευή. Επίσης παρατηρούμε ένα κενό της τάξεως των 0.7V που υπάρχει μεταξύ της εξόδου της μίας συσκευής και της εισόδου της άλλης. Αυτό συνήθως το ονομάζουμε θόρυβο (noise margin).

Ομοίως, το μέγιστο επίπεδο τάσης LOW εξόδου (V_{OL}) είναι 0.4V. Αυτό σημαίνει ότι μία συσκευή που προσπαθεί να στείλει ένα λογικό 0 θα έχει πάντα τάση μικρότερη των 0.4V. Το μέγιστο επίπεδο τάσης εισόδου LOW (V_{IL}) είναι 0.8. Οπότε οποιοδήποτε σήμα εισόδου με τάση μικρότερη των 0.8V θα θεωρείται λογικό 0 (LOW) όταν διαβάζεται από την συσκευή (Εικόνα 4-10).

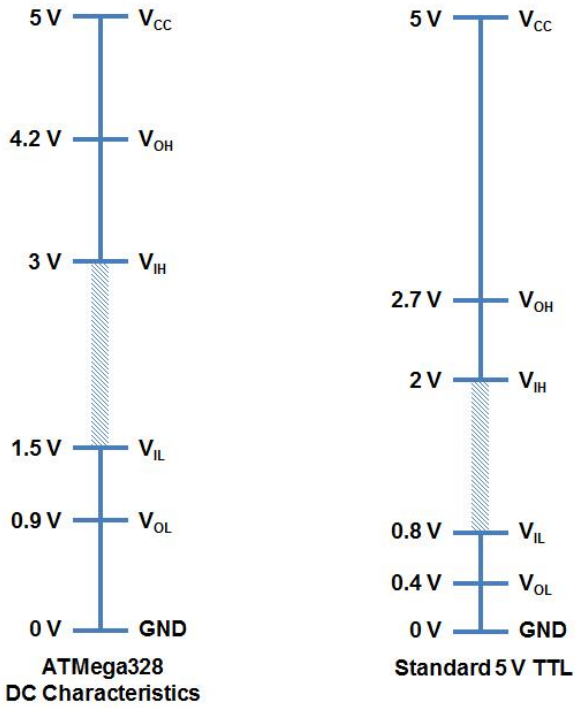
Στην περίπτωση που η τάση είναι ανάμεσα στα 0.8V και 2V, επειδή το φάσμα αυτό είναι απροσδιόριστο, οδηγούμαστε σε μη έγκυρη κατάσταση που συχνά αναφέρεται ως κυμαινόμενη. Μπορεί να προκαλέσει μέχρι και αυθαίρετη εναλλαγή μεταξύ των καταστάσεων HIGH και LOW.



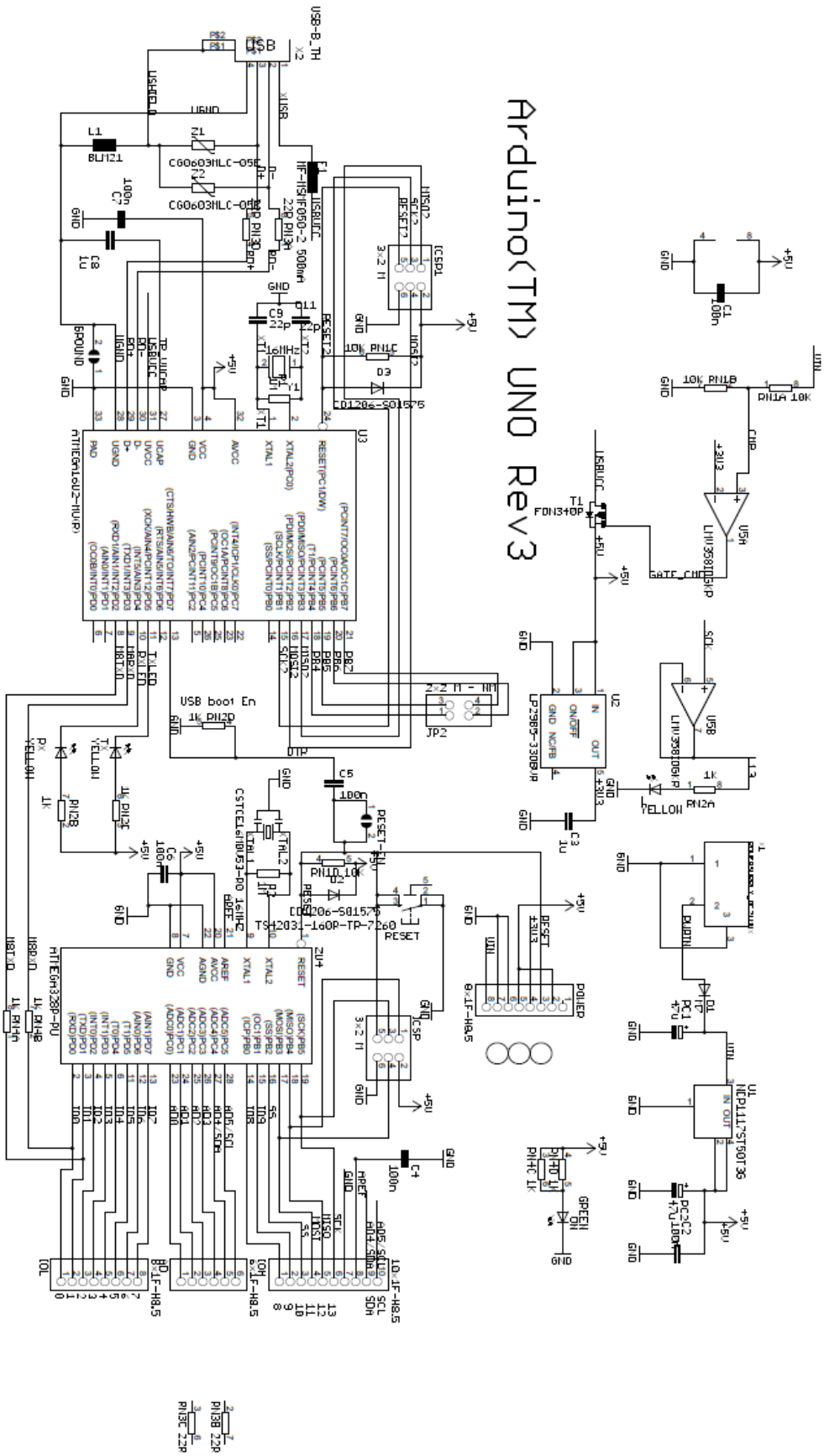
Εικόνα 4-9: TTL Gate Signal Levels

Κοιτάζοντας τα τεχνικά χαρακτηριστικά του Atmega328 θα παρατηρήσουμε ότι τα επίπεδα τάσης είναι ελαφρώς διαφορετικά (Εικόνα 4-11). Το Arduino είναι κατασκευασμένο σε μία ισχυρή πλατφόρμα. Η πιο αξιοσημείωτη διαφορά είναι ότι η άκυρη περιοχή των τάσεων κυμαίνεται μονάχα μεταξύ 1.5V και 3V. Τα όρια του θορύβου είναι

μεγαλύτερα για το Arduino και υπάρχει μεγαλύτερη πιθανότητα να προκύψει ένα LOW σήμα. Το γεγονός αυτό καθιστά την κατασκευή διεπαφών και τη συνεργασία με άλλα εξαρτήματα πολύ πιο απλή.



Εικόνα 4-10: Το ATmega328 TTL συγκριτικά με το Standard 5V TTL



Εικόνα 4-11: Το επίσημο σχηματικό διάγραμμα του Arduino Uno R3

http://arduino.cc/en/uploads/Main/Arduino_Uno_Rev3-schematic.pdf

4.7 Προγραμματίζοντας το Arduino

Το πρόγραμμα μας χωρίζεται σε τρία βασικά μέρη, την δομή (Structure), τις μεταβλητές και σταθερές (Variables και Constants, Πίνακας 4-2 και 4-3) και τις συναρτήσεις (Functions, Πίνακας 4-2 και 4-4). Οι συναρτήσεις αποτελούν βασικό παράγοντα στον προγραμματισμό του μικροελεγκτή καθώς αυτές είναι που συνδέουν το Arduino με τα υπόλοιπά στοιχεία του κυκλώματος όπως Led, κουμπιά, διακόπτες, οθόνες Lcd κ.α. Παρακάτω θα δούμε από ποιες εντολές αποτελείτε κάθε μέρος και θα εξηγήσουμε τις πιο βασικές από αυτές (πίνακας 4-2).

Πίνακας 4-2: Η βασική δομή ενός προγράμματος

Structure

- setup()
- loop()

Control Structures

- if
- if...else
- for
- switch case
- while
- do... while
- break
- continue
- return
- goto

Further Syntax

- ; (semicolon)
- {} (curly braces)
- // (single line comment)
- /* */ (multi-line comment)
- #define
- #include

Arithmetic Operators

- = (assignment operator)
- + (addition)
- - (subtraction)
- * (multiplication)
- / (division)
- % (modulo)

Comparison Operators

Variables Constants

- HIGH | LOW
- INPUT | OUTPUT | INPUT_PULLUP
- LED_BUILTIN
- true | false
- integer constants
- floating point constants

Data Types

- void
- boolean
- char
- unsigned char
- byte
- int
- unsigned int
- word
- long
- unsigned long
- short
- float
- double
- string - char array
- String - object
- array

Conversion

- char()
- byte()
- int()
- word()
- long()
- float()

Functions Digital I/O

- pinMode()
- digitalWrite()
- digitalRead()

Analog I/O

- analogReference()
- analogRead()
- analogWrite() - *PWM*

Due only

- analogReadResolution()
- analogWriteResolution()

Advanced *I/O*

- tone()
- noTone()
- shiftOut()
- shiftIn()
- pulseIn()

Time

- millis()
- micros()
- delay()
- delayMicroseconds()

Math

- min()
- max()
- abs()
- constrain()
- map()

- == (equal to)
- != (not equal to)
- < (less than)
- > (greater than)
- <= (less than or equal to)
- >= (greater than or equal to)

Boolean Operators

- && (and)
- || (or)
- ! (not)

Pointer Access Operators

- * dereference operator
- & reference operator

Bitwise Operators

- & (bitwise and)
- | (bitwise or)
- ^ (bitwise xor)
- ~ (bitwise not)
- << (bitshift left)
- >> (bitshift right)

Compound Operators

- ++ (increment)
- -- (decrement)
- += (compound addition)
- -= (compound subtraction)
- *= (compound multiplication)
- /= (compound division)
- &= (compound bitwise and)
- |= (compound bitwise or)

Variable Scope & Qualifiers

- variable scope
- static
- volatile
- const

Utilities

- sizeof()

- pow()
- sqrt()

Trigonometry

- sin()
- cos()
- tan()

Random Numbers

- randomSeed()
- random()

Bits and Bytes

- lowByte()
- highByte()
- bitRead()
- bitWrite()
- bitSet()
- bitClear()
- bit()

External Interrupts

- attachInterrupt()
- detachInterrupt()

Interrupts

- interrupts()
- noInterrupts()

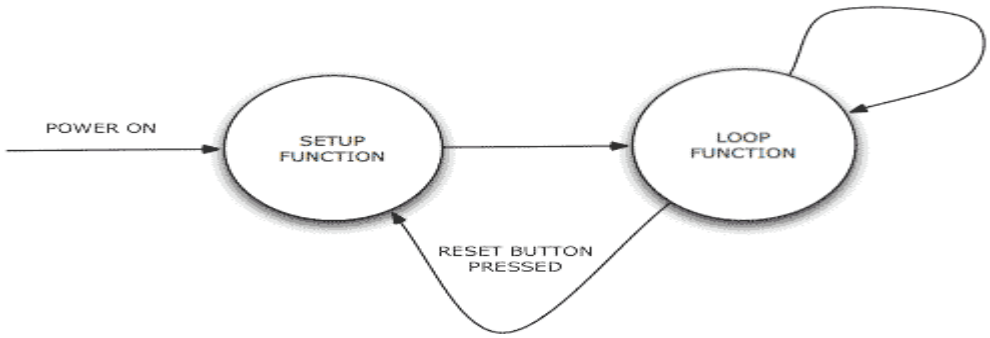
Communication

- Serial
- Stream

USB (Leonardo and Due only)

- Keyboard
- Mouse

Η συνάρτηση setup() καλείται μια φορά όταν το sketch ξεκινά ή μετά από κάποιο Reset του Arduino Uno. Σε αυτήν γίνονται κάποιες βασικές λειτουργίες όπως αρχικοποίηση μεταβλητών, ρύθμιση της κατάστασης των pin (pin mode) και η προετοιμασία των βιβλιοθηκών. Αντίθετα η συνάρτηση loop() καλείται ξανά και ξανά επιτρέποντας στο πρόγραμμα μας να ανταποκρίνεται σε εξωτερικές αλλαγές. Τόσο η setup() όσο και η loop() πρέπει να περιλαμβάνονται αναγκαστικά στο sketch και ας μην περιέχουν κώδικα στο εσωτερικό τους (Εικόνα 4-13).



Εικόνα 4-12: Arduino Flow Chart

Πίνακας 4-3: Οι σταθερές

Όρισμα	Τύπος	Παράμετροι	Περιγραφή
Low	Int	Δεν υπάρχουν	Έχει την τιμή 0 και είναι αντίστοιχη του λογικού false.
High	Int	Δεν υπάρχουν	Έχει την τιμή 1 και είναι αντίστοιχη του λογικού true.
Input	Int	Δεν υπάρχουν	Έχει την τιμή 0 και είναι αντίστοιχη του λογικού false.
Output	Int	Δεν υπάρχουν	Έχει την τιμή 1 και είναι αντίστοιχη του λογικού true.

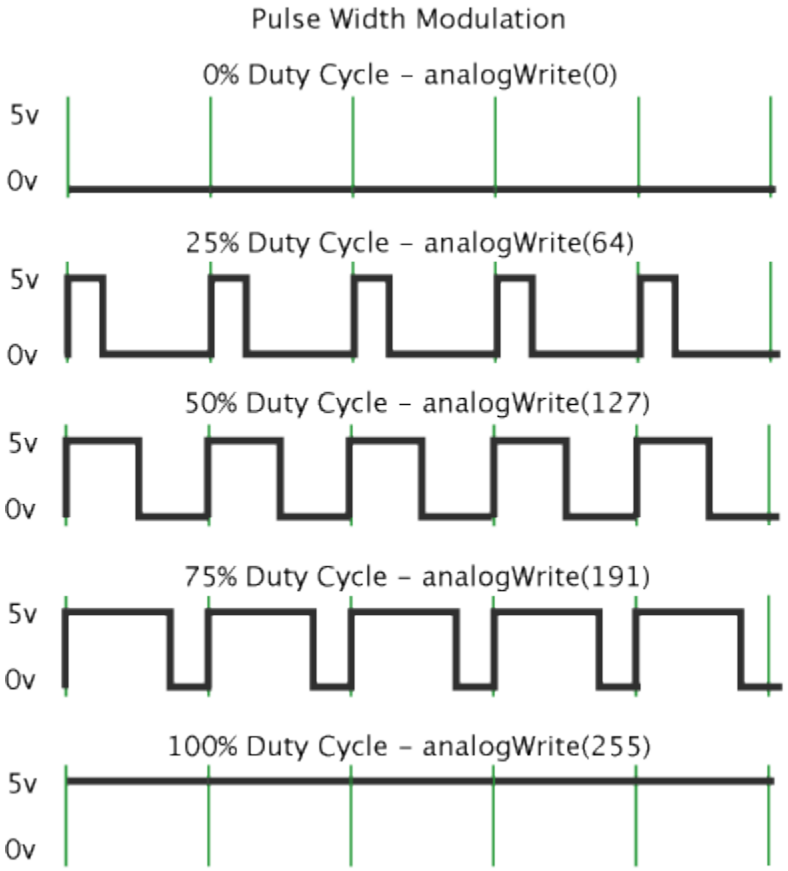
Πίνακας 4-4: Οι συναρτήσεις

Όρισμα	Τύπος	Παράμετροι	Περιγραφή
pinMode	..	(pin,mode)	Καθορίζει αν το συγκεκριμένο pin, θα είναι εισόδου ή εξόδου, εάν δηλαδή θα λάβει η θα στείλει κάποιο σήμα, ανάλογα με την τιμή που δίνεται στο mode (INPUT/OUTPUT) π.χ `pinMode(13, OUTPUT)`.
digitalWrite	..	(pin,status)	Θέτει τη κατάσταση pinStatus (High ή Low) στο συγκεκριμένο ψηφιακό pin π.χ `digitalWrite(13, LOW)`.
digitalRead	..	(pin)	Επιστρέφει τη κατάσταση του συγκεκριμένου ψηφιακού pin (0 για Low και 1 για High), εφόσον αυτό είναι pin εισόδου π.χ `digitalRead(13)`.
analogRead	..	(pin)	Την συνάρτηση αυτή την χρησιμοποιούμε όταν θέλουμε να διαβάσουμε μια αναλογική τιμή από ένα αισθητήριο π.χ αισθητήρα θερμοκρασίας, υπερήχων, υπέρυθρων. Επιστρέφει έναν ακέραιο από 0 έως 1023, ανάλογα με την τάση που τροφοδοτείται το συγκεκριμένο pin αναλογικής εισόδου. Τα διαθέσιμα analog pins του arduino Uno είναι από A0 εως A5 π.χ `val = analogRead(analogPin).`
analogWrite	..	(pin,value)	Θέτει το συγκεκριμένο pin σε κατάσταση ψευδοαναλογικής εξόδου (PWM). Η παράμετρος value καθορίζει το πλάτος του παλμού σε σχέση με την περίοδο του παραγόμενου σήματος, σε κλίμακα από 0 εως 255. Η λειτουργία αυτή λέγεται pulse width modulation και στο Arduino μπορούμε να την χρησιμοποιήσουμε στα pin 3, 5, 6, 9, 10 και 11. Αυτό που κάνει είναι να στέλνει παλμούς, μπορούμε να το φανταστούμε σαν ένα διακόπτη που ανοιγοκλείνει πολύ γρήγορα. Αν π.χ θε-

			λήσουμε να περιστρέψουμε ένα μοτέρ με την μισή ταχύτητα θα κάνουμε χρήση της συνάρτησης ως εξής `analogWrite(3,127)` και αντίστοιχα με την μέγιστη ταχύτητα `analogWrite(3,255)`.
delay	..	(time)	Σταματά προσωρινά τη ροή του προγράμματος για όσο ορίζει η παράμετρος time, η οποία ορίζεται ως unsigned long (0-8192).
Millis	..	..	Μετρητής που επιστρέφει το χρονικό διάστημα σε ms από την στιγμή που άρχισε η εκτέλεση του προγράμματος. Να λάβουμε υπόψη ότι λόγω του τύπου μεταβλητής (unsigned long δηλ. 32bit) θα γίνει overflow σε 2^{32} ms δηλαδή περίπου σε 50 μέρες, οπότε ο μετρητής θα ξεκινήσει πάλι από το μηδέν.
analogReference	..	(type)	Δέχεται τις τιμές DEFAULT, INTERNAL ή EXTERNAL στην παράμετρο `type` για να καθορίσει την τάση αναφοράς (V_{ref}) των αναλογικών εισόδων (5V, 1.1V ή η εξωτερική τάση με την οποία τροφοδοτείται το pin AREF αντίστοιχα).
attachInterrupt	..	(interrupt, function, triggermode)	Θέτει σε λειτουργία το συγκεκριμένο `interrupt`, ώστε να ενεργοποιεί την συνάρτηση `function`, κάθε φορά που ικανοποιείται η συνθήκη που ορίζεται από την παράμετρο `triggermode`: • LOW (ενεργοποίηση όταν η κατάσταση του pin που αντιστοιχεί στο συγκεκριμένο interrupt γίνει LOW). • RISING (όταν από LOW γίνει HIGH). • FALLING (όταν από HIGH γίνει LOW). • CHANGE (όταν αλλάξει κατάσταση γενικά).
DetachInterrupt	..	(interrupt),(pin)	Απενεργοποιεί το συγκεκριμένο `interrupt`.
noInterrupts	..	..	Σταματά προσωρινά την λειτουργία όλων των `interrupt`.
Interrupts	..	..	Επαναφέρει την λειτουργία των `interrupt` που διακόπηκε προσωρινά από μια εντολή noInterrupts.

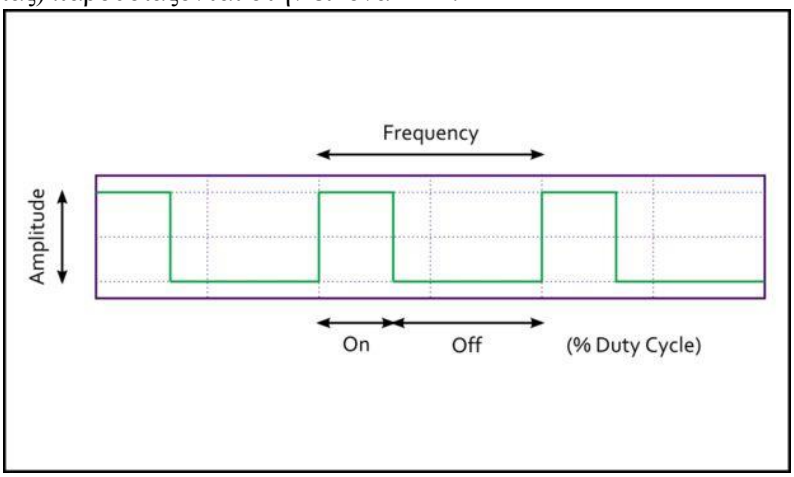
Πίνακας 4-5: Μέθοδοι Κλάσης

Όρισμα	**Τύπος**	**Παράμετροι**	**Περιγραφή**
Serial.begin	..	(baudrate)	Θέτει το ρυθμό μεταφοράς δεδομένων της σειριακής επικοινωνίας (σε baud) π.χ `Serial.begin(9600)`.
Serial.print	..	(data)	Διοχετεύει τα δεδομένα data για αποστολή μέσω της σειριακής επικοινωνίας. Η παράμετρος data είναι είτε αριθμός είτε αλφαριθμητικό π.χ `Serial.print(x, DEC)`.



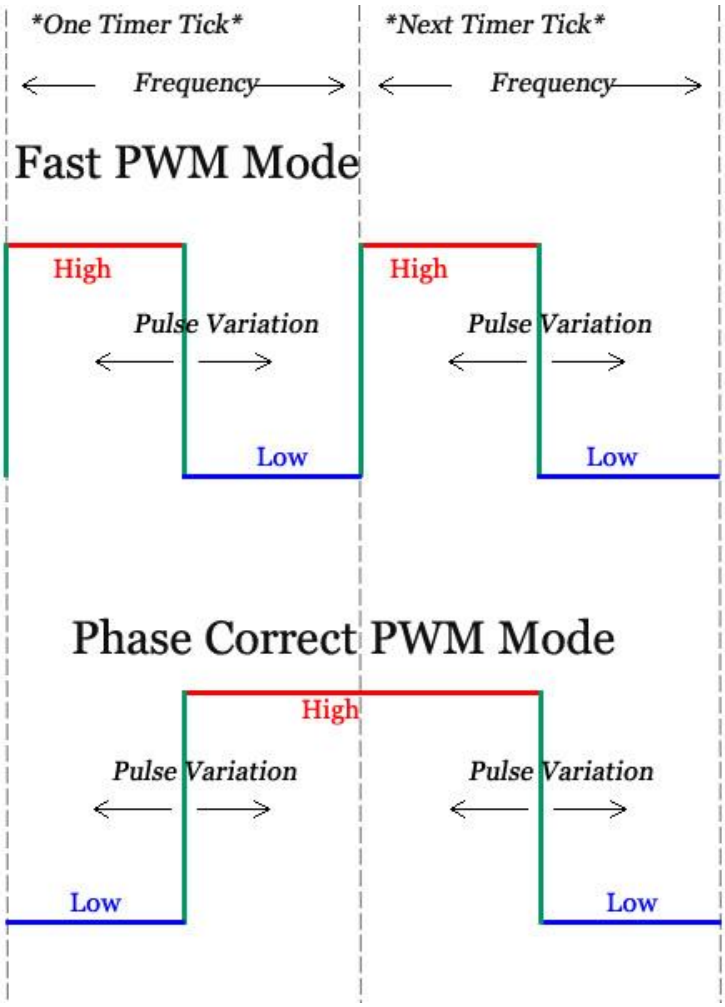
Εικόνα 4-13: Pulse Width Modulation (PWM)

Στην εικόνα 4-13 διακρίνουμε τον τρόπο λειτουργίας του PWM (Pulse Width Modulation), όπως ήδη αναφέραμε στον πίνακα 4-4 στην συνάρτηση analogWrite. Μία PWM κυματομορφή αποτελεί μία περιοδική κυματομορφή με δύο τμήματα. Το τμήμα ON στο οποίο η κυματομορφή έχει την μέγιστη τιμή της και το τμήμα OFF στο οποίο έχει την τιμή 0. Το ON τμήμα ονομάζεται Duty Cycle και μετριέται είτε σε μονάδες χρόνου είτε σε ποσοστό επί της περιόδου. Εφαρμόζοντας μία PWM κυματομορφή στην τροφοδοσία ενός φορτίου επιτυγχάνουμε να ελέγξουμε το ποσοστό της ισχύος που πέφτει πάνω στο φορτίο. Στην περίπτωση που το φορτίο είναι ένας κινητήρας, όπως στην δική μας περίπτωση, αυτό συνεπάγεται τον έλεγχο των στροφών του κινητήρα. Στην παραπάνω εικόνα βλέπουμε το Duty Cycle σε ποσοστό επί της εκατό 0, 25, 50, 75 και 100 αντίστοιχα. Τα τρία βασικά χαρακτηριστικά του PWM (πλάτος, συχνότητα και κύκλος λειτουργίας) παρουσιάζονται στην εικόνα 4-14.



Εικόνα 4-14: Τα βασικά χαρακτηριστικά του PWM

Υπάρχει άλλος ένας τρόπος για την δημιουργία PWM σήματος, αυτός αφορά την πρόσβαση στον PWM καταχωρητή του μικροελεγκτή με άμεσο τρόπο. Σε αυτή την περίπτωση μπορούμε να αλλάξουμε την ανάλυση και το είδος του PWM (γρήγορη λειτουργία PWM, λειτουργία διόρθωσης φάσης ή λειτουργία διόρθωσης φάσης και συχνότητας). Ο τρόπος αυτός παρουσιάζεται στην εικόνα 4-15.



Εικόνα 4-15: Access the MCU's PWM register in a direct way

5 Πειραματικό Μέρος

Στο προηγούμενο κεφάλαιο παρουσιάσαμε το Arduino, την υπολογιστική πλατφόρμα που χρησιμοποιούμε για την κατασκευή και τον έλεγχο του τηλεχειριζόμενου οχήματος. Τώρα, σε αυτό το κεφάλαιο θα μιλήσουμε για τον υπόλοιπο εξοπλισμό που χρειαζόμαστε, θα κάνουμε σημαντικές αναφορές σε θεωρητικό επίπεδο, για τον τρόπο λειτουργίας τους και την τεχνολογία που χρησιμοποιούν.

Αυτός αποτελείται από το σασί, μοντέλο ZK-4WD (Εικόνα 5-1), το L298N Dual H-Bridge Motor Driver (Εικόνα 5-5), που χρησιμοποιείται για την λειτουργία των DC μοτέρ, τον αισθητήρα απόστασης Ultrasonic Ranging Sensor HC-SR04 (Εικόνα 5-10), τον σερβοκινητήρα Tower Pro Micro SG90 (Εικόνα 5-13), το Bluetooth module HC-06 (Εικόνα 5-6), την LCD οθόνη τύπου Hitachi HD44780 (Εικόνα 5-8), τις μπαταριές που χρειαστήκαμε, 14 AA στο σύνολο (χρησιμοποιούμε αλκαλικές μπαταρίες αλλά θα μπορούσαν να χρησιμοποιηθούν και λιθίου για καλύτερη απόδοση), και τις βάσεις στις οποίες τοποθετήθηκαν (Battery Holders, Εικόνα 5-15). Επίσης χρησιμοποιούμε δύο Toggle διακόπτες στην τροφοδοσία του Arduino και του Motor Shield αντίστοιχα. Δύο κλέμες (Screw Terminals), αρκετοί πλαστικοί αποστάτες (Standoffs), πολλά καλώδια κ.α. Θα παρουσιάσουμε τον τρόπο που συνδέεται κάθε ένα ξεχωριστά με το Arduino δίνοντας σχηματικά παραδείγματα και εξηγώντας την λειτουργία τους. Προς το τέλος του κεφαλαίου θα ενώσουμε τα κομμάτια του παζλ και η κατασκευή μας, το τηλεχειριζόμενο όχημα, θα ολοκληρωθεί.

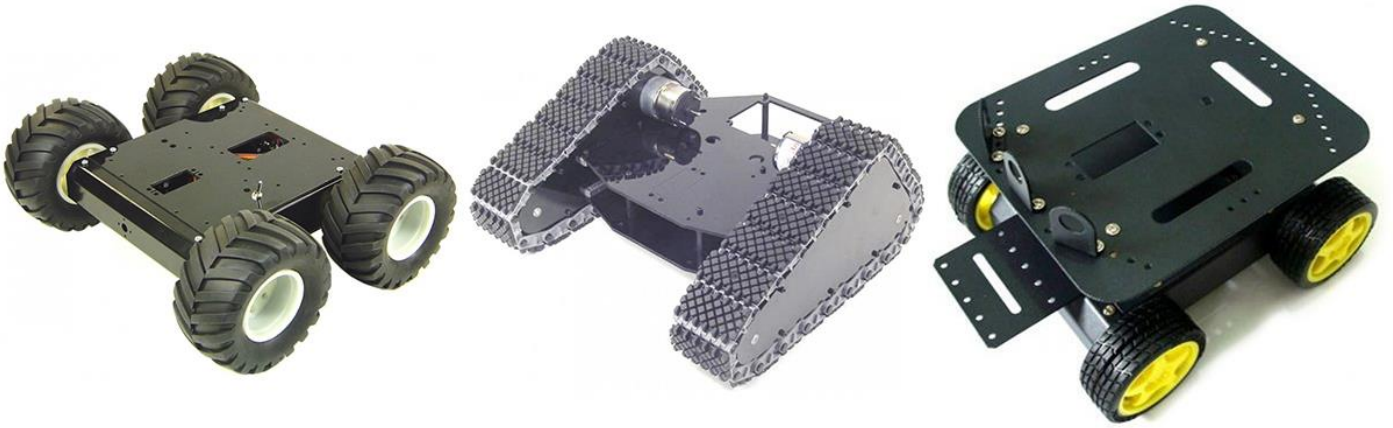
5.1 Το Σασί του Οχήματος

Το μοντέλο του σασί μας είναι το ZK-4WD με διαστάσεις 155mm (W) x 260mm (L) και αποτελεί την πλατφόρμα πιάνω στην οποία θα τοποθετήσουμε όλο τον εξοπλισμό που χρησιμοποιούμε (Εικόνα 5-1). Είναι κατασκευασμένο από δύο κομμάτια Plexiglass, κάτω και επάνω μέρος, τα οποία συνδέονται μεταξύ τους με την χρήση αποστατών μήκους 30 mm . Στο ενδιάμεσο βρίσκονται τέσσερα DC μοτέρ, ένα για κάθε τροχό του οχήματος μας. Επίσης σε κάθε ένα τροχό υπάρχει προσαρμοσμένος ένας κωδικοποιητής στροφών (rotary encoder) που σε συνδυασμό με ένα IR infrared αισθητήριο ταχύτητας μπορεί να μας επιστρέψει την ταχύτητα περιστροφής του κάθε τροχού, εμείς δεν το χρησιμοποιούμε αυτό. Το plexiglass είναι διάτρητο πράγμα που μας βοηθά στην τοποθέτηση του εξοπλισμού μας επάνω του κυρίως με την χρήση αποστατών (μήκους 5mm), αλλά μπορούμε εφόσον χρειάζεται να το τρυπήσουμε και να δημιουργήσουμε νέα ή περισσότερα ανοίγματα σε σημεία που μας βολεύουν καλύτερα. Στον παρακάτω πίνακα παρουσιάζονται κάποια τεχνικά χαρακτηριστικά του συγκεκριμένου μοντέλου.

Πίνακας 5-1: Τεχνικά χαρακτηριστικά του ZK-4WD

Τάση	DC 3V	DC 5V	DC 6V
Ρεύμα	100 MA	100MA	120MA
Ποσοστό μείωσης (Reduction rate)		48:1	
Στροφές ανά λεπτό (RPM)	100	190	240
Διάμετρος ελαστικού		66mm	
Ταχύτητα οχήματος (Μ/λεπτό)	20	39	48
Βάρος μοτέρ (g)		50	
Μέγεθος μοτέρ		70mm*22mm*18mm	
Θόρυβος		<65dB	

Εικόνα 5-1: Το σασί που χρησιμοποιούμε, πριν και μετά την συναρμολόγηση



Εικόνα 5-2: Άλλες διαθέσιμες πλατφόρμες για το Arduino

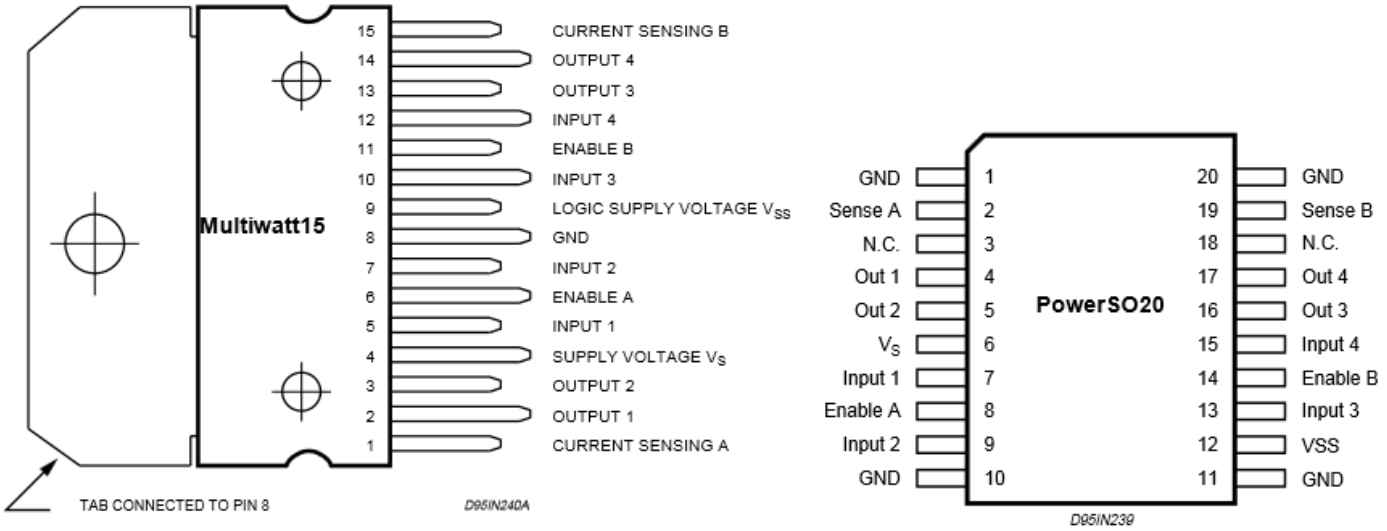
5.2 Το Ολοκληρωμένο L298N

Το L298N, το οποίο κατασκευάζεται από την εταιρία STMicroelectronics, είναι ένα ολοκληρωμένο μονολιθικό κύκλωμα που έρχεται σε δύο κατηγορίες οι οποίες είναι η 15-lead Multiwatt και η POWERSO20 (Εικόνα 5-3). Είναι ένας οδηγός διπλής γέφυρας (Dual H-Bridge Driver) υψηλής τάσης και ρεύματος που έχει σχεδιαστεί για να δέχεται TTL λογικά επίπεδα και για να οδηγεί επαγωγικά φορτία, όπως ρελέ, πηνία, DC και βηματικούς κινητήρες (Stepper Motors). Δύο είσοδοι παρέχονται για να ενεργοποιήσουμε τη συσκευή ανεξάρτητα από τα σήματα εισόδου. Οι εκπομποί των κατώτερων τρανζίστορ κάθε γέφυρας συνδέονται μαζί και το αντίστοιχο εξωτερικό τερματικό μπορεί να χρησιμοποιηθεί για τη σύνδεση ενός εξωτερικού αισθητηρίου πίεσης (Sensing resistor). Μία επιπλέον είσοδος τροφοδοσίας παρέχεται έτσι ώστε το λογικό κύκλωμα να λειτουργεί σε χαμηλότερη τάση. Στον παρακάτω πίνακα παρουσιάζονται βασικά χαρακτηριστικά του ολοκληρωμένου.

Εικόνα 5-3: L298N(Multiwatt15 Vertical), L298P(POWERSO20) και L298HN(Multiwatt15 Horizontal)

Πίνακας 5-2: Χαρακτηριστικά του L298N

Κατασκευαστής	STMicroelectronics
Τύπος	H-Bridge
Τάση λειτουργίας Vs	4.8 V εως 46 V
Λογική τάση τροφοδοσίας Vss	7V
Συνολικό ρεύμα DC	Έως 4A
Παροχή ρεύματος Is	13 mA έως 22 mA
Θερμοκρασία λειτουργίας	- 25 C μέχρι + 130 C
Συνολική κατανάλωση ισχύος	25W
Στυλ τοποθέτησης	Through Hole
Κατηγορία	Multiwatt-15V
Αριθμός εξόδων	4



Εικόνα 5-4: Multiwatt15 και POWERSO20 pin connections

5.2.1 Dual H-Bridge Motor Driver

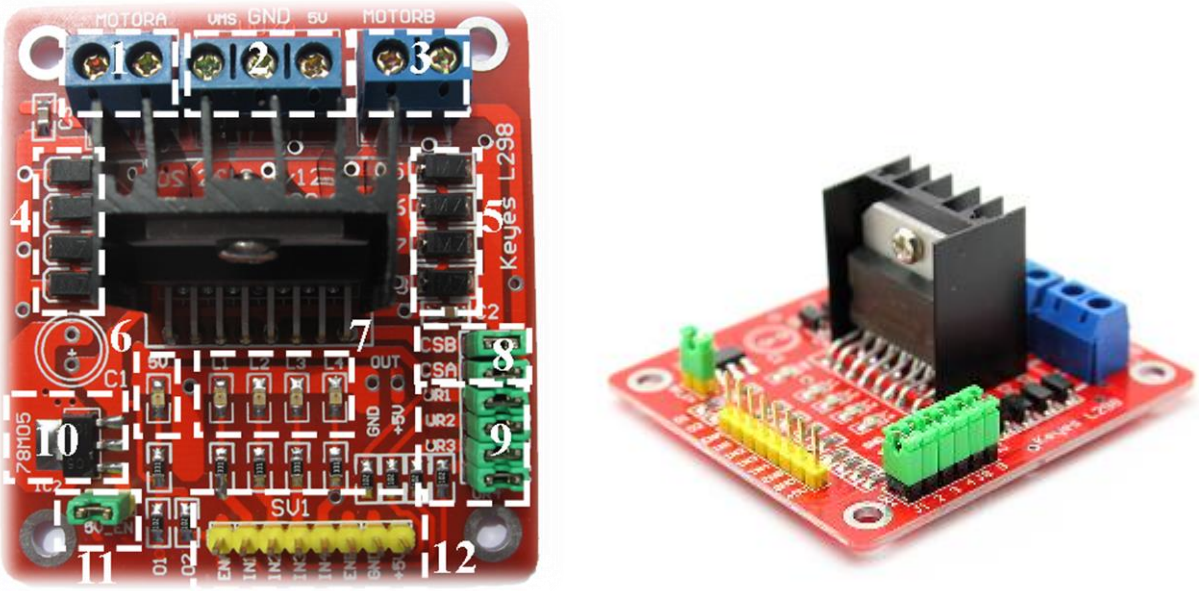
Το Motor Shield που χρησιμοποιούμε βασίζεται στο ολοκληρωμένο L298N για το οποίο μιλήσαμε προηγουμένως και χρησιμοποιείτε για τον έλεγχο οποιουδήποτε τύπου ρομποτικού μηχανισμού που περιλαμβάνει είτε DC είτε stepper (βηματικό) μοτέρ. Συγκεκριμένα με αυτό το Motor Shield μπορούμε να θέσουμε σε λειτουργία είτε δύο DC είτε ένα stepper μοτέρ. Η ταχύτητα και η περιστροφή κάθε DC μοτέρ ελέγχεται ξεχωριστά.

Είναι ελαφρύ με μικρές διαστάσεις, όλες οι γραμμές προστατεύονται από Back-EMF (Counter Electromotive Force). Μία ψήκτρα βοηθάει το L298N να διατηρήσει χαμηλή θερμοκρασία. Επίσης πάνω στην πλακέτα υπάρχουν 4 Leds (L1, L2, L3 και L4) που μας ενημερώνουν για την κατεύθυνση κάθε μοτέρ ξεχωριστά. Άλλο ένα Led (5V Led) ανάβει όταν ως τάση εξόδου έχουν επιλεγεί τα 5V. Η τοποθέτηση του πάνω στο σασί μας θα γίνει εύκολα χρησιμοποιώντας τέσσερις αποστάτες των 5mm.

Πίνακας 5-3: Χαρακτηριστικά του Dual H-Bridge Motor Driver

Ολοκληρωμένο	L298N
Τροφοδοσία	+6 ~ +35V
Μέγιστο ρεύμα εξόδου ανά κανάλι	2A

Λογική τάση εξόδου Vss	+5~+7V (Εσωτερική τροφοδοσία +5V)
Λογικό ρεύμα λειτουργίας	0~36mA
Έλεγχος επιπέδου	Low -0.3V~1.5V, high: 2.3V~Vss
Ενεργοποίηση επιπέδου σήματος	Low -0.3V~1.5V, high: 2.3V~Vss
Μέγιστη ισχύς κίνησης	25W (Temperature 75 ℃)
Θερμοκρασία λειτουργίας	-25℃~+130℃
Διαστάσεις	60mm*54mm
Βάρος	~48g



Εικόνα 5-5: L298N Dual H-Bridge Motor Driver

Στην εικόνα 5-5 παρουσιάζεται το L298N Dual H-Bridge Motor driver με αριθμημένα τα βασικά του στοιχεία τα οποία και θα εξηγήσουμε. Με τον αριθμό (1) βλέπουμε την έξοδο για το πρώτο DC μοτέρ (MOTORA) και αντίστοιχα με τον αριθμό (3) την έξοδο για το δεύτερο DC μοτέρ (MOTORB). Στον αριθμό (2) αντιστοιχούν τα VMS, GND και 5V. Στο VMS θα συνδέσουμε τον θετικό πόλο της πηγής μας και στο GND τον αρνητικό, δεξιά από τη γείωση βλέπουμε μία έξοδο των 5V. Με τον αριθμό (4) και (5) βλέπουμε κάποιες διόδους οι οποίες χρησιμοποιούνται για προστασία από την υπέρταση του ρεύματος (FWD, FreeWheeling Diode Protection). Με τον αριθμό (6) το Led με την ένδειξη λειτουργίας στα 5V και με τον αριθμό (7) τα ενδεικτικά Led L1, L2, L3 και L4 τα οποία μας ενημερώνουν για την κατεύθυνση του κάθε μοτέρ ξεχωριστά. Το CSA αριθμός (8) χρησιμοποιείται για τον έλεγχο του ηλεκτρικού ρεύματος της γέφυρας Α και το CSB αντίστοιχα για τον έλεγχο του ηλεκτρικού ρεύματος στην γέφυρα Β. Στο (9) βρίσκονται αντιστάτες ανάτασης (Pull-Up Resistors) τους οποίου ενεργοποιούμε η απενεργοποιούμε με την χρήση των jumpers (βραχυκυκλωτήρες). Το LM78M05, αριθμός (10), είναι ένας ρυθμιστής τάσης με έξοδο 5V. Ο Βραχυκυκλωτήρας στον αριθμό (11) ενεργοποιεί το LM78M05. Τέλος, στο (12) έχουμε ξεκινώντας από αριστερά, το ENA που αποτελεί TTL συμβατή είσοδο για την ενεργοποίηση της γέφυρας Α, αν λοιπόν την θέσουμε σε κατάσταση HIGH θα ενεργοποιήσουμε την γέφυρα Α που αποτελείται από τις TTL συμβατές εισόδους IN1 και IN2, τις οποίες αν θέσουμε σε κατάσταση IN1=HIGH και IN2=LOW η φορά περιστροφής θα είναι δεξιόστροφη, ενώ αντίθετα αν IN1=LOW και IN2=HIGH η φορά περιστροφής θα είναι αριστερόστροφη. Το ίδιο ακριβώς ισχύει για το ENB και τις TTL λογικές εισόδους IN3 και IN4. Στον πίνακα 5-4 παρουσιάζεται μία σύνοψη των παραπάνω στοιχείων.

Πίνακας 5-4: Βασικά στοιχεία του L298N Dual H-Bridge Motor Driver

VMS και GND	Σύνδεση σε εξωτερική πηγή τροφοδοσίας 6V-35V
Βραχυκυκλωτήρας 5V	Αν είναι συνδεδεμένος ενεργοποιεί το LM78M05

5V και +5V	Έξοδοι 5V
MOTORA	Έξοδος της γέφυρας A
MOTORB	Έξοδος της γέφυρας B
ENA	TTL συμβατή είσοδο ενεργοποίησης της γέφυρας A
IN1	TTL συμβατή είσοδο για την γέφυρα A
IN2	TTL συμβατή είσοδο για την γέφυρα A
ENB	TTL συμβατή είσοδο ενεργοποίησης της γέφυρας B
IN3	TTL συμβατή είσοδο για την γέφυρα B
IN4	TTL συμβατή είσοδο για την γέφυρα B
CSA/CSB	Έλεγχος του ηλεκτρικού ρεύματος της γέφυρας A και B
UR1 UR2 UR3 UR4	Αντιστάτες ανάτασης

5.3 Το Bluetooth Module

Η συσκευή Bluetooth που χρησιμοποιούμε είναι η HC-06 (Εικόνα 5-6), τύπου slave. Κάθε Bluetooth Module έχει ένα όνομα, έναν κωδικό πρόσβασης και έναν ρυθμό μετάδοσης (Baud rate), τα οποία μπορούμε εάν θέλουμε να τα αλλάξουμε, θα εξηγήσουμε παρακάτω πως γίνεται αυτό. Τα σήματα του ακροδέκτη που υπάρχουν στο Bluetooth module έχουν την ακόλουθη σημασία:

Πίνακας 5-5: Τα pins του Bluetooth Module

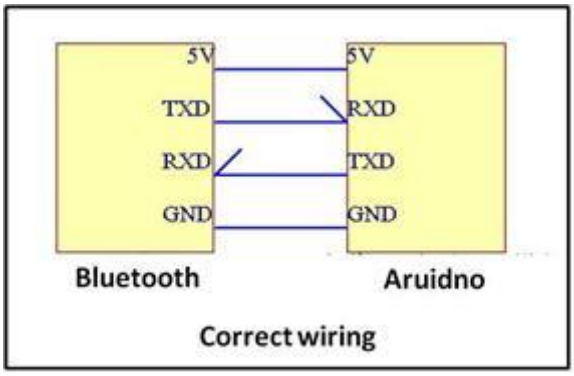
Αριθμός PIN	**Όνομα**	**Λειτουργία**
1	KEY	HIGH: Προγραμματισμός LOW: Λειτουργία
2	VCC	5V
3	GND	Γείωση
4	TXD	TX pin εκπομπής
5	RXD	RX pin λήψης
6	STATE	Κατάσταση σύνδεσης

Εικόνα 5-6: Bluetooth Module HC-06

5.3.1 Η Συνδεσμολογία του Bluetooth Module

Για να γίνει σωστά η επικοινωνία θα πρέπει να ενώσουμε το RXD του Bluetooth με το TXD του Arduino και το TXD του Bluetooth με το RXD του Arduino αντίστοιχα. Όταν περνάμε κάποιο πρόγραμμα στο Arduino, θα πρέπει το RXD και TXD του Bluetooth να τα αποσυνδέουμε από το Arduino Board ή να αποσυνδέουμε το VCC του Bluetooth

απ' τα 5V του Arduino (δεν απαιτείται στην πλατφόρμα Arduino Uno R3, ωστόσο σε παλαιότερες π.χ στην R2, αυτό το βήμα είναι απαραίτητο).



Εικόνα 5-7: Συνδεσμολογία του HC-06

5.4 Η Οθόνη LCD

Στο σασί του οχήματος μας έχει προσαρμοστεί μία LCD οθόνη (τύπου Hitachi HD44780) δεκαέξι χαρακτήρων και δύο γραμμών με πράσινο φόντο και μαύρους χαρακτήρες. Για την λειτουργία της οθόνης είναι απαραίτητη η προσθήκη της βιβλιοθήκης LiquidCrystal την οποία θα δούμε σε επόμενο κεφάλαιο στον κώδικα του Arduino.

Εικόνα 5-8: Οθόνη LCD 16 χαρακτήρων και 2 γραμμών

5.4.1 Η Συνδεσμολογία της Οθόνης LCD

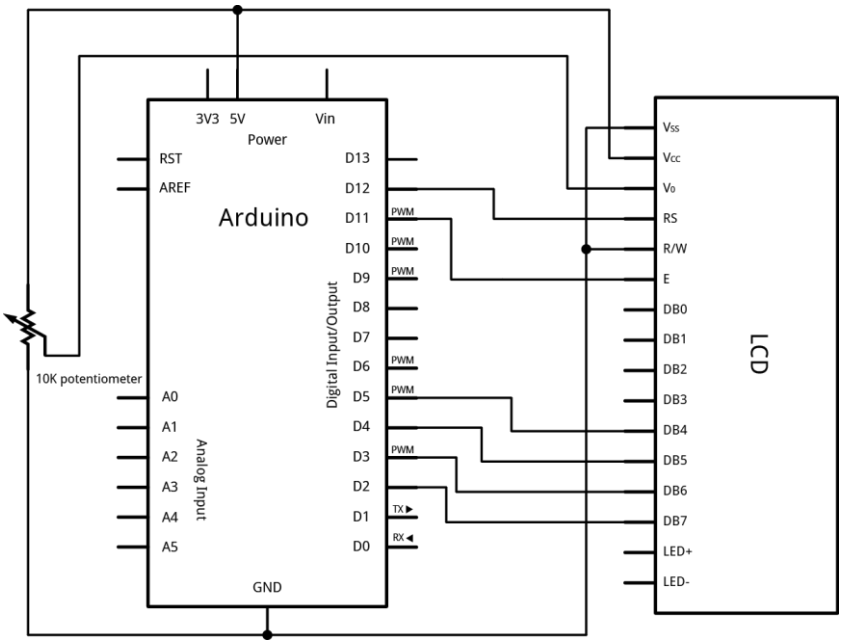
Στον παρακάτω πίνακα θα δούμε την συνδεσμολογία που πρέπει να κάνουμε για την σωστή λειτουργία της οθόνης. Η συνδεσμολογία αυτή απευθύνεται στις περισσότερες LCD οθόνες συμβατές με πλατφόρμες Arduino.

Πίνακας 5-6: Τα pins της οθόνης LCD

LCD Pin	**Σύμβολο**	**Arduino Pin**
Lcd Pin 1	Rss	Στη γείωση του Arduino
Lcd Pin 2	Vdd	Στα 5V του Arduino
Lcd Pin 3	Vo	Μεσαίο pin του ποτενσιόμετρου (10K)
Lcd Pin 4	Rs	Arduino Pin 12
Lcd Pin 5	Rw	Στη γείωση του Arduino
Lcd Pin 6	E	Arduino Pin 11
Lcd Pin 7-10	-	Δεν χρησιμοποιούνται
Lcd Pin 11	D4	Arduino Pin 5
Lcd Pin 12	D5	Arduino Pin 4
Lcd Pin 13	D6	Arduino Pin 3
Lcd Pin 14	D7	Arduino Pin 2
Lcd Pin 15	A	Στα 3.3V του Arduino

Lcd Pin 16	K	Στη γείωση του Arduino

Το ποτενσιόμετρο (Εικόνα 5-10) το χρησιμοποιούμε για να ρυθμίσουμε το Contrast (αντίθεση) της οθόνης. Επίσης τα δύο άλλα pin του ποτενσιόμετρου (αριστερό και δεξί) πρέπει και αυτά να συνδεθούν στην γείωση και στο ρεύμα αντίστοιχα. Το διάγραμμα που παρουσιάζεται παρακάτω θα μας δώσει μία γενική εικόνα του πως πρέπει να είναι η συνδεσμολογία.

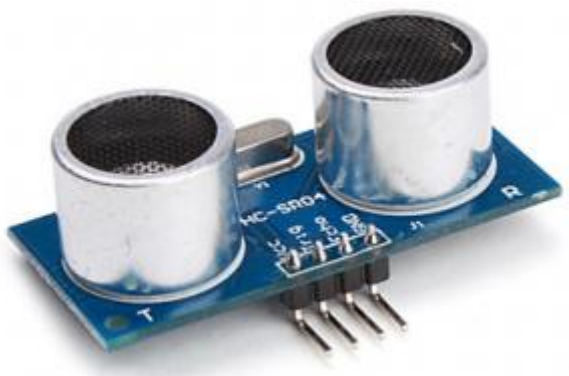


Εικόνα 5-9: Συνδεσμολογία οθόνης LCD με Arduino

Εικόνα 5-10: Ποτενσιόμετρο αντίστασης 10 ΚΩ

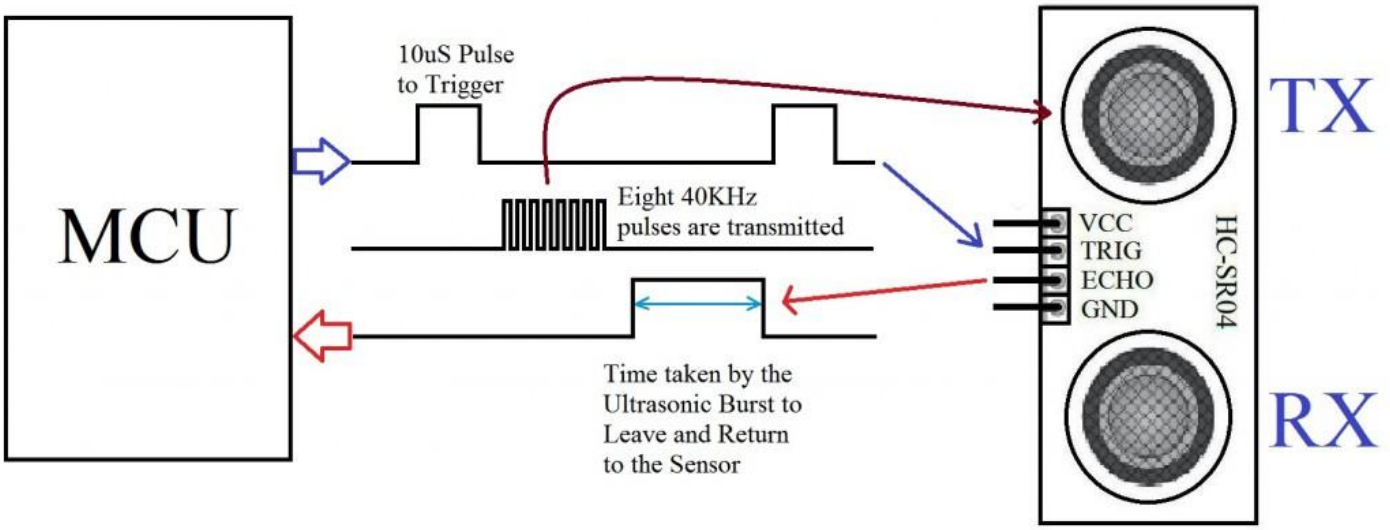
5.5 Ο Αισθητήρας Υπερήχων

Ο αισθητήρας υπερήχων (Ranging Detector) HC-SR04 που χρησιμοποιούμαι είναι ένας αρκετά δημοφιλής και με χαμηλό κόστος μετρητής αποστάσεων. Μπορεί να υπολογίσει απόσταση από δύο εκατοστά έως τετρακόσια εκατοστά με ακρίβεια ενός εκατοστού. Η βασική του λειτουργία θα είναι η διακοπή της κίνησης των τροχών του οχήματος μας σε περίπτωση που αντιληφθεί κάποιο εμπόδιο μπροστά για την αποφυγή της σύγκρουσης. Να αναφέρουμε ότι είναι απαραίτητη η χρήση της βιβλιοθήκης με όνομα Ping (την τελευταία έκδοση την βρίσκουμε ως NewPing) την οποία θα δούμε σε επόμενο κεφάλαιο.



Εικόνα 5-11: Ο αισθητήρας υπερήχων HC-SR04

Ο υπέρηχος είναι ένας υψηλής συχνότητας ήχος (συνήθως χρησιμοποιούνται 40 KHz). Μία γρήγορη ριπή ηχητικών κυμάτων (συχνά μόνο οκτώ κύκλων) αποστέλλεται από τον μετατροπέα εκπομπής (ο αριστερός στην επάνω εικόνα). Στη συνέχεια ο μετατροπέας λήψης (ο δεξιός) περιμένει να ακούσει την ηχώ. Στην παρακάτω εικόνα βλέπουμε πως ακριβώς λειτουργεί ο αισθητήρας υπέρηχων.



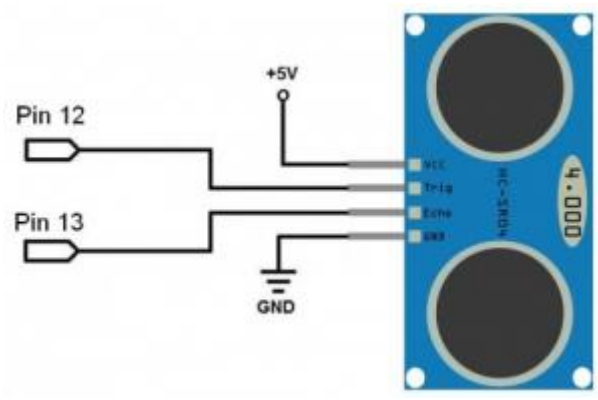
Εικόνα 5-12: Η λειτουργία του HC-SR04

5.5.1 Η Συνδεσμολογία του Αισθητήρα Υπερήχων

Η συνδεσμολογία του HC-SR04 με το Arduino παρουσιάζεται στον παρακάτω πίνακα. Τα Trig και Echo pins χρησιμοποιούνται για την διασύνδεση του αισθητήρα με μία μονάδα μικροελεγκτή, με το Arduino στην δική μας περίπτωση.

Πίνακας 5-7: Συνδεσμολογία του αισθητήρα υπερήχων

HC-SR04 Pin	**Σύμβολο**	**Arduino Pin**
Sensor Pin 1	Gnd	Στην γείωση του Arduino
Sensor Pin 2	Echo	Arduino Pin 11
Sensor Pin 3	Trig	Arduino Pin 12
Sensor Pin 4	Vcc	Στα 5V του Arduino



Εικόνα 5-13: Η συνδεσμολογία του HC-SR04

5.6 Ο Σερβοκινητήρας

Ο αισθητήρας υπερήχων για τον οποίο μιλήσαμε στην προηγούμενη υποενότητα, είναι προσαρμοσμένος επάνω σε ένα σερβοκινητήρα ο οποίος έχει δυνατότητα στρέψης εκατόν ογδόντα μοιρών, ενενήντα αριστερά και ενενήντα δεξιά. Ο σερβοκινητήρας ακολουθεί την κίνηση του οχήματος μας, δηλαδή εάν στρίβουμε αριστερά το όχημα μας τότε ο σερβοκινητήρας στρίβει και αυτός αριστερά δίνοντας ουσιαστικά την δυνατότητα στον αισθητήρα υπερήχων να κοιτά προς την κατεύθυνση στην οποία κινούμαστε. Εάν στρίβουμε δεξιά το όχημα μας τότε ο σερβοκινητήρας στρίβει δεξιά αντίστοιχα. Είναι απαραίτητη η χρήσης της βιβλιοθήκης με όνομα Servo.

Εικόνα 5-14: Σερβοκινητήρας TowerPro Micro SG90

Το μοντέλο του σερβοκινητήρα που χρησιμοποιούμε είναι το TowerPro Micro SG90, το βλέπουμε στην παραπάνω φωτογραφία (Εικόνα 5-14). Το TowerPro Micro SG90 είναι αρκετά ελαφρύ, γρήγορο και πολύ υψηλής ποιότητας. Έχει πλήθος χρήσεων όπως σε τηλεκατευθυνόμενα αεροπλάνα, ελικόπτερα, αυτοκίνητα, βάρκες κτλ. Τα χαρακτηριστικά του παρουσιάζονται παρακάτω (Πίνακας 5-8).

Πίνακας 5-8: Τα χαρακτηριστικά του σερβοκινητήρα

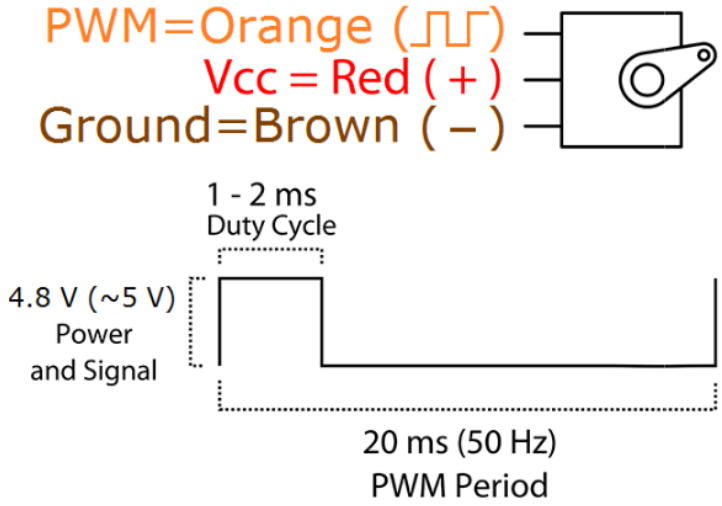
Μέγιστη ροπή όταν οι στροφές του είναι μηδέν (Stall torque)	1.5kg/cm στα 4.8V
Ταχύτητα λειτουργίας χωρίς βάρος	0.3sec/60° στα 4.8V
Μέγεθος	23*12.2*29mm
Βάρος	9g
Τάση λειτουργίας	4.2-6V
Γωνία περιστροφής	180°
Θερμοκρασία λειτουργίας	0 ºC – 55 ºC
Μήκος καλωδίου	250mm

5.6.1 Η Συνδεσμολογία του Σερβοκινητήρα

Για να συνδέσουμε ένα σερβοκινητήρα με το Arduino πρέπει να ακολουθήσουμε την συνδεσμολογία που παρουσιάζεται παρακάτω (Πίνακας 5-9).

Πίνακας 5-9: Η συνδεσμολογία του σερβοκινητήρα

TowerPro Micro SG90	**Arduino**
Servo +5V	Arduino +5V
Servo Ground	Arduino Ground
Servo Signal	Arduino Pin 9



Εικόνα 5-15: Η συνδεσμολογία του TowerPro Micro SG90

5.7 Η Πηγή Τροφοδοσίας

Στο όχημα μας υπάρχουν δύο διαφορετικές πηγές τροφοδοσίας, μία για το Arduino και μία για το Motor Shield. Το Arduino τροφοδοτείται με 9V, δηλαδή με έξι μπαταρίες AA των 1,5V η κάθε μία και το Motor Shield με οκτώ μπαταρίες AA των 1,5V δηλαδή με 12V. Οι μπαταρίες μας είναι τοποθετημένες σε ειδικές θήκες (Battery Holders), των έξι και οκτώ θέσεων αντίστοιχα. Στην παρακάτω εικόνα διακρίνουμε αριστερά το Battery holder έξι θέσεων (τρεις επάνω και τρεις κάτω) και δεξιά το Battery holder οκτώ θέσεων (τέσσερις επάνω και τέσσερις κάτω). Στα συγκεκριμένα Battery holders υπάρχουν επαφές επάνω στις οποίες μπορούμε να κολλήσουμε το καλώδιο τροφοδοσίας.

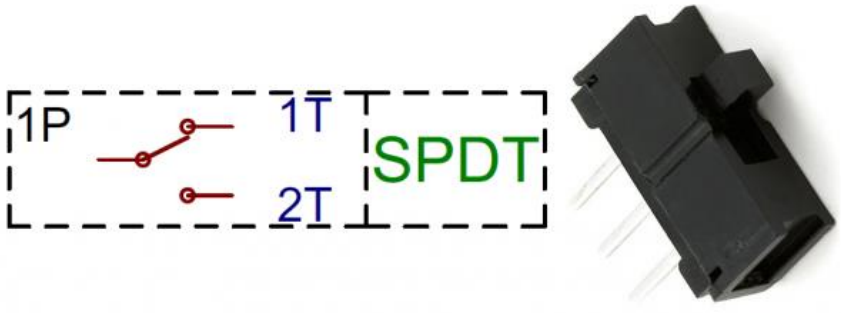
Εικόνα 5-16: Battery holder έξι και οκτώ θέσεων αντίστοιχα

5.7.1 Διακόπτες Toggle

Δύο διακόπτες, Toggle Switches, τύπου SPST (Single Pole-Single Throw) χρησιμοποιούνται για να ανοίξουμε και να κλείσουμε αντίστοιχα το ρεύμα τροφοδοσίας. Ο ένας διακόπτης έχει τοποθετηθεί για να ανοίγει και να κλείνει την τροφοδοσία του Arduino και ο άλλος αντίστοιχα για την τροφοδοσία του Motor Shield. Η συνδεσμολογία τους είναι αρκετά απλή, απλά παρεμβάλουμε το καλώδιο του ρεύματος τροφοδοσίας στα δύο pin του διακόπτη. Ο διακόπτης μας φαίνεται παρακάτω (Εικόνα 5-17).

Εικόνα 5-17: SPST Toggle Switch

Αξίζει να αναφέρουμε μερικούς ακόμα τύπους Toggle διακοπτών που είναι διαθέσιμοι στην αγορά. Εκτός λοιπόν από τους SPST, υπάρχουν οι SPDT (Single pole, double throw) οι οποίοι είναι κατάλληλοι όταν θέλουμε να επιλέξουμε ανάμεσα σε δύο τάσεις (Εικόνα 5-18).



Εικόνα 5-18: SPDT Slide Switch 1

Οι DPST (Double-pole Single-throw), είναι απλά δύο SPST διακόπτες μαζί. Επίσης υπάρχουν οι DPDT (Double-pole, Double-throw), είναι ουσιαστικά δύο SPDT διακόπτες μαζί. Τέλος υπάρχουν ακόμα πιο πολύπλοκοι διακόπτες με περισσότερους από δύο πόλους όπως ο 4PDT (Four-pole, Double-throw) ο οποίος μπορεί να ελέγχει τέσσερα διαφορετικά κυκλώματα (Εικόνα 5-19).

Εικόνα 5-19: 4PDT Toggle Switch

5.8 Υπόλοιπος Εξοπλισμός

Σε αυτή την παράγραφο θα ασχοληθούμε λίγο παραπάνω με τον υπόλοιπο εξοπλισμό που χρησιμοποιήσαμε, που μπορεί να έχουμε ήδη αναφέρει στις προηγούμενές, υποενότητες αλλά δεν ασχοληθήκαμε τόσο μαζί του. Όπως προηγούμενα είχα αναφέρει, χρησιμοποιούνται αποστάτες για την τοποθέτηση του Arduino και του Motor Shield στην πλατφόρμας μας. Ο ρόλος του αποστάτη είναι να δημιουργείται η απαραίτητη απόσταση μεταξύ της εκάστοτε πλατφόρμας και της επιφάνειας τοποθέτησης. Οι αποστάτες που χρησιμοποιούμε είναι πλαστικοί 5mm (Εικόνα 5-20). Αν παρατηρήσουμε τον αποστάτη θα δούμε ότι έχει υποδοχή για πλαστική βίδα, εκεί βιδώνουμε την πλατφόρμα μας, και αντίστοιχα με ένα πλαστικό παξιμάδι βιδώνεται ο αποστάτης στην επιφάνεια του οχήματος.

Εικόνα 5-20: Αποστάτης πλαστικός 5mm

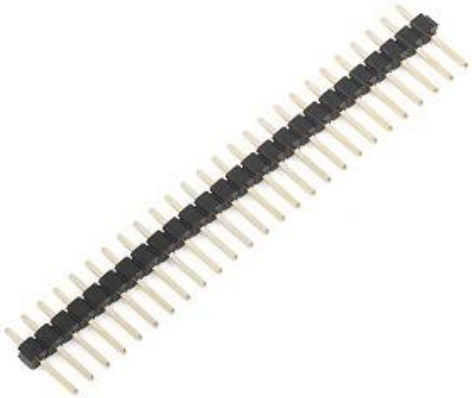
Το Arduino τροφοδοτείται με εξωτερική πηγή ενέργειας μέσω του Power jack ή ίσως να το γνωρίζουμε οι περισσότεροι ως Barrel jack plug. Το καλώδιο τροφοδοσίας του Arduino ξεκινά από το Battery Holder με τις έξι μπαταρίες, συνδέετε σε ένα διακόπτη Toggle, όπως έχει ήδη αναφερθεί, και έπειτα καταλήγει στο Barrel Jack Socket του Arduino χρησιμοποιώντας τον Barrel jack αντάπτορα (Εικόνα 5-21).

Εικόνα 5-21: Αντάπτορας Barrel Jack

Επίσης στο Battery Holder του Motor Shield υπάρχει το λεγόμενο «9V Battery Clip» (Εικόνα 5-22). Αυτό προσαρμόζεται στο Battery Holder του Motor Shield και στην συνέχεια το καλώδιο αυτού καταλήγει στον Toggle διακόπτη για το άνοιγμα ή το κλείσιμο αντίστοιχα της τροφοδοσίας.

Εικόνα 5-22: 9V Battery Clip

Πάρα πολύ σημαντική βοήθεια μας προσφέρουν τα Pin Headers (Εικόνα 5-23). Σε κάθε καλώδιο του οχήματος μας, το οποίο κουμπώνει στις υποδοχές του Arduino, έχει κολληθεί στην άκρη του ένα Pin Header. Στην εικόνα παρατηρούμε ότι υπάρχουν ειδικές οπές μεταξύ των pins, αυτές μας δίνουν την δυνατότητα εύκολα να κόψουμε των αριθμό των pins που χρειαζόμαστε, από ένα έως σαράντα.



Εικόνα 5-23: Pin Header Αρσενικό

Το θερμοσυστελόμενο μακαρόνι (Εικόνα 5-24) τοποθετείται μεταξύ της γυμνής επιφάνειας του καλωδίου και του σημείου που έχει γίνει η κόλληση μεταξύ καλωδίου και pin header. Εφαρμόζοντας το απαραίτητο ποσοστό θερμότητας στο θερμοσυστελόμενο αυτό έχει την ιδιότητα να συρρικνώνεται με αποτέλεσμα να προστατεύει το σημείο που είχε γίνει η κόλληση στο καλώδιο. Διατίθεται σε διάφορα μεγέθη, αναλόγως το μέγεθος του καλωδίου που χρησιμοποιούμε και μπορούμε να το κόψουμε εύκολα στο επιθυμητό μήκος. Χρησιμοποιείται παντού στο τηλεχειριζόμενο όχημα για να προστατεύσουμε τα γυμνά καλώδια από τυχόν βραχυκύκλωμα.

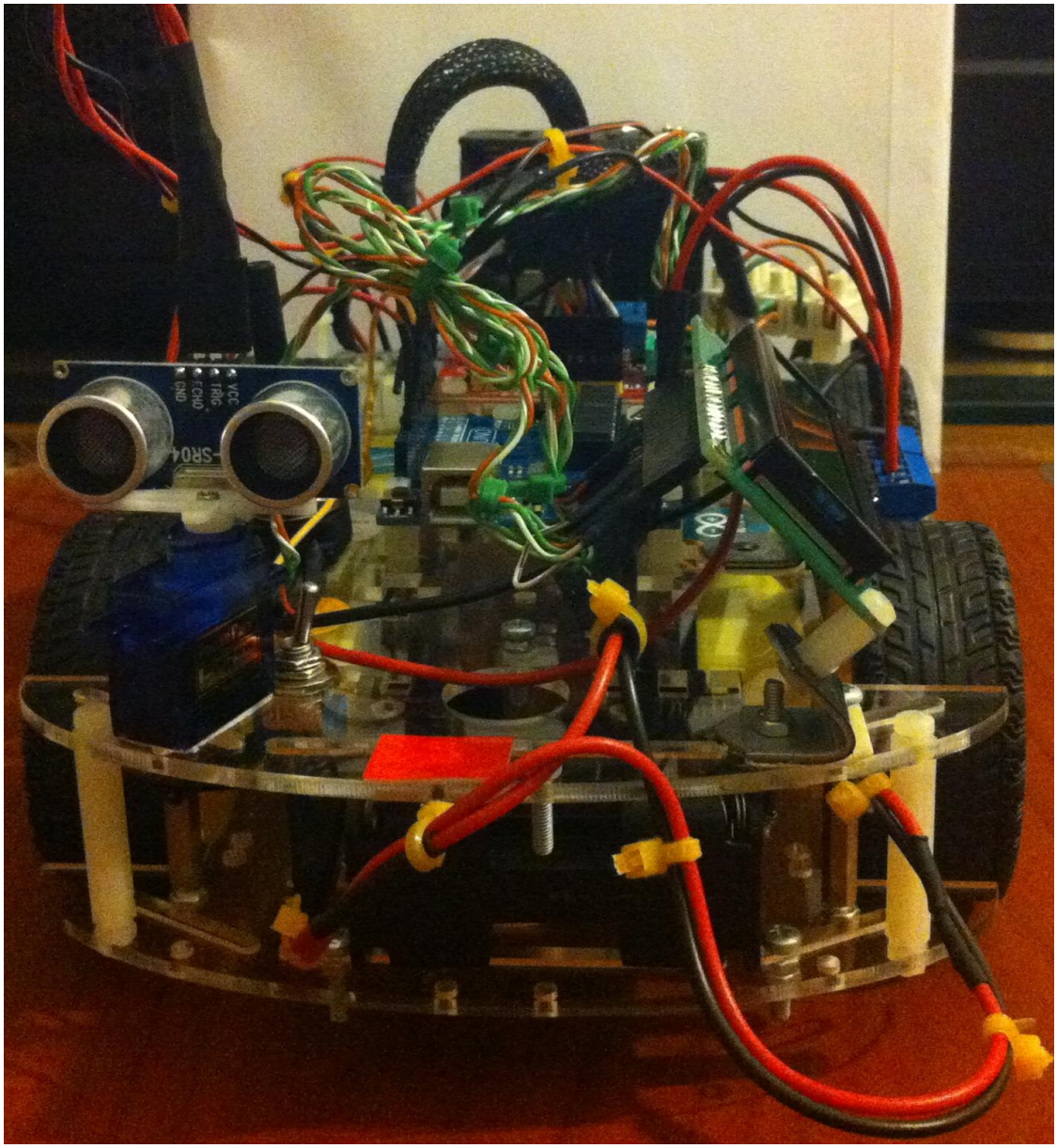
Εικόνα 5-24: Θερμοσυστελόμενο μακαρόνι

Τέλος στο τηλεχειριζόμενο όχημα θα διακρίνουμε τις λεγόμενες κλέμες ή ακόμα καλύτερα τα Terminal Blocks (Εικόνα 5-25). Εκεί καταλήγουν καλώδια, όσες είναι οι είσοδοι άλλες τόσες είναι και οι έξοδοι. Στο τηλεχειριζόμενο όχημα χρησιμοποιούνται ως δια μοιραστές τροφοδοσίας για ένα μέρος του εξοπλισμού.

Εικόνα 5-25: Terminal Block

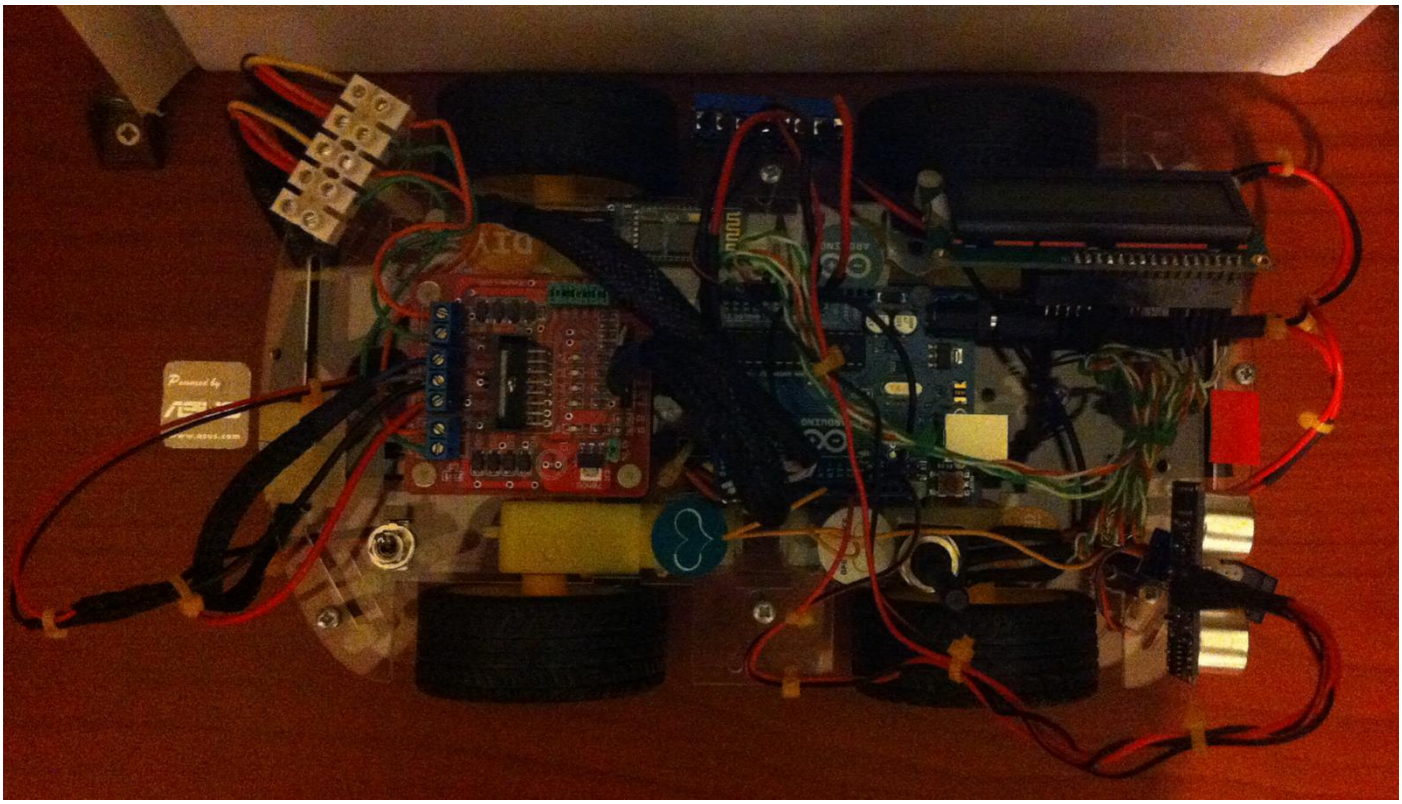
5.9 Η Κατασκευή

Έχοντας ολοκληρώσει την απαραίτητη εισαγωγή σχετικά με τον εξοπλισμό που χρειαζόμαστε και την συνδεσμολογία του, ήρθε η ώρα να παρουσιάσουμε την τελική κατασκευή, δηλαδή το τηλεχειριζόμενο όχημα πλέον ολοκληρωμένο. Στις παρακάτω φωτογραφίες (Εικόνα 5-26 και 5-27) βλέπουμε το τηλεχειριζόμενο όχημα, έχοντας επάνω του το Arduino, το Motor Shield, το Bluetooth, την οθόνη LCD, τον αισθητήρα απόστασης και τον σερβοκινητήρα, δηλαδή όλα για όσα μιλήσαμε προηγουμένως.



Εικόνα 5-26: Μπροστινή όψη του τηλεχειριζόμενου οχήματος

Πιο συγκεκριμένα, στην εικόνα 5-26 διακρίνουμε την μπροστινή όψη του οχήματος. Είναι ορατός ο αισθητήρας απόστασης, η πλαϊνή όψη της LCD οθόνης, το Battery Holder ανάμεσα στα δύο κομμάτια του Plexiglass του οχήματος, ο σερβοκινητήρας, ο διακόπτης Toggle στα δεξιά του σερβοκινητήρα και αρκετά καλώδια.

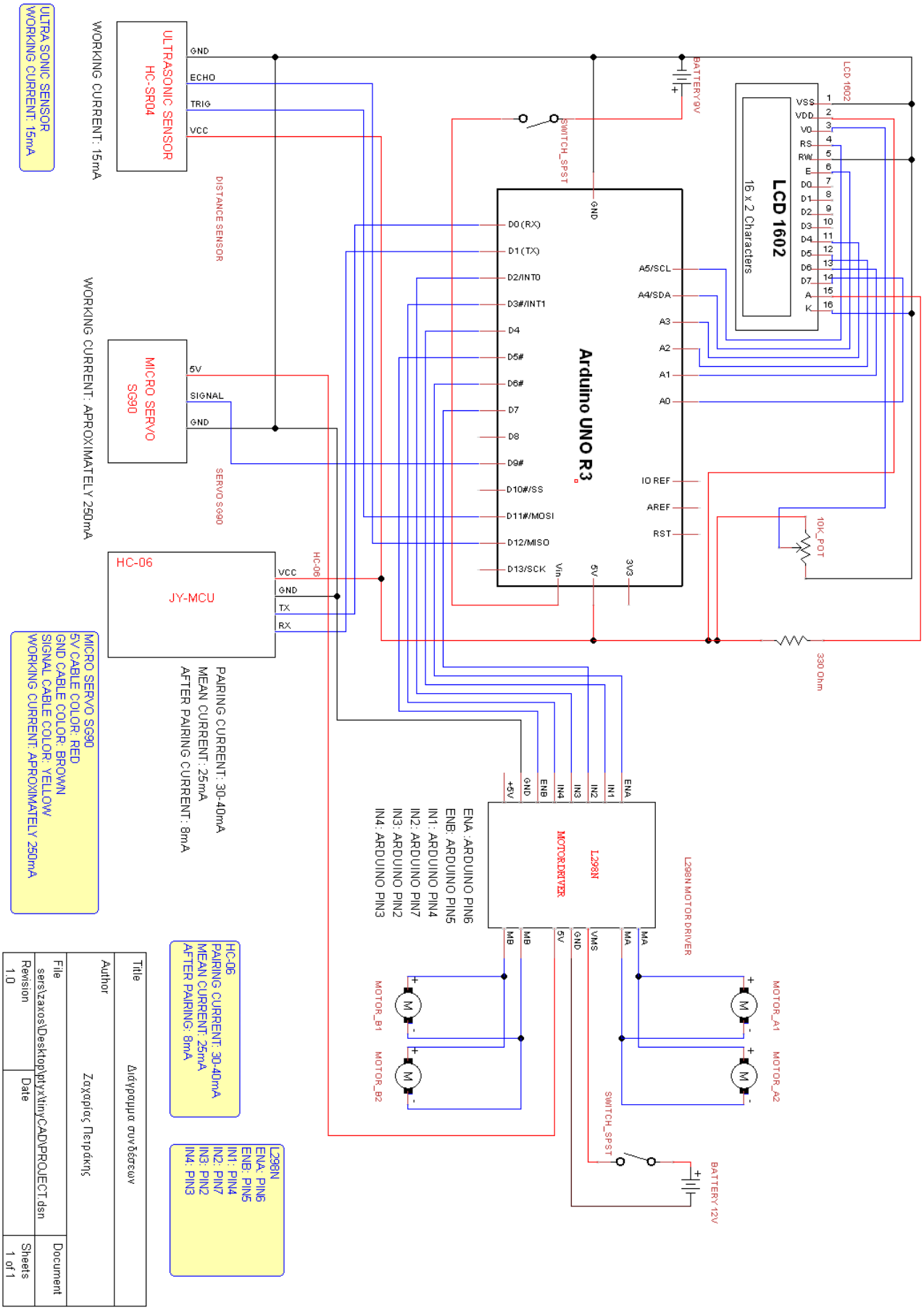


Εικόνα 5-27: Πανοραμική όψη του τηλεχειριζόμενου οχήματος

Τέλος στην εικόνα 5-27 βλέπουμε μία ωραία πανοραμική όψη του οχήματος. Είναι εύκολο να διακρίνουμε όλο τον εξοπλισμό που διαθέτει το όχημα μας. Το Arduino, το Motor Shield, το Bluetooth (αριστερά από το Motor Shield), το ποτενσιόμετρο (δεξιά από το Arduino) αλλά και όλα όσα νωρίτερα αναφέραμε.

5.9.1 Ολοκληρωμένο Διάγραμμα

Εδώ παρουσιάζουμε ένα πολύ ωραίο διάγραμμα που αφορά την συνολική συνδεσμολογία, δηλαδή όλες τις απαραίτητες συνδέσεις που πρέπει να γίνουν έτσι ώστε ο συνολικός εξοπλισμός που χρησιμοποιούμε να δουλεύει σωστά. Το διάγραμμα αυτό έχει δημιουργηθεί στο πρόγραμμα TinyCAD, χρησιμοποιείται για την σχεδίαση ηλεκτρικών κυκλωμάτων σχηματικών κτλ. Το TinyCAD είναι δωρεάν και μπορούμε να το βρούμε στον παρακάτω σύνδεσμο http://sourceforge.net/projects/tinycad/ . Ακολουθεί το ολοκληρωμένο διάγραμμα.



Εικόνα 5-28: Ολοκληρωμένο διάγραμμα της συνδεσμολογίας του εξοπλισμού

6 Προγραμματισμός του Arduino

Σε αυτό το κεφάλαιο θα παρουσιάσουμε τον συνολικό κώδικα τον οποίο φορτώνουμε στο Arduino για την εκτέλεση των απαραίτητων λειτουργιών κίνησης του τηλεχειριζόμενου οχήματος. Πιο συγκεκριμένα θα δούμε τον τρόπο που γίνεται λήψη των σειριακών δεδομένων από το Arduino, πως αποφασίζει την κίνηση που θα ακολουθήσει, την λειτουργία του σερβοκινητήρα και του αισθητήρα απόστασης κτλ. Πρόκειται για ένα εύκολο και αρκετά κατανοητό κώδικα τον οποίο θα εξηγήσουμε παρακάτω.

6.1 Ο Κώδικας

Θα ξεκινήσουμε με το παρακάτω μπλοκ κώδικα (Πίνακας 6-1). Χρησιμοποιώντας την εντολή `#include` μας δίνεται η δυνατότητα να εισάγουμε εξωτερικές βιβλιοθήκες στο sketch μας. Αυτό δίνει στον προγραμματιστή την δυνατότητα πρόσβασης σε ένα μεγάλο αριθμό στάνταρ βιβλιοθηκών (γραμμένες στην γλώσσα C), πρόκειται ουσιαστικά για προκατασκευασμένες μεθόδους. Επίσης μπορούμε να εισάγουμε βιβλιοθήκες γραμμένες συγκεκριμένα για το Arduino.

Οι βιβλιοθήκες που εμείς έχουμε εισάγει είναι η `LiquidCrystal`, η `NewPing` και η `Servo`. Η βιβλιοθήκη `LiquidCrystal` μας επιτρέπει να χειριστούμε οποιαδήποτε LCD οθόνη η οποία είναι συμβατή με τον HD44780 driver. Η βιβλιοθήκη `NewPing` μας βοηθά να χειριστούμε τον αισθητήρα απόστασης και τέλος η βιβλιοθήκη `Servo` μας βοηθά στον έλεγχο του σερβοκινητήρα μας. Και οι τρεις αυτές βιβλιοθήκες είναι απαραίτητες στην εφαρμογή μας, η εισαγωγή τους γίνεται πατώντας το Sketch στη μπάρα εργαλείων του Arduino IDE, μετά επιλέγουμε το Import Library και στην λίστα που εμφανίζεται επιλέγουμε τις βιβλιοθήκες μας. Να σημειώσουμε πως η βιβλιοθήκη `NewPing` δεν υπάρχει στην λίστα οπότε θα πρέπει να την κατεβάσουμε από τον σύνδεσμο http://playground.arduino.cc/Code/NewPing και μετά να πάμε στο Sketch >Import Library>Add Library. Από εκεί πρέπει να επιλέξουμε την θέση στην οποία αποθηκεύσαμε την βιβλιοθήκη ώστε να εισαχθεί

Πίνακας 6-1: Εισαγωγή βιβλιοθηκών

```
#include <LiquidCrystal.h>
#include <NewPing.h>
#include <Servo.h>
```

Η εντολή `#define` (πίνακας 6-2) αποτελεί ένα σημαντικό συστατικό της γλώσσας C που μας επιτρέπει να δώσουμε ένα όνομα σε μία σταθερή τιμή, πριν το πρόγραμμα μεταγλωττιστεί. Για παράδειγμα, στον κώδικα μας ο μεταγλωττιστής θα αντικαταστήσει οποιαδήποτε αναφορά υπάρχει με όνομα `TRIGGER_PIN` με την τιμή 11 κατά την διαδικασία της μεταγλώττισης. Το ίδιο ισχύει για τις σταθερές `ECHO_PIN` και `MAX_DISTANCE` οι οποίες κατά την μεταγλώττιση θα αντικατασταθούν με τις τιμές 12 και 200 αντίστοιχα. Η μεταβλητή `MAX_DISTANCE` ορίζει την μέγιστη απόσταση στην οποία θα γίνεται ping από τον αισθητήρα μας.

Πίνακας 6-2: Δήλωση μεταβλητών με την define

```
#define TRIGGER_PIN  11
#define ECHO_PIN     12
#define MAX_DISTANCE 200
```

Στη συνέχεια ακολουθεί η δήλωση των υπολοίπων μεταβλητών του προγράμματος μας (Πίνακας 6-3). Όπως έχουμε αναφέρει σε προηγούμενη ενότητα το όχημα μας αποτελείται από τέσσερα DC μοτέρ. Εμείς αυτά τα μοτέρ τα διαχειριζόμαστε σε ομάδες των δύο σετ. Δηλαδή τα δύο δεξιά μοτέρ αποτελούν το ένα σετ και τα δύο αριστερά το άλλο σετ. Έτσι στη δήλωση των μεταβλητών μας οι μεταβλητές `MOTOR_1_1`, `MOTOR_1_PWM` και `MOTOR_1_2` αφορούν το πρώτο σετ των μοτέρ και έχουν πάρει τις τιμές 4, 6 και 7 αντίστοιχα, οι οποίες συμβολίζουν τα pins του Arduino και οι μεταβλητές `MOTOR_2_1`, `MOTOR_2_PWM` και `MOTOR_2_2` αφορούν το δεύτερο σετ με τιμές 2, 5 και 3 αντίστοιχα. Επίσης έχει δηλωθεί η μεταβλητή `myservo` που είναι τύπου `Servo`, με την δήλωση `NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE);` αρχικοποιείται η `NewPing` με τις τιμές 11, 12 και 200 αντίστοιχα. Με την χρήση της μεταβλητής `sonar` μπορούμε να καλέσουμε τις διαθέσιμες μεθόδους όπως την sonar.ping(); που θα δούμε παρακάτω. Η δήλωση `LiquidCrystal lcd(A1, A0, A2, A3, A4, A5);` χρησιμοποιείται για την αρχικοποίηση της βιβλιοθήκης

με τα pins με τα οποία αλληλοεπιδρά με το Arduino, τα οποία είναι τα `A1, A0, A2, A3, A4, A5` (Analog pins) στην δική μας περίπτωση. Τέλος αρχικοποιούμε την μεταβλητή `incomingByte` με 0, στην μεταβλητή αυτή θα αποθηκεύεται η εισερχόμενη ροή δεδομένων. Την μεταβλητή `speed_val` με 255, που όπως γνωρίζουμε σύμφωνα με το PWM η τιμή 255 αναφέρεται σε κύκλο εργασίας 100%, δηλαδή πλήρης ταχύτητα. Τέλος η μεταβλητή `dis` με τιμή 200 χρησιμοποιείται για έλεγχο σχετικό με την απόσταση, θα την δούμε παρακάτω.

Πίνακας 6-3: Δήλωση μεταβλητών (συνέχεια)

```
int MOTOR_1_1 = 4;
int MOTOR_1_PWM = 6;
int MOTOR_1_2 = 7;
int MOTOR_2_1 = 2;
int MOTOR_2_PWM = 5;
int MOTOR_2_2 = 3;
Servo myservo;
NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE);
LiquidCrystal lcd(A1, A0, A2, A3, A4, A5);
int incomingByte = 0;
int speed_val = 255 ;
int dis=200;
```

Η συνάρτηση setup εκτελείται μία και μόνο φορά, εδώ βάζουμε όλες τις εντολές που πρέπει να τρέξουν μία φορά, όταν ενεργοποιείται η μονάδα μας (Πίνακας 6-4). Συνήθως μπαίνουν αρχικοποιήσεις τιμών μεταβλητών και οπωσδήποτε ο χαρακτηρισμός των εισόδων/εξόδων που θα χρησιμοποιήσουμε (αν δηλαδή ένα συγκεκριμένο Pin θα είναι είσοδος ή έξοδος).

Μέσα στην `setup()` ξεκινούμε αρχικοποιώντας των αριθμό των στηλών και των γραμμών της LCD `lcd.begin(16, 2)`, το 16 σημαίνει δεκαέξι στήλες και το 2 σημαίνει δύο γραμμές. Με την εντολή `setCursor` μπορούμε να τοποθετήσουμε τον κέρσορα στην επιθυμητή στήλη και γραμμή της οθόνης και με την `print` τυπώνουμε το μήνυμα που θέλουμε στην LCD. Με την εντολή `myservo.attach(9);` δηλώνουμε ότι ο σερβοκινητήρας είναι συνδεδεμένος στο pin 9 του Arduino. Με την `Serial.begin(9600)`, ανοίγουμε μία σειριακή θύρα και ορίζουμε τον ρυθμό μετάδοσης των δεδομένων στα 9600 bit ανά δευτερόλεπτο (Baud Rate). Με την εντολή pinMode(pin, mode), ορίζουμε τα pin που αντιστοιχούν στα μοτέρ ως εξόδους. Η συναρτήσεις `MOTOR_1_STOP()` και `MOTOR_2_STOP()`εξασφαλίζουν ότι τα μοτέρ δεν θα είναι σε λειτουργία κατά την εκκίνηση του Arduino, θα τις δούμε αναλυτικά παρακάτω. Τέλος με την `myservo.write(90)` ορίζουμε την γωνία στρέψης του σερβοκινητήρα, 90 μοίρες αρχικά και στην συνέχεια 40, 160 και πάλι 90, αυτό πρακτικά βοηθά τον αισθητήρα απόστασης που είναι προσαρμοσμένος πάνω στον σερβοκινητήρα να ρίξει μία ματιά δεξιά και αριστερά. Τέλος με το `delay` εκφράζουμε καθυστέρηση σε millisecond (ms).

Πίνακας 6-4: Η συνάρτηση Setup

```
void setup(){
lcd.begin(16, 2);
lcd.setCursor(0, 0);
lcd.print("Uno ");
lcd.setCursor(0, 1);
lcd.print("Z.P");
lcd.setCursor(8, 0);
lcd.print(":)");
lcd.setCursor(11, 0);
lcd.print("Enjoy");
lcd.setCursor(5, 0);
lcd.print("PWM");

myservo.attach(9);
Serial.begin(9600);

pinMode(MOTOR_1_1, OUTPUT);
pinMode(MOTOR_1_PWM, OUTPUT);
```

```
pinMode(MOTOR_1_2, OUTPUT);
pinMode(MOTOR_2_1, OUTPUT);
pinMode(MOTOR_2_PWM, OUTPUT);
pinMode(MOTOR_2_2, OUTPUT);

MOTOR_1_STOP();
MOTOR_2_STOP();
myservo.write(90);
delay(1000);
myservo.write(40);
delay(500);
myservo.write(160);
delay(500);
myservo.write(90);
}
```

Σειρά έχει η συνάρτηση `loop(),`εδώ γράφουμε το πρόγραμμά μας (Πίνακας 6-5). Οι εντολές που υπάρχουν μέσα σε αυτήν θα τρέξουν και όταν φτάσει στο τέλος θα ενεργοποιηθεί ξανά η loop(), συνεχίζοντας από την αρχή της, και ξανά. Αυτό θα συμβαίνει συνεχώς, όσο έχει ρεύμα το Arduino ή μέχρι να πατηθεί το πλήκτρο reset.

Εδώ γίνεται έλεγχος για την ύπαρξη σειριακών δεδομένων κάνοντας χρήση της `Serial.available()` . Εφόσον υπάρχουν, λαμβάνεται ένας αριθμός από Bytes τα οποία είναι διαθέσιμα για ανάγνωση από την σειριακή θύρα. Τα δεδομένα αυτά αποθηκεύονται στον σειριακό buffer ο οποίος έχει την δυνατότητα να κρατά σύνολο 64 Byte. Στη συνέχεια διαβάζεται η εισερχόμενη ροή δεδομένων και αποθηκεύεται στην μεταβλητή `incomingByte`.

Πίνακας 6-5: Η συνάρτηση Loop (λήψη σειριακών δεδομένων)

```
void loop(){
if (Serial.available() > 0 ) {
incomingByte = Serial.read();
```

Εάν η μεταβλητή incomingByte, δηλαδή τα δεδομένα σε Bytes που λήφθηκαν, ισούται με τον αριθμό "70" δηλαδή με το γράμμα "F" στον κώδικα ASCII, τότε θα κληθεί η συνάρτηση `MOTOR_1_GO_FORWARD` και `MOTOR_2_GO_FORWARD` με όρισμα την ταχύτητα (μεταβλητή `speed_val`), η οποία είναι 255, ο σερβοκινητήρας θα πάρει γωνιακή κλίση 90 μοιρών (θα κοιτά ευθεία εμπρός), στη συνέχει γίνεται κλήση της συνάρτησης distSens, η οποία υπολογίζει την απόσταση του οχήματος μας από τυχόν εμπόδια, και τέλος το γράμμα "F" τυπώνεται στην LCD οθόνη. Το παρακάτω μπλοκ κώδικα αναγκάζει το όχημα μας να κινηθεί ευθεία εμπρός (Πίνακας 6-6).

Πίνακας 6-6: Κίνηση οχήματος εμπρός

```
if (incomingByte == 'F'){
MOTOR_1_GO_FORWARD(speed_val);
MOTOR_2_GO_FORWARD(speed_val);
myservo.write(90);
delay(21);
distSens();
lcd.setCursor(4, 1);
lcd.print(" F");
}
```

Αντίστοιχα (Πίνακας 6-7) σε περίπτωση που τα δεδομένα ισούνται με τον αριθμό "82", δηλαδή το γράμμα R θα κληθούν οι συναρτήσεις `MOTOR_1_GO_REVERSE` και `MOTOR_2_GO_FORWARD` με όρισμα και πάλι την ταχύτητα η οποία είναι 255 επίσης, ο σερβοκινητήρας θα πάρει μία γωνιακή κλίση 40 μοιρών (θα κοιτά προς τα δεξιά), γίνεται κλήση της συνάρτησης distSens, η οποία υπολογίζει την απόσταση, και τέλος το γράμμα "R" τυπώνεται στην LCD. Το παρακάτω μπλοκ κώδικα αναγκάζει το όχημα μας να κινηθεί δεξιά.

Πίνακας 6-7: Κίνηση οχήματος δεξιά

```
else if (incomingByte == 'R'){
MOTOR_1_GO_REVERSE(speed_val);
MOTOR_2_GO_FORWARD(speed_val);
myservo.write(40);
delay(21);
distSens();
lcd.setCursor(4, 1);
lcd.print(" R");
}
```

Αντίστοιχα (Πίνακας 6-8) σε περίπτωση που τα δεδομένα ισούνται με τον αριθμό "76", δηλαδή το γράμμα "L" θα κληθούν οι συναρτήσεις MOTOR_1_GO_FORWARD και MOTOR_2_GO_REVERSE με όρισμα και πάλι την ταχύτητα η οποία είναι 255 επίσης, ο σερβοκινητήρας θα πάρει μία γωνιακή κλίση 160 μοιρών (κοιτά αριστερά), γίνεται κλήση της συνάρτησης distSens, η οποία υπολογίζει την απόσταση, και τέλος το γράμμα "L" τυπώνεται στην LCD. Το παρακάτω μπλοκ κώδικα αναγκάζει το όχημα μας να κινηθεί αριστερά.

Πίνακας 6-8: Κίνηση οχήματος αριστερά

```
else if (incomingByte == 'L'){
MOTOR_1_GO_FORWARD(speed_val);
MOTOR_2_GO_REVERSE(speed_val);
myservo.write(160);
delay(21);
distSens();
lcd.setCursor(4, 1);
lcd.print(" L");
}
```

Αντίστοιχα (Πίνακας 6-9) σε περίπτωση που τα δεδομένα ισούνται με τον αριθμό "68", δηλαδή το γράμμα "D" θα κληθούν οι συναρτήσεις MOTOR_1_GO_REVERSE και MOTOR_2_GO_REVERSE με όρισμα και πάλι την ταχύτητα η οποία είναι 255 επίσης, ο σερβοκινητήρας θα πάρει μία γωνιακή κλίση 90 μοιρών (θα κοιτά ευθεία εμπρός), γίνεται κλήση της συνάρτησης distSens, η οποία υπολογίζει την απόσταση, και τέλος το γράμμα "D" τυπώνεται στην LCD. Το παρακάτω μπλοκ κώδικα αναγκάζει το όχημα μας να κινηθεί προς τα πίσω.

Πίνακας 6-9: Κίνηση οχήματος πίσω

```
else if (incomingByte == 'D'){
MOTOR_1_GO_REVERSE(speed_val);
MOTOR_2_GO_REVERSE(speed_val);
myservo.write(90);
delay(21);
distSens();
lcd.setCursor(4, 1);
lcd.print(" D");
}
```

Σε περίπτωση που ληφθεί το γράμμα "E" (αριθμός 69) γίνεται κλήση της συνάρτησης MOTOR_1_STOP() και MOTOR_2_STOP() οι οποίες σταματούν την κίνηση των μοτέρ. Το γράμμα "E" τυπώνεται στην LCD (Πίνακας 6-10).

Πίνακας 6-10: Ακινητοποίηση οχήματος με ενέργεια του χρήστη

```
else if (incomingByte == 'E'){
MOTOR_1_STOP();
MOTOR_2_STOP();
lcd.setCursor(4, 1);
lcd.print(" E");
```

```
}
```

Σε περίπτωση που δεν ληφθεί κάποιο από τα παραπάνω γράμματα, τότε από προεπιλογή στέλνεται το γράμμα "S" (αριθμός 83, έτσι έχει προγραμματιστεί η Android εφαρμογή μας) οπότε και γίνεται κλήση της συνάρτησης `MOTOR_1_STOP()` και `MOTOR_2_STOP()` οι οποίες σταματούν την κίνηση των μοτέρ του οχήματος μας (Πίνακας 6-11).

Πίνακας 6-11: Αυτόματη ακινητοποίηση του οχήματος

```
else if (incomingByte == 'S') {
MOTOR_1_STOP();
MOTOR_2_STOP();
}
}
```

Επίσης μπορούμε να ρυθμίσουμε την ταχύτητα του οχήματος μας, από 255 που είναι η προεπιλεγμένη μέγιστη ταχύτητα, σε 64. Έτσι θα πετύχουμε μία πολύ αργή κίνηση του οχήματος. Αυτό γίνεται όταν στέλνεται το γράμμα "Z" από την Android εφαρμογή μας, δηλαδή ο αριθμός 90 (Πίνακας 6-12). Η τιμή αυτή τυπώνεται στην οθόνη.

Πίνακας 6-12: PWM 64

```
else if(incomingByte == 'Z'){
speed_val=64;
lcd.setCursor(5, 0);
lcd.print("064");
}
```

Αντίστοιχα μπορούμε να ρυθμίσουμε την ταχύτητα στο 85 στέλνοντας το γράμμα "X" από την εφαρμογή μας, δηλαδή τον αριθμό 88. Έτσι θα πετύχουμε μία χαμηλή ταχύτητα του οχήματος (Πίνακας 6-13). Η τιμή αυτή τυπώνεται στην οθόνη.

Πίνακας 6-13: PWM 85

```
else if(incomingByte == 'X'){
speed_val=85;
lcd.setCursor(5, 0);
lcd.print("085");
}
```

Σε περίπτωση που ληφθεί το γράμμα "C", δηλαδή ο αριθμός 67, τότε η ταχύτητα από 255 που είναι η προεπιλεγμένη τιμή, θα πέσει στο 200 (Πίνακας 6-14). Η τιμή αυτή τυπώνεται στην οθόνη.

Πίνακας 6-14: PWM 200

```
else if(incomingByte == 'C'){
speed_val=200;
lcd.setCursor(5, 0);
lcd.print("200");
}
```

Τέλος σε περίπτωση που ληφθεί το γράμμα "V", δηλαδή ο αριθμός 86, τότε η ταχύτητα επανέρχεται στην προεπιλεγμένη τιμή, δηλαδή 255 (Πίνακας 6-15). Η τιμή αυτή τυπώνεται στην οθόνη.

Πίνακας 6-15: PWM 255

```
else if(incomingByte == 'V'){
speed_val=255;
lcd.setCursor(5, 0);
lcd.print("255");
}
}
```

Σε περίπτωση που δεν υπάρχει εισερχόμενη ροή δεδομένων, δηλαδή δεν έχει ληφθεί κανένα από τα παραπάνω γράμματα τότε εξασφαλίζουμε από προεπιλογή ότι δεν θα υπάρχει κίνηση στα μοτέρ καλώντας και πάλι τις συναρτήσεις `MOTOR_1_STOP()` και `MOTOR_2_STOP()`(Πίνακας 6-16).

Πίνακας 6-16: Προεπιλεγμένη ακινητοποίηση του οχήματος

```
else {
MOTOR_1_STOP();
MOTOR_2_STOP();
}
```

Εδώ τυπώνουμε στην LCD οθόνη ένα Counter ο οποίος μετράει σε λεπτά το χρόνο λειτουργίας της εφαρμογής μας (Πίνακας 6-17).

Πίνακας 6-17: Μετρητής χρόνου

```
lcd.setCursor(9, 1);
lcd.print("runT:");
lcd.print(millis()/1000/60);
lcd.print("m");
}
```

Την συνάρτηση distSens την καλέσαμε αρκετές φορές παραπάνω στον κώδικα μας (Πίνακας 6-18). Αρχικά ελέγχουμε εάν η απόσταση μας είναι μεγαλύτερη των 5 εκατοστών, εφόσον είναι τότε ανά 5 microsecond γίνεται ping καλώντας την ping() και επιστρέφεται το echo (ηχώ) σε microsecond (επιστρέφει 0 αν δεν υπάρξει ηχώ) όπου αποθηκεύεται στην μεταβλητή uS. Στην συνέχει καλώντας την convert_cm() γίνεται μετατροπή των microsecond σε απόσταση (εκατοστά, cm). Η απόσταση αυτή αποθηκεύεται στην μεταβλητή dis.

Πίνακας 6-18: Η συνάρτηση distSens, μέτρηση απόστασης

```
void distSens(){
if (dis>5){
delayMicroseconds(5);
unsigned int uS = sonar.ping();
Serial.print("Ping: ");
Serial.print(sonar.convert_cm(uS));
dis= sonar.convert_cm(uS);
Serial.println("cm");
}
```

Σε περίπτωση που η απόσταση είναι μικρότερη των 5 εκατοστών τότε σημαίνει ότι το όχημα μας είναι επικίνδυνα κοντά σε κάποιο εμπόδιο οπότε και τυπώνεται στην LCD το μήνυμα "CRASH" δηλαδή σύγκρουση και ταυτόχρονα καλούνται οι συναρτήσεις `MOTOR_1_STOP` και `MOTOR_2_STOP` ώστε να σταματήσει η κίνηση του οχήματος και να αποφύγουμε την σύγκρουση (Πίνακας 6-19).

Πίνακας 6-19: Η συνάρτηση distSens, παύση κίνησης

```
else{
lcd.setCursor(11, 0);
lcd.print("Crash");
Serial.println("less than five");
MOTOR_1_STOP();
MOTOR_2_STOP();
delay(1000);
dis=200;
```

Χρησιμοποιώντας την παρακάτω εντολή αποφεύγουμε να γίνεται buffering των δεδομένων κατά την διαδικασία εκτέλεσης της else (Πίνακας 6-20).

Πίνακας 6-20: Διακοπή του Buffering

```
while (Serial.read() != -1);
```

Μετά την τύπωση του «CRASH» στην οθόνη και με καθυστέρηση ενός δευτερολέπτου τυπώνεται το μήνυμα «ENJOY» στην LCD (Πίνακας 6-21).

Πίνακας 6-21: Τύπωση στην οθόνη

```
lcd.setCursor(11, 0);
lcd.print("Enjoy");
}
}
```

Η παρακάτω συνάρτηση καλείται όταν θέλουμε το σετ 1 των μοτέρ να εκτελέσει κίνηση προς τα πίσω (Πίνακας 6-22). Για παράδειγμα η συνάρτηση `MOTOR_1_GO_REVERSE(int x)` λειτουργεί με τον εξής απλό τρόπο, θέτουμε το pin 7 που αντιστοιχεί στην μεταβλητή `MOTOR_1_2` σε κατάσταση LOW, δηλαδή 0Volt, και ταυτόχρονα το pin 4 σε κατάσταση HIGH δηλαδή 5Volt αναγκάζοντας έτσι το μοτέρ να περιστρέφεται προς τα πίσω (αριστερόστροφα). Να προσέξουμε ότι τεράστια σημασία έχει και η συνδεσμολογία του μοτέρ για την περιστροφή που θα εφαρμόσει. Σε παρόμοια λογική βασίζονται όλες οι επόμενες συναρτήσεις.

Πίνακας 6-22: Η συνάρτηση MOTOR_1_GO_REVERSE

```
void MOTOR_1_GO_REVERSE(int x){
digitalWrite(MOTOR_1_2, LOW);
digitalWrite(MOTOR_1_1, HIGH);
analogWrite(MOTOR_1_PWM, x);
}
```

Καλείται όταν θέλουμε το σετ 2 των μοτέρ να εκτελέσει κίνηση προς τα πίσω (Πίνακας 6-23).

Πίνακας 6-23: Η συνάρτηση MOTOR_2_GO_REVERSE

```
void MOTOR_2_GO_REVERSE(int y){
digitalWrite(MOTOR_2_1, LOW);
digitalWrite(MOTOR_2_2, HIGH);
analogWrite(MOTOR_2_PWM, y);
}
```

Καλείται όταν θέλουμε το σετ 1 των μοτέρ να εκτελέσει κίνηση προς τα εμπρός (Πίνακας 6-24).

Πίνακας 6-24: Η συνάρτηση MOTOR_1_GO_FORWARD

```
void MOTOR_1_GO_FORWARD(int x){
digitalWrite(MOTOR_1_1, LOW);
digitalWrite(MOTOR_1_2, HIGH);
analogWrite(MOTOR_1_PWM, x);
}
```

Καλείται όταν θέλουμε το σετ 2 των μοτέρ να εκτελέσει κίνηση προς τα εμπρός (Πίνακας 6-25).

Πίνακας 6-25: Η συνάρτηση MOTOR_2_GO_FORWARD

```
void MOTOR_2_GO_FORWARD(int y){
digitalWrite(MOTOR_2_2, LOW);
digitalWrite(MOTOR_2_1, HIGH);
analogWrite(MOTOR_2_PWM, y);
}
```

Καλείται όταν θέλουμε το σετ 1 των μοτέρ να εκτελέσει παύση κίνησης (Πίνακας 6-26).

Πίνακας 6-26: Η συνάρτηση MOTOR_1_STOP

```
void MOTOR_1_STOP(){
digitalWrite(MOTOR_1_2, LOW);
digitalWrite(MOTOR_1_1, LOW);
digitalWrite(MOTOR_1_PWM, LOW);
}
```

Καλείται όταν θέλουμε το σετ 2 των μοτέρ να εκτελέσει παύση κίνησης (Πίνακας 6-27).

Πίνακας 6-27: Η συνάρτηση MOTOR_2_STOP

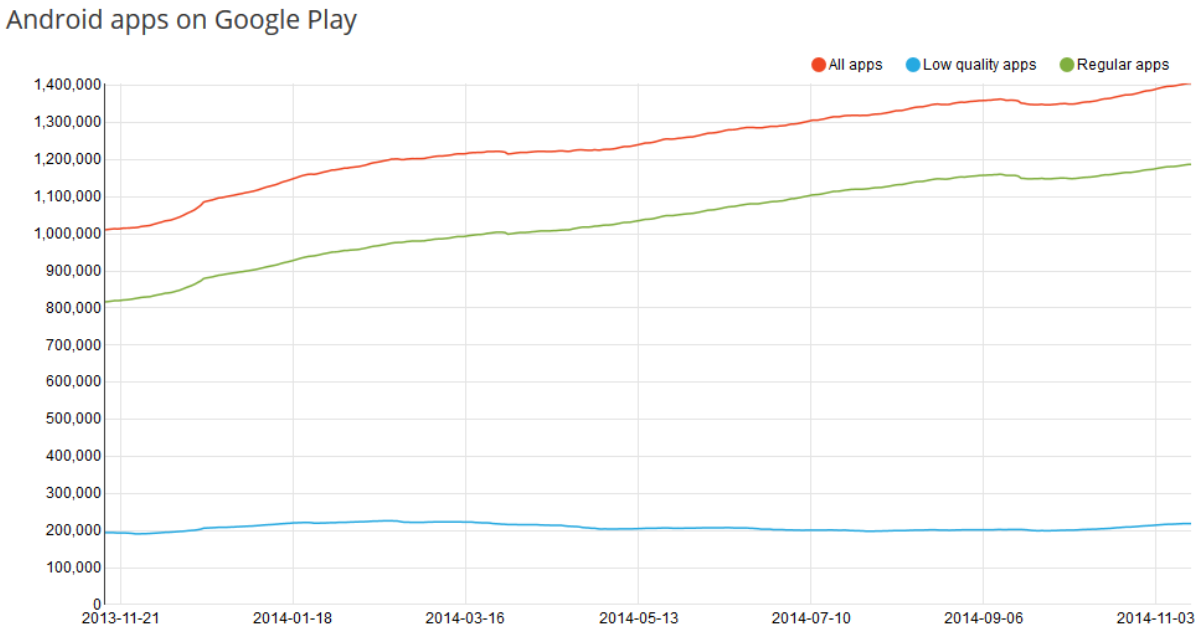
```
void MOTOR_2_STOP(){
digitalWrite(MOTOR_2_2, LOW);
digitalWrite(MOTOR_2_1, LOW);
digitalWrite(MOTOR_2_PWM, LOW);
}
```

7 Το Λειτουργικό Σύστημα Android

Το Android είναι το λειτουργικό σύστημα για κινητά που δημιούργησε η Google, γι' αυτό και τα κινητά που το χρησιμοποιούν συχνά τα αποκαλούμε Android κινητά. Σαν πλατφόρμα είναι η πιο γρήγορα αναπτυσσόμενη σήμερα και χρησιμοποιείται από πολλούς κατασκευαστές κινητών τηλεφώνων, ενώ στην αγορά υπάρχουν πολλές διαθέσιμες συσκευές σε όλες τις κατηγορίες τιμών. Σήμερα το Android θεωρείται σε παγκόσμιο επίπεδο το ίδιο δημοφιλές με το IOS της Apple. Παρακάτω παρουσιάζονται κάποια από τα πλεονεκτήματα του λειτουργικού συστήματος Android.

Εικόνα 7-1: Το λογότυπο του Android

- Υπάρχει μια Android συσκευή για όλες τις απαιτήσεις και κατηγορίες τιμών από διάφορους κατασκευαστές όπως η Samsung, η HTC, η Motorolla, η Sony Ericsson, η LG, η Lenovo κ.α.
- Μεγάλη γκάμα εφαρμογών. Αυτή την στιγμή ο αριθμός των εφαρμογών στο Google store ανέρχεται στις 1.405.737 (Εικόνα 7-2). Μόνο στο μήνα Νοέμβριο ο αριθμός των νέων εφαρμογών έφθασε περίπου τις 20.000. Η αύξηση είναι συνεχής.
- Συγχρονίζεται με τις υπηρεσίες της Google, όπως το Gmail, Contacts, Google Maps κ.α.
- Πρόκειται για μία ολοκληρωμένη, ανοιχτή και δωρεάν πλατφόρμα.
- Αρκετά εύκολο στη χρήση του.



Εικόνα 7-2: Ο αριθμός των Android εφαρμογών στο Google Play

7.1 Η Ιστορία του Android

Τον Οκτώβριο του 2003 στο Palo Alto, πολιτεία της Καλιφόρνιας των ΗΠΑ, ιδρύθηκε η εταιρία Android Inc. από τους Andy Rubin συνιδρυτή του Danger, Rich Miner συνιδρυτή του Wildfire Communications, Inc., Nick Sears αντιπρόεδρο της T-Mobile και Chris White επικεφαλή του σχεδιασμού και της ανάπτυξης διεπαφής στο WebTV. Οι πρώτες προθέσεις της εταιρίας ήταν να αναπτύξει ένα προηγμένο λειτουργικό σύστημα για ψηφιακές φωτογραφικές μηχανές, όταν όμως έγινε αντιληπτό πως η αγορά για αυτού του τύπου τις συσκευές δεν είναι τόσο μεγάλη, ξεκίνησαν νέες προσπάθειες για την παραγωγή ενός λειτουργικού συστήματος για συσκευές smartphones, που τότε κατείχαν πολύ μικρό μερίδιο της αγοράς τηλεφώνων, ικανό να ανταγωνιστεί εκείνα του Symbian και Windows Mobile. Δύο χρόνια αργότερα, τον Αύγουστο του 2005, η Google αγοράζει την Android Inc.

Το 2007 δημιουργήθηκε η Open Handset Alliance (OHA), ένας συνεταιρισμός που αποτελούνταν από τη Google, την HTC, την Samsung, την Qualcomm (κατασκευαστές επεξεργαστών) και άλλους με σκοπό να κάνουν το Android ένα ανοικτό λογισμικό και να κυκλοφορήσουν τα πρώτα smartphones με Android. Πράγματι, το πρώτο smartphone με λειτουργικό Android κυκλοφόρησε ένα χρόνο μετά από την HTC και ονομαζόταν HTC Dream (γνωστό και ως T-Mobile G1). Από εκεί και μετά η ιστορία είναι λίγο πολύ γνωστή. Με πρώτη τη Samsung, δεκάδες εταιρείες υιοθέτησαν το Android ως λειτουργικό και κατάφεραν να το κάνουν το κυρίαρχο λειτουργικό αυτή τη στιγμή στην αγορά. Η αναφορά της Samsung δεν είναι τυχαία, αφού είναι η πρώτη εταιρεία στις πωλήσεις κινητών παγκοσμίως και η εξάπλωση του Android οφείλεται κατά μεγάλο βαθμό σε αυτήν.

Εικόνα 7-3: Η ανάπτυξη του λειτουργικού συστήματος Android σε ενεργοποιήσεις συσκευών

7.2 Οι Εκδόσεις του Λειτουργικού Συστήματος Android

Όλες οι σημαντικές εκδόσεις του Android φέρουν ονομασίες διαφόρων γλυκών επιδορπίων, οι οποίες μάλιστα ακολουθούν και αλφαβητική σειρά. Παρακάτω θα δούμε όλες τις εκδόσεις ξεκινώντας από την παλαιότερη.

- Android 1.0 με ονομασία Apple Pie ,η πρώτη έκδοση κυκλοφόρησε στης 23 Σεπτεμβρίου του 2008, μαζί με τη συσκευή, «HTC Hero». Τόσο το λειτουργικό σύστημα όσο και η συσκευή HTC έλαβαν θετικές κριτικές. Το όνειρο ενός Open-Source λειτουργικού για κινητά τηλέφωνα τελικά έγινε πραγματικότητα. Βασικά χαρακτηριστικά αποτελούν η λήψη και οι ενημερώσεις (Updates) εφαρμογών μέσω του Android Market, Web Browser, υποστήριξη κάμερας, Gmail, επαφές, συγχρονισμός της Agenda του Google, Google maps, εφαρμογή YouTube.
- Έκδοση 1.1 με ονομασία Banana Bread, κυκλοφόρησε στης 9 Φεβρουαρίου του 2009. Βασικά χαρακτηριστικά αποτελούν η δυνατότητα εμφάνισης και απόκρυψης του αριθμητικού πληκτρολόγιού και η αποθήκευση των συνημμένων SMS.
- Android 1.5 με ονομασία Cupcake (Εικόνα 7-4), Κυκλοφόρησε στις 30 Απριλίου του 2009, αυτή ήταν η πρώτη σημαντική αναβάθμιση του Android, προαναγγέλλοντας την έναρξη της "σειράς dessert". Με Cupcake, χαρακτηριστικά,

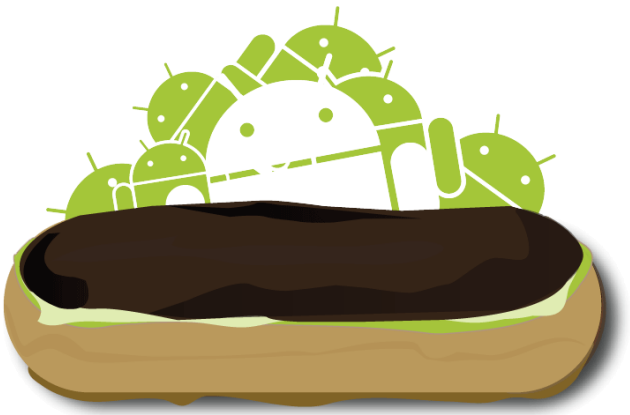
όπως η εγγραφή και παρακολούθηση βίντεο, η πρόβλεψη του κειμένου, η ασύρματη μετάδοση μουσικής και η υποστήριξη Bluetooth A2DP, AVRCP .

Εικόνα 7-4: Λογότυπο Android Cupcake 1.5

- Android 1.6 με ονομασία Donut (Εικόνα 7-5), Κυκλοφόρησε στις 15 Σεπτεμβρίου του 2009, Η έκδοση Donut ήρθε με σημαντικές αναβαθμίσεις, το αποκορύφωμα της οποίας ήταν το turn-by-turn navigation και Gesture framework. Επίσης ενίσχυσε τις δυνατότητες φωτογραφίας και βίντεο.

Εικόνα 7-5: Λογότυπο Android Donut 1.6

- Android 2.0 με ονομασία Éclair (Εικόνα 7-6), Η έκδοση 2.0 κυκλοφόρησε στις 26 Δεκεμβρίου του 2009, ακολουθούμενη από την 2.1 στις 12 Ιανουαρίου του 2010. Η αναβάθμιση αυτή είχε σημαντικές βελτιώσεις επιτρέποντας διάφορες δυνατότητες Bluetooth 2.1, multi-touch υποστήριξη και live wallpapers, HTML, ψηφιακό zoom, υποστήριξη του Microsoft exchange και αναβαθμισμένο UI μεταξύ των άλλων χαρακτηριστικών.



Εικόνα 7-6: Λογότυπο Android Eclair 2.0

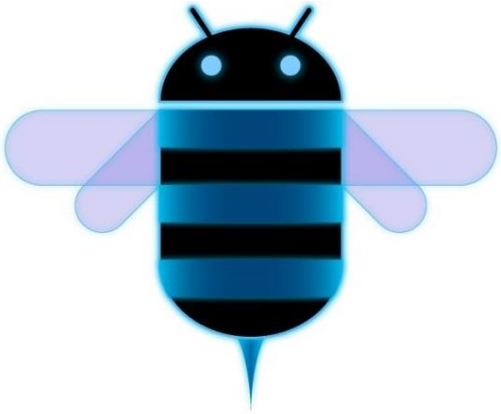
- Android 2.2 με ονομασία Froyo (Εικόνα 7-7), Συντόμευση για το "Frozen Yoghurt", αυτή η έκδοση κυκλοφόρησε στις 20 Μαΐου του 2010. Έφερε βελτιωμένη ταχύτητα στο android, με την υποστήριξη υψηλής ευκρίνειας σε αναλύσεις οθόνης και Adobe Flash 10.1, επιτρέποντας στους χρήστες να βλέπουν ροή βίντεο στο κινητό τους μέσω browsers. Προστέθηκε υποστήριξη για Wi-Fi hotspot συνδεσιμότητα, Usb tethering, εγκατάσταση εφαρμογών στην μνήμη επέκτασης, υποστήριξη upload στο πρόγραμμα περιήγησης, κινούμενες εικόνες GIF.

Εικόνα 7-7: Λογότυπο Android Froyo 2.2

- Android 2.3 με ονομασία Gingerbeard (Εικόνα 7-8), Αυτή η έκδοση έκανε το ντεμπούτο της στις 6 Δεκεμβρίου του 2010. Το βασικό χαρακτηριστικό που έγινε διαθέσιμο με αυτή την έκδοση ήταν το Near Field Communication (NFC), επιτρέποντας στους χρήστες να εκτελούν διάφορα, όπως πληρωμές μέσω κινητού τηλεφώνου και την ανταλλαγή δεδομένων μέσω αυτών περνώντας τα κινητά τους τηλέφωνα πάνω από μια ετικέτα. Προσέθεσε επίσης την υποστήριξη για περισσότερες από μία κάμερες και διάφορους άλλους αισθητήρες. Διαθέτει αναβαθμισμένο UI, βελτιωμένη λειτουργία copy/paste, καλύτερη διαχείριση ενέργειας, χαρακτηριστικά κοινωνικής δικτύωσης, υποστήριξη VOIP/SIP και βιντεοκλήσης.

Εικόνα 7-8: Λογότυπο Android Gingerbeard 2.3

- Android 3.0 με ονομασία Honeycomp (Εικόνα 7-9), Κυκλοφόρησε στις 22 Φεβρουαρίου του 2011 και ακολούθησε γρήγορα η 3.1 και 3.2 τον Μάιο και Ιούλιο αντίστοιχά του ίδιου έτους. Αυτή η έκδοση είχε βελτιστοποιηθεί για ταμπλέτες (Tablets). Βασικά χαρακτηριστικά της νέας έκδοσης αποτελούν η υποστήριξη πολλαπλών πυρήνων, η προσαρμόσιμη οθόνη υποδοχής, η προβολή πρόσφατων εφαρμογών, το Google Talk video chat, τα Google eBooks, η ιδιωτική περιήγηση στον ιστό, και το HTTP Live Streaming.



Εικόνα 7-9: Λογότυπο Android Honeycomp 3.0

- Android 4.0 με ονομασία Ice Cream Sandwich (Εικόνα 7-10), Κυκλοφόρησε στις 19 Οκτωβρίου του 2011. Χαρακτηριστικά της έκδοσης αυτής αποτελούν η βελτιωμένη εισαγωγή κειμένου και ο ορθογραφικός έλεγχος, οι εφαρμογές e-mail υποστηρίζουν EAS v14, άμεση πρόσβαση σε Wi-Fi, Bluetooth health device profile.

Εικόνα 7-10: Λογότυπο Android Ice Cream Sandwich 4.0

- Android 4.1 με ονομασία Jelly Bean (Εικόνα 7-11), Η τελευταία μεγάλη αναβάθμιση του Android κυκλοφόρησε στις 9 Ιουλίου του 2012. Ακόμη πιο όμορφο το Android UI, αλλά και εξεζητημένο λογισμικό, επιτρέποντας στις Android συσκευές να τρέξουν πιο γρήγορα. Οι 4.x εκδόσεις επέτρεψαν στους προγραμματιστές να δημιουργήσουν εφαρμογές με ποιότητα, επιβεβαιώνοντας τη θέση της ως το λειτουργικό σύστημα που προτιμούν οι χρήστες. Πρόσθετα χαρακτηριστικά αποτελούν η φωνητική αναζήτηση, βελτιώσεις στην εφαρμογή της κάμερας, βελτιώσεις ταχύτητας, επιλογή gesture η οποία επιτρέπει την σύνδεση πληκτρολογίου μπράιγ.

Εικόνα 7-11: Λογότυπο Android Jelly Bean 4.1

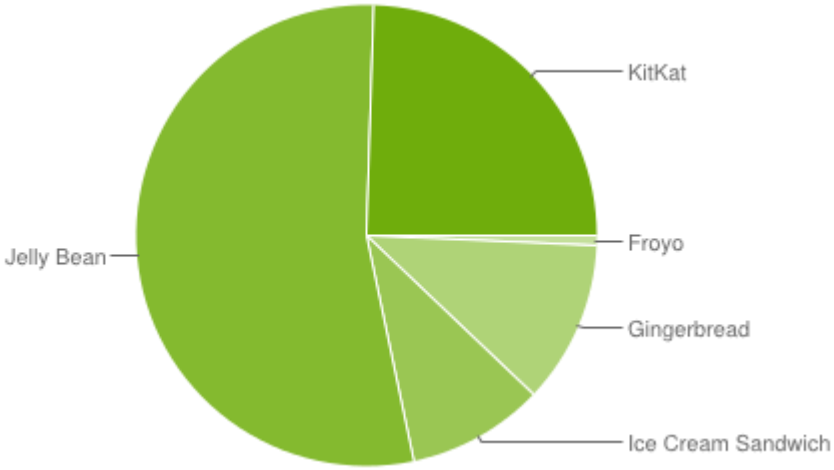
- Android 4.4 με ονομασία KitKat (Εικόνα 7-12), κυκλοφόρησε στις 31 Οκτωβρίου του 2013. Η νέα έκδοση του Android 4.4 έχει τόσες σημαντικές και πολλές βελτιώσεις που αξίζει να σταθούμε στα βασικά. Το UI και UX έχουν αλλάξει και έχουν βελτιωθεί στα σημεία. Αυτό σημαίνει ότι τα γραφικά είναι λίγο διαφορετικά και η εμπειρία χρήσης του Android KitKat καλύτερη από ποτέ. Οι διαφορές θα γίνουν αμέσως ορατές όταν οι κάτοχοι Nexus 4 κάνουν αναβάθμιση στο Android 4.4 γιατί για όλους τους άλλους κατόχους, Samsung smartphones και LG smartphones με custom skins, τα πράγματα είναι διαφορετικά. Σημαντική αλλαγή που φαίνεται από την πρώτη ματιά είναι η απώλεια των notification bar και control bar, έχουν εξφανιστεί και έχουν γίνει διάφανες.

Εικόνα 7-12: Λογότυπο Android KitKat 4.4

Πίνακας 7-1: Συγκεντρωτικός πίνακας εκδόσεων Android, ξεκινώντας από την πιο πρόσφατη

Έκδοση Android	**Όνομα Έκδοσης**	**Ημερομηνία Κυκλοφορίας**	**Ποσοστά Χρήσης**	**Επίπεδο API**
Android 4.4.4	KitKat	23-6-2014	13.6 % (4.4 - 4.4.4)	19
Android 4.4.3	KitKat	14-4-2014	13.6 % (4.4 - 4.4.4)	19
Android 4.4.2	KitKat	9-12-2013	13.6 % (4.4 - 4.4.4)	19
Android 4.4.1	KitKat	5-12-2013	13.6 % (4.4 - 4.4.4)	19
Android 4.4	KitKat	31-10-2013	13.6 % (4.4 - 4.4.4)	19
Android 4.3	Jelly Bean	24-7-2013	10.3 % (4.3)	18
Android 4.2.2	Jelly Bean	11-2-2013	19.1 % (4.2 - 4.2.2)	17
Android 4.2.1	Jelly Bean	27-11-2012	19.1 % (4.2 - 4.2.2)	17
Android 4.2	Jelly Bean	13-11-2012	19.1 % (4.2 - 4.2.2)	17
Android 4.1.2	Jelly Bean	9-10-2012	29 % (4.1 - 4.1.2)	16
Android 4.1.1	Jelly Bean	23-7-2012	29 % (4.1 - 4.1.2)	16
Android 4.1	Jelly Bean	9-7-2012	29 % (4.1 - 4.1.2)	16
Android 4.0.4	Ice Cream Sandwich	28-3-2012	12.3 % (4.0.3 - 4.0.4)	15
Android 4.0.3	Ice Cream Sandwich	16-12-2011	12.3 % (4.0.3 - 4.0.4)	15
Android 4.0.2	Ice Cream Sandwich	28-11-2011	0 %	14
Android 4.0.1	Ice Cream Sandwich	19-10-2011	0 %	14
Android 4.0	Ice Cream Sandwich	19-10-2011	0 %	14
Android 3.2.6	Honeycomb	15-2-2012	0 %	13
Android 3.2.4	Honeycomb	15-12-2011	0 %	13
Android 3.2.2	Honeycomb	30-9-2011	0 %	13
Android 3.2.1	Honeycomb	20-9-2011	0 %	13
Android 3.2	Honeycomb	15-7-2011	0 %	13
Android 3.1	Honeycomb	10-5-2011	0 %	12
Android 3.0	Honeycomb	22-2-2011	0 %	11

Android 2.3.4	Gingerbread	10-5-2011	14.9 % (2.3.3 - 2.3.7)	10
Android 2.3.7	Gingerbread	21-9-2011	14.9 % (2.3.3 - 2.3.7)	10
Android 2.3.6	Gingerbread	2-9-2011	14.9 % (2.3.3 - 2.3.7)	10
Android 2.3.5	Gingerbread	25-7-2011	14.9 % (2.3.3 - 2.3.7)	10
Android 2.3.3	Gingerbread	9-2-2011	14.9 % (2.3.3 - 2.3.7)	10
Android 2.3	Gingerbread	6-12-2010	0 % (2.3 - 2.3.2)	9
Android 2.2	Froyo	20-5-2010	0.8 % (2.2)	8
Android 2.1	Eclair	12-1-2010	0 %	7
Android 2.0.1	Eclair	3-12-2009	0 %	6
Android 2.0	Eclair	26-10-2009	0 %	5
Android 1.6	Donut	15-9-2009	0 %	4
Android 1.5	Cupcake	30-4-2009	0 %	3
Android 1.1	Banana bread	9-2-2009	0 %	2
Android 1.0	Apple pie	23-9-2008	0 %	1
Android 0.9		18-8-2008	0 %	

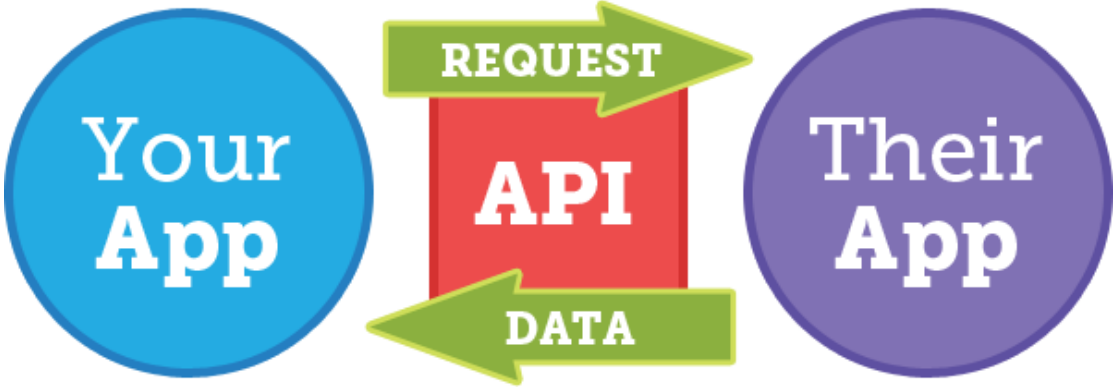


Εικόνα 7-13: Δημοφιλέστερες εκδόσεις

https://developer.android.com/about/dashboards/index.html

7.2.1 Η Διεπαφή Προγραμματισμού Εφαρμογών

Η διεπαφή προγραμματισμού εφαρμογών (API) είναι ένα συγκεκριμένο σύνολο κανόνων και προδιαγραφών που μία εφαρμογή μπορεί να ακολουθήσει για να έχει πρόσβαση σε υπηρεσίες και πόρους, και να κάνει χρήση αυτών, τα οποία παρέχονται από κάποια άλλη συγκεκριμένη εφαρμογή που υλοποιεί αυτή τη διεπαφή. Χρησιμεύει ως διεπαφή μεταξύ διαφορετικών εφαρμογών και διευκολύνει την αλληλοεπίδραση τους, παρόμοια με τον τρόπο που η διεπαφή χρήστη διευκολύνει την αλληλοεπίδραση μεταξύ των ανθρώπων και των υπολογιστών. Με απλά λόγια αποτελεί τον τρόπο επικοινωνίας μίας εφαρμογής με το λειτουργικό σύστημα ή κάποια άλλη εφαρμογή (Εικόνα 7-14).

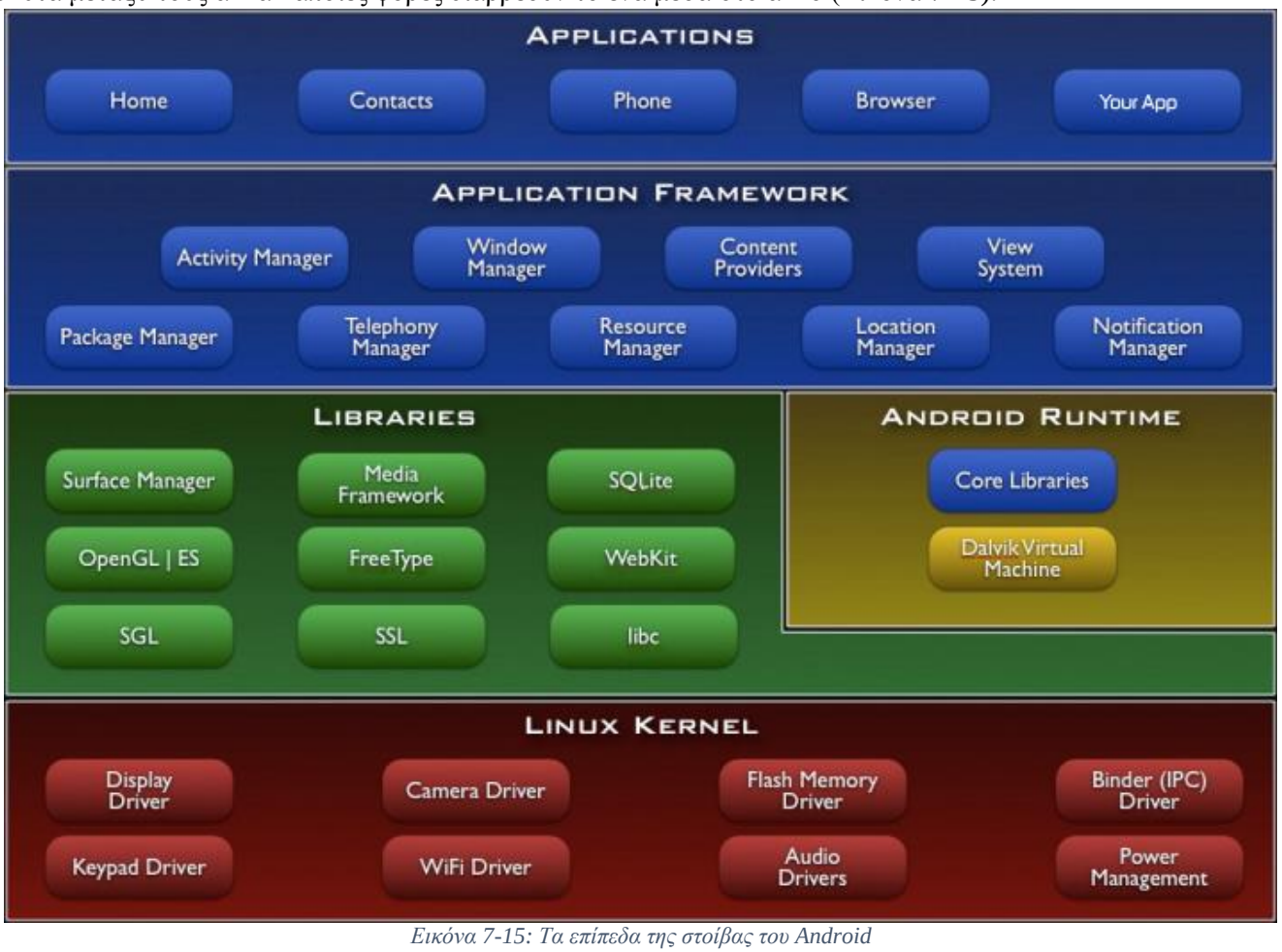


Εικόνα 7-14: Application Programming Interface

Το API Level που εμφανίζεται σε κάθε έκδοση Android, είναι ένας ακέραιος αριθμός που προσδιορίζει μοναδικά την έκδοση πλαισίου του API το οποίο προσφέρεται από μία έκδοση της πλατφόρμας Android.

7.3 Η Αρχιτεκτονική του Λειτουργικού Συστήματος Android

Θα μπορούσαμε να παρουσιάσουμε το λειτουργικό σύστημα Android σαν μία τούρτα που αποτελείται από διάφορα επίπεδα (layers). Κάθε layer έχει τα δικά του χαρακτηριστικά και σκοπό, τα οποία όμως δεν διαχωρίζονται απόλυτα μεταξύ τους αλλά κάποιες φορές διαρρέουν το ένα μέσα στο άλλο (Εικόνα 7-15).



Εικόνα 7-15: Τα επίπεδα της στοίβας του Android

7.3.1 Linux Kernel

Το βασικό επίπεδο είναι ο πυρήνας Linux (Linux Kernel). Όλο το λειτουργικό σύστημα Android είναι χτισμένο πάνω στον πυρήνα του Linux έκδοσης 2.6 με κάποιες περαιτέρω αλλαγές πάνω στην αρχιτεκτονική η οποίες έγιναν από την Google. Το Linux είναι αυτό που αλληλοεπιδρά με το Hardware και περιέχει όλα τα απαραίτητα προγράμματα οδήγησης για την επικοινωνία και τον έλεγχο του υλικού. Ως παράδειγμα μπορούμε να σκεφτούμε την λειτουργία του Bluetooth, ο πυρήνας (kernel) πρέπει να διαθέτει τους απαραίτητους οδηγούς του Bluetooth οι οποίοι θα καταστήσουν δυνατή την επικοινωνία με το hardware κομμάτι του Bluetooth. Το Android χρησιμοποιεί το Linux για όλες τις βασικές λειτουργίες όπως την διαχείριση της μνήμης, τη διαχείριση των διαδικασιών (processes), τη δικτύωση, τις ρυθμίσεις ασφαλείας κτλ. Τέλος υπάρχουν αρκετοί καλοί λόγοι για τους οποίους έχει γίνει η επιλογή του Linux ως βάση της στοίβας του λειτουργικού συστήματος Android. Κάποιοι από τους βασικότερους είναι η φορητότητα, η ασφάλεια και το πλήθος των διαθέσιμων χαρακτηριστικών.

7.3.2 Βιβλιοθήκες

Το επόμενο επίπεδο αποτελείται από τις βιβλιοθήκες του λειτουργικού συστήματος Android. Είναι αυτό το επίπεδο που επιτρέπει στη συσκευή να χειριστεί διαφορετικούς τύπους δεδομένων. Είναι γραμμένες στη γλώσσα

C/C++, συχνά λαμβάνονται από την κοινότητα ανοιχτού κώδικα προκειμένου να παρέχουν τις απαραίτητες υπηρεσίες στο επίπεδο εφαρμογής του λειτουργικού συστήματος Android. Οι σημαντικότερες βιβλιοθήκες είναι οι εξής :

- Surface Manager. Η βιβλιοθήκη αυτή χρησιμοποιείται για τη διαχείριση της οθόνης. Είναι υπεύθυνη για τη σύνθεση διαφορετικών επιφανειών σχεδίασης στην οθόνη. Επιτρέπει σε διάφορες διαδικασίες τη σύνθεση γραφικών 2D και 3D επιπέδου.
- Graphic Libraries. Η Open GL|ES και η SGL είναι δύο βασικές βιβλιοθήκες γραφικών. Η OpenGL|ES είναι μία βιβλιοθήκη 3D γραφικών, ενώ η SGL αποτελεί βιβλιοθήκη για 2D γραφικά.
- Media Framework. Το media framework (πλαίσιο εφαρμογής) περιλαμβάνει όλα τα codecs που απαιτούνται για την εγγραφή και αναπαραγωγή πολυμέσων.
- SSL. Χρησιμοποιείται για την ασφάλεια στο διαδίκτυο.
- SQLite. Είναι η μηχανή της βάσης δεδομένων που χρησιμοποιείται στο android για τους σκοπούς της αποθήκευσης δεδομένων.
- WebKit. Μηχανή περιήγησης ανοιχτού κώδικα.

7.3.3 Η Εικονική Μηχανή Dalvik

Ο Dalvik είναι μία συγκεκριμένου σκοπού εικονική μηχανή σχεδιασμένη συγκεκριμένα για το λειτουργικό σύστημα του Android, αναπτύχθηκε από τον Dan Bornstein και την ομάδα του στην εταιρία της Google. Το όνομα προέρχεται από το ψαροχώρι του Dalvik στην Eyjafjörður, Ισλανδία.

Κάθε Android εφαρμογή εκτελείται σε μία ξεχωριστή διεργασία, με το δικό της στιγμιότυπο της εικονικής μηχανής Dalvik. Ο τύπος μεταγλώττισης που χρησιμοποιεί ονομάζεται Just in-Time ή αλλιώς JIT. Με τον JIT μεταγλωττιστή, κάθε φορά που εκτελείτε μία εφαρμογή, μεταφράζει δυναμικά ένα μέρος του Dalvik bytecode σε κώδικα μηχανής. Καθώς η εκτέλεση συνεχίζεται, περισσότερο bytecode μεταγλωττίζεται και αποθηκεύεται προσωρινά.

Η εικονική μηχανή της Java σχεδιάστηκε για να είναι μία «γενική» λύση και η ομάδα Dalvik θεώρησε ότι θα μπορούσε να κάνει μία καλύτερη δουλειά εστιάζοντας αποκλειστικά σε κινητές συσκευές. Εξέτασαν τους περιορισμούς εκείνους που αφορούν τις κινητές συσκευές και οι οποίοι είναι το λιγότερο πιθανό να αλλάξουν στο κοντινό μέλλον. Ένας από αυτούς είναι η διάρκεια ζωής της μπαταρίας, και ο άλλος η επεξεργαστική ισχύς. Το Dalvik χτίστηκε από το μηδέν για την αντιμετώπιση αυτών των περιορισμών.

Ένας άλλος λόγος για την αντικατάσταση της εικονικής μηχανής Java με την εικονική μηχανή Dalvik είναι η αδειοδότηση. Ενώ η γλώσσα Java, τα εργαλεία Java και οι βιβλιοθήκες της Java είναι δωρεάν η εικονική μηχανή της Java δεν είναι. Αυτό ήταν ένα μεγάλο πρόβλημα το 2005 όταν ξεκίνησε η προετοιμασία του Dalvik. Σήμερα υπάρχουν ενναλακτικές πηγές ανοικτού κώδικα, αντί της εικονικής μηχανής Java, με ονομασία OpenJDK και Apache Harmony.

Με την ανάπτυξη μίας πραγματικά ανοικτού κώδικα και «δωρεάν» εικονικής μηχανής, το Android παρέχει για μία ακόμη φορά μία πλήρως εξοπλισμένη πλατφόρμα που άλλοι ενθαρρύνονται να υιοθετήσουν για ένα πλήθος συσκευών χωρίς να χρειάζεται να ανησυχούν για την αδειοδότηση.

7.3.4 ART versus Dalvik

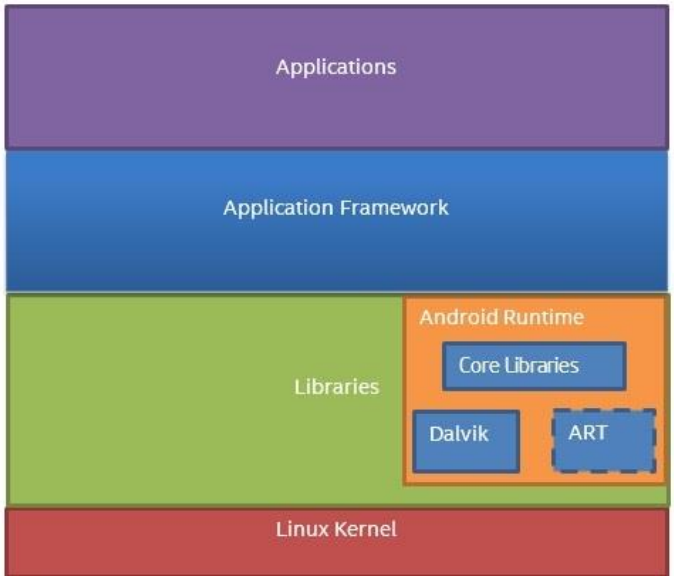
Ένα από τα σημαντικότερα χαρακτηριστικά που μας έφερε η έκδοση Android KitKat 4.4 είναι ο νέος Ahead-of-Time μεταγλωττιστής ή αλλιώς ART. Σε αντίθεση λοιπόν με τον Dalvik, ο ART κατά την διάρκεια της εγκατάστασης μίας εφαρμογής, μεταφράζει στατικά το DEX bytecode σε κώδικα μηχανής και το αποθηκεύει σε χώρο αποθήκευσης της συσκευής. Αυτό το γεγονός συμβαίνει μόνο μία φορά και αποκλειστικά κατά την διάρκεια της εγκατάστασης της εφαρμογής και έτσι δε χρειάζεται να επαναληφθεί η διαδικασία κάθε φορά που ανοίγουμε την εφαρμογή μας.

Εικόνα 7-16: ART vs Dalvik

Το σημαντικότερο όφελος του ART runtime έναντι του JIT είναι ότι η εφαρμογή θα τρέξει ταχύτερα στο ART. Επειδή το DEX bytecode έχει μεταφραστεί σε κώδικα μηχανής κατά η διάρκεια της εγκατάστασης, δεν χρειάζεται επιπλέον χρόνος για μεταγλώττιση κατά τη διάρκεια εκτέλεσης της εφαρμογής. Έτσι οι εφαρμογές μας ανοίγουν πιο γρήγορα σε σχέση με το Dalvik. Επίσης το Dalvik εκμεταλλεύεται μεγαλύτερο μέρος της μνήμης λόγο της ανάγκης για προσωρινή αποθήκευση του JIT κώδικα, πράγμα που δεν συμβαίνει με το ART. Επίσης με το ART έχουμε μεγαλύτερη διάρκεια ζωής της μπαταρίας.

Ωστόσο υπάρχουν και τα αρνητικά του ART. Ο χρόνος εγκατάστασης μίας εφαρμογής με ART runtime είναι μεγαλύτερος. Όπως είπαμε και παραπάνω, το ART κατά την διάρκεια της εγκατάστασης μίας εφαρμογής, μεταφράζει στατικά το DEX bytecode σε κώδικα μηχανής και το αποθηκεύει σε χώρο αποθήκευσης της συσκευής που σημαίνει μεγαλύτερη κατανάλωση αποθηκευτικού χώρου σε σχέση με το Dalvik.

Τέλος, να αναφέρουμε πως το ART είναι ακόμα σε beta έκδοση αλλά η Google το δουλεύει εδώ και 2 χρόνια και σύντομα θα αντικαταστήσει το Dalvik ως default επιλογή (Εικόνα 7-17).



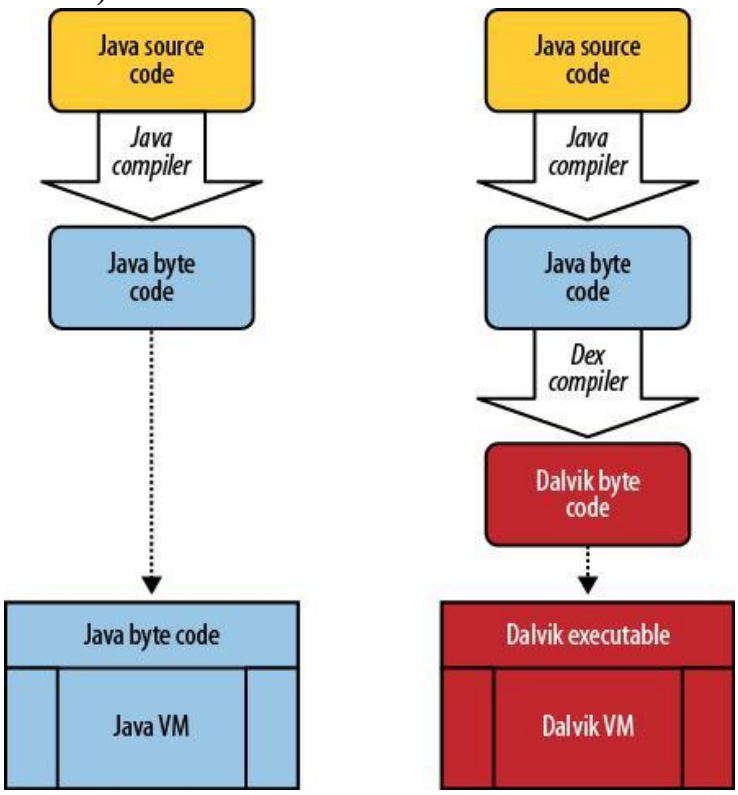
Εικόνα 7-17: Η πειραματική προσθήκη του "ART" στην έκδοση Android KitKat 4.4

7.3.5 Το Android και η Java

Στην Java, γράφουμε το πηγαίο αρχείο μας, το κάνουμε compile σε Java bytecode χρησιμοποιώντας το μεταγλωττιστή της Java και στη συνέχεια εκτελούμε αυτόν τον κώδικα από byte (Java Byte Code) στην εικονική μηχανή της Java. Ωστόσο στο Android τα πράγματα είναι διαφορετικά.

Εξακολουθούμε να γράφουμε το πηγαίο αρχείο μας σε Java, και εξακολουθούμε να το κάνουμε compile χρησιμοποιώντας τον compiler της Java. Αλλά σε αυτό το σημείο, το μεταγλωττίζουμε ξανά χρησιμοποιώντας τον Dalvik

compiler για Dalvik bytecode. Ο Dalvik bytecode (αρχεία «.dex») είναι αυτός που στη συνέχεια εκτελείται στην Dalvik εικονική μηχανή (Εικόνα 7-18).



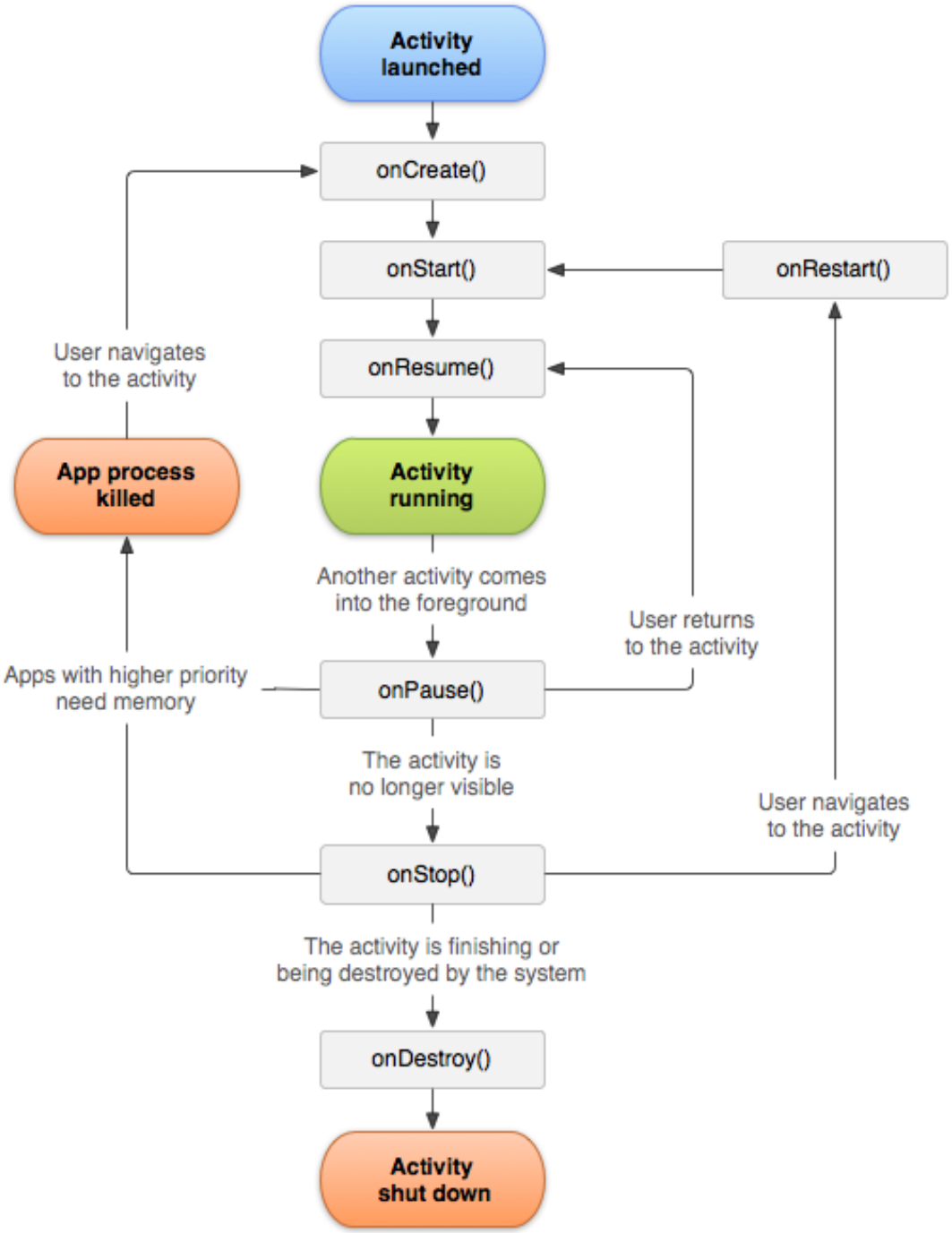
Εικόνα 7-18: Διαφορά μεταξύ Java και Dalvik

7.3.6 Το Πλαίσιο Εφαρμογής

Το application framework (πλαίσιο εφαρμογής) είναι ένα πλούσιο περιβάλλον που παρέχει πολλές υπηρεσίες για να βοηθήσει τον προγραμματιστή να τελειώση την εφαρμογή του. Το επίπεδο αυτό δίνει τη δυνατότητα στους προγραμματιστές να γίνουν δημιουργικοί και να φέρουν φανταστικές εφαρμογές στην αγορά.

Στο επίπεδο αυτό, υπάρχουν πολλές βιβλιοθήκες της Java ειδικά κατασκευασμένες για το Android. Επίσης θα βρούμε πολλές υπηρεσίες όπου η εφαρμογή μας μπορεί να αξιοποιήση, όπως τοποθεσία, αισθητήρες, Wi-Fi, τηλεφωνία κτλ. Τα σημαντικότερα συστατικά στοιχεία του επιπέδου αυτού είναι τα παρακάτω:

- Activity Manager: Διαχειρίζεται τον κύκλο ζωής όλων των εφαρμογών.
- Packages Manager: Παρακολουθεί ποιες εφαρμογές είναι εγκατεστημένες στην συσκευή μας.
- Window Manager: Διαχειρίζεται τα διάφορα παράθυρα.
- Telephony Manager: Διαχειρίζεται όλες τις φωνητικές κλήσεις. Το χρησιμοποιούμε αν θέλουμε να έχουμε πρόσβαση σε φωνητικές κλήσεις στην εφαρμογής μας.
- Content Providers: Επιτρέπει σε εφαρμογές να μοιράζονται δεδομένα με άλλες εφαρμογές.
- Resources Manager: Χρησιμοποιείται για την διαχείριση των πόρων μίας εφαρμογής.
- View System: Περιέχει δομικά στοιχεία του Android UI.
- Location Manager: Διαχείριση περιοχής, χρησιμοποιώντας GPS ή πύργο κινητής τηλεφωνίας.
- Notification Manager: Διαχειρίζεται διαφορετικές ειδοποιήσεις.
- XMPP Manager: Χρησιμοποιείται για τη διαχείριση της υπηρεσίας XMPP (Extensible Messaging and Presence Protocol).



Εικόνα 7-19: Κύκλος ζωής μίας Android εφαρμογής

http://developer.android.com/guide/components/activities.html

Πίνακας 7-2: Σύντομη περιγραφή των μεθόδων του κύκλου ζωής μίας εφαρμογής

Μέθοδος	Περιγραφή	Μπορεί να καταστραφεί	Επόμενη Μέθοδος
onCreate()	Καλείται όταν η δραστηριότητα δημιουργείται για πρώτη φορά. Εδώ κάνουμε όλες τις στατικές αρχικοποιήσεις. Πάντα ακολουθεί η onStart().	Όχι	onStart()
onRestart()	Καλείται μετά που η δραστηριότητα έχει σταματήσει, ακριβώς πριν ξεκινήσει ξανά. Πάντα ακολουθεί η onStart().	Όχι	onStart()
onStart()	Καλείται ακριβώς πριν η δραστηριότητα γίνει εμφανής στον χρήστη. Ακολουθεί η onResume() αν η δραστηριότητα γίνεται στο προσκήνιο, ή η onStop() αν γίνεται κρυφά.	Όχι	onResume() ή onStop()
onResume()	Καλείται ακριβώς πριν αρχίσει η αλληλοεπίδραση της δραστηριότητας με τον χρήστη. Πάντα ακολουθεί η onPause().	Όχι	onPause()
onPause()	. Καλείται όταν το σύστημα είναι έτοιμο να ξεκινήσει την επανάληψη μιας άλλης δραστηριότητας. Σταματάει οτιδήποτε περιττό καταναλώνει πόρους από την CPU. Ακολουθεί η onResume αν η δραστηριότητα επιστρέψει στο προσκήνιο, αλλιώς η onStop() αν παραμείνει αόρατη προς τον χρήστη.	Ναι	onResume() ή onStop()
onStop()	Καλείται όταν η δραστηριότητα δεν είναι πλέον ορατή στον χρήστη, αυτό μπορεί να συμβεί είτε διότι καταστρέφεται είτε επειδή μία άλλα δραστηριότητα έχει επαναληφθεί και την καλύπτει. Ακολουθεί η onRestart αν η δραστηριότητα επιστρέφει για να αλληλοεπιδράσει με το χρήστη ή η onDestroy() αν πρόκειται να τερματιστεί.	Ναι	onRestart() ή onDestroy()
onDestroy()	Καλείται πριν η δραστηριότητα καταστραφεί. Αυτή είναι η τελευταία κλήση που θα λάβει η δραστηριότητα. Η μέθοδος αυτή καλείται είτε γιατί η δραστηριότητα τερματίζεται(έγινε κλήση της finish()), είτε γιατί το σύστημα την καταστρέφει προσωρινά για να εξοικονομήσει χώρο.	Ναι	Καμία

7.3.7 Εφαρμογές

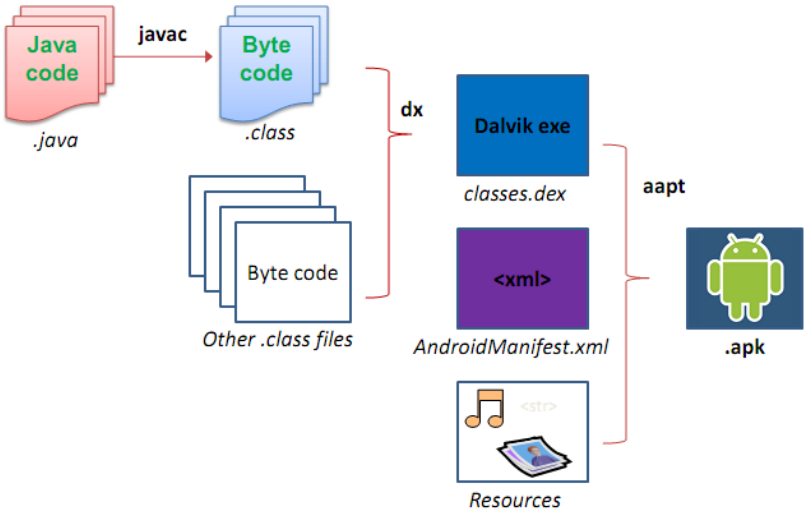
Τέλος, σε αυτό το επίπεδο βρίσκονται οι εφαρμογές που δημιουργούμε εμείς και άλλοι προγραμματιστές. Έρχονται προ εγκατεστημένες στη συσκευή ή μπορούμε να τις κατεβάσουμε από το Android Market.

Εικόνα 7-20: Το επίπεδο ανάπτυξης εφαρμογών

7.3.7.1 Το Αρχείο APK

Μία εφαρμογή είναι ένα ενιαίο αρχείο πακέτου εφαρμογών. Ένα αρχείο APK έχει περίπου τρείς κύριες συνιστώσες.

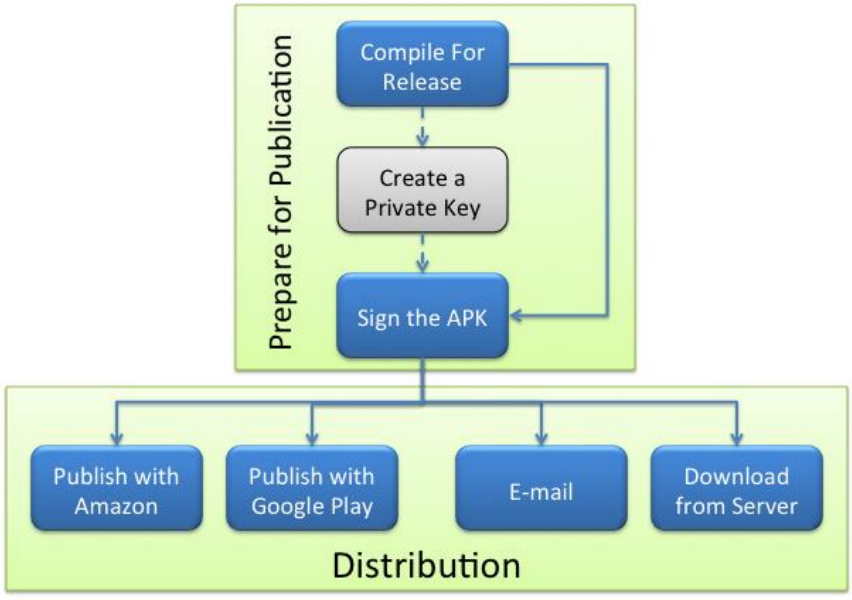
- Το εκτελέσιμο αρχείο Dalvik: Περιλαμβάνει όλο τον πηγαίο κώδικα σε Java μεταγλωττισμένο σε εκτελέσιμο αρχείο Dalvik (Dalvik executable, «.dex»). Αυτός είναι ο κώδικας που τρέχει την εφαρμογή μας.
- Πόροι (Resources): Οι πόροι είναι ότι δεν είναι κώδικας. Η εφαρμογή μας μπορεί να περιέχει ένα αριθμό από εικόνες, βίντεο, ήχο καθώς και πολυάριθμα αρχεία XML που περιγράφουν layouts, πακέτα γλώσσας κτλ. Συλλογικά, αυτά τα στοιχεία είναι οι πόροι.
- Βιβλιοθήκες: Προαιρετικά, η εφαρμογή μας μπορεί να περιλαμβάνει κάποιο εγγενή κώδικα, όπως οι βιβλιοθήκες C/C++. Αυτές οι βιβλιοθήκες μπορούν να πακεταριστούν μαζί με το αρχείο APK.



Εικόνα 7-21: Δομή του αρχείου .apk

7.3.7.2 Η Υπογραφή μίας Εφαρμογής

Το Android απαιτεί όλες οι εφαρμογές να υπογραφούν ψηφιακά με ένα πιστοποιητικό έτσι ώστε να μπορέσουν να εγκατασταθούν (Εικόνα 7-22). Το Android χρησιμοποιεί αυτό το πιστοποιητικό για να προσδιορίσει τον δημιουργό της εφαρμογής, και το πιστοποιητικό δεν χρειάζεται να έχει υπογραφεί από κάποια αρχή έκδοσης πιστοποιητικών. Οι Android εφαρμογές συχνά χρησιμοποιούν αυτό-υπογεγραμμένα πιστοποιητικά. Ο προγραμματιστής κατέχει το ιδιωτικό κλειδί του πιστοποιητικού.

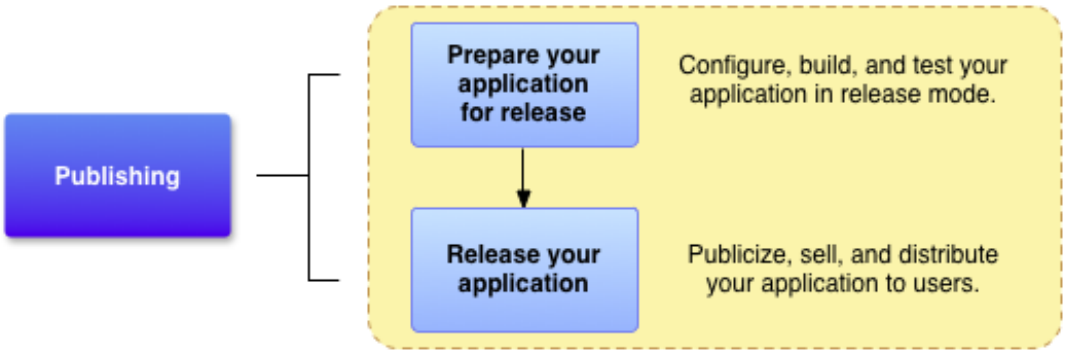


Εικόνα 7-22: Η υπογραφή μίας εφαρμογής και η δημοσίευση της

7.3.7.3 Δημοσίευση μίας εφαρμογής

Το Android διαφέρει στον τρόπο που διανέμει τις εφαρμογές του σε σχέση με άλλες εταιρίες. Για παράδειγμα στο IOS, το λειτουργικό σύστημα του IPhone, το μονοπώλιο για την διανομή των εφαρμογών το κατέχει η ίδια η Apple μέσω του App Store. Στο Android υπάρχουν πολλά διαφορετικά καταστήματα ή αγορές με κάθε μία να ακολουθεί την δική της πολιτική. Ως εκ τούτου, το Android αποτελεί μία ελεύθερη αγορά.

Στην πράξη, η μεγαλύτερη αγορά αυτή τη στιγμή είναι το Android Market, το οποίο διαχειρίζεται η ίδια η Google. Οι εφαρμογές μπορούν να διανεμηθούν και μέσω του διαδικτύου. Όταν κάνουμε λήψη ενός αρχείου APK από μία ιστοσελίδα, η εφαρμογή μας εγκαθίσταται αυτόματα στην συσκευή μας.



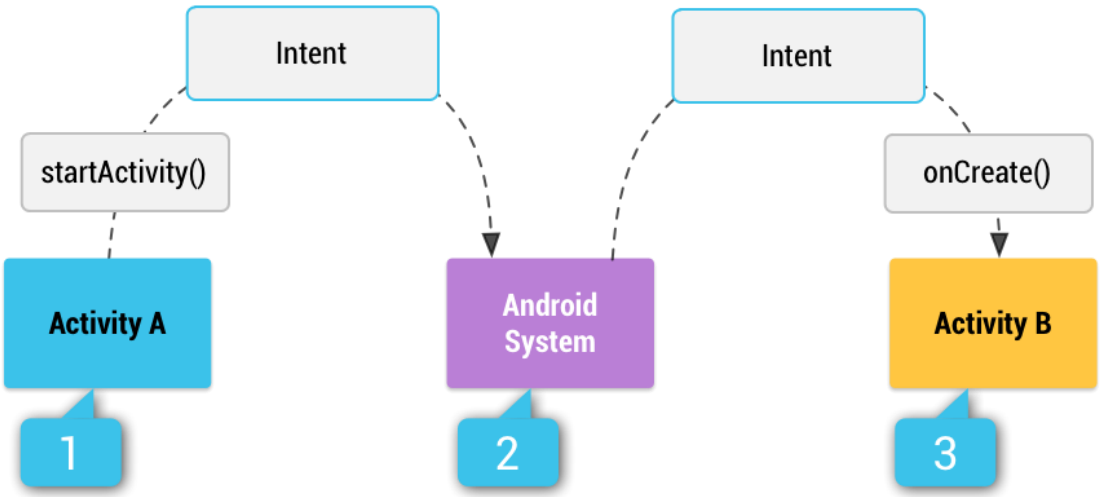
Εικόνα 7-23: Η δημοσίευση μίας Android εφαρμογής.

http://developer.android.com/tools/publishing/publishing_overview.html#publishing-prepare

7.4 Τα Συστατικά μίας Εφαρμογής

Οι εφαρμογές στο Android αποτελούνται από κάποια συστατικά που συνδέονται μεταξύ τους από το *AndroidManifest.xml*, το οποίο περιγράφει κάθε συστατικό αλλά και πως όλα τα συστατικά αλληλοεπιδρούν μεταξύ τους. Επίσης, στο *AndroidManifest.xml*, προσδιορίζονται τα μεταδεδομένα της εφαρμογής, οι απαιτήσεις σε Hardware και σε χαρακτηριστικά της πλατφόρμας, εξωτερικές βιβλιοθήκες και δικαιώματα.

- Intents: Αποτελούν ένα από τα πιο σημαντικά συστατικά μίας εφαρμογής (Εικόνα 7-24). Τα intents περιγράφουν μηνύματα συστήματος τα οποία παράγονται και τρέχουν στο σύστημα καθ' όλη τη διάρκεια λειτουργίας της συσκευής και ενημερώνουν τις εφαρμογές για κάθε είδους συμβάν που λαμβάνει χώρα ανά πάσα χρονική στιγμή. Υπάρχουν τρεις βασικές περιπτώσεις χρήσης των intents: Για την εκκίνηση μίας δραστηριότητας, έναρξη μίας υπηρεσίας και για την παράδοση μίας εκπομπής.

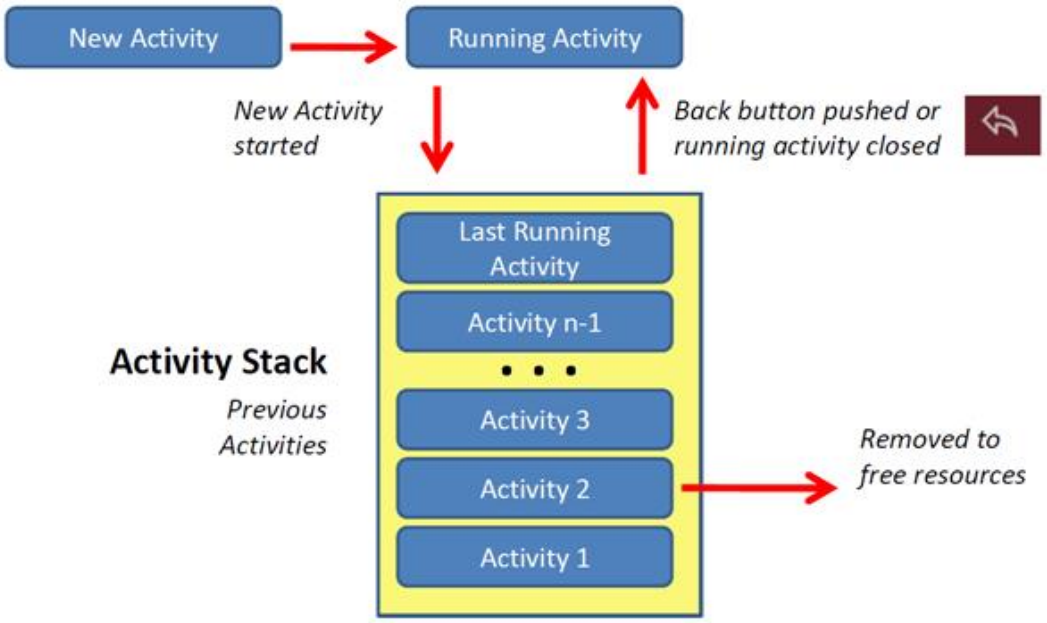


Εικόνα 7-24: Παράδειγμα εκκίνησης μίας νέας δραστηριότητας με την χρήση intents

- Activities: Ένα Activity αποτελεί βασικό συστατικό μίας εφαρμογής το οποίο αναπαριστά μία οθόνη με την οποία οι χρήστες μπορούν να αλληλοεπιδράσουν με σκοπό να κάνουν κάτι, όπως μία κλήση, το τράβηγμα μίας φωτογραφίας, αποστολή e-mail κτλ. Για παράδειγμα, μία εφαρμογή e-mail θα μπορούσε να έχει ένα activity στο οποίο θα φαίνονται τα νέα e-mail, ένα άλλο activity για την αποστολή ενός e-mail και ένα ακόμα για την ανάγνωση των e-mails.

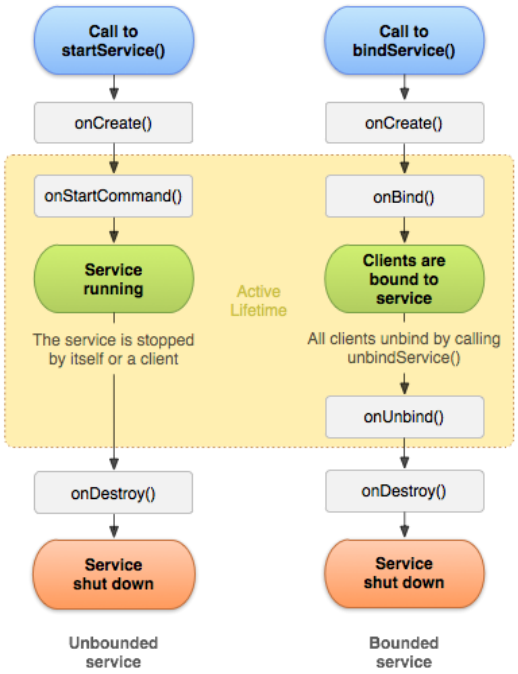
Μία εφαρμογή αποτελείται από πολλαπλά activities τα οποία είναι χαλαρά συνδεδεμένα μεταξύ τους. Συνήθως ένα activity σε μία εφαρμογή ορίζεται ως το «κύριο» activity, το οποίο και εμφανίζεται στο χρήστη όταν εκτελεί την εφαρμογή για πρώτη φορά. Κάθε activity μπορεί στην συνέχεια να αρχίσει ένα νέο activity το οποίο θα εκτελεί κάποιες άλλες λειτουργίες στην εφαρμογή μας. Κάθε φορά που ξεκινά ένα νέο activity, το προηγούμενο αυτόματα σταματά, ωστόσο το σύστημα μας διατηρεί το activity αυτό σε μία στοίβα «back stack». Όταν ένα νέο activity ξεκινά, ωθείται στη στοίβα και παρουσιάζεται στο χρήστη. Η «back stack» τηρεί το βασικό μηχανισμό λειτουργεία μας στοίβας, δηλαδή το «last in, first out», οπότε όταν ο χρήστης έχει τελειώσει με το παρών activity και πατήσει το κουμπί «πίσω», το activity αυτό εξάγεται από τη στοίβα και καταστρέφεται, και το προηγούμενο activity συνεχίζεται.

Όταν το activity σταματά επειδή ξεκινά κάποιο νέο, ειδοποιείται για αυτή την αλλαγή κατάστασης μέσω των μεθόδων επανάκλησης του κύκλου ζωής μίας δραστηριότητας. Υπάρχουν αρκετές τέτοιες μέθοδοι που μπορεί να λάβει ένα activity, λόγο της αλλαγής μίας κατάστασης, και κάθε μέθοδος μας δίνει την ευκαιρία να εκτελέσουμε μία συγκεκριμένη εργασία κατάλληλη για αυτή την αλλαγή κατάστασης.



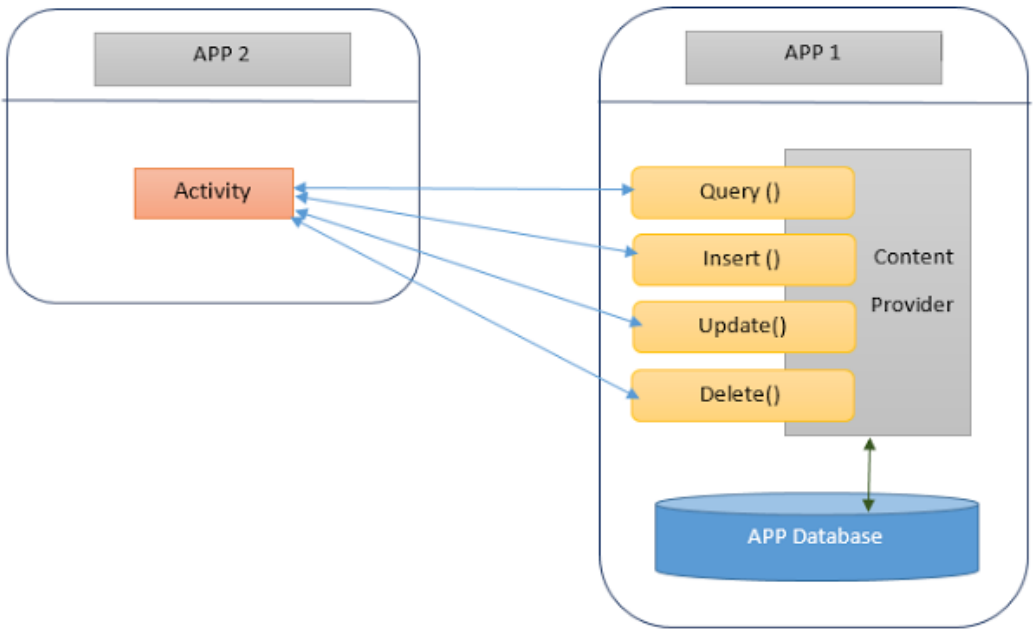
Εικόνα 7-25: Η στοίβα των activities

- Services: Το Service είναι ένα συστατικό μίας εφαρμογής το οποίο μπορεί να εκτελεί μακροχρόνιες εργασίες στο παρασκήνιο χωρίς κάποιο γραφικό περιβάλλον. Ένα άλλο συστατικό μίας εφαρμογής μπορεί να ξεκινήσει κάποιο service, το οποίο θα συνεχίσει να τρέχει στο παρασκήνιο ακόμα και αν ο χρήστης μεταβεί σε άλλη εφαρμογή. Επιπλέον, ένα συστατικό μπορεί να συνδεθεί σε ένα service για να αλληλοεπιδρά με αυτό και ακόμα μπορούν να εκτελέσουν διαδιεργασιακή επικοινωνία (interprocess communication, IPC) . Για παράδειγμα ένα service μπορεί να διαχειρίζεται διαδικτυακές συναλλαγές, να παίζει μουσική, μεταφορά αρχείων κτλ. όλα στο παρασκήνιο. Ένα service μπορεί να έχει δύο μορφές, Started και Bound. Στο αριστερό κομμάτι της εικόνας 7-26 διακρίνουμε τον κύκλο ζωής μιας υπηρεσίας με την χρήση της μεθόδου startService() και δεξιά με την μέθοδο bindService().



Εικόνα 7-26: Κύκλος ζωής μίας υπηρεσίας.

- Content Providers: Διαχειρίζονται την πρόσβαση σε μία κεντρική αποθήκη δεδομένων. Αποτελούν μέρος μίας Android εφαρμογής και συχνά προσφέρουν το δικό τους user interface για την εργασία με τα δεδομένα. Ωστόσο, οι Content Providers προορίζονται κατά κύριο λόγο για να χρησιμοποιηθούν από άλλες εφαρμογές, οι οποίες έχουν πρόσβαση στον Provider και αντλούν δεδομένα. Στην εικόνα 7-27 η εφαρμογή 1 αποθηκεύει τα δεδομένα της σε μία βάση δεδομένων και παρέχει έναν Content Provider. Η εφαρμογή 2 επικοινωνεί με τον Provider της εφαρμογής 1 για να αποκτήσει πρόσβαση στα δεδομένα της.

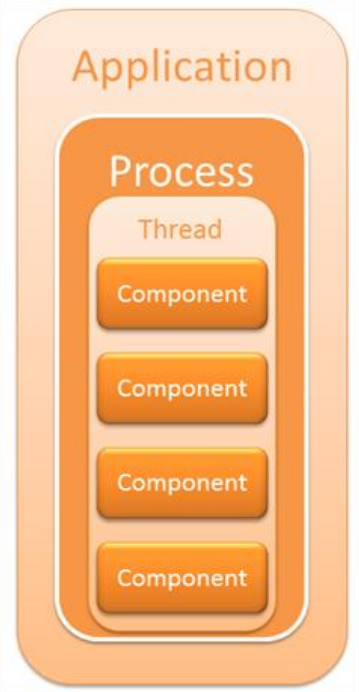


Εικόνα 7-27: Content Providers

- App Widgets: Αποτελούν μικρογραφίες εφαρμογών οι οποίες μπορούν να ενσωματωθούν σε άλλες εφαρμογές και να λαμβάνουν περιοδικές ενημερώσεις. Μία εφαρμογή η οποία φιλοξενεί κάποιο App Widget ονομάζεται App Widget host.

Εικόνα 7-28: App Widget καιρού, στην οθόνη υποδοχής της συσκευής μας

- Processes and Threads: Όταν ξεκινά ένα στοιχείο μίας εφαρμογής και η εφαρμογή δεν έχει κάποιο άλλο στοιχείο να τρέχει, το σύστημα ξεκινά μία νέα διεργασία Linux για την εφαρμογή με ένα μόνο νήμα εκτέλεσης. Από προεπιλογή, όλα τα στοιχεία της εφαρμογής τρέχουν στην ίδια διεργασία και νήμα («κύριο» νήμα). Αν ένα στοιχείο μίας εφαρμογής ξεκινήσει και υπάρχει ήδη διεργασία που να τρέχει για την εφαρμογή αυτή, τότε το στοιχείο ξεκινά εντός της υπάρχουσας διεργασίας και χρησιμοποιεί το ίδιο νήμα εκτέλεσης. Ωστόσο, μπορούμε να φροντίσουμε διαφορετικά στοιχεία της εφαρμογής μας να τρέχουν σε ξεχωριστές διεργασίες, και μπορούμε να δημιουργήσουμε επιπλέον νήματα για κάθε διεργασία.



Εικόνα 7-29: Η διεργασία και το «κύριο» νήμα.

7.5 Εγκατάσταση Περιβάλλοντος Υλοποίησης Εφαρμογών

Για να αναπτύξουμε μία Android εφαρμογή χρειαζόμαστε φυσικά κάποιο προγραμματιστικό περιβάλλον. Όπως έχουμε ήδη αναφέρει όλη η ανάπτυξη των Android εφαρμογών στηρίζεται στην γλώσσα προγραμματισμού Java, πράγμα που σημαίνει ότι θα πρέπει να εγκαταστήσουμε στον υπολογιστή μας το πιο πρόσφατο Java™ Platform, Standard Edition Development Kit, (JDK™), εφόσον δεν είναι ήδη εγκατεστημένο. Σαν πρώτο βήμα, θα επισκεφτούμε την επίσημη ιστοσελίδα της Oracle απ' όπου θα κατεβάσουμε την τελευταία έκδοση της Java(Java SE Development Kit 8) (Εικόνα 7-30).

Java SE Development Kit 8u20

You must accept the Oracle Binary Code License Agreement for Java SE to download this software.

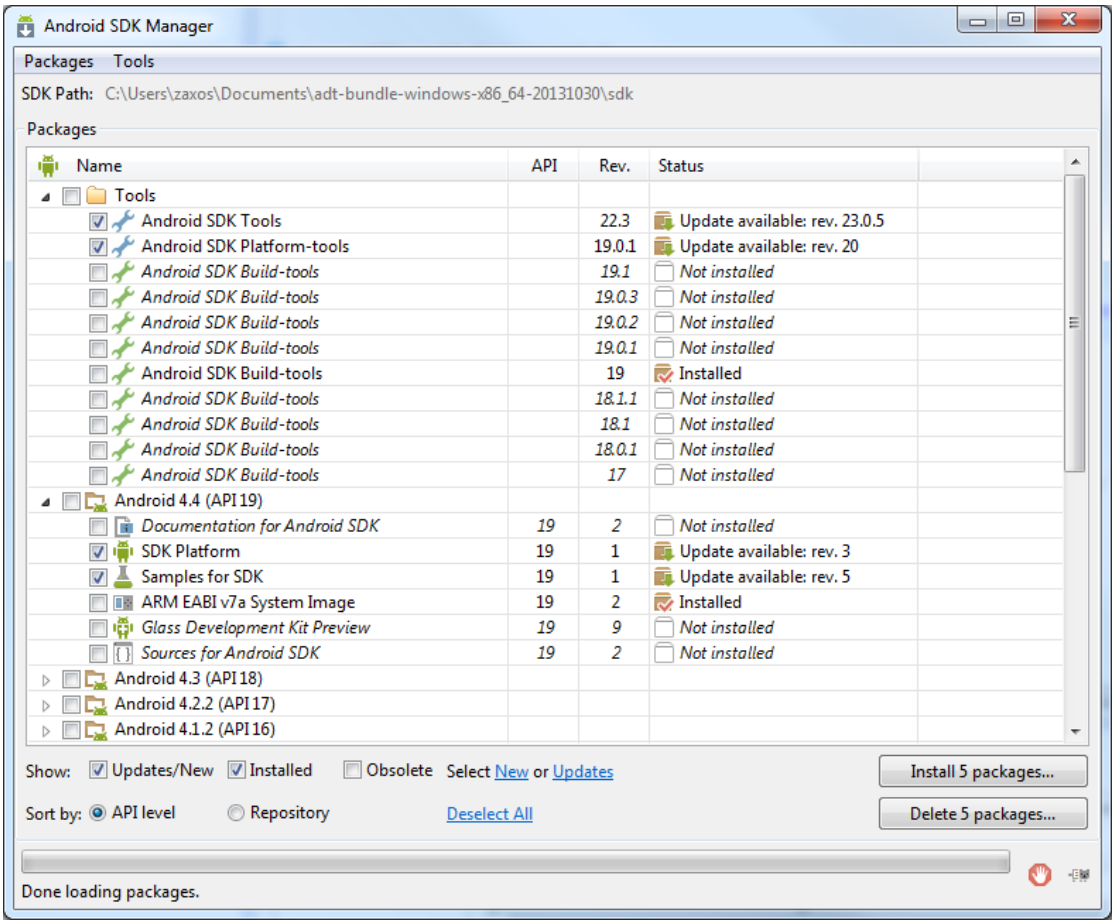
Thank you for accepting the Oracle Binary Code License Agreement for Java SE; you may now download this software.

Product / File Description	File Size	Download
Linux x86	135.24 MB	jdk-8u20-linux-i586.rpm
Linux x86	154.87 MB	jdk-8u20-linux-i586.tar.gz
Linux x64	135.6 MB	jdk-8u20-linux-x64.rpm
Linux x64	153.42 MB	jdk-8u20-linux-x64.tar.gz
Mac OS X x64	209.11 MB	jdk-8u20-macosx-x64.dmg
Solaris SPARC 64-bit (SVR4 package)	137.02 MB	jdk-8u20-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	97.09 MB	jdk-8u20-solaris-sparcv9.tar.gz
Solaris x64 (SVR4 package)	137.16 MB	jdk-8u20-solaris-x64.tar.Z
Solaris x64	94.22 MB	jdk-8u20-solaris-x64.tar.gz
Windows x86	161.08 MB	jdk-8u20-windows-i586.exe
Windows x64	173.08 MB	jdk-8u20-windows-x64.exe

Εικόνα 7-30: Java SE Development Kit 8 Downloads

Ανοίγουμε τον παραπάνω υπερσύνδεσμο, κάνουμε κλικ στην καρτέλα Downloads, επιλέγουμε Java Platform (JDK) 8u20 (η 8u20 είναι η τελευταία έκδοση αυτή τη στιγμή) και μας εμφανίζεται η παραπάνω λίστα με όλες τις διαθέσιμες εκδόσεις ανάλογα το λειτουργικό σύστημα που διαθέτουμε. Κάνουμε κλικ στο link που αντιστοιχεί στην πλατφόρμα του λειτουργικού μας συστήματος για να αρχίσει το κατέβασμα του αρχείου. Μετά την λήψη προχωρούμε στην εγκατάσταση του JDK.

Το επόμενο βήμα είναι η εγκατάσταση του Android SDK (Software Development Kit) στον υπολογιστή μας. Αποτελεί το επίσημο εργαλείο από την Google, ένα πακέτο το οποίο περιλαμβάνει όλα όσα χρειαζόμαστε για να αρχίσουμε να αναπτύσσουμε Android εφαρμογές. Πιο συγκεκριμένα το πακέτο αυτό περιέχει το Eclipse και κάποια plugins, Android SDK εργαλεία, εργαλεία για την πλατφόρμα του Android, μία έκδοση της πλατφόρμας του Android και έναν εξωμότη μίας έκδοσης του Android. Πάμε στην επίσημη σελίδα http://developer.android.com/sdk/index.html του Android, κάνουμε κλικ στο εικονίδιο Download Eclipse ADT with the Android SDK for Windows, στην σελίδα που ακολουθεί αποδεχόμαστε τους όρους, επιλέγουμε την έκδοση του λειτουργικού μας συστήματος (32bit ή 64bit) και πατάμε λήψη. Μόλις τελειώσει η λήψη μετακινούμαστε στο φάκελο λήψης και τρέχουμε το αρχείο SDK Manager.exe ή ανοίγουμε το Eclipse,το περιβάλλον πάνω στο οποίο θα αναπτύξουμε την εφαρμογή μας, κάνουμε κλικ στο εικονίδιο **SDK Manager** που βρίσκεται επάνω αριστερά. Το παρακάτω παράθυρο εμφανίζεται (Εικόνα 7-31).



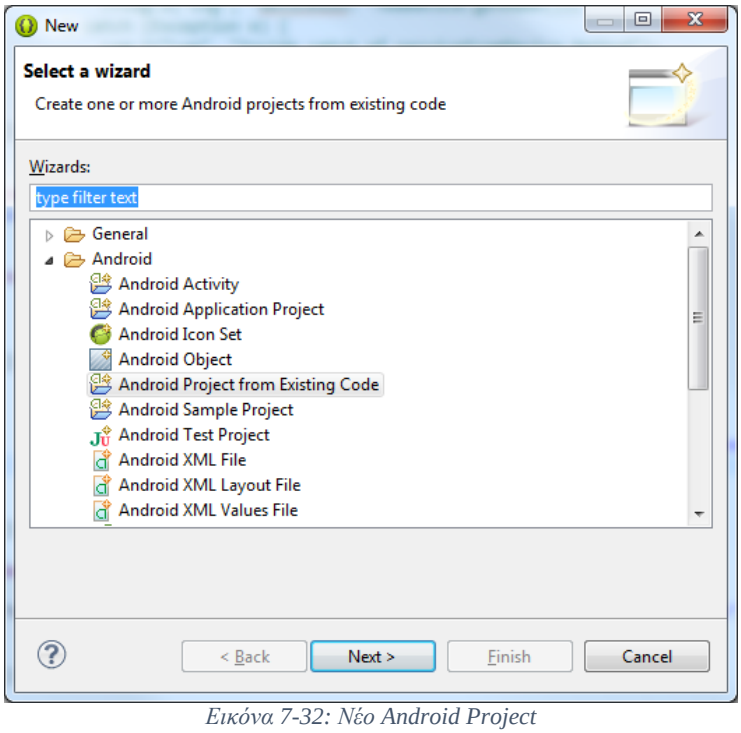
Εικόνα 7-31: Android SDK Manager

Όπως βλέπουμε παραπάνω κάποια πακέτα είναι ήδη εγκατεστημένα, σε κάποια υπάρχουν διαθέσιμες ενημερώσεις και κάποια άλλα δεν τα έχουμε εγκαταστήσει καθόλου. Κατά κανόνα, όταν ξεκινάμε για πρώτη φορά το SDK πρέπει οπωσδήποτε πρώτον, να εγκαταστήσουμε από την καρτέλα «Tools» τα Android SDK Tools, Android SDK Platform-tools και τα Android SDK Build-tools (τελευταία έκδοση) και δεύτερον από την τελευταία διαθέσιμη έκδοση του Android (4.4 API 19 όπως φαίνεται και στην εικόνα) να εγκαταστήσουμε το SDK Platform και το ARM EABI v7a System Image (εικόνα του συστήματος για τον εξομοιωτή). Εφόσον επιλέξουμε τα διαθέσιμα πακέτα κάνουμε κλικ στο «Install packages». Είναι σωστό συχνά να ανοίγουμε τον Android SDK Manager και να εγκαθιστούμε όλες τις απαραίτητες ενημερώσεις.

7.6 Δημιουργία ενός Android Project με το Eclipse

Σε αυτή την υποενότητα θα δούμε πως θα δημιουργήσουμε ένα Android Project σε επτά απλά βήματα.

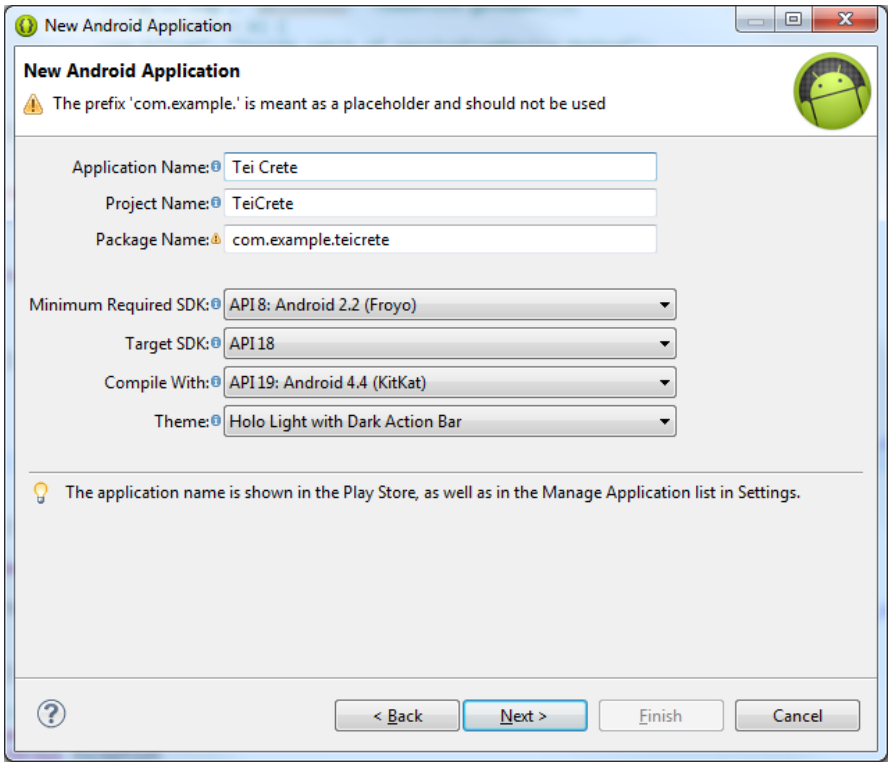
1. Ανοίγουμε την εφαρμογή Eclipse, κάνουμε κλικ στο εικονίδιο New που βρίσκεται πάνω αριστερά στη μπάρα εργαλείων.
2. Στο παράθυρο που εμφανίζεται, ανοίγουμε το φάκελο με ονομασία Android, επιλέγουμε Android Application Project, και πατάμε next (Εικόνα 7-32).



Εικόνα 7-32: Νέο Android Project

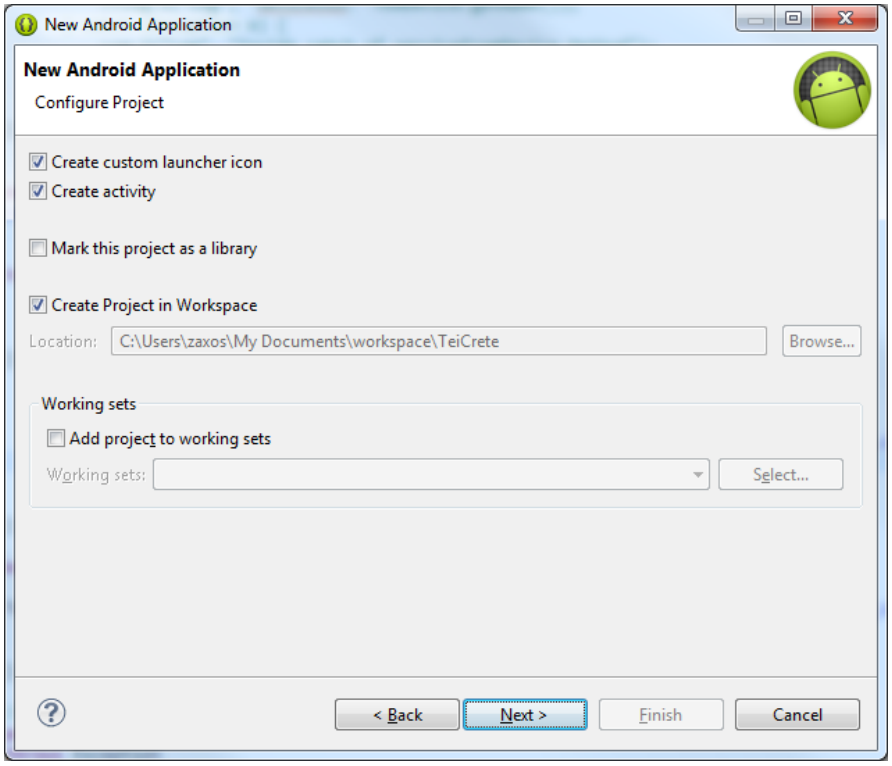
3. Συμπληρώνουμε τη φόρμα που εμφανίζεται (Εικόνα 7-33):

- Το πεδίο Application Name είναι το όνομα της εφαρμογής που θα εμφανίζεται στους χρήστες. Για το παράδειγμα αυτό, θα χρησιμοποιήσουμε το «Tei Crete».
- Το πεδίο Project Name είναι το όνομα για το directory του project και το όνομα που είναι ορατό στο Eclipse.
- Το πεδίο Package Name είναι ο χώρος των ονομάτων των κλάσεων της εφαρμογής. Ακολουθεί τους ίδιους κανόνες με τα πακέτα της Java. Το όνομα του πακέτου πρέπει να είναι μοναδικό ανάμεσα σε όλα τα άλλα πακέτα που είναι εγκατεστημένα στο σύστημα. Για το λόγο αυτό είναι προτιμότερο να χρησιμοποιούμε ένα όνομα που ξεκινά με το αντίστροφο domain name του οργανισμού η της ιστοσελίδας μας. Για το project αυτό μπορούμε να χρησιμοποιήσουμε το «com.example.teicrete». Ωστόσο αν θέλουμε να δημοσιεύσουμε τη εφαρμογή αυτή στο Google Play θα πρέπει να δώσουμε στο πακέτο ένα πραγματικό όνομα χωρίς να περιέχεε το «com.example».
- Το minimum Required SDK είναι η χαμηλότερη έκδοση του Android που υποστηρίζει η εφαρμογή μας και μπορεί να λάβει σαν τιμή κάποιο από τα API Levels. Για να υποστηρίξουμε όσο το δυνατό περισσότερες συσκευές, θα πρέπει να ορίσουμε το μικρότερο API Level που υποστηρίζει τις δυνατότητες της εφαρμογής μας. Αν οποιοδήποτε χαρακτηριστικό της εφαρμογής μας υποστηρίζεται μόνο σε κάποιο από τα πρόσφατα Levels και δεν είναι σημαντικό για την όλη εφαρμογή, μπορούμε να το ενεργοποιούμε μόνο στις εκδόσεις που το υποστηρίζουν. Μπορούμε να το αφήσουμε στη default επιλογή.
- Το πεδίο Target SDK ορίζει το υψηλότερο επίπεδο με το οποίο δοκιμάσαμε την εφαρμογή μας. Καθώς διατίθενται οι νέες εκδόσεις του Android, θα πρέπει να δοκιμάζουμε την εφαρμογή στην νέα έκδοση και να ενημερώσουμε αυτή την τιμή για να ταυτίζεται με την τελευταία έκδοση ή να εκμεταλλεύεται τα νέα χαρακτηριστικά της πλατφόρμας.
- Το πεδίο Compile With είναι η έκδοση της πλατφόρμας με βάση την οποία πρόκειται η εφαρμογή μας να μεταγλωττιστεί. Από προεπιλογή, τίθεται στην τελευταία διαθέσιμη έκδοση του SDK (4.1 ή μεγαλύτερη).
- Το πεδίο Theme ορίζει το στυλ του UI που θα εφαρμοστεί στη εφαρμογή μας. Έπειτα κάνουμε κλικ στο Next.



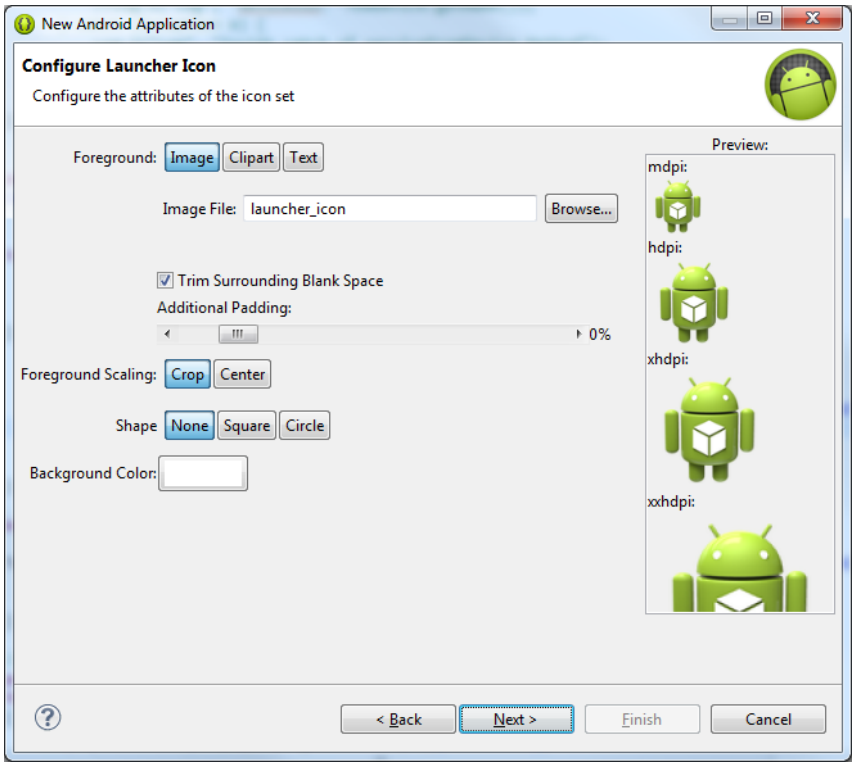
Εικόνα 7-33: Ονομασία και λοιπές ρυθμίσεις

4. Σε αυτό το παράθυρο αφήνουμε τις προεπιλεγμένες επιλογές και πατάμε Next (Εικόνα 7-34).



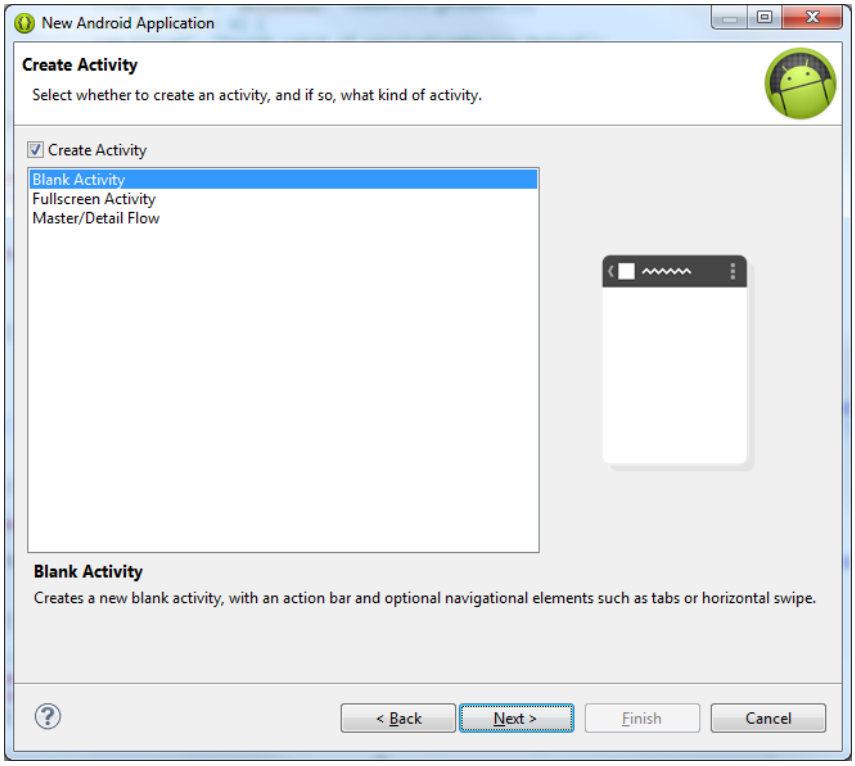
Εικόνα 7-34: Προεπιλεγμένες επιλογές της εφαρμογής

5. Σε αυτό το παράθυρο μπορούμε να δημιουργήσουμε ένα εικονίδιο εκκίνησης για την εφαρμογή μας. Μπορούμε να προσαρμόσουμε ένα εικονίδιο με διάφορους τρόπους και το εργαλείο παράγει ένα εικονίδιο για όλες τις πυκνότητες οθόνης. Πριν δημοσιεύσουμε την εφαρμογή μας, θα πρέπει να βεβαιωθούμε ότι το εικονίδιο πληροί τις προδιαγραφές που ορίζονται στον οδηγό σχεδίασης εικονιδίων. Πατάμε πάλι Next (Εικόνα 7-35).



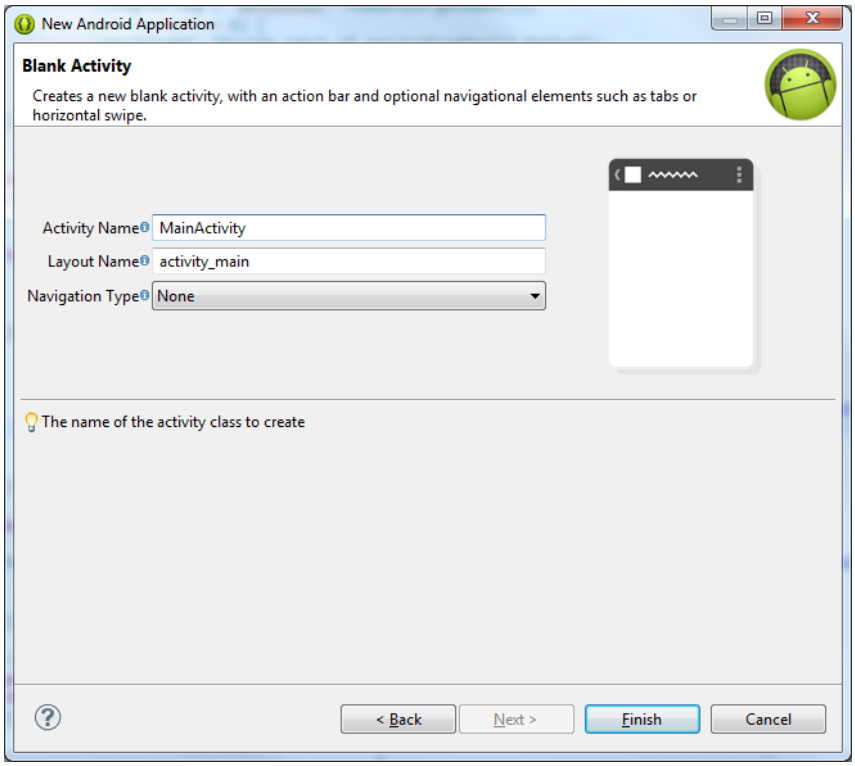
Εικόνα 7-35: Το εικονίδιο εκκίνησης

6. Τώρα μπορούμε να επιλέξουμε ένα activity template απ' όπου μπορούμε να ξεκινήσουμε την ανάπτυξη της εφαρμογής μας (Εικόνα 7-36).



Εικόνα 7-36: Επιλογή του template

7. Αφήνουμε όλες τις λεπτομέρειες για το activity στις προεπιλεγμένες επιλογές και πατάμε finish (Εικόνα 7-37).



Εικόνα 7-37: Τελευταίο στάδιο δημιουργίας του Project

7.7 Η Αρχειακή Δομή ενός Android Project

Ο τρόπος που θα εκτελέσουμε την εφαρμογή μας εξαρτάται από δύο παράγοντες: αν έχουμε μία πραγματική Android συσκευή και αν χρησιμοποιούμε το Eclipse. Εδώ θα δούμε πως μπορούμε να τρέξουμε την εφαρμογή μας τόσο σε μία συσκευή όσο και στον εξομοιωτή του Eclipse. Πριν γίνει όμως αυτό θα κάνουμε μία αναφορά στην αρχειακή δομή ενός Android project.

- AndroidManifest.xml: Κάθε εφαρμογή πρέπει να έχει το αρχείο AndroidManifest.xml (με ακριβώς αυτό το όνομα) στο root directory της. Το αρχείο manifest παρουσιάζει βασικές πληροφορίες σχετικά με την εφαρμογή μας στο σύστημα του Android, πληροφορίες που το σύστημα υποχρεωτικά πρέπει να έχει για να μπορέσει να τρέξει οποιοδήποτε κώδικα της εφαρμογής μας. Μεταξύ άλλων το αρχείο manifest κάνει τα παρακάτω:
    - Ονομάζει το πακέτο Java της εφαρμογής. Το όνομα του πακέτου χρησιμεύει ως ένα μοναδικό αναγνωριστικό για την εφαρμογή.
    - Περιγράφει τα στοιχεία της εφαρμογής. Τα activities, services, broadcast receivers, και content providers από τα οποίο αποτελείται η εφαρμογή μας και για τα οποία έχει γίνει αναφορά σε προηγούμενη ενότητα.
    - Καθορίζει ποιες διεργασίες θα φιλοξενήσουν συστατικά της εφαρμογής.
    - Δηλώνει ποια δικαιώματα πρέπει να έχει η εφαρμογή προκειμένου να έχει πρόσβαση σε προστατευμένα μέρη του API και να αλληλοεπιδρά με άλλες εφαρμογές.
    - Δηλώνει επίσης τα δικαιώματα που άλλοι πρέπει να έχουν για να αλληλοεπιδρούν με τα στοιχεία της εφαρμογής.
    - Απαριθμεί τις Instrumentation κλάσεις οι οποίες παρέχουν χαρακτηριστικά και άλλες πληροφορίες καθώς η εφαρμογή εκτελείται. Οι δηλώσεις αυτές είναι παρούσες μόνο στο manifest αρχείο καθώς η εφαρμογή αναπτύσσεται και τεστάρετε. Αφαιρούνται πριν την δημοσίευση της εφαρμογής.
    - Δηλώνει το ελάχιστο επίπεδο του Android API που απαιτεί η εφαρμογή.
    - Παραθέτει τις βιβλιοθήκες με τις οποίες συνδέεται η εφαρμογή.
- src/: Ο φάκελος που περιέχει τον πηγαίο κώδικα της εφαρμογής μας. Από προεπιλογή, περιέχει μία Activity κλάση η οποία τρέχει μόλις εκτελεστεί η εφαρμογή μας.
- res/: Περιέχει αρκετούς υποφακέλους για πόρους της εφαρμογής μας. Όπως:

- drawable-hdpi/: Φάκελος που περιέχει τις εικόνες που χρησιμοποιούνται στην εφαρμογή μας, σχεδιασμένες για high-density(hdpi) οθόνες. Αντίστοιχα υπάρχουν οι φάκελοι drawable-ldpi/ για low-density οθόνες, drawable-mdpi/ για medium density οθόνες, drawable-xhdpi/ για extra high density οθόνες.
- layout/: Φάκελος για αρχεία που ορίζουν το UI της εφαρμογής.
- values/: Φάκελος για διάφορα άλλα αρχεία XML που περιέχουν μία συλλογή από πόρους που χρησιμοποιεί η εφαρμογή όπως χρώματα και strings.

7.8 Εκτέλεση Εφαρμογής σε Συσκευή Android

Για να εγκαταστήσουμε και να τρέξουμε μία εφαρμογή σε μία πραγματική συσκευή Android, αρχικά θα την συνδέσουμε με τον υπολογιστή μας χρησιμοποιώντας το usb καλώδιο που περιείχε η συσκευή με την αγορά της. Αν χρησιμοποιούμε Windows θα χρειαστεί να εγκαταστήσουμε τους απαραίτητους Usb οδηγούς για την συσκευή μας (Πίνακας 7-3). Για παράδειγμα στα Windows 7 πρέπει να ακολουθήσουμε την εξής διαδικασία:

1. Όπως προαναφέραμε συνδέουμε την Android συσκευή στη θύρα Usb του υπολογιστή μας.
2. Κάνουμε δεξί κλικ στο εικονίδιο Computer στην επιφάνειά εργασίας και επιλέγουμε manage.
3. Επιλέγουμε Device Manager από το αριστερό πάνελ.
4. Κάνουμε επέκταση στην κατηγορία Other device.
5. Κάνουμε δεξί κλικ στην συσκευή μας και επιλέγουμε Update Driver Software.
6. Στο παράθυρο που εμφανίζεται πατάμε Browse my computer for driver software και στη συνέχεια next.
7. Πατάμε Browse και πηγαίνουμε στο φάκελο όπου βρίσκονται οι drivers για την συσκευή μας.
8. Πατάμε Next για την εγκατάσταση των drivers.

Πίνακας 7-3: Διαθέσιμοι Usb drivers ανά κατασκευαστή

Κατασκευαστής	**Διεύθυνση οδηγού**
Acer	http://www.acer.com/worldwide/support/mobile.html
ALCATEL ONE TOUCH	http://www.alcatelonetouch.com/global-en/support/faq/usbdriver.html
Asus	http://support.asus.com/download/
Dell	http://support.dell.com/support/downloads/index.aspx?c=us&cs=19&l=en&s=dhs&~ck=anavml
Foxconn	http://drivers.cmcs.com.tw/
Fujitsu	http://www.fmworld.net/product/phone/sp/android/develop/
Garmin-Asus	https://www.garminasus.com/en_US/support/pcsync/
Hisense	http://app.hismarttv.com/dss/resourcecontent.do?method=viewResourceDetail&resourceId=16&type=5
HTC	http://www.htc.com
Huawei	http://www.huaweidevice.com/worldwide/downloadCenter.do?method=index
Intel	http://www.intel.com/software/android
KT Tech	http://www.kttech.co.kr/cscenter/download05.asp for EV-S100 (Take)
Kyocera	http://www.kyocera-wireless.com/support/phone_drivers.htm
Lenovo	http://developer.lenovomm.com/developer/download.jsp
LGE	http://www.lg.com/us/mobile-phones/mobile-support/mobile-lg-mobile-phone-support.jsp
Motorola	http://developer.motorola.com/docstools/USB_Drivers/
MTK	http://online.mediatek.com/Public%20Documents/MTK_Android_USB_Driver.zip
Oppo	http://www.oppo.com/index.php?q=software/view&sw_id=631
Pantech	http://www.isky.co.kr/cs/software/software.sky?fromUrl=index
Pegatron	http://www.pegatroncorp.com/download/New_Duke_PC_Driver_0705.zip (ZIP download)
Samsung	http://www.samsung.com/us/support/downloads
Sharp	http://k-tai.sharp.co.jp/support/
SK Telesys	http://www.sk-w.com/service/wDownload/wDownload.jsp

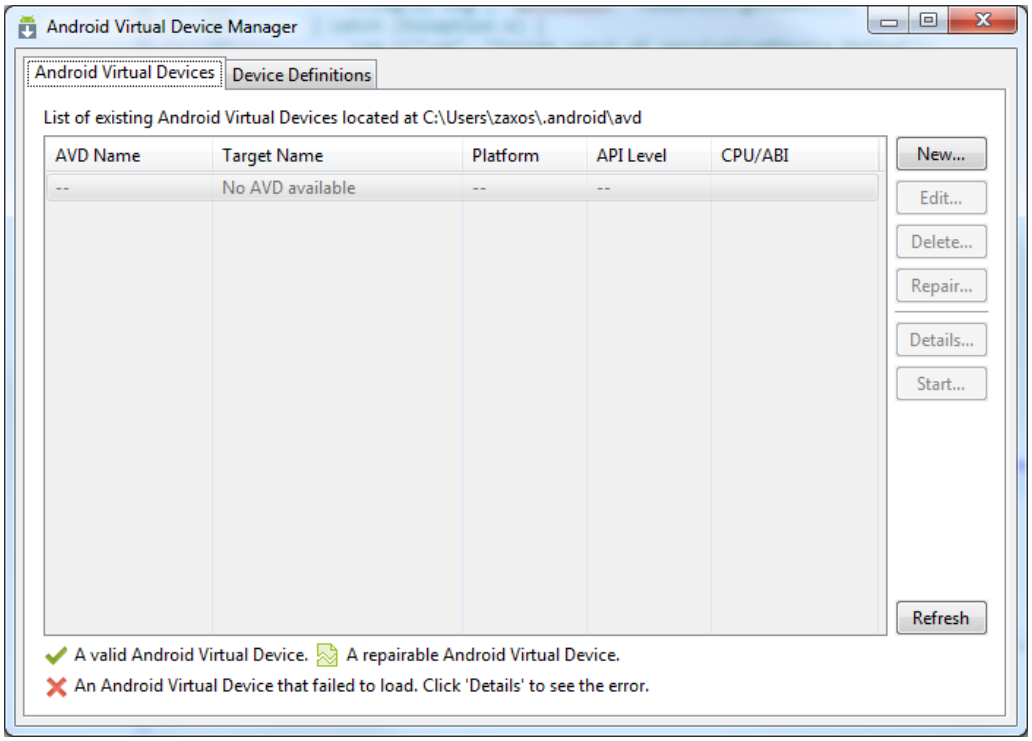
Sony Mobile Communications	http://developer.sonymobile.com/downloads/drivers/
Teleepoch	http://www.teleepoch.com/android.html
Toshiba	http://support.toshiba.com/sscontent?docId=4001814
Yulong Coolpad	http://www.yulong.com/product/product/product/downloadList.html#downListUL
Xiaomi	http://www.xiaomi.com/c/driver/index.html
ZTE	http://support.zte.com.cn/support/news/NewsDetail.aspx?newsId=1000442

Στη συνέχεια ενεργοποιούμε το USB debugging στη συσκευή μας. Στις περισσότερες συσκευές που τρέχουν την έκδοση Android 3.2 ή παλαιότερη, βρίσκουμε την επιλογή αυτή πηγαίνοντας στο Settings>Applications>Development. Σε περίπτωση που χρησιμοποιούμε την έκδοση Android 4.0 ή μεταγενέστερη, η επιλογή αυτή βρίσκεται στο Settings>Developer options. Να σημειώσουμε ότι στις εκδόσεις από την 4.2 και μετά, η επιλογή Developer options είναι κρυφή. Για να την ενεργοποιήσουμε, πάμε Settings>About phone και πατάμε Build number επτά φορές. Πηγαίνουμε στην προηγούμενη οθόνη για να βρούμε την επιλογή Developer options. Στη συνέχεια κάνουμε κλικ στο εικονίδιο Run στη μπάρα εργαλείων του Eclipse, στο παράθυρο Run As που εμφανίζεται επιλέγουμε Android Application και πατάμε OK. Το Eclipse εγκαθιστά την εφαρμογή στην συνδεδεμένη στον υπολογιστή μας συσκευή και την εκτελεί.

7.9 Εκτέλεση Εφαρμογής σε Εικονική Συσκευή Android

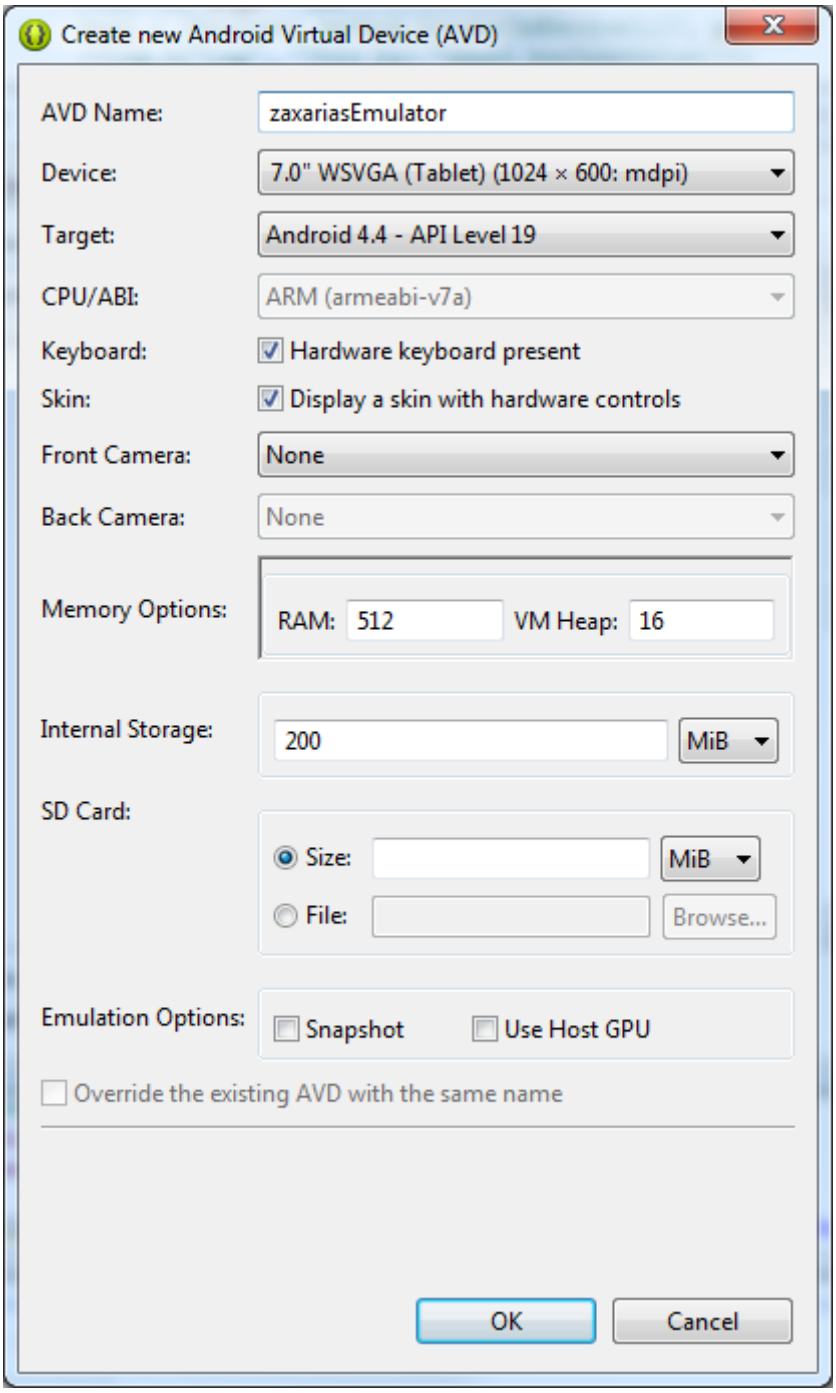
Για να τρέξουμε την εφαρμογή μας στον εξομοιωτή θα πρέπει να ορίσουμε μία εικονική συσκευή (AVD, Android Virtual Device). Η εικονική αυτή συσκευή θα χρησιμοποιείται για τον έλεγχο όλων των εφαρμογών μας από την στιγμή που την δημιουργήσαμε και στο εξής. Για να δημιουργήσουμε μία εικονική συσκευή ακολουθούμε την παρακάτω διαδικασία:

1. Στη μπάρα εργαλείων του Eclipse κάνουμε κλικ στο εικονίδιο Virtual Device Manager .
2. Στο παράθυρο που εμφανίζεται κάνουμε κλικ στο New για να δημιουργήσουμε την εικονική συσκευή (Εικόνα 7-38).



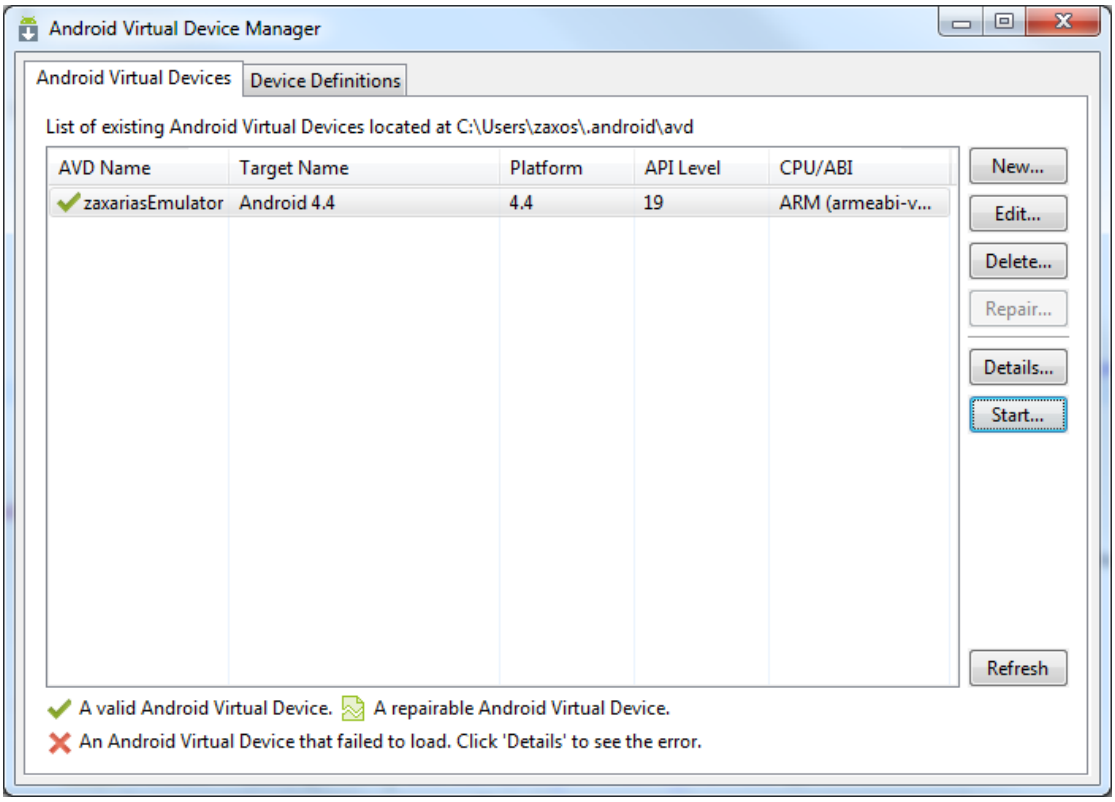
Εικόνα 7-38: Android Virtual Device Manager

3. Το παρακάτω παράθυρο (Εικόνα 7-39) θα εμφανιστεί. Στο πεδίο AVD Name ορίζουμε το όνομα της εικονικής συσκευής. Στο πεδίο Device τον τύπο της εικονικής συσκευής που θέλουμε να χρησιμοποιήσουμε και στο Target την έκδοση του λειτουργικού για την οποία προορίζεται ο εξομοιωτής.



Εικόνα 7-39: Δημιουργία του Virtual Device

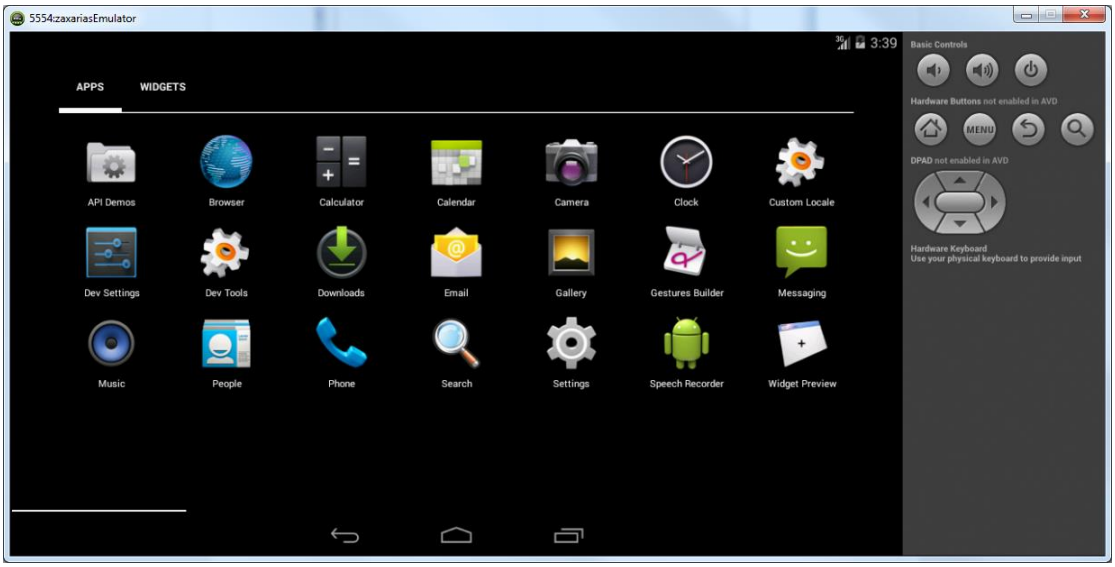
4. Πατώντας OK ολοκληρώνεται η διαδικασία δημιουργίας της εικονικής μας συσκευής (Εικόνα 7-40).



Εικόνα 7-40: Ολοκλήρωση δημιουργίας του Device Manager

5. Στο παραπάνω παράθυρο (Εικόνα 7-40) επιλέγουμε την εικονική συσκευή που μόλις δημιουργήσαμε και κάνουμε κλικ στο Start ώστε να τρέξει ο εξομοιωτής.
6. Μόλις φορτώσει ο εξομοιωτής η παρακάτω οθόνη εμφανίζεται (Εικόνα 7-41). Κάνοντας Unlock μπορούμε να δούμε τις εφαρμογές που είναι εγκατεστημένες στην εικονική μας συσκευή (Εικόνα 7-42).

Εικόνα 7-41: Το interface της εικονικής συσκευής



Εικόνα 7-42: Οι εφαρμογές που βρίσκονται εγκατεστημένες στην εικονική συσκευή

Τώρα για να τρέξουμε μία εφαρμογή στην εικονική μας συσκευή, κάνουμε δεξί κλικ στο root του Project μας στον Package Explorer και στη συνέχεια Run As>Android Application. Το Eclipse εγκαθιστά την εφαρμογή στην εικονική συσκευή και έπειτα την εκτελεί.

8 Υλοποίηση της Εφαρμογής μας στο Android

Σε αυτή την ενότητα θα δούμε αναλυτικά όλα τα στάδια για την δημιουργία της Android εφαρμογής για τον ασύρματο έλεγχο του τηλεχειριζόμενου οχήματος. Πριν από κάθε στάδιο θα παρέχεται οποιαδήποτε σημαντική θεωρητική εισαγωγή, εφόσον αυτό κρίνεται απαραίτητο, έτσι ώστε τα βήματα που ακολουθούμε να γίνουν απολύτως κατανοητά.

8.1 Κλάσεις Διάταξης στο Android

Οι διεπαφές χρήστη εφαρμογών μπορούν να είναι απλές ή σύνθετες περιλαμβάνοντάς πολλές διαφορετικές ή και μόνο λίγες οθόνες. Στο Android η δημιουργία αρχείων διάταξης στην XML γίνεται με την μορφή πόρων που παρέχονται στον κατάλογο res/layout. Αυτός είναι ένας πρακτικός και χρήσιμος τρόπος δημιουργίας διεπαφών χρήστη στο Android. Ωστόσο υπάρχει και η προγραμματιστική δημιουργία διατάξεων η οποία όμως είναι καλύτερο να χρησιμοποιείται σε πιο περίπλοκες καταστάσεις. Οι τύποι διατάξεων που ενσωματώνονται στο Android SDK είναι οι εξής:

- LinearLayout
- RelativeLayout
- FrameLayout
- TableLayout

Το LinearLayout είναι ένα View Group το οποίο εμφανίζει κάθε θυγατρικό View σε μία στήλη ή μία γραμμή (Εικόνα 8-1). Όταν εφαρμόζεται σε όλη την οθόνη τότε κάθε θυγατρικό View σχεδιάζεται κάτω από το προηγούμενο View αν ο προσανατολισμός είναι κατακόρυφος ή δεξιά από το προηγούμενο View εάν ο προσανατολισμός είναι οριζόντιος. Είναι χρήσιμη για την δημιουργία φορμών.

Εικόνα 8-1: Linear Layout

Το RelativeLayout, βάση του οποίου είναι σχεδιασμένη και η δική μας εφαρμογή, είναι ένα View Group που μας επιτρέπει να καθορίσουμε την σχέση των στοιχείων ελέγχου child view μεταξύ τους. Για παράδειγμα, μπορούμε να ορίσουμε αν κάποιο θυγατρικό view θα τοποθετηθεί πάνω, κάτω, δεξιά ή αριστερά ενός άλλου view αναφέροντας απλά το μοναδικό αναγνωριστικό του.

Εικόνα 8-2: Relative Layout

Το FrameLayout είναι ένα View Group το οποίο σχεδιάστηκε ώστε να εμφανίζει μία στοίβα θυγατρικών view. Μπορούμε να το χρησιμοποιήσουμε ώστε να εμφανίσουμε πολλαπλές εικόνες μέσα στην ίδια περιοχή (Εικόνα 8-3).

Εικόνα 8-3: Frame Layout

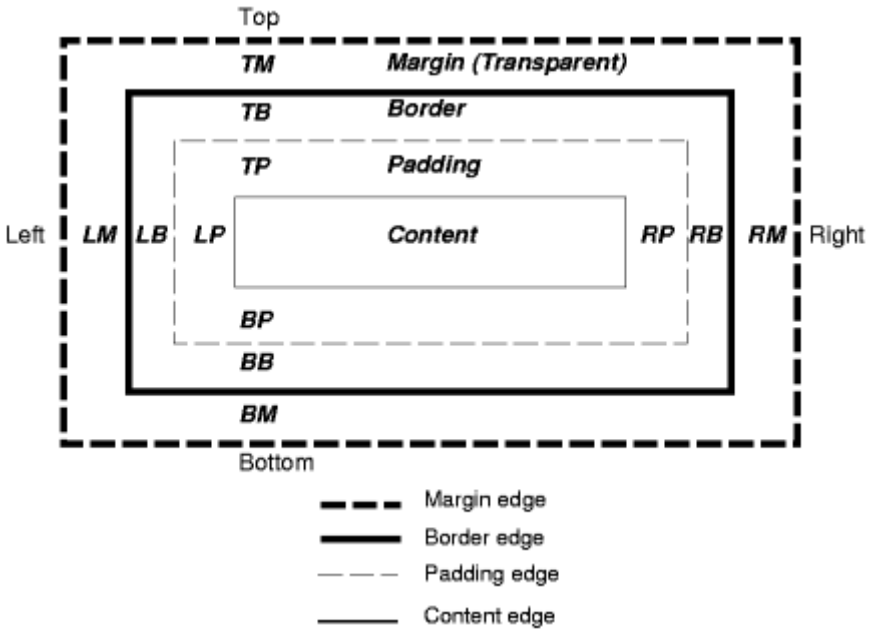
Το TableLayout είναι ένα View Group το οποίο οργανώνει τα παιδιά σε γραμμές. Προσθέτουμε μεμονωμένα αντικείμενα View μέσα σε κάθε γραμμή του πίνακα χρησιμοποιώντας ένα View διάταξης Table Row για κάθε γραμμή του πίνακα.

Εικόνα 8-4: Table Layout

8.1.1 Η Διάταξη και Σχεδίαση της Εφαρμογής μας

Στην δική μας εφαρμογή γίνεται χρήση του RelativeLayout και LinearLayout. Πηγαίνοντας στον κατάλογο res/layout/activity_bluetooth_class.xml του Package Explorer θα δούμε την διάταξη της εφαρμογής μας μέσω πόρων XML.

Στον κώδικα που ακολουθεί (Πίνακας 8-1) δημιουργούμε ένα RelativeLayout και ρυθμίζουμε τις παραμέτρους του όπως, διαστάσεις, του δίνουμε ένα όνομα/id, «*RelativeLayout1*» στην δική μας περίπτωση, το background που χρησιμοποιεί η διάταξη φορτώθηκε κάνοντας χρήση του `android:background`. Το `xmlns:android` είναι απαραίτητο και προσδιορίζει τον χώρο των ονομάτων του XML ο οποίος πρέπει να είναι ο *"http://schemas.android.com/apk/res/android"*. Το `tools:context` προσδιορίζει το Activity με το οποίο συνδέεται η εκάστοτε διάταξη. Για να διασφαλίσουμε την ευελιξία της διάταξης μας και την δυνατότητα να προσαρμόζεται σε διαφορετικά μεγέθη οθόνης πρέπει να χρησιμοποιούμε είτε το *"match_parent"* είτε το *"wrap_content"* για τον καθορισμό τους πλάτους και ύψους των στοιχείων της εφαρμογής μας. Εάν χρησιμοποιήσουμε το *"wrap_content"* το πλάτος ή το ύψος του αντικειμένου έχει οριστεί στο ελάχιστο αναγκαίο μέγεθος για να χωρέσει το περιεχόμενο μέσα σε αυτή την όψη, ενώ το *"match_parent"* κάνει το αντικείμενο να επεκταθεί ώστε να ταιριάζει με το μέγεθος της μητρικής όψης.



Εικόνα 8-5: Περιθώρια Margin,Border, και Padding

Στην παραπάνω φωτογραφία (Εικόνα 8-1) βλέπουμε τα περιθώρια των Margin, Border και Padding ενός στοιχείου. Τα περιθώρια Margin και Padding τα χρησιμοποιούμε αρκετά συχνά στην διάταξη μας για να ορίσουμε αποστάσεις.

Πίνακας 8-1: Relative Layout

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/RelativeLayout1"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/wall"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".BluetoothClass" >
```

Εσωτερικά του RelativeLayout δημιουργούμε ένα LinearLayout το οποίο περιέχει δύο λίστες, `<ListView>` (Πίνακας 8-2). Αντίστοιχα όπως κάναμε στο RelativeLayout και εδώ ονοματίζουμε το LinearLayout με όνομα «*linearLayout1*» και ορίζουμε τις διαστάσεις του. Επίσης ορίζουμε τις διαστάσει που επιθυμούμε και για τις λίστες. Τα ονόματα που δώσαμε στις λίστες, όπως φαίνεται και από τον κώδικα είναι «*listViewDetected*» για την πρώτη λίστα και «*listViewPaired*» αντίστοιχα για την δεύτερη. Στην πρώτη λίστα θα εμφανίζονται οι ασύρματες συσκευές Bluetooth που εντοπίστηκαν από την συσκευή μας και στην δεύτερη εκείνες οι οποίες έχουν δημιουργήσει ζεύξη με την συσκευή μας (έχουν συνδεθεί με αυτήν).

Πίνακας 8-2: Linear Layout

```
  <LinearLayout
        android:id="@+id/linearLayout1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentRight="true"
        android:layout_alignParentTop="true"
        android:layout_marginTop="60dp" >

        <ListView
            android:id="@+id/listViewDetected"
```

```
            android:layout_width="131dp"
            android:layout_height="match_parent" >
        </ListView>

        <ListView
            android:id="@+id/listViewPaired"
            android:layout_width="251dp"
            android:layout_height="120dp" >
        </ListView>
    </LinearLayout>
```

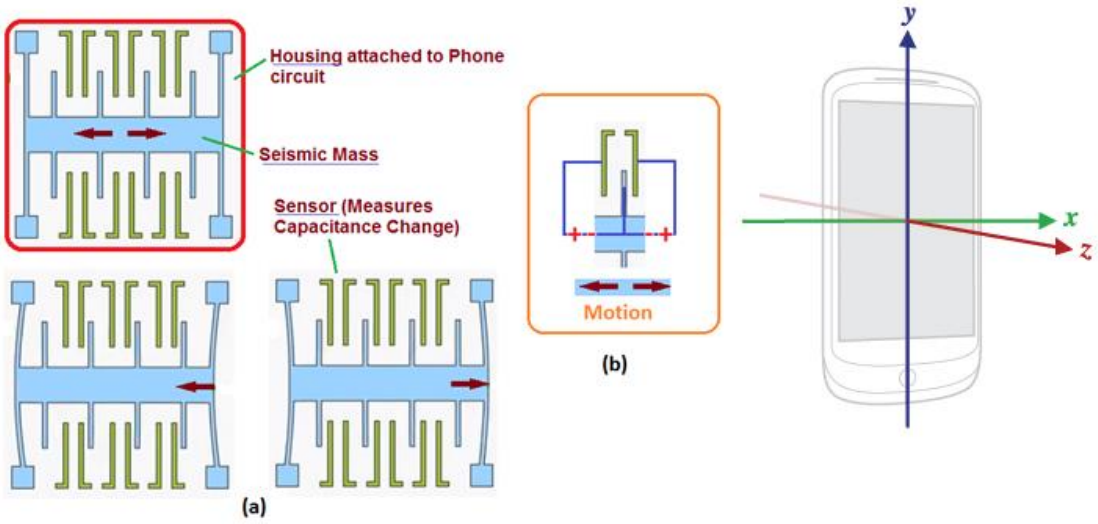
Δημιουργούμε πέντε κουμπιά με ονόματα «*imageButton1*», «*imageButton2*», «*imageButton3*», «*imageButton4*» και «*imageButton5*» αντίστοιχα (Πίνακας 8-3). Όλα τα κουμπιά έχουν τις ίδιες διαστάσεις, έχουν τοποθετηθεί με τέτοιον τρόπο, με τις ιδιότητες του RelativeLayout, ώστε να σχηματίζουν σταυρό μεταξύ τους ο οποίος θα χρησιμοποιηθεί για τον έλεγχο του τηλεκατευθυνόμενου οχήματος για τις κινήσεις εμπρός, πίσω, αριστερά και δεξιά. Σε κάθε κουμπί έχουμε φορτώσει μία εικόνα ως background για καλύτερο οπτικό αποτέλεσμα.

Πίνακας 8-3: Image Buttons

```
    <ImageButton
        android:id="@+id/imageButton1"
        android:layout_width="120dip"
        android:layout_height="120dip"
        android:layout_alignParentLeft="true"
        android:layout_centerVertical="true"
        android:background="@drawable/my_selector"
        android:contentDescription="@string/left" />

    <ImageButton
        android:id="@+id/imageButton2"
        android:layout_width="120dip"
        android:layout_height="120dip"
        android:layout_alignLeft="@+id/imageButton1"
        android:layout_alignParentBottom="true"
        android:layout_marginBottom="47dp"
        android:layout_marginLeft="115dp"
        android:background="@drawable/my_selector"
        android:contentDescription="@string/down" />

    <ImageButton
        android:id="@+id/imageButton3"
        android:layout_width="120dip"
        android:layout_height="120dip"
        android:layout_alignLeft="@+id/imageButton1"
        android:layout_alignParentTop="true"
        android:layout_marginLeft="115dp"
        android:layout_marginTop="47dp"
        android:background="@drawable/my_selector"
        android:contentDescription="@string/up" />

    <ImageButton
        android:id="@+id/imageButton4"
        android:layout_width="120dip"
        android:layout_height="120dip"
        android:layout_alignLeft="@+id/imageButton1"
        android:layout_alignParentLeft="true"
        android:layout_centerVertical="true"
        android:layout_marginLeft="228dp"
        android:background="@drawable/my_selector"
```

```
        android:contentDescription="@string/right" />

    <ImageButton
        android:id="@+id/imageButton5"
        android:layout_width="120dip"
        android:layout_height="120dip"
        android:layout_alignParentRight="true"
        android:layout_centerVertical="true"
        android:background="@drawable/my_selector"
        android:contentDescription="@string/stop" />
```

Με τον παρακάτω «κώδικα» δημιουργούμε τρία TextView με ονόματα «*xval*», «*yval*» και «*zval*» (Πίνακας 8-4). Με το στοιχείο ελέγχου TextView μπορούμε πολύ απλά να σχεδιάσουμε κείμενο στην οθόνη. Το μέγεθος του κειμένου ορίζεται με το `android:textSize` και είναι 15sp (scale-independent pixel). Αυτό που εμείς θα εμφανίζουμε στα πεδία αυτά είναι οι τιμές του ενσωματωμένου στη συσκευή μας επιταχυνσιόμετρου. Το επιταχυνσιόμετρο αποτελεί ένα αισθητήριο το οποίο μας επιτρέπει να ανιχνεύσουμε την κίνηση της συσκευής μας ως προς το χώρο. Οι τιμές θα εμφανίζονται σε m/s^2. Πιο συγκεκριμένα το επιταχυνσιόμετρο μετράει τη δύναμη της επιτάχυνσης που εφαρμόζεται στη συσκευή στους τρεις φυσικού άξονες (x,y και z), συμπεριλαμβανομένης της δύναμης της βαρύτητας. Στο «*xval*» θα εμφανίζεται η επιτάχυνση της συσκευής ως προς τον άξονα των X, στο «*yval*» ως προς τον άξονα των Y και στο «*zval*» ως προς τον Z άξονα. Στην εικόνα 8-6 παρουσιάζεται η αρχή λειτουργίας του επιταχυνσιόμετρο σε μία συσκευή.



Εικόνα 8-6: Το επιταχυνσιόμετρο

Το σχήμα (b) απόκομμα του σχήματος (a) δείχνει την αλλαγή στην χωρητικότητα ως αποτέλεσμα της αλλαγής στη θέση της σεισμική μάζας όταν η συσκευή αλλάζει προσανατολισμό. Αυτό θα αναγνωρίσει την αλλαγή στη βαρυτική έλξη αλλάζοντας την τρέχουσα ισοδύναμη σε μεταβολή της χωρητικότητας. Ο αισθητήρας αυτός ονομάζεται MEMS (Micro Electro Mechanical System), μικρο-ηλεκτρομηχανικό σύστημα.

Τα TextView έχουν τοποθετηθεί κάτω δεξιά στην εφαρμογή μας ώστε να μας ενημερώνουν διακριτικά για τις τιμές που λαμβάνει το επιταχυνσιόμετρο. Η εφαρμογή μας θα έχει την δυνατότητα να κινεί το τηλεκατευθυνόμενο όχημα με το σταυρό αλλά και με την χρήση του επιταχυνσιόμετρου. Επίσης έχει τοποθετηθεί και ένα TextView με όνομα «*seektxt*»το οποίο μας ενημερώνει για την τρέχουσα τιμή του PWM στο Arduino. Τέλος, ένα κουμπί, το «*imageButton6*», μας επιτρέπει την ενεργοποίηση και απενεργοποίηση αντίστοιχα της δυνατότητας να ελέγχουμε το τηλεχειριζόμενο όχημα εκμεταλλευόμενοι το επιταχυνσιόμετρο της συσκευής μας.

Πίνακας 8-4: Text Views

```
<TextView
        android:id="@+id/xval"
        android:layout_width="140sp"
        android:layout_height="wrap_content"
```

```
        android:layout_alignParentBottom="true"
        android:layout_alignParentRight="true"
        android:layout_marginBottom="60dip"
        android:layout_marginRight="10dp"
        android:textColor="#FFFFFF"
        android:textSize="15sp" />

    <TextView
        android:id="@+id/yval"
        android:layout_width="140sp"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_alignParentRight="true"
        android:layout_marginBottom="40dip"
        android:layout_marginRight="10dp"
        android:textColor="#FFFFFF"
        android:textSize="15sp" />

    <TextView
        android:id="@+id/zval"
        android:layout_width="140sp"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_alignParentRight="true"
        android:layout_centerVertical="true"
        android:layout_marginBottom="20dip"
        android:layout_marginRight="10dp"
        android:textColor="#FFFFFF"
        android:textSize="15sp" />

    <ImageButton
        android:id="@+id/imageButton6"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_marginRight="50dp"
        android:layout_toLeftOf="@+id/imageButton5"
        android:background="@drawable/my_selector2"
        android:contentDescription="@string/logo" />

    <TextView
        android:id="@+id/seektxt"
        android:layout_width="140sp"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_marginBottom="70dp"
        android:layout_marginLeft="230dp"
        android:layout_toRightOf="@+id/imageButton2"
        android:textColor="#FFFFFF"
        android:textSize="15sp" />
```

Επίσης έχουν τοποθετηθεί τέσσερα κουμπιά, κάτω στο κέντρο, τα οποία μας δίνουν την δυνατότητα να επιλέξουμε την επιθυμητή τιμή PWM για το Arduino (Πίνακας 8-5). Εδώ τελειώνει το Relative Layout.

Πίνακας 8-5: PWM Buttons

```
<Button
        android:id="@+id/button1"
        style="?android:attr/buttonStyleSmall"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```

```
        android:layout_alignParentBottom="true"
        android:layout_marginLeft="120dp"
        android:layout_toRightOf="@+id/imageButton2"
        android:text="@string/first" />

    <Button
        android:id="@+id/button2"
        style="?android:attr/buttonStyleSmall"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignBaseline="@+id/button1"
        android:layout_alignBottom="@+id/button1"
        android:layout_marginLeft="25dp"
        android:layout_toRightOf="@+id/button1"
        android:text="@string/second" />

    <Button
        android:id="@+id/button3"
        style="?android:attr/buttonStyleSmall"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignBottom="@+id/button2"
        android:layout_marginLeft="25dp"
        android:layout_toRightOf="@+id/button2"
        android:text="@string/third" />

    <Button
        android:id="@+id/button4"
        style="?android:attr/buttonStyleSmall"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignBottom="@+id/button3"
        android:layout_marginLeft="25dp"
        android:layout_toRightOf="@+id/button3"
        android:text="@string/forth" />

</RelativeLayout>
```

Στην παρακάτω φωτογραφία (Εικόνα 8-7) βλέπουμε το τελικό αποτέλεσμα του σχεδιασμού μας στο οποίο απεικονίζεται πολύ ωραία η διάταξη των αντικειμένων στο χώρο που έχει γίνει με το RelativeLayout. Αριστερά είναι ο σταυρός, πάνω δεξιά οι δύο λίστες, κάτω δεξιά τα τρία «TextView», αριστερά από τα «TextViews» είναι το λογότυπο του TEI το οποίο αποτελεί επίσης και κουμπί για την ενεργοποίηση και απενεργοποίηση αντίστοιχα της δυνατότητα τηλεχειρισμού του οχήματος αναλόγως με την κίνηση της συσκευής. Το πράσινο κουμπί κάτω από τις λίστες αφορά επίσης την διεύθυνση του οχήματος και τέλος τα τέσσερα κουμπιά που βρίσκονται στο κέντρο και κάτω (1^{st}, 2^{nd}, 3^{rd} και 4^{th}).

Εικόνα 8-7: Το κύριο layout της εφαρμογής μας

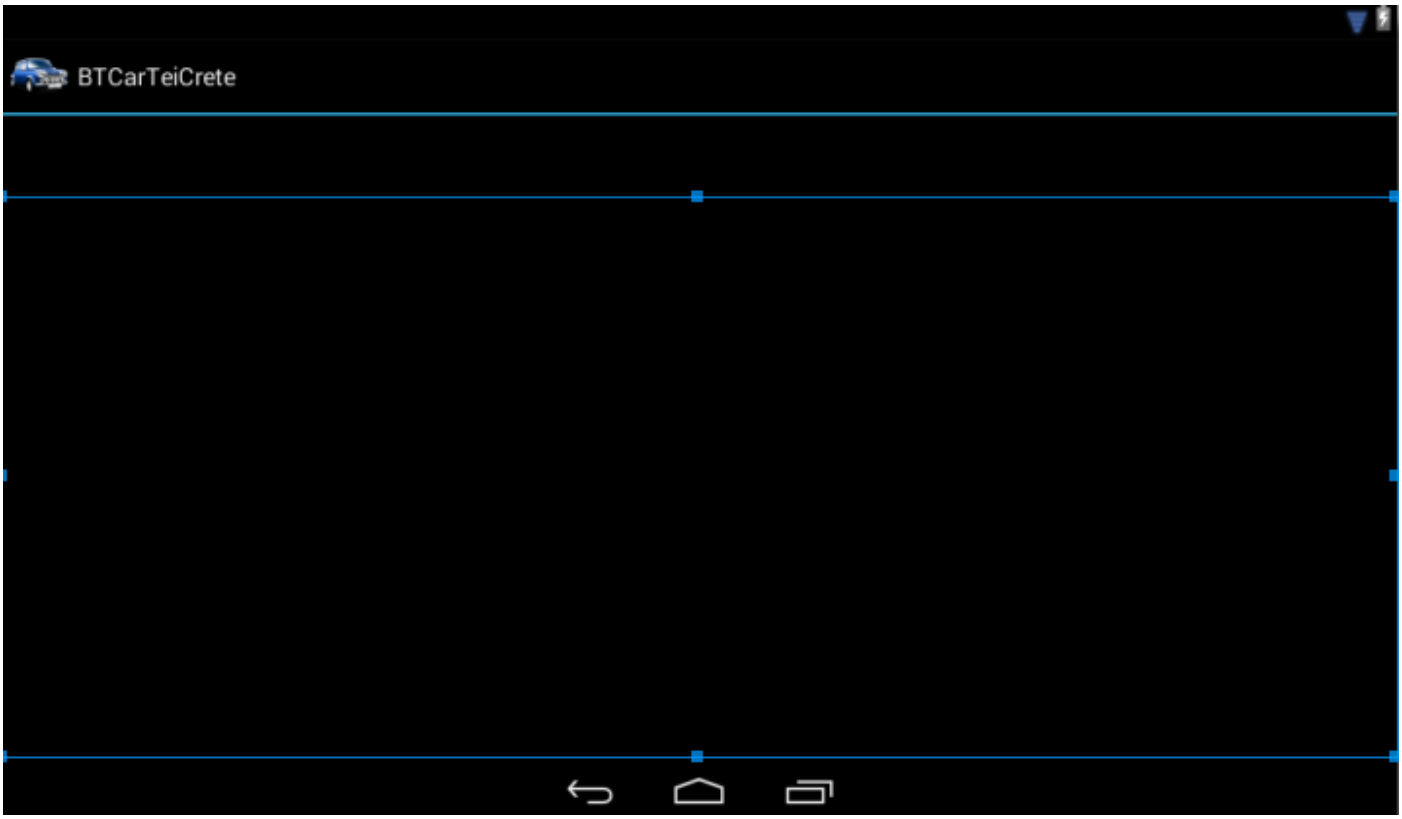
Στον κατάλογο res/layout τους Package Explorer της εφαρμογής μας έχουμε δημιουργήσει άλλο ένα XML αρχείο το activity_bluetooth_info.xml. Στο κύριο Activity της εφαρμογής μας, όπως θα δούμε αργότερα, καλούμε μία νέα κλάση, την οποία φυσικά έχουμε ήδη δημιουργήσει, η οποία θα μας οδηγεί σε μία νέα οθόνη η οποία θα περιλαμβάνει πληροφορίες για την εφαρμογή μας. Στο activity_bluetooth_info.xml λοιπόν, παρουσιάζεται η διάταξη του νέου αυτού Activity το οποία θα ενεργοποιείται με το πάτημα ενός κουμπιού. Ακολουθεί ο «κώδικας» της διάταξης αυτής.

Η έκδοση της XML είναι η 1.0 και η κωδικοποίηση που χρησιμοποιείται είναι η *"utf-8"*. Η γραμμή αυτή συμπληρώνεται αυτόματα (Πίνακας 8-6). Όπως βλέπουμε χρησιμοποιούμε την διάταξη LinearLayout στην οποία έχουμε δημιουργήσει ένα TextView με όνομα «*TextView1*» στο οποίο θα εμφανίζουμε πληροφορίες σχετικά με την εφαρμογή μας

Πίνακας 8-6: Linear Layout (infoClass)

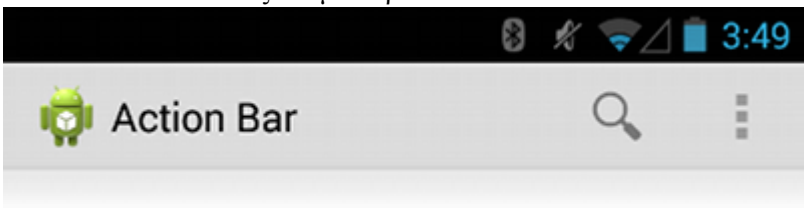
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_marginTop="60dp"
    android:orientation="vertical" >

    <TextView
        android:id="@+id/TextView1"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="0.88"
        android:ems="10" >

        <requestFocus />
    </TextView>

</LinearLayout>
```

Στην παρακάτω φωτογραφία βλέπουμε το τελικό αποτέλεσμα του σχεδιασμού μας (Εικόνα 8-8). Περιλαμβάνει ένα TextView.



Εικόνα 8-8: Το δευτερεύον layout της εφαρμογής μας

Στην εφαρμογή μας έχουν τοποθετηθεί κάποια κουμπιά στη μπάρα ενεργειών. Η μπάρα ενεργειών βρίσκεται στο επάνω μέρος της οθόνης της συσκευής μας όπως φαίνεται π.χ στην παρακάτω φωτογραφία (Εικόνα 8-9). Τα κουμπιά αυτά μας δίνουν εύκολη πρόσβαση σε απαραίτητες ενέργειες, όπως την δυνατότητα μετάβασης στο μενού αναζήτησης Bluetooth συσκευών, την ενεργοποίηση και απενεργοποίηση του Bluetooth, την αποσύνδεση της συσκευής μας από άλλη συσκευή, την ενεργοποίηση της δυνατότητας ανίχνευσης της συσκευής μας από άλλες συσκευές, την αναζήτηση συσκευών, την εμφάνιση των συνδεδεμένων με τη συσκευή μας συσκευών, την μετάβαση στην οθόνη πληροφοριών για την εφαρμογή μας και τέλος παρέχεται η δυνατότητα ελαχιστοποίησής της εφαρμογής. Όλες αυτές οι επιλογές είναι προσβάσιμες με ένα απλό κλικ. Θα τις δούμε παρακάτω.



Εικόνα 8-9: Η μπάρα ενεργειών του Android.

Για την δημιουργία των κουμπιών αυτών στην μπάρα ενεργειών, πήγαμε στον κατάλογο res/menu και δημιουργήσαμε ένα XML αρχείο με όνομα bluetooth_class.xml (Πίνακας 8-8). Το XML αρχείο που ακολουθεί, ξεκινά με το `<menu>`, αποτελεί τον αρχικό κόμβο και είναι απαραίτητο. Το `xmlns:android` είναι επίσης απαραίτητο και προσδιορίζει τον χώρο των ονομάτων του XML ο οποίος πρέπει να είναι ο *`"http://schemas.android.com/apk/res/android"`*, το `<item />` συμβολίζει την δημιουργία ενός νέου στοιχείου στο μενού, κουμπί στην δική μας περίπτωση. Όπως βλέπουμε έχουν δημιουργηθεί εννέα τέτοια κουμπιά με ονόματα/id *`action_minimize`*, *`action_info`*, *`action_paired`*, *`buttonSearch`*, *`buttonDesc`*, *`buttonDisconnect`*,

buttonOff, *buttonOn* και *settings*. Κάθε κουμπί έχει το δικό του εικονίδιο π.χ για το πρώτο κουμπί φορτώνουμε το εικονίδιο με όνομα *ic_action_minimize* με την εντολή `android:icon="@drawable/ic_action_minimize"`. Το `android:showAsAction` μπορεί να πάρει τις εξής τιμές:

Πίνακας 8-7: Τοποθέτηση στοιχείου στη μπάρα ενεργειών

Τιμή	**Περιγραφή**
`ifRoom`	Το στοιχείο τοποθετείται στη μπάρα ενεργειών μόνο αν υπάρχει ο διαθέσιμος χώρος για αυτό.
`withText`	Να περιλαμβάνει τον τίτλο κείμενο που προσδιορίζεται από το `android:title` μαζί με το στοιχείο.
`never`	Να μην τοποθετηθεί το στοιχείο στην μπάρα ενεργειών.
`always`	Το στοιχείο να τοποθετείται πάντα στη μπάρα ενεργειών.
`collapseActionView`	Η ενέργεια που σχετίζεται με το στοιχείο είναι πτυσσόμενη.

Εμείς χρησιμοποιούμε το συνδυασμό *"always|withText"*. Τέλος το `android:orderInCategory` είναι ένας ακέραιος αριθμός που προσδιορίζει την σειρά προτεραιότητας ενός στοιχείου μέσα σε ένα group. Εμείς ξεκινούμε από τα δεξιά με το κουμπί ελαχιστοποίησης της εφαρμογής και καταλήγουμε κινούμενοι προς τα αριστερά στο κουμπί ρυθμίσεων.

Πίνακας 8-8: Android Action bar Menu

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

    <item
        android:id="@+id/action_minimize"
        android:icon="@drawable/ic_action_minimize"
        android:orderInCategory="100"
        android:showAsAction="always|withText"
        android:title="@string/action_minimize"/>
    <item
        android:id="@+id/action_info"
        android:icon="@drawable/ic_action_btinfo"
        android:orderInCategory="90"
        android:showAsAction="always|withText"
        android:title="@string/action_info"/>
    <item
        android:id="@+id/action_paired"
        android:icon="@drawable/ic_action_paired"
        android:orderInCategory="80"
        android:showAsAction="always|withText"
        android:title="@string/action_paired"/>
    <item
        android:id="@+id/buttonSearch"
        android:icon="@drawable/ic_action_btsearch"
        android:orderInCategory="70"
        android:showAsAction="always|withText"
        android:title="@string/ButtonSearch"/>
    <item
        android:id="@+id/buttonDesc"
        android:icon="@drawable/ic_action_btdisc"
        android:orderInCategory="60"
        android:showAsAction="always|withText"
        android:title="@string/Discover"/>
    <item
        android:id="@+id/buttonDisconnect"
        android:icon="@drawable/ic_action_disconnect"
        android:orderInCategory="50"
```

```
        android:showAsAction="always|withText"
        android:title="@string/Disconnect"/>
    <item
        android:id="@+id/buttonOff"
        android:icon="@drawable/ic_action_btoff"
        android:orderInCategory="40"
        android:showAsAction="always|withText"
        android:title="@string/ButtonOff"/>
    <item
        android:id="@+id/buttonOn"
        android:icon="@drawable/ic_action_bt"
        android:orderInCategory="30"
        android:showAsAction="always|withText"
        android:title="@string/ButtonOn"/>
    <item
        android:id="@+id/settings"
        android:icon="@drawable/ic_action_sets"
        android:orderInCategory="20"
        android:showAsAction="always|withText"
        android:title="@string/settings"/>

</menu>
```

Το όνομα του πρώτου στοιχείου του μενού δίνεται από την εντολή `android:title="@string/action_minimize"`. Από πού όμως φορτώνονται οι χαρακτήρες με όνομα `action_minimize`? Στον κατάλογο res/values του Package Explorer υπάρχουν τρία αρχεία XML τα οποία δημιουργούνται αυτόματα κατά την δημιουργία ενός νέου Android Project. Το dimens.xml, strings.xml και styles.xml. Στο dimens.xml μπορούμε να ορίσουμε τα περιθώρια της οθόνης. Από προεπιλογή είναι ως εξής (Πίνακας 8-9).

Πίνακας 8-9: Dimens XML

```
<resources>
    <dimen name="activity_horizontal_margin">16dp</dimen>
    <dimen name="activity_vertical_margin">16dp</dimen>
</resources>
```

Στο styles.xml μπορούμε να κάνουμε αλλαγές που αφορούν στο θέμα της εφαρμογής μας. Η αλλαγή που έχει γίνει εδώ είναι η διαφανοποίηση της μπάρας ενεργειών. Το συγκεκριμένο αρχείο έχει ως εξής (Πίνακας 8-10).

Πίνακας 8-10: Styles XML

```
<resources>
    <style name="AppBaseTheme" parent="android:Theme.Light">
    </style>
    <style name="AppTheme"  parent="@android:style/Theme.Holo">
        <item name="android:windowActionBarOverlay">true</item>
    </style>
</resources>
```

Τέλος στο αρχείο strings.xml περιέχονται τα ονόματα-τίτλοι που έχουν αποδοθεί σε στοιχεία της εφαρμογής μας π.χ στα κουμπιά (Πίνακας 8-11). Για παράδειγμα, στο στοιχείο με id `action_minimize` σύμφωνα με τις παρακάτω εντολές θα δοθεί το όνομα `minimize.`

Πίνακας 8-11: Strings XML

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <string name="app_name">BTCarTeiCrete</string>
```

```
    <string name="action_minimize">minimize</string>
    <string name="action_info">info</string>
    <string name="hello_world">Hello world!</string>
    <string name="ButtonOn">On</string>
    <string name="ButtonOff">Off</string>
    <string name="Discover">Discover</string>
    <string name="ButtonSearch">Search</string>
    <string name="Bookmark">ButtonSearch</string>
    <string name="Save">ButtonSearch</string>
    <string name="Search">ButtonSearch</string>
    <string name="Disconnect">Disconnect</string>
    <string name="second_activity">Useful Information</string>
    <string name="settings">Settings</string>
    <string name="action_paired">Paired</string>
    <string name="left">left</string>
    <string name="right">right</string>
    <string name="up">up</string>
    <string name="down">down</string>
    <string name="stop">stop</string>
    <string name="first">1st</string>
    <string name="second">2nd</string>
    <string name="third">3rd</string>
    <string name="forth">4th</string>
    <string name="logo">logo</string>
    <string name="todo">TODO</string>

</resources>
```

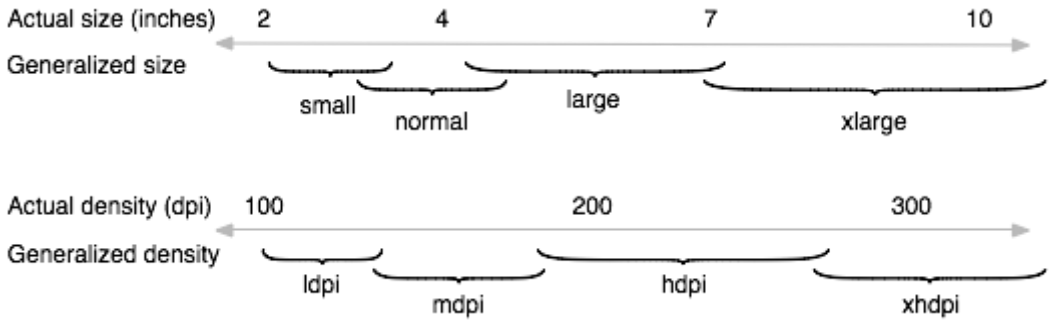
Στις εικόνες 8-10 και 8-11 βλέπουμε την διεπαφή της εφαρμογής μας στο τελικό τους στάδιο, όπως ακριβώς φαίνονται στην Tablet συσκευή.

Εικόνα 8-10: Το UI της εφαρμογής μας

Εικόνα 8-11: Το δευτερεύον Activity της εφαρμογής μας

Όσον αφορά την σχεδίαση της εφαρμογής μας ενδιαφέρον παρουσιάζουν οι υποφάκελοι drawable-hdpi, drawable-ldpi, drawable-mdpi, drawable-xdpi και drawable-xxhdpi οι οποίοι βρίσκονται στον κατάλογο res/. Όπως έχουμε αναφέρει σε προηγούμενη ενότητα οι φάκελοι αυτοί περιέχουν όλες τις εικόνες που χρησιμοποιούνται από την εφαρμογή μας διαθέσιμες για διάφορες πυκνότητες οθόνης. Για τους σκοπούς της εργασίας αυτής χρησιμοποιήθηκες μία Tablet συσκευή Lenovo IdeaTab A1000-F. Θα ήταν καλό να βρούμε την πυκνότητα της οθόνης αυτής έτσι ώστε να γνωρίζουμε εκ των προτέρων σε ποιον φάκελο drawable δουλεύουμε. Για να βρούμε λοιπόν την πυκνότητα της οθόνης μας, η οποία μετριέται σε PPI (pixels per inch), ακολουθούμε την παρακάτω διαδικασία:

1. Βρίσκουμε την διαγώνια ανάλυση σε pixel, την οποία συμβολίζουμε με dp, εφαρμόζοντας το πυθαγόρειο θεώρημα. Άρα dp=$\sqrt{w^2 + h^2}$, όπου w και h είναι η ανάλυση του πλάτους και ύψους αντίστοιχα σε pixels. Η ανάλυση του πλάτους της συσκευής μας είναι 600 pixels και η ανάλυση του ύψους 1024 pixels. Έχουμε λοιπόν, dp=$\sqrt{600^2 + 1024^2}$=1186.8 pixels.
2. Υπολογίζουμε τα pixels per inch που δίνονται από τον τύπο, PPI=$\frac{dp}{di}$, όπου di είναι το μέγεθος της διαγώνιου, το dp το έχουμε είδη βρει. Η διαγώνιος μας είναι 7 ίντσες οπότε, $\frac{1186.8}{7}$=169.5 PPI.



Εικόνα 8-12: Αντιστοίχηση του μεγέθους της οθόνης με την πυκνότητα(προσεγγιστικά).

Αυτό σημαίνει πως η οθόνης μας είναι μεσαίας πυκνότητας, ο φάκελος στον οποίο δουλεύουμε είναι ο drawable-mdpi. Ωστόσο, οι εικόνες που χρησιμοποιούμε στην εφαρμογής μας πρέπει να φροντίζουμε να υπάρχουν και στους άλλους drawable φακέλους, καθότι η εφαρμογή μπορεί να χρησιμοποιηθεί σε συσκευή με διαφορετική πυκνότητα.

8.2 Το Android Manifest

Απαραίτητο και πολύ βασικό αρχείο το οποίο περιγράφει την εφαρμογή μας. Καθορίζει τις δυνατότητες και τα δικαιώματα της εφαρμογής μας καθώς και τον τρόπο λειτουργίας της. Τα στοιχεία που μπορούν να εμφανιστούν στο manifest αρχείο αναφέρονται παρακάτω με αλφαβητική σειρά(δεν μπορούμε να προσθέσουμε δικά μας στοιχεία ή χαρακτηριστικά):

- `<action>`
- `<activity>`
- `<activity-alias>`
- `<application>`
- `<category>`
- `<data>`
- `<grant-uri-permission>`
- `<instrumentation>`
- `<intent-filter>`
- `<manifest>`
- `<meta-data>`
- `<permission>`
- `<permission-group>`
- `<permission-tree>`
- `<provider>`
- `<receiver>`
- `<service>`
- `<supports-screens>`
- `<uses-configuration>`
- `<uses-feature>`
- `<uses-library>`
- `<uses-permission>`
- `<uses-sdk>`

Το AndroidManifest.xml της εφαρμογής μας έχει ως εξής (Πίνακας 8-12). Στην πρώτη γραμμή αναφέρεται η έκδοση της XML και η κωδικοποίηση που χρησιμοποιείται στο αρχείο αυτό, η `"utf-8"`. Το `<manifest>` αποτελεί την ρίζα του αρχείου και προσδιορίζει το `xmlns:android`, που όπως έχουμε νωρίτερα αναφέρει είναι απαραίτητο και προσδιορίζει τον χώρο των ονομάτων του XML, ο οποίος πρέπει να είναι ο `"http://schemas.android.com/apk/res/android"`. Το `package` προσδιορίζει το όνομα του πακέτου της εφαρμογής μας το οποίο είναι το `"com.example.bluetoothcarteicrete"`. Το `android:versionCode` είναι ένας ακέραιος αριθμός που αναπαριστά την έκδοση της εφαρμογής μας, όσο βελτιώνομε την εφαρμογή πρέπει να αυξάνουμε και τον ακέραιο αυτό. Εμείς χρησιμοποιούμε τον αριθμό 1 ως έκδοση της εφαρμογής μας. Το `android:versionName` είναι μία συμβολοσειρά που αναπαριστά την έκδοση του κώδικα της εφαρμογής μας, όπως θα φαίνεται στους χρήστες .

Πίνακας 8-12: Android Manifest

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.bluetoothcarteicrete"
    android:versionCode="1"
    android:versionName="1.0" >
```

Με το <uses-sdk> εκφράζουμε την συμβατότητα μίας εφαρμογής με μία ή περισσότερες εκδόσεις στο Android, μέσω του API Level (Πίνακας 8-13). Χαρακτηριστικά του αποτελούν το android:minSdkVersion το android:targetSdkVersion και το android:maxSdkVersion. Το minSdkVersion καθορίζει το χαμηλότερο επίπεδο API το οποίο υποστηρίζει η εφαρμογή μας, στην δική μας περίπτωση το API 14. Το targetSdkVersion καθορίζει το βέλτιστο επίπεδο API που υποστηρίζει η εφαρμογή μας, σε εμάς είναι το 19, αναπαριστά την έκδοση του Android SDK για την οποία κατασκευάστηκε και στην οποία δοκιμάστηκε η εφαρμογή μας. Τέλος το android:maxSdkVersion, το οποίο δεν χρησιμοποιούμε στο δικό μας manifest, καθορίζει το υψηλότερο επίπεδο API που υποστηρίζει η εφαρμογή. Τέλος το <uses-permission> χρησιμοποιείται για να ζητήσουμε κάποια συγκεκριμένη άδεια, που η εφαρμογή πρέπει να κάνει δεκτή, προκειμένου να λειτουργήσει σωστά. Οι άδειες χορηγούνται από τον χρήστη, δηλαδή εμάς, κατά την εγκατάσταση της εφαρμογής, όχι όταν τρέχει. Γι' αυτό και πρέπει να περιλαμβάνονται οι απαραίτητες άδειες στο manifest αρχείο. Η άδειες που χορηγήσαμε στην δική μας εφαρμογή είναι η *"android.permission.BLUETOOTH"* και η *"android.permission.BLUETOOTH_ADMIN"*. Η πρώτη δίνει την δυνατότητα στην εφαρμογή μας να συνδεθεί με ήδη συζευγμένες συσκευές Bluetooth και η δεύτερη επιτρέπει στην εφαρμογή μας να ανακαλύψει και να σχηματίσει ζεύγος με συσκευές.

Πίνακας 8-13: Android Manifest (συνέχεια)

```
<uses-sdk
        android:minSdkVersion="14"
        android:targetSdkVersion="19" />
    <uses-permission android:name="android.permission.BLUETOOTH" />
    <uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
```

Το <application> αποτελεί την δήλωση της εφαρμογής μας (Πίνακας 8-14). Αυτό το στοιχείο περιέχει υποστοιχεία τα οποία δηλώνουν κάθε ένα από τα συστατικά της εφαρμογής μας και έχει χαρακτηριστικά που μπορούν να επηρεάσουν όλα τα στοιχεία της εφαρμογής. Το android:allowBackup ορίζει αν θα επιτραπεί στην εφαρμογή να συμμετάσχει σε διαδικασία δημιουργίας αντιγράφων ασφαλείας (Backup) ή σε διαδικασία επαναφοράς ρυθμίσεων. Από προεπιλογή είναι στο *"true"*. Το android:icon ορίζει το εικονίδιο της εφαρμογής μας. Το android:label χρησιμοποιείται για να φορτώσουμε το όνομα της εφαρμογής μας. Τέλος με το android:theme γίνεται αναφορά στο θέμα που χρησιμοποιείται στην εφαρμογή μας.

Πίνακας 8-14: Android Manifest (συνέχεια)

```
    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
```

Στο android:name ορίζουμε ποιο είναι το όνομα της κλάσης που υλοποιεί το Activity, σε εμάς είναι το *"com.example.bluetoothcarteicrete.BluetoothClass"* (Πίνακας 8-15) . Στο android:label ορίζουμε το όνομα που θα φαίνεται στον χρήστη για το συγκεκριμένο Activity. Με το android:screenOrientation ρυθμίζουμε τον τρόπο που θα εμφανίζεται το συγκεκριμένο Activity στην οθόνη της συσκευής μας, *"landscape"* σε εμάς. Με το android:configChanges ορίζουμε ποιες αλλαγές το Activity θα εξετάσει από μόνο του, *"keyboardHidden|orientation|screenSize"*. Το <intent-filter> καθορίζει τους τύπους των intents που ένα Activity, Service ή Broadcast Receiver μπορεί να ανταποκριθεί. Ένα <intent-filter> πρέπει υποχρεωτικά να περιλαμβάνει τουλάχιστον ένα <action>. Το όνομα του <action> όπως βλέπουμε είναι το *"android.intent.action.MAIN"*. Να αναφέρουμε πως κάποιες τυπικές ενέργειες/actions έχουν ήδη οριστεί στην κλάση Intent ως ACTION_string σταθερές. Το ίδιο ισχύει και για το <category>, τυπικές κατηγορίες/categories έχουν ήδη οριστεί στην κλάση Intent ως CATEGORY_name σταθερές.

Πίνακας 8-15: Android Manifest (συνέχεια)

```
<activity
          android:name="com.example.bluetoothcarteicrete.BluetoothClass"
          android:label="@string/app_name"
```

```
            android:screenOrientation="landscape"
            android:configChanges="keyboardHidden|orientation|screenSize" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
</activity>
```

Τέλος, στο manifest δημιουργούμε ένα δεύτερο `<activity>` το οποίο αφορά την κλάση *InfoClass* (Πίνακας 8-16). Η κλάση αυτή θα καλείται μέσα από την κύρια κλάση μας. Η δημιουργία ενός `<activity>` για κάθε κλάση της εφαρμογής μας είναι απαραίτητη. Οι διαφορές που διακρίνουμε με το παραπάνω `<activity>` είναι το `android:parentActivityName`, δηλαδή το όνομα της μητρική κλάσης του `<activity>` αυτού, το οποίο είναι η κλάση *BluetoothClass* άρα το*"com.example.bluetoothcarteicrete.BluetoothClass"* και τα `<meta-data>`, δηλαδή δεδομένα που περιγράφουν άλλα δεδομένα.

Πίνακας 8-16: Android Manifest (συνέχεια)

```
    <activity
        android:name="com.example.bluetoothcarteicrete.InfoClass"
        android:label="@string/second_activity"
        android:parentActivityName="com.example.bluetoothcarteicrete.BluetoothClass"
        android:screenOrientation="landscape"
        android:configChanges="keyboardHidden|orientation|screenSize"  >
        <meta-data
            android:name="android.support.PARENT_ACTIVITY"
            android:value="com.example.bluetoothcarteicrete.BluetoothClass" />
    </activity>
    </application>
</manifest>
```

8.3 Οι Κλάσεις της Εφαρμογής μας

Στην εφαρμογή μας, όπως ήδη έχουμε αναφέρει, υπάρχουν δύο βασικές κλάσεις, η BluetoothClass και η InfoClass. Οι κλάσεις αυτές περιέχουν όλο το απαραίτητο κώδικα σε Java τον οποίο τρέχει η εφαρμογή μας. Στην πρώτη κλάση, την BluetoothClass, γίνονται όλες οι βασικές και απαραίτητες λειτουργίες της εφαρμογής μας που αφορούν το έλεγχο του τηλεχειριζόμενου οχήματος, όπως θα δούμε στον κώδικα παρακάτω, ενώ η InfoClass αποτελεί μία κλάση με σκοπό την πληροφόρηση του εκάστοτε χρήστη σχετικά με τον τρόπο λειτουργία της εφαρμογής, θα την δούμε και αυτήν παρακάτω. Ας ξεκινήσουμε λοιπόν με την BluetoothClass.

8.3.1 Η Κλάση BluetoothClass

Η εφαρμογή μας ξεκινά με το όνομα του πακέτου που βρίσκεται η κλάση μας, αυτό είναι το `com.example.bluetoothcarteicrete`, και στην συνέχεια χρησιμοποιείται η δήλωση **`import`** για να εισάγουμε τις απαραίτητες κλάσεις και διασυνδέσεις (Interfaces). Στον παρακάτω κώδικα φαίνεται το σύνολο των κλάσεων και διασυνδέσεων που εισήχθησαν (Πίνακας 8-17)

Πίνακας 8-17: Κλάση Bluetooth Class

```
package com.example.bluetoothcarteicrete;

import java.io.IOException;
import java.io.OutputStream;
import java.lang.reflect.Method;
import java.text.DecimalFormat;
import java.util.ArrayList;
import java.util.Set;
import java.util.UUID;
```

```
import android.app.Activity;
import android.app.AlertDialog;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.bluetooth.BluetoothSocket;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.IntentFilter;
import android.hardware.Sensor;
import android.hardware.SensorEvent;
import android.hardware.SensorEventListener;
import android.hardware.SensorManager;
import android.media.AudioManager;
import android.media.MediaPlayer;
import android.os.Bundle;
import android.os.Handler;
import android.util.Log;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.MotionEvent;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.View.OnTouchListener;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.ImageButton;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.Toast;
```

Στην συνέχεια ξεκινά η κλάση μας, την έχουμε ορίσει ως **public**, δηλαδή δημόσια, που σημαίνει ότι είναι ορατή σε όλες τις άλλες κλάσεις σε όλα τα πακέτα.

Πίνακας 8-18: Διάφορα επίπεδα ελέγχου πρόσβασης

Επίπεδο πρόσβασης	**Κλάση**	**Πακέτο**	**Υποκλάση**	**Κόσμος**
public	Ναι	Ναι	Ναι	Ναι
protected	Ναι	Ναι	Ναι	Όχι
private	Ναι	Όχι	Όχι	Όχι
Προκαθορισμένο	Ναι	Ναι	Όχι	Όχι

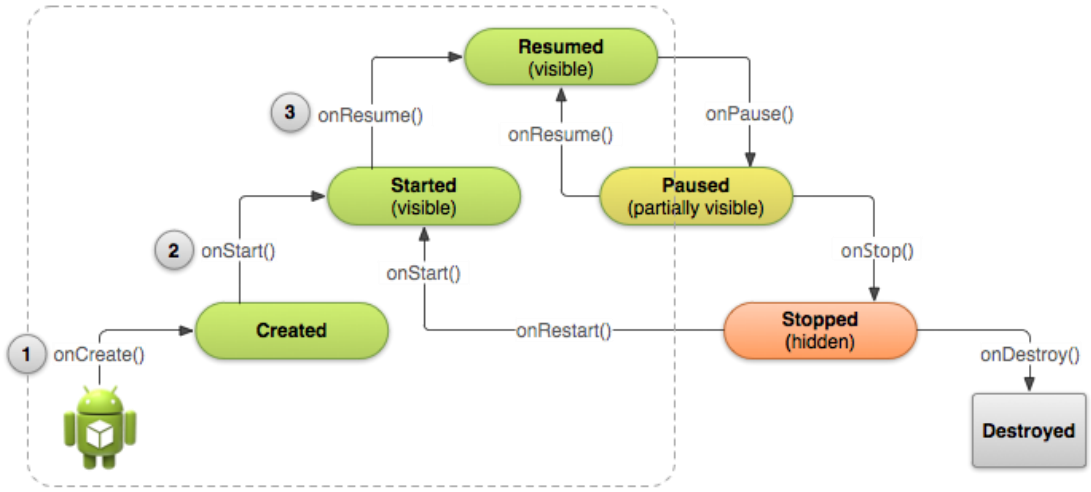
Εδώ πλέον ξεκινά η κλάση μας η οποία ορίζεται με την χρήση της δεσμευμένης λέξης **class** και ακολουθεί το όνομα της το οποίο είναι το `BluetoothClass` (Πίνακας 8-19). Η δεσμευμένη λέξη **extends** χρησιμοποιείται για να υποδείξει την υπερκλάση της `BluetoothClass` η οποία είναι η Activity. Η `BluetoothClass` με απλά λόγια αποτελεί υποκλάση της κλάσης `Activity`, αυτό σημαίνει πως η `BluetoothClass` κληρονομεί όλες τις ιδιότητες και την συμπεριφορά της `Activity`. Επίσης η `BluetoothClass` υλοποιεί την διασύνδεση (Interface) με όνομα `SensorEventListener`, η οποία αφορά την λειτουργία του επιταχυνσιόμετρου. Να θυμηθούμε ότι διασύνδεση είναι μια συλλογή μεθόδων οι οποίες υποδεικνύουν ότι μία κλάση έχει κάποια συμπεριφορά επιπρόσθετα με αυτή που κληρονομεί από τις υπερκλάσεις της. Τέλος ακολουθεί η δήλωση όλων των απαραίτητων μεταβλητών.

Πίνακας 8-19: Δήλωση μεταβλητών (BluetoothClass)

```
public class BluetoothClass extends Activity implements SensorEventListener{
    public static final int REQUEST_ENABLE_BT = 1;
    public String macAdress;
```

```
    public OutputStream outStream = null;
    public BluetoothSocket btSocket = null;
    public BluetoothDevice bdDevice;
    public BluetoothClass bdClass;
    public BluetoothAdapter bluetoothAdapter = null;
    public static final UUID MY_UUID = UUID.fromString("00001101-0000-1000-8000-
00805F9B34FB");
    public boolean connected=false;
    public ImageButton imageButton6;
    public ImageButton imageButton5;
    public ImageButton imageButton4;
    public ImageButton imageButton3;
    public ImageButton imageButton2;
    public ImageButton imageButton1;
    public Button gearbutton1;
    public Button gearbutton2;
    public Button gearbutton3;
    public Button gearbutton4;
    public Button buttonSearch;
    public Button buttonOn;
    public Button buttonDesc;
    public Button buttonOff;
    public ListView listViewPaired;
    public ListView listViewDetected;
    public ArrayList<String> arrayListpaired;
    public ArrayAdapter<String> adapter,detectedAdapter;
    public ArrayList<BluetoothDevice> arrayListPairedBluetoothDevices;
    public ListItemClickedonPaired listItemClickedonPaired;
    public ArrayList<BluetoothDevice> arrayListBluetoothDevices = null;
    public ListItemClicked listItemClicked;
    public SensorManager mSensorManager;
    public Sensor mAccelerometer;
    public TextView title,tv,tv1,tv2,seekshow;
    public float x,y,z;
    public MediaPlayer player, player2;
    public int accelGyro=0;
    public int PWMvalue=255;
```

Η μέθοδος `onCreate()`, όπως έχουμε αναφέρει στην προηγούμενη ενότητα, είναι μία μέθοδος που περιλαμβάνεται στον κύκλο ζωής ενός Activity, και σε αυτήν γίνονται όλες οι στατικές αρχικοποιήσεις. Καλείται όταν εκτελούμε την εφαρμογή μας. Το παρακάτω διάγραμμα είναι αρκετά βοηθητικό για την κατανόηση των βασικών μεθόδων του κύκλου ζωής της εφαρμογής (Εικόνα 8-13).



Εικόνα 8-13: Οι τρείς βασικές κλήσεις μεθόδων κατά την δημιουργία του Activity.

Μόλις κληθεί η `onCreate()` η παρακάτω αρχικοποιήσεις ακολουθούν, απαραίτητες για την λειτουργία της εφαρμογής μας (Πίνακας 8-20).

Πίνακας 8-20: Η συνάρτηση onCreate

```
@Override
	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.activity_bluetooth_class);
		listViewDetected = (ListView) findViewById(R.id.listViewDetected);
		listViewPaired = (ListView) findViewById(R.id.listViewPaired);
		arrayListpaired = new ArrayList<String>();
		bluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
		arrayListPairedBluetoothDevices = new ArrayList<BluetoothDevice>();
		listItemClickedonPaired = new ListItemClickedonPaired();
		arrayListBluetoothDevices = new ArrayList<BluetoothDevice>();
		adapter = new ArrayAdapter<String>(BluetoothClass.this,
				android.R.layout.simple_list_item_1, arrayListpaired);
		detectedAdapter = new ArrayAdapter<String>(BluetoothClass.this,
				android.R.layout.simple_list_item_single_choice);

		listViewDetected.setAdapter(detectedAdapter);
		listItemClicked = new ListItemClicked();
		detectedAdapter.notifyDataSetChanged();
		listViewPaired.setAdapter(adapter);
		buttonSearch = (Button) findViewById(R.id.buttonSearch);
		buttonOn = (Button) findViewById(R.id.buttonOn);
		buttonDesc = (Button) findViewById(R.id.buttonDesc);
		buttonOff = (Button) findViewById(R.id.buttonOff);
		addListenerOnButton();
		getPairedDevices();
		mSensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
		mAccelerometer = mSensorManager
				.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
		tv = (TextView) findViewById(R.id.xval);
		tv1 = (TextView) findViewById(R.id.yval);
		tv2 = (TextView) findViewById(R.id.zval);
		seekshow = (TextView) findViewById(R.id.seektxt);
		mHandler.removeCallbacks(loop);

	}
```

Η μέθοδος `onStart()`, βρίσκεται αμέσως μετά από την `onCreate()`, καλείτε πριν η δραστηριότητα γίνει εμφανής στον χρήστη (Πίνακας 8-21). Στην παρούσα εφαρμογή την χρησιμοποιούμε για να παίξουμε ένα εισαγωγικό ήχο κατά την εκκίνηση της εφαρμογής μας και για να ενεργοποιήσουμε την δυνατότητα επιλογής της συσκευής που θα εμφανίζεται στην εκάστοτε λίστα.

Πίνακας 8-21: Η συνάρτηση onStart

```
@Override
    protected void onStart() {
        super.onStart();
       listViewDetected.setOnItemClickListener(listItemClicked);
       listViewPaired.setOnItemClickListener(listItemClickedonPaired);
        setVolumeControlStream(AudioManager.STREAM_MUSIC);
        player = MediaPlayer.create(this,R.raw.intro);
        player.start();
    }
```

Στην παρακάτω λίστα ακολουθεί ο κώδικας όπου παίρνουμε τις τιμές του επιταχυνσιόμετρου (Πίνακας 8-22). Μόλις πάρουμε τις τιμές από το αισθητήριο αυτό, δημιουργούμε τις απαραίτητες προϋποθέσεις όπου θα μας βοηθήσουν αργότερα, να κινούμε το τηλεχειριζόμενο αυτοκίνητο απλά κινώντας την συσκευή μας αναλόγως προς τα που θέλουμε να κατευθύνουμε το τηλεχειριζόμενο μας. Κάτι πολύ σημαντικό που πρέπει να κατανοήσουμε για το σύστημα συντεταγμένων είναι ότι οι άξονες δεν αλλάζουν όταν αλλάζει ο προσανατολισμός της συσκευής.

Πίνακας 8-22: Η συνάρτηση onAccuracyChanged

```
@Override
      public final void onAccuracyChanged(Sensor sensor, int accuracy) {

      }

      @Override
      public final void onSensorChanged(SensorEvent event) {
            x = event.values[0];
            y = event.values[1];
            z = event.values[2];
            String resultX = String.format("%.1f", x);
            String resultY = String.format("%.1f", y);
            String resultZ = String.format("%.1f", z);
            if (accelGyro == 1) {
                  if (y > 3) {
                        selectBtn = "R";
                        mHandler2.removeCallbacks(loop2);
                        loop2.run();
                  }
                  else if (y < -3) {
                        selectBtn = "L";
                        mHandler2.removeCallbacks(loop2);
                        loop2.run();
                  }
                  else if (z > 8) {
                        selectBtn = "F";
                        mHandler2.removeCallbacks(loop2);
                        loop2.run();
                  }
                  else if (z < -4) {
                        selectBtn = "D";
```

```
                        mHandler2.removeCallbacks(loop2);

                        loop2.run();
                  } else {
                        mHandler2.removeCallbacks(loop2);
                  }
            }
            tv.setText("X axis" + "\t" + resultX + " m/s2");
            tv1.setText("Y axis" + "\t" + resultY + " m/s2");
            tv2.setText("Z axis" + "\t" + resultZ + " m/s2");
      }
```

Επίσης σημαντικές, οι μέθοδοι onRestart(), onResume(),onPause(), onStop() και onDestroy() χρησιμοποιούνται όποτε κρίνεται απαραίτητο (Πίνακας 8-23). Οι onResume() και onPause() χρησιμοποιούνται για την ενεργοποίηση και απενεργοποίηση αντίστοιχα του επιταχυνσιόμετρου. Στην onStop(), σταματάει ο ήχος κατά την ελαχιστοποίηση της εφαρμογής, δεν κρίνεται τόσο απαραίτητο καθώς ο ήχος που χρησιμοποιούμε είναι μικρής διάρκειας, ωστόσο βοηθά στο να καταλάβουμε πως λειτουργεί. Τις onRestart(), και onDestroy() δεν κρίθηκε απαραίτητο να τις χρησιμοποιήσουμε, αυτό όμως δεν αναιρεί την σπουδαιότητα τους, για αυτό και τις έχουμε γράψει ώστε να φαίνονται στον κώδικα μας.

Πίνακας 8-23: Η συνάρτηση onRestart

```
    @Override
    protected void onRestart()
    {
       super.onRestart();
    }

    @Override
    protected void onResume()
    {
    super.onResume();
    mSensorManager.registerListener(this,                                    mAccelerometer,
SensorManager.SENSOR_DELAY_NORMAL);
    }

    @Override
    protected void onPause()
    {
    super.onPause();
    mSensorManager.unregisterListener(this);
    }

    @Override
    protected void onStop()
    {
       super.onStop();
       player.stop();
    }
    @Override
    protected void onDestroy()
    {
       super.onDestroy();
    }
```

Η μέθοδος `onCreateOptionsMenu()`, είναι απαραίτητη και χρησιμοποιείται για την αρχικοποίηση των περιεχομένων του μενού επιλογών της εφαρμογής μας (Πίνακας 8-24). Καλείται μία και μόνο φορά, την στιγμή που εμφανίζεται το μενού για πρώτη φορά στον χρήστη. Παράμετρος του είναι το `menu`, δηλαδή το μενού επιλογών στο οποίο τοποθετούμε τα στοιχεία μας.

Πίνακας 8-24: Η μέθοδος onCreateOptionsMenu

```
  @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        MenuInflater inflater = getMenuInflater();
        inflater.inflate(R.menu.bluetooth_class, menu);
        return super.onCreateOptionsMenu(menu);
    }
```

Ο βασικός σκοπός της μεθόδου `getPairedDevices()`, είναι να εμφανίσουμε στην κατάλληλη λίστα, δηλαδή στην «*ListViewPaired*» (η δεξιά λίστα εκ των δύο που υπάρχουν στην εφαρμογή μας) τις συσκευές που έχουν ήδη σχηματίσει ζεύξη με την συσκευή μας. Αυτό επιτυγχάνεται με τον παρακάτω κώδικα (Πίνακας 8-25).

Πίνακας 8-25: Η συνάρτηση getPairedDevices

```
private void getPairedDevices() {

        adapter.clear();
        adapter.notifyDataSetChanged() ;
        Set<BluetoothDevice> pairedDevice =           bluetoothAdapter.getBondedDevices();
        if(pairedDevice.size()>0)
        {
            for(BluetoothDevice device : pairedDevice)
            {
                 arrayListpaired.add(device.getName()+"\n"+device.getAddress());
               arrayListPairedBluetoothDevices.add(device);
            }
        }
        adapter.notifyDataSetChanged();
    }
```

Στην εφαρμογή μας, θα χρειαστεί να κάνουμε αναζήτηση συσκευών, πιο συγκεκριμένα θα χρειαστεί να βρούμε το ασύρματο Bluetooth module, το οποίο βρίσκεται στο Arduino, να συνδεθούμε σε αυτό και κατόπιν να έχουμε τον ασύρματο έλεγχο του οχήματος (Πίνακας 8-26). Με την παρακάτω κλάση, την `ListItemClicked`, η οποία υλοποιεί τη διασύνδεση `OnItemClickListener`, μας δίνεται η δυνατότητα μόλις εντοπιστεί η ασύρματη συσκευή, να κάνουμε κλικ στο όνομα της που θα εμφανιστεί στην ανάλογη λίστα, την «*ListViewDetected*» (εκ των δύο λιστών στην εφαρμογή μας, είναι η αριστερή), και κατόπιν θα κληθεί η συνάρτηση `connect(bdDevice)`, θα την δούμε παρακάτω, στην οποία θα δημιουργηθεί η σύνδεση με την ασύρματη Bluetooth συσκευή.

Πίνακας 8-26: Η κλάση ListItemClicked

```
class ListItemClicked implements OnItemClickListener
    {
        @Override
        public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
            bdDevice = arrayListBluetoothDevices.get(position);
            Log.i("Log", "The dvice : "+bdDevice.toString());
            connect(bdDevice);
        }
    }
```

Όπως αναφέρω παραπάνω, υπάρχει η κατάλληλη λίστα στην οποία εμφανίζονται τυχόν συσκευές με τις οποίες η συσκευή μας έχει σχηματίσει ζεύγος, με την κλάση `ListItemClickedonPaired` μας δίνεται η δυνατότητα να αφαιρέσουμε το δεσμό που έχει γίνει μεταξύ των δύο αυτών συσκευών (Πίνακας 8-27). Κάνουμε κλικ στην συσκευή που θέλουμε να κάνουμε «unpair» και κατόπιν καλείτε η συνάρτηση `removeBond(bdDevice)` η οποία αναλαμβάνει την διαδικασία αυτή.

Πίνακας 8-27: Η κλάση ListItemClickedonPaired

```
  class ListItemClickedonPaired implements OnItemClickListener
    {
        @Override
        public void onItemClick(AdapterView<?> parent, View view, int position,long id) {
            bdDevice = arrayListPairedBluetoothDevices.get(position);
            try {
                Boolean removeBonding = removeBond(bdDevice);
                if(removeBonding)
                {
                    arrayListpaired.remove(position);
                    adapter.notifyDataSetChanged();
                }
                Log.i("Log", "Removed"+removeBonding);
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    }
```

Αρκετά σημαντικός ο ρόλος της μεθόδου `connect(BluetoothDevice bdDevice)`, εδώ είναι το σημείο όπου η δύο ασύρματες Bluetooth συσκευές συνδέονται πραγματικά (Πίνακας 8-28). Να υπενθυμίσω πως η συγκεκριμένη συνάρτηση καλείται μόλις εμείς επιλέξουμε από την λίστα την ασύρματη συσκευή με την οποία επιθυμούμε να συνδεθούμε. Η σύνδεση επιτυγχάνεται χρησιμοποιώντας το mac Address της ασύρματης συσκευής και εν συνεχεία δημιουργώντας ένα RFCOMM Bluetooth Socket, `createRfcommSocketToServiceRecord(MY_UUID)`, έτοιμο να εξασφαλίσει εξερχόμενη σύνδεση με την απομακρυσμένη συσκευή. Μετά καλείτε η `connect()` η οποία θα εξασφαλίσει την σύνδεση με την ασύρματη συσκευή. Να αναφερθούμε λίγο στο μπλοκ **`try`**, χρησιμεύει στην προστασία του κώδικα που περιέχει την μέθοδο η οποία μπορεί να προκαλέσει μία εξαίρεση. Τις εξαιρέσεις τις χειριζόμαστε μέσα στο μπλοκ **`catch`**.

Πίνακας 8-28: Η συνάρτηση connect

```
public void connect(BluetoothDevice bdDevice) {
        Log.d("log", macAdress);
        BluetoothDevice device = bluetoothAdapter.getRemoteDevice(macAdress);
        Log.d("log", "Connecting to ... " + device);
        bluetoothAdapter.cancelDiscovery();
        try {
                btSocket = device.createRfcommSocketToServiceRecord(MY_UUID);
                btSocket.connect();
                Log.d("log", "Connection made.");
                 connected=true;
        } catch (IOException e) {
                try {
                        connected=false;
                        btSocket.close();
                        Toast.makeText(getApplicationContext(),       "Closing       Socket",
Toast.LENGTH_SHORT).show();
                } catch (IOException e2) {
                        Toast.makeText(getApplicationContext(),  "Unable  To  Close  Socket",
Toast.LENGTH_SHORT).show();
```

```
                        connected=false;
                        Log.d("log", "Unable to end the connection");
                }
                 Toast.makeText(getApplicationContext(),    "Socket    Creation    Failed",
Toast.LENGTH_LONG).show();
                Log.d("log", "Socket creation failed");
                callThread();
        }
    }
```

Η συνάρτηση `disConnect()`, χρησιμοποιείται για τον τερματισμό της σύνδεσης με την ασύρματη συσκευή εφόσον αυτό κριθεί απαραίτητο (Πίνακας 8-29). Αυτό πραγματοποιείται κλείνοντας το socket που είχε δημιουργηθεί κατά την κλήσης της συνάρτησης `connect(bdDevice)`, με την εντολή `btSocket.close();`.

Πίνακας 8-29: Η συνάρτηση disConnect

```
public void disConnect() {
     if (connected==true) {
      try {
             btSocket.close();
    } catch (IOException e) {
            try {
                    btSocket.close();
            } catch (IOException e2) {
                 Toast.makeText(getApplicationContext(),   "Unable   To   End   Connection",
Toast.LENGTH_SHORT).show();
                    Log.d("log", "Unable to end the connection");
            }
    }

} else{

    Toast.makeText(getApplicationContext(),    "No    Socket    Connection    Is    Available",
Toast.LENGTH_LONG).show();

}
    }
```

Η συγκεκριμένη μέθοδος χρησιμοποιείται από την κλάση `ListItemClickedonPaired` για την αφαίρεση του δεσμού μεταξύ δύο συσκευών (Πίνακας 8-30). Όταν βρίσκουμε μία ασύρματη Bluetooth συσκευή για πρώτη φορά, θα πρέπει να περάσουμε από μία διαδικασία πιστοποίησης έτσι ώστε να μας δοθεί πρόσβαση σε αυτήν. Αυτό γίνεται πληκτρολογώντας, έναν τετραψήφιο συνήθως κωδικό ή αλλιώς passkey. Η διαδικασία αυτή ονομάζεται pairing και η οποία πρέπει να επαναληφθεί σε περίπτωση που είχε προηγουμένως γίνει unpair ή σε περίπτωση που οι δύο συσκευές σχηματίζουν δεσμό για πρώτη φορά.

Πίνακας 8-30: Η συνάρτηση removeBond

```
  public boolean removeBond(BluetoothDevice btDevice)
    throws Exception
    {
        Class btClass = Class.forName("android.bluetooth.BluetoothDevice");
        Method removeBondMethod = btClass.getMethod("removeBond");
        Boolean returnValue = (Boolean) removeBondMethod.invoke(btDevice);
        return returnValue.booleanValue();
    }
```

Η συνάρτηση `startSearching()`, όπως λέει και το όνομα της, ξεκινά την διαδικασία αναζήτησης ασύρματης Bluetooth συσκευής (Πίνακας 8-31). Αυτό φυσικά γίνεται μόλις πατηθεί το κουμπί search που υπάρχει στη μπάρα ενεργειών της εφαρμογής μας. Κατόπιν θα εμφανιστεί σε εμάς ένας διάλογος όπου θα πρέπει να επιβεβαιώσουμε την έναρξη της διαδικασίας σάρωσης πατώντας «yes» ή «no» σε περίπτωση που θέλουμε να ακυρώσουμε την διαδικασία. Η διαδικασία της αναζήτησης ξεκινά καλώντας την `startDiscovery()`. Πρόκειται για μία ασύγχρονη διεργασία και η μέθοδος θα επιστρέψει μία λογική τιμή σε περίπτωση που η αναζήτηση ξεκίνησε επιτυχώς. Η διαδικασία της αναζήτησης κρατά περίπου δώδεκα δευτερόλεπτα. Πολύ σημαντικό να γνωρίζουμε, η εφαρμογή μας πρέπει να υλοποιεί έναν `BroadcastReceiver` για το *`ACTION_FOUND`* intent προκειμένου να λαμβάνει πληροφορίες για κάθε συσκευή που βρέθηκε, τον οποίο θα δούμε παρακάτω.

Πίνακας 8-31: Η συνάρτηση startSearching

```
private void startSearching() {
       new AlertDialog.Builder(this)
       .setTitle("Searching Devices")
       .setMessage("Start searching for Bluetooth devices ?")
       .setPositiveButton("Yes", new DialogInterface.OnClickListener() {
           public void onClick(DialogInterface dialog, int which) {
                if(bluetoothAdapter.isEnabled()){
                Log.i("Log", "in the start searching method");
                    bluetoothAdapter.startDiscovery();
                    IntentFilter              intentFilter              =              new
IntentFilter(BluetoothDevice.ACTION_FOUND);
                    BluetoothClass.this.registerReceiver(myReceiver, intentFilter);

                    detectedAdapter.clear();
                    detectedAdapter.notifyDataSetChanged();
                   Toast.makeText(getApplicationContext(),        "Searching        Started",
Toast.LENGTH_LONG).show();

                }
                else{
                   Toast.makeText(getApplicationContext(),      "Bluetooth      Is      Off",
Toast.LENGTH_LONG).show();
                }
           }
        })
       .setNegativeButton("No", new DialogInterface.OnClickListener() {
           public void onClick(DialogInterface dialog, int which) {
              if(bluetoothAdapter.isEnabled())
                {
                    bluetoothAdapter.disable();
                    Toast.makeText(getApplicationContext(),   "Closing   Bluetooth   Adapter",
Toast.LENGTH_LONG).show();
                }
              else{
                 Toast.makeText(getApplicationContext(),          "Search          Canceled",
Toast.LENGTH_LONG).show();
                }
           }
        })
        .show();
 }
```

Εδώ υλοποιείται ο `BroadcastReceiver` της εφαρμογής μας (Πίνακας 8-32). Δουλειά του είναι να ανιχνεύει αντικείμενα `Intent` που εκπέμπονται από την εφαρμογή μας, πιο συγκεκριμένα από την μέθοδο `startSearching()` κάνοντας χρήση του `IntentFilter`, δηλαδή φίλτρο προθέσεως, μέσα από την ενέργεια *`ACTION_FOUND`* τύπου intent. Μόλις βρεθεί η συσκευή, η `startSearching()` θα εκπέμψει το *`ACTION_FOUND`* intent, ο `BroadcastReceiver` θα λάβει την εκπομπή αυτή και θα κινηθεί αναλόγως τον τρόπο που έχει προγραμματιστεί.

Πίνακας 8-32: Ο BroadcastReceiver

```
    private BroadcastReceiver myReceiver = new BroadcastReceiver() {
        @Override
        public void onReceive(Context context, Intent intent) {
            Message msg = Message.obtain();
            String action = intent.getAction();
            if(BluetoothDevice.ACTION_FOUND.equals(action)){
             btDestroy();
                BluetoothDevice                              device                              =
intent.getParcelableExtra(BluetoothDevice.EXTRA_DEVICE);
                macAdress=device.getAddress();
                try
                {

                    device.getClass().getMethod("setPairingConfirmation",
boolean.class).invoke(device, true);
                    device.getClass().getMethod("cancelPairingUserInput",
boolean.class).invoke(device);
                }
                catch (Exception e) {
                    Log.i("Log", "Inside the exception: ");
                    e.printStackTrace();
                }

                if(arrayListBluetoothDevices.size()<1)
                {
            detectedAdapter.add(device.getName()+"\n"+device.getAddress());
                    arrayListBluetoothDevices.add(device);
                    detectedAdapter.notifyDataSetChanged();
                }
                else
                {

                    boolean flag = true;
                    for(int i = 0; i<arrayListBluetoothDevices.size();i++)
                    {

if(device.getAddress().equals(arrayListBluetoothDevices.get(i).getAddress()))
                        {
                            flag = false;
                        }
                    }
                    if(flag == true)
                    {

       detectedAdapter.add(device.getName()+"\n"+device.getAddress());
                        arrayListBluetoothDevices.add(device);
                        detectedAdapter.notifyDataSetChanged();

                    }
```

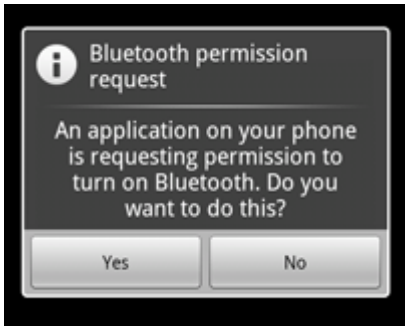
```
                }
            }
        }
    };
```

Τη μέθοδο `btDestroy()` την καλούμε αμέσως μετά που θα βρεθεί κάποια ασύρματη Bluetooth συσκευή (Πίνακας 8-33). Σκοπός της μεθόδου αυτής είναι να σταματήσει η αναζήτηση, αφού βρέθηκε η ασύρματη συσκευή και δεύτερον κάνουμε διαγραφή του Broadcast Receiver.

Πίνακας 8-33: Η συνάρτηση btDestroy

```
protected void btDestroy() {
      if (bluetoothAdapter != null) {
          Toast.makeText(getApplicationContext(),   "Device   Found   canceling   discovery",
Toast.LENGTH_LONG).show();
        bluetoothAdapter.cancelDiscovery();

      }
      unregisterReceiver(myReceiver);

    }
```

Η συνάρτηση `onBluetooth()` χρησιμοποιείται για την ενεργοποίησης του Bluetooth, και καλείται μόλις πατήσουμε το αντίστοιχο κουμπί «ON» από την μπάρα ενεργειών (Πίνακας 8-34). Θα εμφανιστεί ένας διάλογος (Εικόνα 8-14) που θα μας ενημερώνει ότι μία εφαρμογή επιθυμεί την ενεργοποίηση του Bluetooth, φυσικά συμφωνούμε, κατόπιν το Bluetooth ενεργοποιείται. Η αίτηση για την ενεργοποίησης του Bluetooth γίνεται καλώντας το *REQUEST_ENABLE_BT* intent.



Εικόνα 8-14: Αίτηση από την εφαρμογή μας για την ενεργοποίηση του Bluetooth.

Πίνακας 8-34: Η συνάρτηση onBluetooth

```
    private void onBluetooth() {
        if(!bluetoothAdapter.isEnabled())
        {
            Intent enableBtIntent = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
            startActivityForResult(enableBtIntent, REQUEST_ENABLE_BT);
            Log.i("Log", "Bluetooth is Enabled");
        }
    }
```

Η συνάρτηση `offBluetooth()` χρησιμεύει στην απενεργοποίηση του Bluetooth, μόλις πατηθεί το κουμπί «OFF» που βρίσκεται στην μπάρα ενεργειών (Πίνακας 8-35). Τότε είναι δηλαδή που θα κληθεί η συνάρτηση αυτή. Η απενεργοποίηση γίνεται καλώντας την `disable()`.

Πίνακας 8-35: Η συνάρτηση offBluetooth

```
    private void offBluetooth() {
        if(bluetoothAdapter.isEnabled())
        {
             bluetoothAdapter.disable();
             Toast.makeText(getApplicationContext(),        "Bluetooth        Is        Off",
Toast.LENGTH_LONG).show();
        }
        else{

             Toast.makeText(getApplicationContext(),        "Bluetooth        Is        Off",
Toast.LENGTH_LONG).show();
        }
    }
```

Με την `minimize()` γίνεται ελαχιστοποίηση του παραθύρου της εφαρμογής μας. Η συνάρτηση αυτή εκτελείται με το πάτημα του κουμπιού «MINIMIZE» στη μπάρα ενεργειών (Πίνακας 8-36).

Πίνακας 8-36: Η συνάρτηση minimize

```
private void minimize(){
       BluetoothClass.this.moveTaskToBack(true);
       Toast.makeText(BluetoothClass.this,
                    "Hiding", Toast.LENGTH_LONG).show();
   }
```

Η συνάρτηση `makeDiscoverable()` καθιστά την συσκευή μας ανιχνεύσιμη από άλλες συσκευές Bluetooth. Δεν είναι απαραίτητη στην παρούσα εργασία καθώς δεν πρόκειται άλλη συσκευή να συνδεθεί με την δική μας. Ωστόσο έχει ενδιαφέρον να δούμε τον τρόπο που υλοποιείται (Πίνακας 8-37).

Πίνακας 8-37: Η συνάρτηση makeDiscoverable

```
private void makeDiscoverable() {
        Intent discoverableIntent = new Intent(BluetoothAdapter.ACTION_REQUEST_DISCOVERABLE);
        discoverableIntent.putExtra(BluetoothAdapter.EXTRA_DISCOVERABLE_DURATION, 300);
        startActivity(discoverableIntent);
        Log.i("Log", "Discoverable ");
    }
```

Υπάρχουν δύο βασικές χρήσεις ενός `Handler`. Η πρώτη είναι για να προγραμματίσουμε μηνύματα και runnables τα οποία θα εκτελεστούν μελλοντικά, και δεύτερον για να τοποθετήσουμε στην ουρά ενέργειες που πρέπει να εκτελεστούν σε ένα διαφορετικό νήμα από το δικό μας. Εδώ εμείς κάνουμε χρήση του `Runnable`, το οποίο αποτελεί μία διασύνδεση (Interface) και περιέχει μία μόνο μέθοδο την `run()`, μέσα στην οποία βρίσκεται το κομμάτι του κώδικα το οποίο θα εκτελεστεί (Πίνακας 8-38). Να σημειώσουμε ότι η `Runnable` χρησιμοποιείται συχνά μαζί με νήματα (Threads) και περιγράφει την λειτουργία τους, δηλαδή το τι ακριβώς πρέπει να κάνουν. Το παρακάτω κομμάτι κώδικα καλείται μόλις πατάμε τα κουμπιά πλοήγησης της εφαρμογής μας, δηλαδή τα κουμπιά που κατευθύνουν το τηλεχειριζόμενο όχημα. Η συνάρτηση `sendData` είναι αυτή που αποστέλλει τα δεδομένα για την κίνηση του οχήματος στο Arduino. Κάθε κουμπί όπως θα δούμε παρακάτω είναι ανάλογο της κίνησης που πρέπει να εκτελεί το όχημα μας.

Πίνακας 8-38: Handler και Runnable

```
    private String selectBtn="X";
    private Handler mHandler = new Handler();
    private Runnable loop = new Runnable()
    {
        public void run()
        {
             if (selectBtn=="L"){
            Log.i("repeatBtn", "repeat click L");
            mHandler.postDelayed(this,19);
            sendData("L",connected);
             }

             if (selectBtn=="R"){
                Log.i("repeatBtn", "repeat click R");
                mHandler.postDelayed(this,19);
                sendData("R",connected);
             }


             if (selectBtn=="F"){
                Log.i("repeatBtn", "repeat click F");
                mHandler.postDelayed(this,19);
                sendData("F",connected);
             }

             if (selectBtn=="D"){
                Log.i("repeatBtn", "repeat click D");
                mHandler.postDelayed(this,19);
                sendData("D",connected);
             }

             if (selectBtn=="E"){
                Log.i("repeatBtn", "repeat click E");
                mHandler.postDelayed(this,19);
                sendData("E",connected);
             }
        }
    };
```

Όμοια με παραπάνω, το συγκεκριμένο Runnable χρησιμοποιείται για την κίνηση που πρόκειται να πραγματοποίηση το τηλεχειριζόμενο όχημα καλώντας την μέθοδο `sendData()` η οποία είναι υπεύθυνη να αποστείλει στο Arduino τα δεδομένα για την κίνηση που πρόκειται να εφαρμόσει (Πίνακας 8-39). Η διαφορά με το παραπάνω είναι πως εδώ το `Runnable` χρησιμοποιείται για να κινήσουμε το όχημα μέσω του επιταχυνσιόμετρου της συσκευής μας και όχι μέσω του σταυρού πλοήγησης που έχουμε δημιουργήσει στην εφαρμογή.

Πίνακας 8-39: Handle και Runnable (2)

```
    private Handler mHandler2 = new Handler();
    private Runnable loop2 = new Runnable()
    {
        public void run()
        {
             if (selectBtn=="L"){
                Log.i("repeatBtn", "repeat click L");
                sendData("L",connected);
             }
```

```
            if (selectBtn=="R"){
                  Log.i("repeatBtn", "repeat click R");
                  sendData("R",connected);
                  }
            if (selectBtn=="F"){
                  Log.i("repeatBtn", "repeat click F");
                  sendData("F",connected);
                  }
            if (selectBtn=="D"){
                  Log.i("repeatBtn", "repeat click D");
                  sendData("D",connected);
                  }
            if (selectBtn=="E"){
                  Log.i("repeatBtn", "repeat click E");
                  sendData("E",connected);
                  }
        }
    };
```

Η συνάρτηση `sendData()` είναι αυτή που θα αναλάβει να αποστείλει την ροή δεδομένων στο Arduino μέσω του Bluetooth (Πίνακας 8-40). Καλείτε όπως είδαμε παραπάνω από την μέθοδο `run()` της διεπαφής `Runnable`. Η `sendData()` μας δίνει μία εξερχόμενη ροή η οποία συνδέεται με το Socket που προηγούμενα είχαμε δημιουργήσει, κατά την σύνδεση με την ασύρματη Bluetooth συσκευή, καλώντας την `getOutputStream()`. Στη συνέχεια, τα δεδομένα που επιθυμούμε να αποστείλουμε εκχωρούνται σε ένα πίνακα από Byte και μετέπειτα χρησιμοποιώντας την `write()` γράφουμε τα δεδομένα αυτά στην εξερχόμενη ροή δεδομένων.

Πίνακας 8-40: Η συνάρτηση sendData

```
    private void sendData (String data,Boolean conn) {
      if(conn==true){
        try {
                outStream = btSocket.getOutputStream();
        } catch (IOException e) {
                Log.d("log", "Bug BEFORE Sending stuff", e);
        }

        String message = data;
        byte[] msgBuffer = message.getBytes();

        try {
                outStream.write(msgBuffer);
        } catch (IOException e) {
                Log.d("log", "Bug while sending stuff", e);
        }
      }
      else {
      }
}
```

Στη μέθοδο `addListenerOnButton()` αρχικοποιούμε τα κουμπιά που χρησιμοποιούνται για τον έλεγχο του τηλεχειριζόμενου οχήματος (Πίνακας 8-41). Αρχικά γίνεται η αντιστοίχηση του κάθε κουμπιού με εκείνο που το αντιπροσωπεύει από την διάταξη της εφαρμογής μας π.χ `imageButton5 = (ImageButton) findViewById(R.id.imageButton5);` και στη συνέχεια ορίζουμε την λειτουργεία που κάθε ένα από αυτά θα πρέπει να εκτελεί. Όπως παρατηρούμε στον παρακάτω κώδικα, χρησιμοποιούμε τον `OnTouchListener()` για τα κουμπιά πλοήγησης του οχήματος και τον `OnClickListener()` για το κουμπί ενεργοποίησης της δυνατότητας ελέγχου του τηλεχειριζόμενου οχήματος μέσω του επιταχυνσιόμετρου. Η διαφορά αυτών των δύο είναι ότι με τον πρώτο μπορώ να

ρυθμίσω τις ενέργειες που θα εκτελεί το κουμπί σε παρατεταμένο πάτημα, *ACTION_DOWN*, το οποίο μας βολεύει ιδιαίτερα σε αντίθεση με τον `OnClickListener` που αναγνωρίζει απλά το πάτημα (όχι παρατεταμένο) ενός κουμπιού. Πιο συγκεκριμένα, κατά το παρατεταμένο πάτημα ενός κουμπιού, καλείται η μέθοδος `run()` του `Runnable()` που είδαμε παραπάνω η οποία θε εκτελεστεί για την τιμή που έχει πάρει η μεταβλητή `selectBtn` και η οποία θα αποστείλει στην μέθοδο `sendData` τα δεδομένα προς αποστολή στο Arduino. Μόλις αφήσουμε κάποιο από τα κουμπιά πλοήγησης, *ACTION_UP*, σταματάει η κλήση της `run()` άρα και η αποστολή δεδομένων και οποιαδήποτε τυχόν δεδομένα είχαν μπει σε ουρά προς αποστολή αφαιρούνται καλώντας την `removeCallbacks(loop)`.

Πίνακας 8-41: Η μέθοδος addListenerOnButton

```
public void addListenerOnButton() {
              imageButton6 = (ImageButton) findViewById(R.id.imageButton6);
              imageButton5 = (ImageButton) findViewById(R.id.imageButton5);
              imageButton4 = (ImageButton) findViewById(R.id.imageButton4);
              imageButton3 = (ImageButton) findViewById(R.id.imageButton3);
              imageButton2 = (ImageButton) findViewById(R.id.imageButton2);
              imageButton1 = (ImageButton) findViewById(R.id.imageButton1);

              gearbutton1 = (Button) findViewById(R.id.button1);
              gearbutton2 = (Button) findViewById(R.id.button2);
              gearbutton3 = (Button) findViewById(R.id.button3);
              gearbutton4 = (Button) findViewById(R.id.button4);
              player2 = MediaPlayer.create(this, R.raw.click);

              imageButton1.setOnTouchListener(new OnTouchListener() {
                     @Override
                     public boolean onTouch(View v, MotionEvent event) {
                            if (event.getAction() == MotionEvent.ACTION_DOWN) {
                                   v.setPressed(true);
                                   selectBtn = "L";
                                   mHandler.removeCallbacks(loop);
                                   loop.run();
                            } else if (event.getAction() == MotionEvent.ACTION_UP) {
                                   v.setPressed(false);
                                   mHandler.removeCallbacks(loop);
                                   sendData("S", connected);
                            }
                            return true;
                     }
              });

              imageButton2.setOnTouchListener(new OnTouchListener() {
                     @Override
                     public boolean onTouch(View v, MotionEvent event) {
                            if (event.getAction() == MotionEvent.ACTION_DOWN) {
                                   v.setPressed(true);
                                   selectBtn = "D";
                                   mHandler.removeCallbacks(loop);
                                   loop.run();
                            } else if (event.getAction() == MotionEvent.ACTION_UP) {
                                   v.setPressed(false);
                                   mHandler.removeCallbacks(loop);
                                   sendData("S", connected);
                            }
                            return true;
                     }
```

```
	});

	imageButton3.setOnTouchListener(new OnTouchListener() {
		@Override
		public boolean onTouch(View v, MotionEvent event) {
			if (event.getAction() == MotionEvent.ACTION_DOWN) {
				v.setPressed(true);
				selectBtn = "F";
				mHandler.removeCallbacks(loop);
				loop.run();
			} else if (event.getAction() == MotionEvent.ACTION_UP) {
				v.setPressed(false);
				mHandler.removeCallbacks(loop);
				sendData("S", connected);
			}
			return true;
		}
	});
	imageButton4.setOnTouchListener(new OnTouchListener() {
		@Override
		public boolean onTouch(View v, MotionEvent event) {
			if (event.getAction() == MotionEvent.ACTION_DOWN) {
				v.setPressed(true);
				selectBtn = "R";
				mHandler.removeCallbacks(loop);
				loop.run();
			} else if (event.getAction() == MotionEvent.ACTION_UP) {
				v.setPressed(false);
				mHandler.removeCallbacks(loop);
				sendData("S", connected);
			}
			return true;
		}
	});

	imageButton5.setOnTouchListener(new OnTouchListener() {
		@Override
		public boolean onTouch(View v, MotionEvent event) {
			if (event.getAction() == MotionEvent.ACTION_DOWN) {
				v.setPressed(true);
				selectBtn = "E";
				mHandler.removeCallbacks(loop);
				loop.run();
			} else if (event.getAction() == MotionEvent.ACTION_UP) {
				v.setPressed(false);
				mHandler.removeCallbacks(loop);
				sendData("S", connected);
			}
			return true;
		}
	});

	imageButton6.setOnClickListener(new OnClickListener() {
		@Override
		public void onClick(View arg0) {
			player2.start();
			if (accelGyro == 0) {
			    accelGyro = 1;
			}
```

```
                        else {
                            accelGyro = 0;
                        }

                }
        });

        gearbutton1.setOnClickListener(new OnClickListener() {
                @Override
                public void onClick(View arg0) {
                        sendData("Z", connected);
                        seekshow.setText("PWM 64");
                }
        });
        gearbutton2.setOnClickListener(new OnClickListener() {
                @Override
                public void onClick(View arg0) {
                        sendData("X", connected);
                        seekshow.setText("PWM 85");
                }
        });
        gearbutton3.setOnClickListener(new OnClickListener() {
                @Override
                public void onClick(View arg0) {
                        sendData("C", connected);
                        seekshow.setText("PWM 200");
                }
        });
        gearbutton4.setOnClickListener(new OnClickListener() {
                @Override
                public void onClick(View arg0) {
                        sendData("V", connected);
                        seekshow.setText("PWM 255");
                }
        });
    }
```

Η μέθοδος `onOptionsItemSelected(MenuItem item)` καλείται κάθε φορά που έχει επιλεγεί κάποιο στοιχείο από το μενού της μπάρας ενεργειών μας, δηλαδή κάποιο κουμπί στην δική μας περίπτωση (Πίνακας 8-42). Μόλις πατηθεί κάποιο από τα κουμπιά καλείται και η αντίστοιχη μέθοδος π.χ για το κουμπί *buttonOn* καλείται η συνάρτηση για την ενεργοποίηση του Bluetooth με όνομα `onBluetooth()`. Όλες οι παρακάτω μέθοδοι έχουν αναφερθεί παραπάνω. Τέλος, η κλάση `InfoClass` καλείται επίσης κατά το πάτημα ενός κουμπιού, του *action_info*.

Πίνακας 8-42: Η μέθοδος onOptionsItemSelected

```
@Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case R.id.buttonDesc:
             makeDiscoverable();
             return true;
            case R.id.action_paired:
             getPairedDevices();
             return true;
            case R.id.settings:
             Intent sets =new Intent();
             sets.setAction(android.provider.Settings.ACTION_BLUETOOTH_SETTINGS);
             startActivity(sets);
             return true;
            case R.id.buttonOn:
```

```
            onBluetooth();
            return true;
          case R.id.buttonOff:
            offBluetooth();
            return true;
          case R.id.buttonSearch:
            arrayListBluetoothDevices.clear();
             startSearching();
             return true;
          case R.id.action_minimize:
             minimize();
             return true;
          case R.id.buttonDisconnect:
             disConnect();
             return true;
          case R.id.action_info:
              Toast.makeText(BluetoothClass.this,
                             "test info test", Toast.LENGTH_LONG).show();
              Intent intent = new Intent(this, InfoClass.class);
              startActivity(intent);
              return true;
          default:
              return super.onOptionsItemSelected(item);
      }
  }
```

8.3.2 Η Κλάση InfoClass

Το δεύτερο Activity της εφαρμογή μας ξεκινά με το όνομα του πακέτου που βρίσκεται η κλάση μας, αυτό είναι το `com.example.bluetoothcarteicrete`, και στην συνέχεια χρησιμοποιείται η δήλωση **`import`** για να εισάγουμε τις απαραίτητες κλάσεις και διασυνδέσεις (Interfaces). Στον παρακάτω κώδικα φαίνεται το σύνολο των κλάσεων και διασυνδέσεων που εισήχθησαν (Πίνακας 8-43)

Πίνακας 8-43: Η κλάση InfoClass

```
package com.example.bluetoothcarteicrete;
import android.app.Activity;
import android.os.Build;
import android.os.Bundle;
import android.view.MenuItem;
import android.widget.TextView;
```

Στην συνέχεια ξεκινά η κλάση μας, την έχουμε ορίσει ως **`public`**, δηλαδή δημόσια, που σημαίνει ότι είναι ορατή σε όλες τις άλλες κλάσεις σε όλα τα πακέτα. Η κλάση μας, όπως επίσης έχουμε αναφέρει νωρίτερα για την κλάση BluetoothClass, ορίζεται με την χρήση της δεσμευμένης λέξης **`class`** και ακολουθεί το όνομα της το οποίο είναι το `InfoClass` (Πίνακας 8-44). Η δεσμευμένη λέξη **`extends`** χρησιμοποιείται για να υποδείξει την υπερκλάση της `InfoClass` η οποία είναι και πάλι η Activity. Η `InfoClass` με απλά λόγια αποτελεί υποκλάση της κλάσης `Activity`, αυτό σημαίνει πως η `InfoClass` κληρονομεί όλες τις ιδιότητες και την συμπεριφορά της `Activity`. Επίσης η `InfoClass` υλοποιεί την διασύνδεση (Interface) με όνομα `MenuItem`, η οποία επιτρέπει την πρόσβαση σε στοιχεία του μενού που έχουν δημιουργηθεί νωρίτερα. Με το `setContentView` ορίζουμε το Layout που χρησιμοποιεί το συγκεκριμένο Activity, αυτό είναι το *`activity_bluetooth_info`* το οποίο μπορούμε να το βρούμε στο φάκελο res>layout με όνομα activity_bluetooth_info.xml. Αμέσως μετά ενεργοποιούμε το κουμπί «πίσω» στην μπάρα ενεργειών, το οποίο μας δίνει την δυνατότητα να επιστρέψουμε στο κύριο layout της εφαρμογής μας, αυτό γίνεται καλώντας την `setDisplayHomeAsUpEnabled(true)`. Στη συνέχεια θέτουμε το κείμενο στο *`TextView1`* που νωρίτερα είχαμε δημιουργήσει στο XML αρχείο μας.

Πίνακας 8-44: Ενεργοποίηση του «Home Button»

```
public class InfoClass extends Activity {
	@Override
	protected void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.activity_bluetooth_info);
		if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.HONEYCOMB) {
			getActionBar().setDisplayHomeAsUpEnabled(true);
		}
		TextView et = (TextView) findViewById(R.id.TextView1);
		et.setText("Η εφαρμογή BluetoothCarTeiCrete αναπτύχθηκε το διάστημα 2013-2014 με σκοπό τον τηλεχειρισμό οχήματος ασύρματα χρησιμοποιώντας το πρωτόκολλο Bluetooth. Παρακάτω αναφέρω τον τρόπο λειτουργίας της εφαρμογής.\n\n"
				+ "Αριστερά υπάρχει ο σταυρός, αποτελεί το σύστημα διεύθυνσης του οχήματος. Πάνω δεξιά υπάρχουν δύο λίστες, πατώντας το κουμπί Search για να γίνει αναζήτηση συσκευής, θα εμφανιστούν οι διαθέσιμες συσκευές στην αριστερή λίστα, πατώντας πιάνω στην συσκευή σχηματίζουμε σύνδεση μαζί της. Πατώντας το κουμπί Paired εμφανίζονται οι συσκευές που έχουν γίνει Paired με την συσκευή μας, πατώντας πάνω την συσκευή μπορούμε να αφαιρέσουμε τον δεσμό εφόσον επιθυμούμε. Κάτω δεξιά υπάρχουν οι τιμές του επιταχυνσιόμετρου της συσκευής μας στους άξονες x, y, και z. Επάνω από τις τιμές βρίσκεται ένα πράσινο κουμπί, πατώντας το ακινητοποιούμε το όχημα μας. Το λογότυπο του ΤΕΙ αποτελεί κουμπί για την ενεργοποίηση και την απενεργοποίηση αντίστοιχα της διεύθυνσης του οχήματος, βάση της κίνησης της συσκευής. Τέλος, τα τέσσερα κουμπιά κάτω στο κέντρο ρυθμίζουν την ταχύτητα με την οποία θα κινείται το όχημα (PWM).\n\n"
				+ "Στην μπάρα ενεργειών εκτός από τα κουμπιά Search και Paired, που ανέφερα παραπάνω, υπάρχουν μερικά ακόμα που προσφέρουν κάποιες πρόσθετες δυνατότητες. Πιο συγκεκριμένα, το κουμπί Settings μας δίνει την δυνατότητα να μεταφερθούμε στις ρυθμίσεις της συσκευής, τα κουμπιά ON και OF αντίστοιχα μας δίνουν την δυνατότητα άμεσα μέσα από την δική μας εφαρμογή να ενεργοποιήσουμε και να απενεργοποιήσουμε το Bluetooth. Το κουμπί DISCONNECT τερματίζει την σύνδεση με την εκάστοτε συσκευή Bluetooth, το DISCOVER καθιστά την συσκευή μας ορατή σε άλλες συσκευές και τέλος το MINIMIZE ελαχιστοποιεί το παράθυρο της εφαρμογής μας.");
		et.setKeyListener(null);
	}
```

Στη συνάρτηση `onOptionsItemSelected` (Πίνακας 8-45) ορίζεται η συμπεριφορά του συγκεκριμένου Activity σε περίπτωση που πατηθεί το κουμπί για την επιστροφή στο κεντρικό Activity, που ενεργοποιήσαμε προηγούμενα. Σε περίπτωση λοιπόν που το κουμπί αυτό πατηθεί, θα επιστέψουμε στο προηγούμενο Activity το οποίο είναι και το κύριο της εφαρμογής μας. Αυτό επιτυγχάνεται ουσιαστικά καλώντας την `finish()` η οποία τερματίζει το τρέχον Activity.

Πίνακας 8-45: Επιστροφή στο κύριο Activity

```
   @Override
	public boolean onOptionsItemSelected(MenuItem item) {
	    switch (item.getItemId()) {
	    case android.R.id.home:
	         finish();
	        return true;
	    }
	    return super.onOptionsItemSelected(item);
	}
	    }
```

9 Βιβλιογραφία

9.1 Βιβλία

[1] F.Darwin, Ian, 2012. *Android Cookbook*. First Edition, O'Reilly Media.
[2] Simon, Jonathan, 2011 *Head First Android Development.* First Edition, O'Reilly Media.
[3] Burnette, Ed, 2010. *Hello Android.* Third Edition, Pragmatic Bookshelf.
[4] Gargenta, Marko, 2011. *Learning Android*. First Edition, O'Reilly Media.
[5] Meier, Reto, 2012. *Professional Android 4 Application Development*. Third Edition, Wrox.
[6] Mednieks, Zigurd, 2011. *Programming Android*. First Edition, O'Reilly Media.
[7] Conder, Shane, 2010. *Android Wireless Application Development.* Second Edition, Addison-Wesley Professional.
[8] Matthews, Robbie, 2011. *Beginning Android Tablet Programming.* Fits Edition, Apress.
[9] Jordan, Lucas, 2011. *Practical Android Projects.* First Edition, Apress.
[10] Goransson, Andreas, 2013. *Android Open Accessory*. First Edition, Wrox.
[11] Timmis, Harold, 2011. *Practical Arduino Enginnering.* First Edition, Apress.
[12] Banzi, Massimo, 2011. *Getting Started with Arduino.* Second Edition, Maker Media.
[13] Floyd Kelly, James, 2013. *Arduino Adventures.* First Edition, Apress.
[14] Ramos Melgar, Enrique, 2012. *Arduino and Kinect Projects.* First Edition, Apress.
[15] Margolis, Michael, 2011. *Arduino Cookbook.* Second Edition, O'Reilly Media.
[16] Premeaux, Emery, 2011. *Arduino Projects to Save the World.* First Edition, Apress.
[17] Warren, John David, 2011. *Arduino Robotics.* First Edition, Apress.
[18] Karvinen, Tero, 2011. *Make: Arduino Bots and Gadgets.* First Edition, Maker Media.

9.2 Υπερσύνδεσμοι

[1] http://www.arduino.cc/
[2] https://www.sparkfun.com/
[3] http://batteryuniversity.com/learn/article/serial_and_parallel_battery_configurations
[4] http://www.zbattery.com/Connecting-Batteries-in-Series-or-Parallel
[5] http://technologytom.com/html/batteries_or_cells_.html
[6] http://hardwarefun.com/tutorials/creating-robots-using-arduino-h-bridge
[7] http://arduino.cc/en/Reference/Board
[8] http://www.instructables.com/id/Powering-Arduino-with-a-Battery/
[9] http://www.sengpielaudio.com/calculator-voltagedivider.htm
[10] http://arduino.cc/en/Tutorial/PWM
[11] https://www.bananarobotics.com/shop/How-to-use-the-L298N-Dual-H-Bridge-Motor-Driver
[12] http://arduinobasics.blogspot.gr/2013/01/arduino-basics-bluetooth-tutorial.html
[13] http://www.tautvidas.com/blog/2012/08/distance-sensing-with-ultrasonic-sensor-and-arduino/
[14] http://apcmag.com/arduino-project-8-stompy-the-robot-part-1.htm
[15] http://apcmag.com/arduino-masterclass-part-4-build-a-mini-robot.htm
[16] http://arduino.cc/en/Tutorial/sweep#.UwNnoYVU5zw
[17] http://playground.arduino.cc/Main/UltrasonicSensor#.UwPgV4VU5zw
[18] https://code.google.com/p/arduino-new-ping/
[19] http://forum.arduino.cc/index.php/topic,106043.0.html
[20] http://www.baldengineer.com/blog/2011/01/06/millis-tutorial/
[21] http://forum.pololu.com/viewtopic.php?f=24&t=5617
[22] http://www.instructables.com/id/How-to-add-6-extra-pins-to-your-Arduino-with-no-ex/
[23] http://gadgetmakersblog.com/arduino-power-consumption/
[24] http://www.learningaboutelectronics.com/Articles/Toggle-switch-wiring.php
[25] https://learn.adafruit.com/character-lcds/wiring-a-character-lcd

[26] http://learning.grobotronics.com/2013/07/controlling-lcd-displays-with-the-hitachi-hd44780-driver/
[27] http://www.hobbytronics.co.uk/arduino-tutorial10-lcd
[28] http://www.fibidi.com/arduino-lcd-16x2-characters/
[29] http://playground.arduino.cc/Main/ArduinoPinCurrentLimitations
[30] http://arduino-info.wikispaces.com/QuickRef#pwr
[31] http://arduinoarts.com/2011/09/tutorials-with-arduino-range-sensor-with-lcd-and-7-segment-led-display-arduino/
[32] http://www.gammon.com.au/forum/?id=11473
[33] http://playground.arduino.cc/Main/InternalTemperatureSensor
[34] https://www.sparkfun.com/datasheets/LCD/GDM1602K-Extended.pdf
[35] https://learn.sparkfun.com/tutorials/how-to-read-a-schematic/schematic-symbols-part-1
[36] http://electronicsclub.info/index.htm
[37] http://arduino.cc/en/Tutorial/BlinkWithoutDelay
[38] http://www.raywenderlich.com/32392/arduino-tutorial-for-complete-beginners-using-a-button
[39] https://learn.sparkfun.com/tutorials/using-the-bluesmirf/all
[40] http://www.wisegeek.com/what-is-bluetooth-pairing.htm
[41] http://www.pcmag.com/encyclopedia/term/60868/bluetooth-profiles
[42] http://electrosome.com/hc-sr04-ultrasonic-sensor-pic/
[43] http://www.directron.com/switchinfo.html
[44] https://learn.sparkfun.com/tutorials/switch-basics/poles-and-throws-open-and-closed
[45] https://learn.sparkfun.com/tutorials/capacitors/application-examples
[46] https://android.com/
[47] http://www.appbrain.com/stats/number-of-android-apps
[48] http://developer.android.com/index.html
[49] http://android-developers.blogspot.gr/2011/11/new-layout-widgets-space-and-gridlayout.html
[50] http://developer.android.com/guide/topics/ui/controls/button.html
[51] http://stackoverflow.com/questions/11863329/gridlayout-and-row-column-span-woe
[52] http://developer.android.com/guide/topics/connectivity/bluetooth.html
[53] http://developer.android.com/tools/device.html
[54] http://stackoverflow.com/questions/4805269/programmatically-register-a-broadcast-receiver
[55] http://digitalhacksblog.blogspot.gr/2012/05/android-example-bluetooth-discover-and.html
[56] http://www.tutorialspoint.com/android/android_bluetooth.htm
[57] http://www.basic4ppc.com/android/forum/threads/android-bluetooth-bluetoothadmin-tutorial.14768/
[58] http://www.javatpoint.com/android-bluetooth-list-paired-devices-example
[59] http://www.javatpoint.com/android-bluetooth-tutorial
[60] http://www.javatpoint.com/android-bluetooth-tutorial
[61] http://sonuasher89.blogspot.gr/2013/02/bluetooth-example-which-scans-for.html
[62] http://www.theappguruz.com/blog/android-bluetooth-connection-demo/
[63] http://stackoverflow.com/questions/14228289/android-device-bluetooth-pairing/14254202#14254202
[64] http://stackoverflow.com/questions/4755871/how-to-set-image-button-backgroundimage-for-different-state
[65] http://stackoverflow.com/questions/4903758/android-how-to-refresh-listview-contents
[66] http://www.mkyong.com/android/android-imagebutton-example/
[67] http://karanbalkar.com/2012/09/tutorial-1-android-accelerometer-demo/
[68] http://blog.softteco.com/2011/07/how-to-close-activity-and-all-children.html
[69] http://blog.rochdev.com/2011/05/update-ui-when-holding-button.html
[70] http://stackoverflow.com/questions/17459858/button-down-event-in-eclipse-android
[71] http://stackoverflow.com/questions/5171248/programmatically-connect-to-paired-bluetooth-device
[72] http://alotaboutandroid.blogspot.gr/2011/11/how-android-was-born.html
[73] http://socialcompare.com/en/comparison/android-versions-comparison
[74] http://www.androidcentral.com/android-z-what-jit

[75] http://www.87android.com/what-is-dalvik-virtual-machine-in-android/
[76] http://www.android-app-market.com/android-architecture.html
[77] https://software.intel.com/en-us/blogs/2014/06/18/art-vs-dalvik-introducing-the-new-android-x86-runtime
[78] http://pocketnow.com/2013/11/13/dalvik-vs-art
[79] http://www.anandtech.com/show/8231/a-closer-look-at-android-runtime-art-in-android-l
[80] http://justamomentgoose.wordpress.com/2013/06/04/android-started-note-2-android-file-apk-decompile/
[81] http://developer.android.com/tools/publishing/preparing.html
[82] http://developer.android.com/guide/components/intents-filters.html
[83] http://www.javatpoint.com/android-core-building-blocks
[84] http://developer.xamarin.com/guides/android/deployment,_testing,_and_metrics/publishing_an_application/
[85] http://developer.android.com/guide/components/services.html
[86] http://www.dormantroot.com/2011/04/androidunder-hood.html
[87] http://www.compiletimeerror.com/2013/12/content-provider-in-android.html#.VGuAEskxpMZ
[88] http://www.101apps.co.za/articles/processes-and-threads.html
[89] http://www.pcmag.com/encyclopedia/term/37856/api
[90] https://www.princeton.edu/~achaney/tmve/wiki100k/docs/Application_programming_interface.html
[91] http://www.sitepoint.com/know-your-layouts-in-android/
[92] http://androidplot.com/docs/borders-margins-and-padding/
[93] http://en.wikipedia.org/wiki/Pixel_density#Calculation_of_monitor_PPI
[94] http://developer.android.com/guide/practices/screens_support.html
[95] https://docs.oracle.com/javase/tutorial/java/javaOO/accesscontrol.html
[96] https://docs.oracle.com/javase/tutorial/java/IandI/subclasses.html
[97] http://icons8.com/icons/#!/3271/car
[98] http://stackoverflow.com/questions/3377288/how-to-remove-gravity-factor-from-accelerometer-readings-in-android-3-axis-accel

Επιστροφή στα περιεχόμενα