ΤΕΙ ΠΕΙΡΑΙΑ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΗΛΕΚΤΡΟΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Χειρισμός ηλεκτρικής κλειδαριάς με Arduino μέσω εφαρμογής Android

Γεώργιος Κόλλιας

Κωνσταντίνος Βελής

Εισηγητής: Ιωάννης Έλληνας, Καθηγητής

ΑΙΓΑΛΕΩ

ΜΑΙΟΣ 2014

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Χειρισμός ηλεκτρικής κλειδαριάς με Arduino μέσω εφαρμογής Android

Γεώργιος Κόλλιας

Α.Μ. 38102

Κωνσταντίνος Βελής

Α.Μ. 38460

Εισηγητής:

Ιωάννης Έλληνας, Καθηγητής

Εξεταστική επιτροπή:

Βελώνη Αναστασία

Αλατσαθιανός Σταμάτης

Ημερομηνία εξέτασης: 13/10/14

ΕΥΧΑΡΙΣΤΙΕΣ

Η παρούσα πτυχιακή εργασία ολοκληρώθηκε μετά από επίμονες προσπάθειες, σε ένα ενδιαφέρον γνωστικό αντικείμενο όπως αυτό της ενασχόλησης με τη δημιουργία Android εφαρμογών, αλλά και τον προγραμματισμό της υπολογιστικής πλατφόρμας Arduino. Την προσπάθειά μας αυτή υποστήριξε ο καθηγητής μας Ιωάννης Έλληνας, τον οποίο και θα θέλαμε να τον ευχαριστήσουμε.

Ακόμη θα θέλαμε να ευχαριστήσουμε τις οικογένειες μας που όλη τη διάρκεια της φοιτητικής μας ζωής στάθηκαν δίπλα μας και μας βοήθησαν να αντιμετωπίσουμε τις όποιες δυσκολίες μας παρουσιάστηκαν στις σπουδές μας όλα αυτά τα χρόνια.

ΠΕΡΙΛΗΨΗ

Η παρούσα πτυχιακή εργασία ασχολείται με τον χειρισμό μιας ηλεκτρική κλειδαριάς που είναι συνδεδεμένη με την πλατφόρμα arduino. Ο χειρισμός αυτός επιτυγχάνεται χρησιμοποιώντας ένα κινητό ή tablet με το λογισμικό android. Σκοπός της εργασίας είναι η υλοποίηση μιας απλής και φιλικής προς το χρήστη εφαρμογής, που θα του δίνει τη δυνατότητα να έχει πρόσβαση σε ένα ιδιωτικό χώρο παρέχοντας του ορισμένες πληροφορίες και ασφάλεια. Αντικείμενο της εργασίας είναι ο προγραμματισμός της πλατφόρμας arduino, η δημιουργία εφαρμογής android με τη βοήθεια της πλατφόρμας Amarino που δημιουργεί κανάλι επικοινωνίας μεταξύ του κινητού τηλέφωνου και του εξωτερικού κυκλώματος μέσω της χρήσης της τεχνολογίας Bluetooth.

ΕΠΙΣΤΗΜΟΝΙΚΗ ΠΕΡΙΟΧΗ: Προγραμματισμός Arduino και εφαρμογής Android

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: ηλεκτρική κλειδαριά, arduino, android, amarino, bluetooth

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

ΚΕΦΑΛΑΙΟ 1

ΕΙΣΑΓΩΓΗ

Αρχικά για την υλοποίηση της εργασίας βασικό κομμάτι ήταν η επιλογή και η αγορά της κατάλληλης πλατφόρμας Arduino μέσω της οποίας θα επιτύχουμε τη λειτουργία της κλειδαριάς. Επιπλέον, απαραίτητη ήταν η χρήση κινητού τηλέφωνου με Android λειτουργικό σύστημα και στο οποίο έγινε η εγκατάσταση της εφαρμογής Door lock που δημιουργήθηκε μέσω του προγράμματος eclipse, για το χειρισμό της ηλεκτρικής κλειδαριάς.

Εκτός της πλατφόρμας Arduino και της χρήσης Android εφαρμογής κρίθηκε αναγκαία η χρήση ενός απλού ηλεκτρονικού κυκλώματος για την ασύρματη επικοινωνία μέσω Bluetooth μεταξύ χρήστη και κλειδαριάς το οποίο και συνδέθηκε με το Arduino.

Για την αγορά του Arduino και των εξαρτημάτων για την υλοποίηση του κυκλώματος, μελετήσαμε μέσω διαδικτύου οικονομικές επιλογές εξοπλισμού, όπως για παράδειγμα η αγορά ηλεκτρικής κλειδαριάς και Bluetooth. Οι αγορές πραγματοποιήθηκαν όλες από το διαδίκτυο.

Παρακάτω θα ακολουθήσουν τα βήματα που πραγματοποιήθηκαν για την πτυχιακή εργασία και σε θεωρητικό αλλά και σε πρακτικό μέρος.

ΚΕΦΑΛΑΙΟ 2

2 ΕΙΣΑΓΩΓΗ ΣΤΟ ARDUINO

Το arduino είναι μια ‘ανοιχτού κώδικα’ πλατφόρμα ηλεκτρονικών βασισμένη σε ευέλικτο και εύκολο στη χρήση hardware και software που προορίζεται για οποιονδήποτε έχει στοιχειώδεις γνώσεις ηλεκτρονικών και προγραμματισμού και ενδιαφέρεται να δημιουργήσει διάφορα διαδραστικά αντικείμενα ή περιβάλλοντα.

Στην ουσία, πρόκειται για ένα ηλεκτρονικό κύκλωμα που βασίζεται στον μικροελεγκτή ATmega της Atmel και του οποίου όλα τα σχέδια, καθώς και το software που χρειάζεται για την λειτουργία του, διανέμονται ελεύθερα και δωρεάν ώστε να μπορεί να κατασκευαστεί από τον καθένα. Αφού κατασκευαστεί, μπορεί να συμπεριφερθεί σαν ένας μικροσκοπικός υπολογιστής, αφού ο χρήστης μπορεί να συνδέσει επάνω του πολλαπλές μονάδες εισόδου/εξόδου και να προγραμματίσει τον μικροελεγκτή να δέχεται δεδομένα από τις μονάδες εισόδου, να τα επεξεργάζεται και να στέλνει κατάλληλες εντολές στις μονάδες εξόδου.

Το Arduino, δεν είναι ούτε ο μοναδικός, ούτε και ο καλύτερος δυνατός τρόπος για την δημιουργία μιας οποιασδήποτε διαδραστικής ηλεκτρονικής συσκευής. Όμως το κύριο πλεονέκτημά του είναι η τεράστια κοινότητα που το υποστηρίζει και η οποία έχει δημιουργήσει, συντηρεί και επεκτείνει μια ανάλογου μεγέθους online γνωσιακή βάση.

Επειδή απευθύνεται κυρίως σε αρχάριους των ηλεκτρονικών και επειδή, παρά τις αναλυτικότατες οδηγίες που υπάρχουν, δεν έχουν όλοι τις γνώσεις και τα μέσα να κατασκευάσουν μια ηλεκτρονική πλακέτα, κυκλοφορούν έτοιμες, προκατασκευασμένες πλακέτες Arduino που κοστίζουν περίπου €25. Με λίγα χρήματα παραπάνω μάλιστα, οι περισσότεροι προμηθευτές διαθέτουν Arduino Starter Kit, τα οποία, εκτός από το ίδιο το Arduino, περιέχουν διάφορα άλλα εξαρτήματα και εργαλεία που μπορεί να χρειαστείτε

για τις πρώτες εφαρμογές (όπως το απαραίτητο καλώδιο USB για την σύνδεση με τον υπολογιστή, ράστερ, καλώδια, LED, διακόπτες, ποτενσιόμετρα, αντιστάσεις, διόδους, τρανζίστορ κ.λπ.).

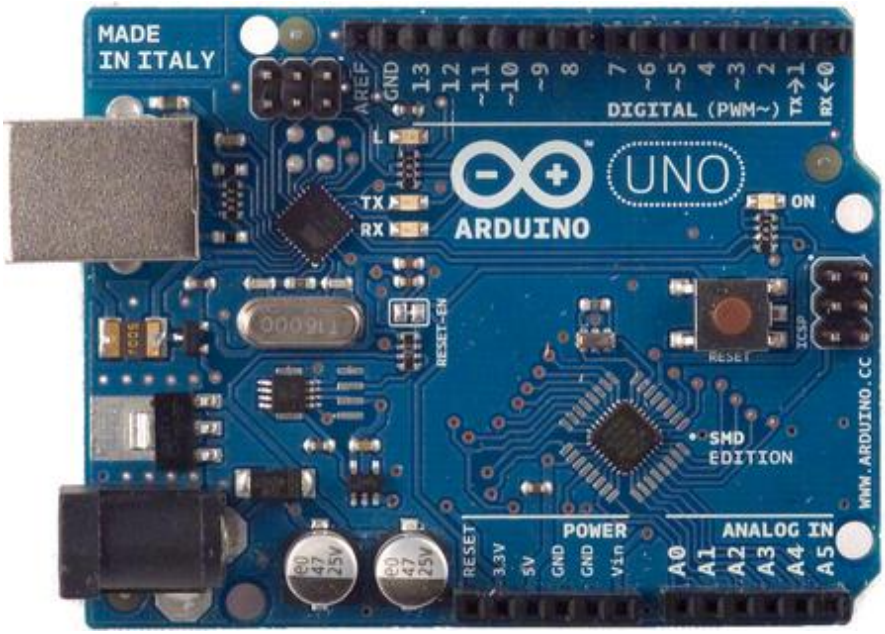
2.1 Μοντέλα Arduino

Στην αγορά διατίθενται πολλά μοντέλα της πλατφόρμας και στην παρούσα εργασία χρησιμοποιήθηκε το Arduino Uno. Μερικά από τα μοντέλα Arduino αναφέρονται παρακάτω:

- Arduino Diecimila
- Arduino Nuova Generarazione
- Arduino Extreme
- Arduino Usb
- Arduino Lilypad
- Arduino Nano
- Arduino Mini
- Arduino Duemilanove
- Arduino Uno
- Arduino Mega

2.2 Χαρακτηριστικά Arduino Uno

Μικροελεγκτής	ATMEGA328
Τάση λειτουργίας	5V
Τάση εισόδου	7-12V
Όρια τάσης εισόδου	6-20V
Ψηφιακοί ακροδέκτες I/O	14(6 εκ των οποίων PWM έξοδο)
Αναλογικοί ακροδέκτες εισόδου	6
Ισχύς συνεχόμενου ρεύματος ανά ακροδέκτη	40mA
Ισχύς συνεχόμενου ρεύματος για ακροδέκτη τάσης 3.3V	50mA
Μνήμη flash	32KB(ATMEGA328)
Μνήμη SRAM	2KB(ATMEGA328)
Μνήμη EEPROM	1KB(ATMEGA328)
Ταχύτητα ρολογιού	16MHz



Εικόνα 2.1 Ηλεκτρονική πλατφόρμα Arduino Uno

2.3 Μικροελεγκτής ATmega328

Το Arduino βασίζεται στον ATmega328, έναν 8-bit RISC μικροελεγκτή, τον οποίο χρονίζει στα 16MHz. Ο ATmega328 διαθέτει ενσωματωμένη μνήμη τριών τύπων:

- 2Kb μνήμης SRAM που είναι η ωφέλιμη μνήμη που μπορούν να χρησιμοποιήσουν τα προγράμματά για να αποθηκεύουν μεταβλητές, πίνακες κ.λπ. κατά το runtime. Όπως και σε έναν υπολογιστή, αυτή η μνήμη χάνει τα δεδομένα της όταν η παροχή ρεύματος στο Arduino σταματήσει ή αν γίνει reset.

- 1Kb μνήμης EEPROM η οποία μπορεί να χρησιμοποιηθεί για εγγραφή/ανάγνωση δεδομένων (χωρίς datatype) ανά byte από τα προγράμματά κατά το runtime. Σε αντίθεση με την SRAM, η EEPROM δεν χάνει τα περιεχόμενά της με απώλεια τροφοδοσίας ή reset οπότε είναι το ανάλογο του σκληρού δίσκου.

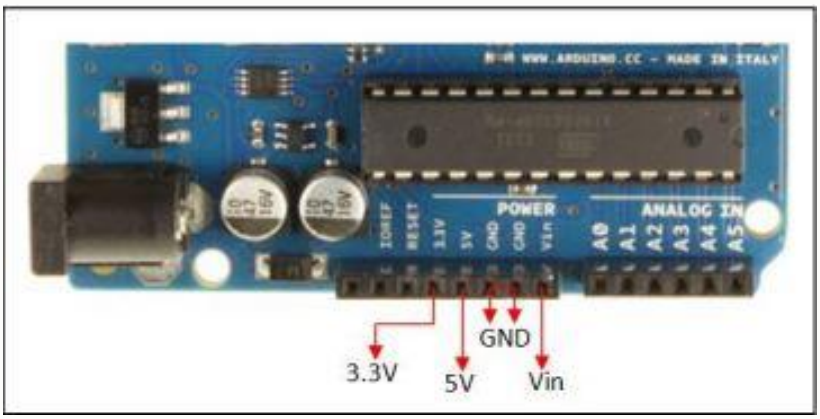
- 32Kb μνήμης Flash, από τα οποία τα 2Kb χρησιμοποιούνται από το firmware του Arduino που έχει εγκαταστήσει ήδη ο κατασκευαστής του. Το firmware αυτό που στην ορολογία του Arduino ονομάζεται bootloader είναι αναγκαίο για την εγκατάσταση προγραμμάτων στον μικροελεγκτή μέσω της θύρας USB, χωρίς δηλαδή να χρειάζεται εξωτερικός hardware programmer. Τα υπόλοιπα 30Kb της μνήμης Flash χρησιμοποιούνται για την αποθήκευση αυτών ακριβώς των προγραμμάτων, αφού πρώτα μεταγλωττιστούν στον υπολογιστή. Η μνήμη Flash, όπως και η EEPROM δεν χάνει τα περιεχόμενά της με απώλεια τροφοδοσίας ή reset. Επίσης, ενώ η μνήμη Flash υπό κανονικές συνθήκες δεν προορίζεται για χρήση runtime μέσα από τα προγράμματά

σας, λόγω της μικρής συνολικής μνήμης που είναι διαθέσιμη σε αυτά (2Kb SRAM + 1Kb EEPROM), έχει σχεδιαστεί μια βιβλιοθήκη που επιτρέπει την χρήση όσου χώρου περισσεύει (30Kb μείον το μέγεθος του προγράμματός σας σε μεταγλωττισμένη μορφή).

2.4 Τροφοδοσία

Το Arduino τροφοδοτείται είτε από εξωτερική τροφοδοσία είτε απευθείας από την θύρα USB. Ως εξωτερική τροφοδοσία ορίζεται είτε μια μπαταρία, είτε μετασχηματιστής των 9Volt από 220V. Η μπαταρία μπορεί να συνδεθεί στις υποδοχές του Arduino Vin και GND όπου τοποθετούνται ο θετικός πόλος και ο αρνητικός αντίστοιχα. Από την άλλη αν τροφοδοτήσουμε με μετασχηματιστή απλά τοποθετούμε το βύσμα στην υποδοχή που υπάρχει με τον θετικό πόλο στο κέντρο.

Η πλακέτα μπορεί να λειτουργήσει με εξωτερική πηγή από 6 έως 20 Volts. Αν τροφοδοτηθεί με λιγότερα από 7 Volt τα pin εξόδου 5Volt δεν θα καταφέρουν να εξάγουν τάση 5 Volts. Αν δώσουμε πάνω από 12 Volts θα υπερθερμανθεί ο σταθεροποιητής τάσης στην πλακέτα και ενδεχομένως να καταστραφεί. Συνεπώς μια ιδανική τάση είναι τα 9 Volts.



Εικόνα 2.2 Ακροδέκτες τροφοδοσίας του Arduino

Οι ακροδέκτες τροφοδοσίας είναι οι εξής:

- **VIN** Ακροδέκτης για μη σταθεροποιημένη τάση. Συνήθως εδώ συνδέεται μια εξωτερική πηγή τροφοδοσίας.
- **5V** Ακροδέκτης σταθεροποιημένης τάσης 5Volt. Χρησιμοποιείται για την τροφοδοσία του μικροελεγκτή ή άλλων ηλεκτρονικών στοιχείων.
- **3.3V** Το ολοκληρωμένο FTDI που βρίσκεται στην πλακέτα του Arduino παράγει τάση των 3.3V με μέγιστο ρεύμα 50mA.
- **GND** Ακροδέκτες Γείωσης

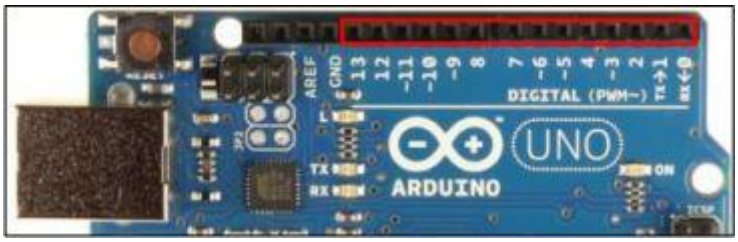
2.5 Είσοδοι - Έξοδοι

Καταρχήν το Arduino διαθέτει σειριακό interface. Ο μικροελεγκτής ATmega υποστηρίζει σειριακή επικοινωνία, την οποία το Arduino προωθεί μέσα από έναν ελεγκτή Serial-over-USB ώστε να συνδέεται με τον υπολογιστή μέσω USB. Η σύνδεση αυτή χρησιμοποιείται για την μεταφορά των προγραμμάτων που σχεδιάζονται από τον υπολογιστή στο Arduino αλλά και για αμφίδρομη επικοινωνία του Arduino με τον υπολογιστή μέσα από το πρόγραμμα την ώρα που εκτελείται.
Επιπλέον, στην πάνω πλευρά του Arduino βρίσκονται 14 θηλυκά pin, αριθμημένα από 0 ως 13, που μπορούν να λειτουργήσουν ως ψηφιακές είσοδοι και έξοδοι. Λειτουργούν στα 5V και καθένα μπορεί να παρέχει ή να δεχτεί το πολύ 40mA.
Ως ψηφιακή έξοδος, ένα από αυτά τα pin μπορεί να τεθεί από το πρόγραμμά σε κατάσταση HIGH ή LOW, οπότε το Arduino θα ξέρει αν πρέπει να διοχετεύσει ή όχι ρεύμα στο συγκεκριμένο pin. Αν πάλι ρυθμίσουμε ένα από αυτά τα pin ως ψηφιακή είσοδο μέσα από το πρόγραμμά, μπορούμε με την κατάλληλη εντολή να διαβάσουμε την κατάστασή του (HIGH ή LOW) ανάλογα με το αν η εξωτερική συσκευή που έχουμε συνδέσει σε αυτό το pin διοχετεύει ή όχι ρεύμα στο pin (με αυτόν τον τρόπο λόγου χάρη μπορούμε να «διαβάζουμε» την κατάσταση ενός διακόπτη).
Μερικά από αυτά τα 14 pin, εκτός από ψηφιακές είσοδοι/έξοδοι έχουν και δεύτερη λειτουργία. Συγκεκριμένα:

- Τα pin 0 και 1 λειτουργούν ως RX και TX της σειριακής όταν το πρόγραμμά ενεργοποιεί την σειριακή θύρα. Έτσι, όταν λόγου χάρη το πρόγραμμά στέλνει δεδομένα στην σειριακή, αυτά προωθούνται και στην θύρα USB μέσω του ελεγκτή Serial-Over-USB αλλά και στο pin 0 για να τα διαβάσει ενδεχομένως μια άλλη συσκευή (π.χ. ένα δεύτερο Arduino στο δικό του pin 1). Αυτό φυσικά σημαίνει ότι αν

στο πρόγραμμά ενεργοποιήσουμε το σειριακό interface, χάνουμε 2 ψηφιακές εισόδους/εξόδους.

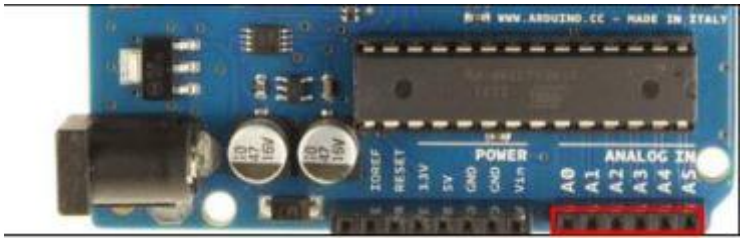
- Τα pin 2 και 3 λειτουργούν και ως εξωτερικά interrupt (interrupt 0 και 1 αντίστοιχα). Με άλλα λόγια, μπορούμε να τα ρυθμίσουμε μέσα από το πρόγραμμά ώστε να λειτουργούν αποκλειστικά ως ψηφιακές είσοδοι στις οποίες όταν συμβαίνουν συγκεκριμένες αλλαγές, η κανονική ροή του προγράμματος σταματάει άμεσα και εκτελείται μια συγκεκριμένη συνάρτηση. Τα εξωτερικά interrupt είναι ιδιαίτερα χρήσιμα σε εφαρμογές που απαιτούν συγχρονισμό μεγάλης ακρίβειας.
- Τα pin 3, 5, 6, 9, 10 και 11 μπορούν να λειτουργήσουν και ως ψευδοαναλογικές έξοδοι με το σύστημα PWM (Pulse Width Modulation), δηλαδή το ίδιο σύστημα που διαθέτουν οι μητρικές των υπολογιστών για να ελέγχουν τις ταχύτητες των ανεμιστήρων. Έτσι, μπορούμε να συνδέσουμε για παράδειγμα ένα LED σε κάποιο από αυτά τα pin και να ελέγξουμε πλήρως την φωτεινότητά του με ανάλυση 8bit (256 καταστάσεις από 0-σβηστό ως 255-πλήρως αναμμένο). Είναι σημαντικό να καταλάβουμε ότι το PWM δεν είναι πραγματικά αναλογικό σύστημα και ότι θέτοντας στην έξοδο την τιμή 127, δεν σημαίνει ότι η έξοδος θα δίνει 2.5V αντί της κανονικής τιμής των 5V, αλλά ότι θα δίνει ένα παλμό που θα εναλλάσσεται με μεγάλη συχνότητα και για ίσους χρόνους μεταξύ των τιμών 0 και 5V.



Εικόνα 2.3 Ψηφιακοί ακροδέκτες του Arduino

Στην κάτω πλευρά του Arduino, με τη σήμανση ANALOG IN, θα βρούμε μια ακόμη σειρά από 6 pin, αριθμημένα από το 0 ως το 5. Το καθένα από αυτά λειτουργεί ως αναλογική είσοδος κάνοντας χρήση του ADC (Analog to Digital Converter) που είναι ενσωματωμένο στον μικροελεγκτή. Για παράδειγμα, μπορούμε να τροφοδοτήσουμε ένα από αυτά με μια τάση την οποία μπορείτε να κυμάνετε με ένα ποτενσιόμετρο από 0V ως μια τάση αναφοράς Vref η οποία, αν δεν κάνουμε κάποια αλλαγή είναι προρυθμισμένη στα 5V. Τότε, μέσα από το πρόγραμμά μπορούμε να «διαβάσουμε» την τιμή του pin ως ένα ακέραιο αριθμό ανάλυσης 10-bit, από 0 (όταν η τάση στο pin

είναι 0V) μέχρι 1023 (όταν η τάση στο pin είναι 5V). Η τάση αναφοράς μπορεί να ρυθμιστεί με μια εντολή στο 1.1V, ή σε όποια τάση μεταξύ 2 και 5V τροφοδοτώντας εξωτερικά με αυτή την τάση το pin με την σήμανση AREF που βρίσκεται στην απέναντι πλευρά της πλακέτας. Έτσι, αν τροφοδοτήσουμε το pin AREF με 3.3V και στην συνέχεια δοκιμάσουμε να διαβάσουμε κάποιο pin αναλογικής εισόδου στο οποίο εφαρμόζετε τάση 1.65V, το Arduino θα σας επιστρέψει την τιμή 512.

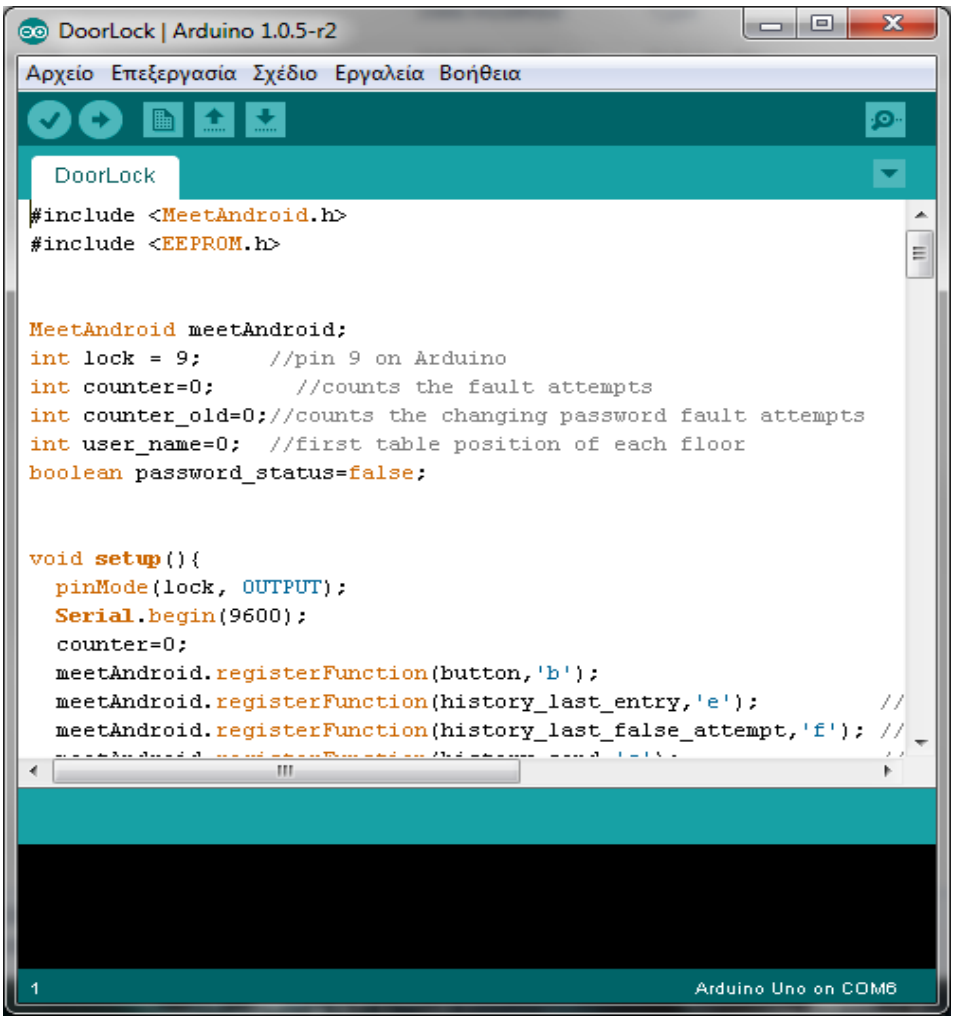


Εικόνα 2.4 Αναλογικοί ακροδέκτες του Arduino

2.6 Ολοκληρωμένο Περιβάλλον Ανάπτυξης του Arduino

Το περιβάλλον ανάπτυξης Arduino περιέχει μια περιοχή επεξεργασίας κειμένου για τη συγγραφή κώδικα, μια περιοχή μηνυμάτων, ένα μενού, μια γραμμή εργαλείων με κουμπιά για κοινές λειτουργίες, καθώς και μια σειρά από μενού. Συνδέεται με το υλικό Arduino για τη φόρτωση προγραμμάτων και για να επικοινωνούν μεταξύ τους.

Ένα ολοκληρωμένο πρόγραμμα συνήθως ονομάζεται sketch. Αυτό το sketch είναι γραμμένο με το πρόγραμμα επεξεργασίας κειμένου. Έχει δυνατότητες για την αντιγραφή/επικόλληση και για την αναζήτηση/αντικατάσταση κειμένου. Η κονσόλα απεικονίζει την έξοδο του κειμένου από το περιβάλλον Arduino συμπεριλαμβάνοντας πλήρη μηνύματα λάθους και άλλες πληροφορίες. Τα κουμπιά της γραμμής εργαλείων επιτρέπουν τον έλεγχο και το ανέβασμα των προγραμμάτων, τη δημιουργία νέου sketch, το άνοιγμα και την αποθήκευση των sketch και άνοιγμα της σειριακής οθόνης.



Εικόνα 2.5 Διεπαφή χρήστη του Arduino IDE

Τα κουμπιά της γραμμής εργαλείων από αριστερά προς τα δεξιά είναι τα εξής:

- Verify/Compile: Πραγματοποιείται ο έλεγχος για λάθη στον κώδικα
- Upload: Περνάει τον κώδικα στον μικροελεγκτή
- New: Δημιουργεί ένα νέο sketch
- Open: Παρουσιάζει ένα μενού με όλα τα sketch κάνοντας κλικ σε ένα από αυτά που θα ανοίξει μέσα στο τρέχον παράθυρο
- Save: Κάνει αποθήκευση το sketch
- Serial Monitor:ανοίγει η σειριακή οθόνη και μας δίνει τη δυνατότητα να δώσουμε τα δεδομένα από το πληκτρολόγιο.

2.7 Η δομή του προγράμματος

Η δομή ενός απλού προγράμματος Arduino είναι η εξής:

```
// δήλωση μεταβλητών
Void setup ()
{
//αρχικοποιήσεις
}
Void loop ()
{
// κώδικας
}
```

Οι δύο ειδικές συναρτήσεις που υπάρχουν σε κάθε sketch του Arduino είναι η setup () και η loop (). Η setup () καλείται μια φορά μόνο και μέσα σε αυτή γίνονται η προετοιμασία των βιβλιοθηκών, η αρχικοποίηση των μεταβλητών και η ρύθμιση κατάστασης των ακίδων. Η loop () αντίθετα καλείται συνεχώς. Απαραίτητο είναι και οι δύο συναρτήσεις να περιλαμβάνονται σε κάθε sketch, έστω και αν το περιεχόμενο τους είναι κενό.

2.8 Βασικές δομές και λειτουργίες

Δομές ελέγχου ροής

- if (δομή ελέγχου μίας συνθήκης)
- if ... else (δομή ελέγχου πολλαπλών συνθηκών)
- for (δομή επαναληπτικού ελέγχου συνθήκης)
- while (δομή επαναληπτικού ελέγχου συνθήκης)
- do ... while (δομή επαναληπτικού ελέγχου συνθήκης)
- switch ... case (δομή ελέγχου περιπτώσεων)
- break (εντολή διακοπής μιας επαναληπτικής δομής)
- continue (εντολή παράλειψης της τρέχουσας επανάληψης)

- return (εντολή επιστροφής από μία συνάρτηση)
- goto (εντολή μετάβασης σε κάποιο σημείο του κώδικα)

Αριθμητικοί τελεστές

- = (τελεστής εκχώρησης)
- + (τελεστής πρόσθεσης)
- - (τελεστής αφαίρεσης)
- * (τελεστής πολλαπλασιασμού)
- / (τελεστής διαίρεσης)
- % (τελεστής υπόλοιπου ακεραίας διαίρεσης)

Λογικοί τελεστές

- && (λογική σύζευξη)
- || (λογική διάζευξη)
- ! (λογική άρνηση)

Δυαδικοί τελεστές

- & (δυαδική σύζευξη)
- | (δυαδική διάζευξη)
- ^ (δυαδική αποκλειστική διάζευξη)
- ~ (δυαδική άρνηση)
- << (δυαδική αριστερή ολίσθηση)
- >> (δυαδική δεξιά ολίσθηση)

Τελεστές αύξησης και μείωσης

- ++ (αύξηση κατά μία ακέραιη μονάδα)
- -- (μείωση κατά μία ακέραιη μονάδα)

Σύνθετοι τελεστές

- +=, -=, *=, /=, %= (σύνθετοι αριθμητικοί τελεστές)
- &=, |=, ^=, ~=, <<=, >>= (σύνθετοι δυαδικοί τελεστές)

Τελεστές σύγκρισης

- == (ισότητα)
- != (ανισότητα)
- < (μικρότερο)
- > (μεγαλύτερο)
- <= (μικρότερο ή ίσο)
- >= (μεγαλύτερο ή ίσο)

Τελεστές δεικτών

- (τελεστής απόκτησης περιεχομένου)
- & (τελεστής απόκτησης διεύθυνσης)

Σταθερές

- HIGH (τιμή υψηλής στάθμης για μία επαφή εισόδου ή εξόδου)
- LOW (τιμή χαμηλής στάθμης για μία επαφή εισόδου ή εξόδου)
- false (λογικό επίπεδο ψεύδους σε μία συνθήκη)
- true (λογικό επίπεδο αλήθειας σε μία συνθήκη)
- INPUT (χρησιμοποιείται για τον ορισμό μίας επαφής ως είσοδο)
- OUTPUT (χρησιμοποιείται για τον ορισμό μίας επαφής ως έξοδο)
- A0, ..., A5 (συμβολοσταθερές για τις αναλογικές επαφές εισόδου)

Τύποι δεδομένων

- boolean (λογική δυαδική τιμή)
- char (προσημασμένος χαρακτήρας 8 ψηφίων)
- unsigned char (μη προσημασμένος χαρακτήρας 8 ψηφίων)
- byte (μη προσημασμένος χαρακτήρας 8 ψηφίων)
- int (προσημασμένος ακέραιος αριθμός 16 ψηφίων)
- unsigned int (μη προσημασμένος ακέραιος αριθμός 16 ψηφίων)
- word (μη προσημασμένος ακέραιος αριθμός 16 ψηφίων)

- long (προσημασμένος ακέραιος αριθμός 32 ψηφίων)
- unsigned long (μη προσημασμένος ακέραιος αριθμός 32 ψηφίων)
- float, double (αριθμός κινητής υποδιαστολής απλής ακρίβειας)
- String (αντικείμενο αλφαριθμητικού με χρήσιμες μεθόδους)
- Ως αλφαριθμητικό μπορεί να θεωρηθεί και ο πίνακας χαρακτήρων

Συναρτήσεις μετατροπής τύπων

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

Συναρτήσεις εισόδου και εξόδου

- pinMode() (ορίζει μια επαφή ως είσοδο ή έξοδο)

Συναρτήσεις ψηφιακής εισόδου και εξόδου

- digitalWrite() (γράφει σε μία ψηφιακή επαφή εξόδου)
- digitalRead() (διαβάζει από μία ψηφιακή επαφή εισόδου)

Συναρτήσεις αναλογικής εισόδου και εξόδου

- analogReference() (ορίζει την τάση αναλογικής αναφοράς)
- analogWrite() (γράφει PWM σήματα σε μία επαφή εξόδου)
- analogRead() (διαβάζει από μία αναλογική επαφή εισόδου)

Προηγμένες συναρτήσεις εισόδου και εξόδου

- tone() (παράγει ένα τετραγωνικό σήμα ορισμένης συχνότητας)
- noTone() (διακόπτει την παραγωγή τετραγωνικών σημάτων)
- shiftOut() (ολισθαίνει τα ψηφία μιας τιμής σε μία επαφή εξόδου)
- pulseIn() (επιστρέφει την διάρκεια σε μs ενός παλμού HIGH ή LOW)

Συναρτήσεις χρόνου

- millis() (διάρκεια εκτέλεσης του προγράμματος σε ms)

- micros() (διάρκεια εκτέλεσης του προγράμματος σε μs)
- delay() (παύση προγράμματος - η διάρκεια δίδεται σε ms)
- delayMicroseconds() (παύση προγράμματος - η διάρκεια δίδεται σε μs)

Μαθηματικές και Τριγωνομετρικές συναρτήσεις

- max() (βρίσκει τον μεγαλύτερο ανάμεσα σε δύο αριθμούς)
- min() (βρίσκει τον μικρότερο ανάμεσα σε δύο αριθμούς)
- abs() (επιστρέφει την απόλυτη τιμή ενός αριθμού)
- constrain() (ελέγχει για υπερχείλιση ή υποχείλιση ορίων)
- map() (πραγματοποιεί γραμμικό μετασχηματισμό ορίων)
- pow() (επιστρέφει το αποτέλεσμα μίας δύναμης)
- sqrt() (επιστρέφει την ρίζα ενός αριθμού)
- sin() (υπολογίζει το ημίτονο ενός αριθμού)
- cos() (υπολογίζει το συνημίτονο ενός αριθμού)
- tan() (υπολογίζει την εφαπτομένη ενός αριθμού)

Συναρτήσεις γεννήτριας ψευδοτυχαίων αριθμών

- random() (δίδεται ένας νέος αριθμός από την γεννήτρια)
- randomSeed() (θέτει τον σπόρο της γεννήτριας παραγωγής)

Συναρτήσεις επεξεργασίας δυαδικών αριθμών

- lowByte() (επιστρέφει το δεξιότερο byte μίας μεταβλητής)
- highByte() (επιστρέφει το αριστερότερο byte μίας μεταβλητής)
- bitRead() (διαβάζει ένα συγκεκριμένο ψηφίο μίας μεταβλητής)
- bitWrite() (γράφει σε ένα συγκεκριμένο ψηφίο μιας μεταβλητής)
- bitSet() (γράφει την τιμή 1 σε κάποιο ψηφίο μίας μεταβλητής)
- bitClear() (γράφει την τιμή 0 σε κάποιο ψηφίο μιας μεταβλητής)
- bit() (υπολογίζει μία συγκεκριμένη δύναμη με βάση το 2)

Συναρτήσεις χρήσης ρουτινών εξυπηρέτησης διακοπών

- attachInterrupt() (ενεργοποιεί μία ρουτίνα εξυπηρέτησης διακοπής)
- detachInterrupt() (απενεργοποιεί μία ρουτίνα εξυπηρέτησης διακοπής)

Συναρτήσεις ενεργοποίησης και απενεργοποίησης διακοπών

- interrupts() (ενεργοποιεί τα σήματα διακοπής)
- noInterrupts() (απενεργοποιεί τα σήματα διακοπής)

Υποστήριξη σειριακής επικοινωνίας

- Serial (αντικείμενο σειριακής επικοινωνίας με χρήσιμες μεθόδους)

ΚΕΦΑΛΑΙΟ 3

3 ΕΙΣΑΓΩΓΗ ΣΤΟ ANDROID

Το android είναι ένα λειτουργικό σύστημα το οποίο στηρίζεται στον πυρήνα του λειτουργικού συστήματος Linux και αφορά κυρίως συσκευές κινητής τηλεφωνίας αλλά και tablets. Στην αρχή αναπτύχθηκε από την Google και αργότερα από την Open Handset Alliance, η οποία αποτελεί μια κοινοπραξία εταιριών λογισμικού και κατασκευής hardware, οι οποίες ασχολούνται με την ανάπτυξη αλλά και την εξέλιξη ανοιχτών προτύπων στις συσκευές κινητής τηλεφωνίας. Το android δίνει τη δυνατότητα στο κάθε προγραμματιστή να συνθέτει κώδικα χρησιμοποιώντας τη γλώσσα προγραμματισμού Java, και να ελέγχει τη συσκευή μέσω βιβλιοθηκών λογισμικού ανεπτυγμένων από τη Google. Στις 5 Νοεμβρίου 2007 πραγματοποιήθηκε η πρώτη παρουσίαση της πλατφόρμας android. Η Google δημοσίευσε το μεγαλύτερο μέρος του κώδικα του android υπό τους όρους της Apache License, μιας ελεύθερης άδειας λογισμικού.

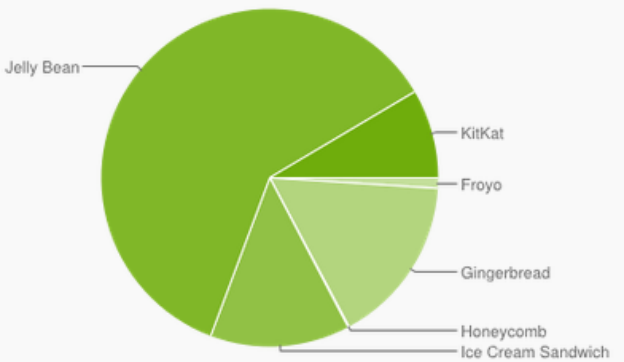
Οι προγραμματιστές android έχουν τη δυνατότητα να δημιουργήσουν διάφορες εφαρμογές με σχεδόν απεριόριστη λειτουργικότητα. Για παράδειγμα, μια εφαρμογή μπορεί να μεταδώσει δεδομένα, όπως για παράδειγμα λίστα με επαφές και φωτογραφίες, από το κινητό μέσω του διαδικτύου και να λάβει όλα όσα μπορεί να χρειαστεί online και να εμφανίζονται στην οθόνη της συσκευής του χρήστη.

Η πλατφόρμα android παρέχει στον προγραμματιστή τη δυνατότητα να χρησιμοποιήσει μια μεγάλης ποικιλίας από βιβλιοθήκες και όλα εκείνα τα χρήσιμα εργαλεία που μπορούν να συνδράμουν για τη δημιουργία ενός πάρα πολύ εξελιγμένου λογισμικού. Αυτή η μεγάλη ποικιλία από έτοιμα εργαλεία βοηθάει πολύ στην αύξηση της παραγωγικότητας των προγραμματιστών android εφαρμογών και τους βοηθά να δημιουργήσουν πλούσιο λογισμικό πολύ γρηγορότερα και με περιορισμένα λάθη.

3.1 Η εξέλιξη του Android

Επειδή το Android είναι ένα λειτουργικό σύστημα ανοιχτού κώδικα η εξέλιξη του λόγω της φύσης του είναι ραγδαία και αυτό φαίνεται κυρίως από το γεγονός ότι σε διάστημα 4,5 ετών έχουν κυκλοφορήσει οι 10 κύριες εκδόσεις του.

Version	Codename	API	Distribution
2.2	Froyo	8	1.0%
2.3.3 - 2.3.7	Gingerbread	10	16.2%
3.2	Honeycomb	13	0.1%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	13.4%
4.1.x	Jelly Bean	16	33.5%
4.2.x		17	18.8%
4.3		18	8.5%
4.4	KitKat	19	8.5%



Εικόνα 3.1 Ποσοστά των εγκατεστημένων εκδόσεων στις android συσκευές εως 1 Μαΐου 2014

Στην πληροφορική συνηθίζεται τα προϊόντα software και hardware να κυκλοφορούν εκτός από τον αριθμό έκδοσής τους και με μια κωδική ονομασία. Η κωδική ονομασία μπορεί να είναι για παράδειγμα ονόματα πόλεων, ονόματα ζώων, στην περίπτωση όμως του Android τα κωδικά ονόματα έρχονται στη μορφή επιδορπίου.

Συνοπτικά οι εκδόσεις Android :

- Android 1.5 Cupcake
- Android 1.6 Donut
- Android 2.0/2.1 Eclair
- Android 2.2 Froyo
- Android 2.3 Gingerbread
- Android 3.0 Honeycomb
- Android 4.0 Ice Cream Sandwich
- Android 4.1/4.2/4.3 Jelly Bean
- Android 4.4 Kitkat

3.1.1 Android 1.5 Cupcake

Εικόνα 3.2 Λογότυπο του Android 1.5 Cupcake

Η έκδοση “Cupcake”, βασισμένη στο Linux Kernel 2.6.27, παρουσιάστηκε στις 30 Απριλίου του 2009. Υποστηρίζει νέες λειτουργίες για την κάμερα τις συσκευής, όπως η καταγραφή και παρακολούθηση βίντεο από την λειτουργία της κάμερας και η άμεση μεταφόρτωση του βίντεο αλλά και των φωτογραφιών στο YouTube και το Picasa αντίστοιχα απευθείας από το τηλέφωνο. Έχει νέο έξυπνο πληκτρολόγιο με πρόβλεψη κειμένου. Υποστηρίζει πρότυπο Bluetooth A2DP και AVRCP ενώ έχει και την ικανότητα να συνδέεται αυτόματα σε μικροσυσκευές Bluetooth από μια συγκεκριμένη απόσταση. Ακόμα στην έκδοση αυτή έχει νέο γραφικό περιβάλλον με κινούμενες μεταβάσεις οθόνης.

3.1.2 Android 1.6 Donut

Εικόνα 3.3 Λογότυπο του Android 1.6 Donut

Η έκδοση “Donut”, βασισμένη στο Linux Kernel 2.6.29, παρουσιάστηκε στις 15 Σεπτεμβρίου του 2009. Έχει ταχύτερη απόκριση σε σχέση με την προηγούμενη έκδοση. Υποστηρίζεται πλέον η επιλογή πολλαπλών αρχείων ταυτόχρονα, έχει ανανεωμένο γκάλερι και φωτογραφική μηχανή, καθώς και βελτιωμένο Android Market. Έχει ανανεωμένη φωνητική αναζήτηση, με ταχύτερη απόκριση και βαθύτερη ολοκλήρωση με εγγενείς (native) εφαρμογές, συμπεριλαμβανομένης της δυνατότητας κλήσης επαφών. Δυνατότητα αναζήτησης σελιδοδεικτών, ιστορικού, επαφών αλλά και στο διαδίκτυο από την αρχική οθόνη. Υποστήριξη για ανάλυση οθονών WVGA. Ανανεωμένη υποστήριξη τεχνολογιών για CDMA/EVDO, 802.1x, VPNs και με μηχανή μετατροπής κειμένου σε ομιλία.

3.1.3 Android 2.0/2.1 Eclair

Εικόνα 3.4 Λογότυπο του Android 2.0/2.1 Éclair

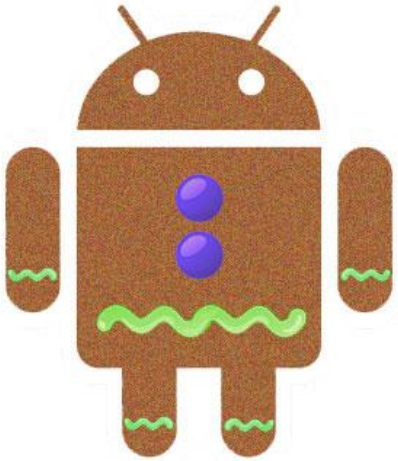
Η έκδοση “Eclair”, βασισμένη και αυτή στον Linux Kernel 2.6.29, παρουσιάστηκε στις 26 Οκτωβρίου του 2009, ενώ τον Ιανουάριο του 2010 επανεκδόθηκε σε Android 2.1 Eclair (MR1).Σε αυτή την έκδοση υπάρχει ακόμα ταχύτερη απόκριση του υλικού σε σχέση με τις δυο προηγούμενες και πλέον υποστηρίζονται περισσότερες οθόνες και αναλύσεις. Υπάρχει νέος browser ο οποίος υποστηρίζει το πρότυπο HTML5, νέο User Interface, και βελτιωμένοι χάρτες Google. Έχει ενσωματωθεί η υποστήριξη φλας για την κάμερα η οποία έχει πλέον και ψηφιακό zoom. Επίσης έχει βελτιωθεί η κλάση MotionEvent ώστε να υπάρχει η δυνατότητα για γεγονότα πολλαπλής αφής (multitouch events). Υποστηρίζεται Bluetooth και έχει βελτιωθεί και το πληκτρολόγιο.

3.1.4 Android 2.2 Froyo

Εικόνα 3.5 Λογότυπο του Android 2.2 Froyo

Η έκδοση “Froyo”, βασισμένη στο Linux Kernel 2.6.32, παρουσιάστηκε στις 20 Μαΐου του 2010. Υπάρχουν βελτιστοποιήσεις στην ταχύτητα γενικά του λειτουργικού συστήματος, στην μνήμη και στην απόδοση. Έχει ενσωματωθεί ο μηχανισμός JavaScript του Chrome V8 στον browser, υπάρχει πλέον Adobe Flash 10.1, ενώ υποστηρίζεται καλύτερα πλέον το Microsoft Exchange. Έχει γίνει ανανέωση του Android Market. Ο χρήστης μπορεί πλέον να ελέγχει αν θα γίνεται ή όχι κίνηση πακέτων δεδομένων από το δίκτυο κινητής τηλεφωνίας. Υπάρχει η δυνατότητα εγκατάστασης εφαρμογών στην κάρτα μνήμης και η μεταφορά τους εκεί από τη μνήμη του τηλεφώνου. Επίσης το τηλέφωνο πλέον μπορεί να μετατραπεί σε Wi-Fi hotspot.

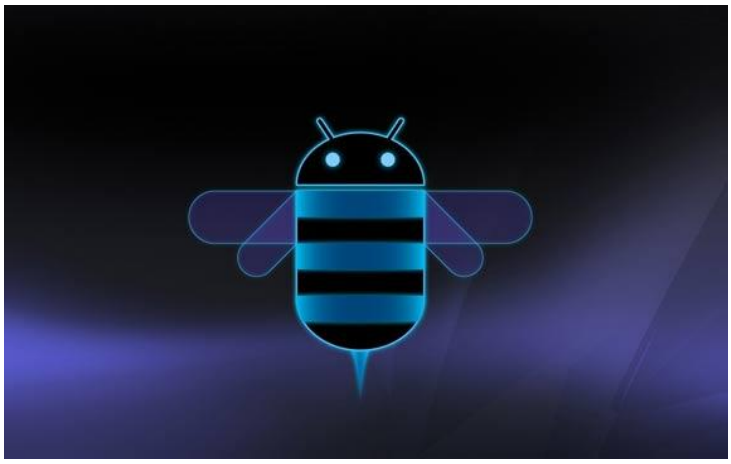
3.1.5 Android 2.3 Gingerbread



Εικόνα 3.6 Λογότυπο του Android 2.3 Gingerbread

Η έκδοση “Gingerbread”, βασισμένη στο Linux Kernel 2.6.35.7, παρουσιάστηκε στις 6 Δεκεμβρίου του 2010, ενώ τον Φεβρουάριο του 2011 επανεκδόθηκε σε Android 2.3.3. Στην έκδοση αυτή υπάρχουν αλλαγές στο User Interface το οποίο έχει γίνει πιο απλό και ταχύ, ενώ υποστηρίζονται πλέον οθόνες μεγάλων μεγεθών και αναλύσεων. Υπάρχει πλέον το πρωτόκολλο SIP για κλήσεις μέσω VoIP, υποστηρίζεται ο τύπος βίντεο WebM/VP8 και ο κωδικοποιητής AAC, έχει βελτιωθεί ο ήχος καθώς και οι λειτουργίες απεικόνισης για την ανάπτυξη παιχνιδιών. Υπάρχει η δυνατότητα για Copy-Paste σε όλο το σύστημα και όχι μόνο στην ίδια εφαρμογή. Υποστηρίζεται το NFC (Near Field Communication) και η ύπαρξη πολλαπλών καμερών. Επίσης, έχει βελτιωθεί η ενεργειακή υποστήριξη και έχει γίνει μετάβαση από το σύστημα αρχείων YAFFS στο ext4 στις νέες συσκευές.

3.1.6 Android 3.0 Honeycomb



Εικόνα 3.7 Λογότυπο του Android 3.0 Honeycomb

Η έκδοση “Honeycomb”, βασισμένη στο Linux Kernel 2.6.36, παρουσιάστηκε στις 9 Μαΐου του 2011, με την ιδιαιτερότητα ότι προοριζόταν αποκλειστικά για tablets. Οι αλλαγές που έγιναν στην έκδοση αυτή έχουν να κάνουν κυρίως με τη βελτίωση της υποστήριξης των tablets. Υπάρχει ένα νέο, εντελώς διαφορετικό, User Interface και υποστηρίζονται διπύρηνοι και τετραπύρηνοι επεξεργαστές. Ακόμα, έχει απλοποιηθεί το multitasking έτσι ώστε ο χρήστης να μπορεί με τη χρήση ενός πλήκτρου (recent apps) να περνάει από μια εφαρμογή σε άλλη. Υπάρχει η δυνατότητα για Video Chat μέσω της εφαρμογής Google Talk καθώς η ανάγνωση βιβλίων μέσω του Google eBooks. Επιπλέον, μπορούν να κρυπτογραφηθούν όλα τα δεδομένα χρήστη.

3.1.7 Android 4.0 Ice Cream Sandwich

Εικόνα 3.8 Λογότυπο του Android 4.0 Ice Cream Sandwich

Η έκδοση “Ice Cream Sandwich”, βασισμένη στο Linux Kernel 3.0.1, παρουσιάστηκε στις 19 Οκτωβρίου του 2011. Για άλλη μια φορά έχει βελτιωθεί η ταχύτητα και η απόδοση του συστήματος. Πλέον στο User Interface, το οποίο είναι και πάλι διαφορετικό, υπάρχουν εικονικά πλήκτρα τα οποία παίρνουν τη θέση των φυσικών ή αφής που υπήρχαν στις συσκευές. Βελτίωση της ασφάλεια του συστήματος με την προσθήκη αναγνώρισης προσώπου για να ξεκλειδώσει η συσκευή. Ο browser μπορεί να ανοίξει ταυτόχρονα μέχρι και 16 καρτέλες. Υπάρχει η δυνατότητα ο χρήστης να τερματίσει εφαρμογές οι οποίες τρέχουν στο background, ενώ μπορεί να θέσει και όρια στην κίνηση πακέτων δεδομένων. Η εφαρμογή Android Beam αξιοποιεί πλέον το NFC αφού επιτρέπει την αποστολή δεδομένων από τη συσκευή σε όσες βρίσκονται εντός μιας μικρής ακτίνας εμβέλειας. Ακόμα με την ύπαρξη του Wi-Fi Direct συσκευές μπορούν να συνδεθούν μεταξύ τους ασύρματα χωρίς την μεσολάβηση κάποιου access point. Τέλος, υποστηρίζεται η εγγραφή βίντεο σε 1080p.

3.1.8 Android 4.1/4.2/4.3 Jelly Bean

Εικόνα 3.9 Λογότυπο του Android 4.1/4.2/4.3 Jelly Bean

Η Google ανακοίνωσε την έκδοση 4.1 στις 27 Ιουνίου 2012. Βασίστηκε στον Linux kernel 3.0.31 και ήταν μια σημαντική ενημέρωση που στόχευε στην βελτίωση της λειτουργικότητας και της εμφάνισης του user interface. Το tablet Nexus 7 ήταν η πρώτη συσκευή που έτρεξε αυτή την έκδοση και παρουσιάστηκε τον Ιούλιο του 2012. Οι νέες λειτουργίες παρουσιάζονται παρακάτω:

- Πλήρως ανανεωμένο το σύστημα ειδοποιήσεων
- Βελτιωμένες λειτουργίες κάμερας και προηγμένη αναζήτηση με ομιλία
- Βελτίωση των γραφικών μέσω της υποστήριξης OpenGL ES 3.0
- Βελτιωμένες δυνατότητες αισθητήρα
- Προηγμένη ασφάλεια και βελτιώσεις στην απόδοση
- Δυνατότητα χρήσης εξωτερικής συσκευής ήχου μέσω USB
- Υποστηρίζονται τα tablets μικρών οθονών με χρήση βελτιωμένης έκδοσης για κινητά τηλέφωνα

3.1.9 Android 4.4 Kit Kat

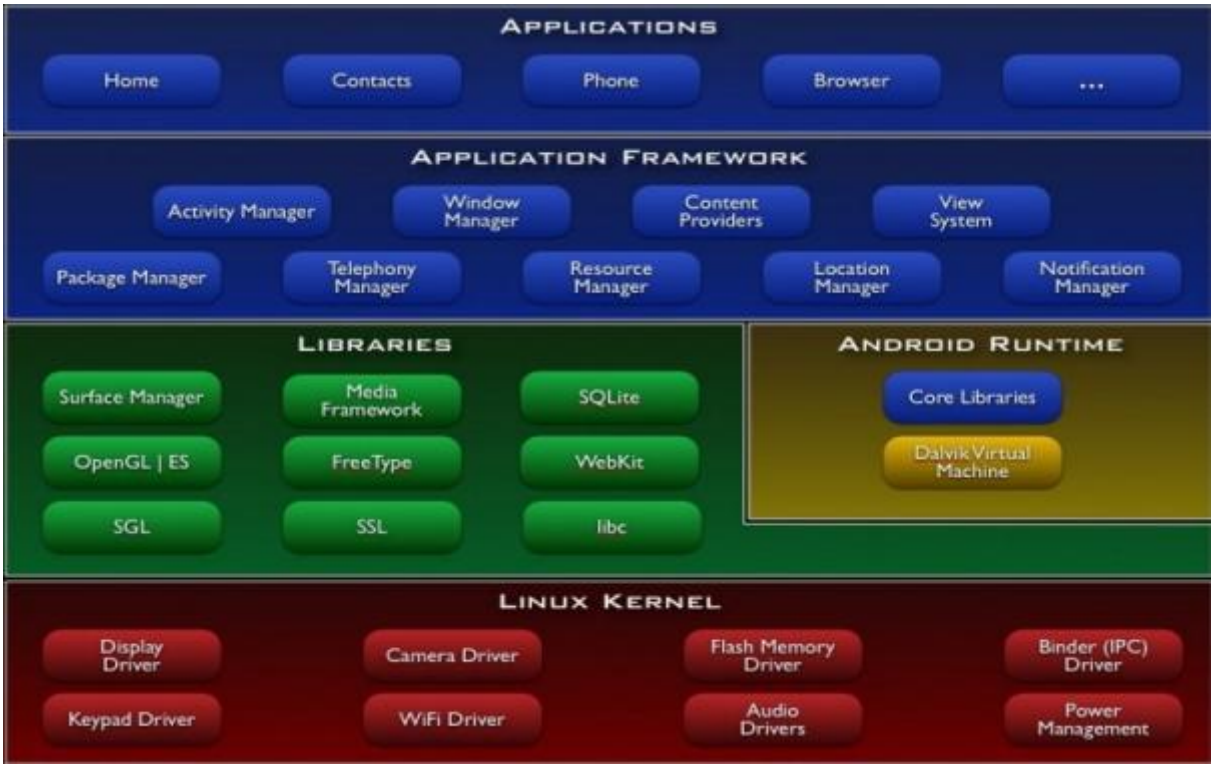
Εικόνα 3.10 Λογότυπο του Android 4.4 Kit Kat

Το Android 4.4 ανακοινώθηκε τον Σεπτέμβρη του 2013 στη συσκευή της Google Nexus 5. Αναμενόταν για καιρό ως Android 5.0 με κωδική ονομασία Key Lime Pie. Αναπτύχθηκε για να τρέχει σε καλύτερο εύρος συσκευών απ' ότι οι προηγούμενες εκδόσεις, έχοντας 512 MB ελάχιστη απαίτηση μνήμης RAM. Βασικά νέα χαρακτηριστικά της ήταν:

- Δυνατότητα εκτύπωσης μέσω ασύρματου δικτύου
- Ανανεωμένος σχεδιασμός με λευκά αντί μπλε στοιχεία
- Δυνατότητα επιλογής εισαγωγής μηνύματος μέσω της εφαρμογής Ρυθμίσεις
- Υποστήριξη Bluetooth Message Access Profile (MAP)
- Απενεργοποιήθηκε η πρόσβαση στα στατιστικά της μπαταρίας από εφαρμογές που δεν έχουν εγκατασταθεί από το Android Market
- Εικονικά κουμπιά του μενού είναι πάντα ορατά ακόμα και σε συσκευές που έχουν φυσικά κουμπιά
- Έγιναν βελτιώσεις στην ασφάλεια και λύθηκαν σφάλματα προηγούμενων εκδόσεων

3.2 Αρχιτεκτονική του Android

Το Android εκτός από ένα λειτουργικό σύστημα είναι και μια στοίβα λογισμικού η οποία αποτελείτε από υπηρεσίες διασύνδεσης με τις εφαρμογές (middleware) και τέλος από τις κύριες (core) εφαρμογές, μεταξύ αυτών, ενός email cient, μιας εφαρμογής διαχείρισης SMS, ενός ημερολογίου, ενός browser, εφαρμογή διαχείρισης επαφών, και άλλες οι οποίες έρχονται δεμένες με την υπόλοιπη στοιβάδα λογισμικού του Android. Στο παρακάτω σχεδιάγραμμα βλέπουμε οπτικά την αρχιτεκτονική αυτή.



Εικόνα 3.11: Η Αρχιτεκτονική του Android 1.4.1

Από ότι παρατηρούμε η αρχιτεκτονική του λειτουργικού συστήματος αποτελείτε από 5 κύρια επίπεδα.

- Τον πυρήνα Linux (Linux Kernel)
- Τις τωρινές και τις προηγμένες βιβλιοθήκες
- Την εικονική μηχανή Dalvik
- Τον χρόνο εκτέλεσης (Android Runtime)
- Το πλαίσιο εφαρμογής (Application Framework)

3.2.1 Πυρήνας Linux (Linux Kernel)

Η βάση της στοίβας λογισμικού του Android είναι ο πυρήνας Linux. Ο τροποποιημένος πυρήνας του συστήματος βασίζεται στην έκδοση 2.6 (και στην έκδοση 3.0.1 για το Android 4.0) του Linux Kernel, η οποία υποστηρίζει όλες τις κύριες λειτουργίες του λειτουργικού συστήματος. Οι λειτουργίες αυτές αφορούν διαχείριση μνήμης, διαχείριση διεργασιών, λειτουργίες δικτύου, ασφάλεια του λειτουργικού, και ένα σύνολο οδηγών υλικού (hardware drivers). Οι οδηγοί αυτοί είναι υπεύθυνοι για την επικοινωνία του software με το hardware της συσκευής. Ο πυρήνας του Android περιέχει:

- Οδηγό προβολής οθόνης
- Οδηγό Wifi και Bluetooth
- Οδηγό κάμερας και άλλα

Ο πυρήνας του Android μπορεί να βασίζεται στον πυρήνα του Linux, αλλά διαφέρει αρκετά από αυτόν. Ο λόγος είναι οι αλλαγές στην αρχιτεκτονική που έχει κάνει η Google για να είναι ελαφρύτερος και βελτιστοποιημένος για χρήση σε κινητές συσκευές. Αυτό σημαίνει ότι παρότι το Android είναι κατά βάση Linux, επί της ουσίας είναι αρκετά δύσκολο να τρέξουν εφαρμογές ή να χρησιμοποιηθούν βιβλιοθήκες από τη μία πλατφόρμα στην άλλη.

3.2.2 Βιβλιοθήκες

Στο δεύτερο επίπεδο της στοίβας έχουμε τις βιβλιοθήκες του Android. Αυτές ουσιαστικά αποτελούν τα APIs που είναι διαθέσιμα στους προγραμματιστές για την ανάπτυξη των εφαρμογών. Οι βιβλιοθήκες ενσωματώνονται και χρησιμοποιούνται από τις εφαρμογές για τις διάφορες λειτουργίες που παρέχει η καθεμία από αυτές. Στην ουσία είναι ένα από τα δομικά υλικά των εφαρμογών, και έτσι αποτελούν αναπόσπαστο κομμάτι τους. Οι δυνατότητες των βιβλιοθηκών του Android γίνονται εμφανείς στους προγραμματιστές στην στοίβα του πλαισίου εφαρμογής. Το σύνολο σχεδόν των βιβλιοθηκών είναι γραμμένο σε C και C++, οι οποίες έχουν μεταγλωττιστεί για τη χρήση τους από το λειτουργικό. Μερικές από τις κύριες βιβλιοθήκες του Android είναι:

- System C library – μια ενσωμάτωση της standard βιβλιοθήκης συστήματος της C (libc) τροποποιημένη για κινητές συσκευές βασισμένες στο Linux.

• Βιβλιοθήκες Πολυμέσων – Υποστηρίζει αναπαραγωγή και εγγραφή πολλών δημοφιλών μέσων ήχου και εικόνας, όπως: MPEG4, H.264, MP3, AAC, AMR, JPG, και PNG

• Surface Manager – διαχειρίζεται την πρόσβαση στο υποσύστημα προβολής, και συνθέτει απρόσκοπτα δισδιάστατα και τρισδιάστατα επίπεδα γραφικών τα οποία προέρχονται από πολλαπλές εφαρμογές.

• LibWebCore – μια μοντέρνα μηχανή υποστήριξης πλοήγηση στο διαδίκτυο (browser engine) η οποία χρησιμοποιείτε και από τον ενσωματωμένο browser του Android αλλά και από τις WebViews που ενσωματώνονται στις εφαρμογές.

• SGL – η γνωστή μηχανή δισδιάστατων γραφικών

• Βιβλιοθήκες 3D – μια υλοποίηση βασισμένη στα APIs του OpenGL ES 1. Οι βιβλιοθήκες χρησιμοποιούν είτε τρισδιάστατη επιτάχυνση υλικού, όπου αυτή είναι διαθέσιμη, είτε μια υψηλά βελτιωμένη τρισδιάστατη επιτάχυνση λογισμικού σε περίπτωση που η πρώτη δεν είναι διαθέσιμη.

• FreeType – παρέχει ευκρίνεια γραφικών στα bitmaps και τις γραμματοσειρές των εφαρμογών του συστήματος.

• SQLite – μια πανίσχυρη και συνάμα πολύ ελαφριά σχεσιακή βάση δεδομένων

3.2.3 Η εικονική μηχανή Dalvik

Σχεδόν το σύνολο των APIs του Android βασίζονται στη γλώσσα προγραμματισμού Java. Στην Java περιέχεται η Java Virtual Machine στην οποία εκτελείτε ο κώδικας bytecode των εφαρμογών. Στο Android υπάρχει η ανάλογη εικονική μηχανή Dalvik.

Η Dalvik λοιπόν είναι η εικονική μηχανή μέσω της οποίας τρέχουν οι εφαρμογές του Android. Η κάθε εφαρμογή τρέχει μέσω τις δικής της εικονικής μηχανής στη δικιά της διεργασία και για αυτό το λόγο καμία εφαρμογή δεν έχει επαφή με την άλλη, ενώ εκτελούνται ταυτόχρονα. Η Dalvik δεν υποστηρίζει τον κώδικα bytecode, αντί αυτού οι κλάσεις της Java γίνονται compile σε αρχεία .dex ώστε να τρέξουν στην VM. Το Android είναι από τη φύση του multitasking λειτουργικό σύστημα και για αυτό επιτρέπει στις εφαρμογές του να τρέχουν σε πολλά νήματα ταυτόχρονα και να απασχολούν πολλές διαδικασίες εάν αυτό είναι αναγκαίο. Για να γίνει αυτό εφικτό η μηχανή Dalvik είναι σχεδιασμένη για να έχει ελάχιστο αντίκτυπο στη χρήση της μνήμης. Χάρη στον λιτό της σχεδιασμό, το σύστημα είναι σε θέση να τρέχει πολλές εικονικές μηχανές ταυτόχρονα.

3.2.4 Χρόνος Εκτέλεσης Εφαρμογής (Android Runtime)

Ο χρόνος εκτέλεσης των εφαρμογών του Android, βρίσκεται στο ίδιο επίπεδο με τις κύριες βιβλιοθήκες και την μηχανή Dalvik. Εδώ βρίσκουμε το κοινό σημείο επαφής μεταξύ των δυνατοτήτων που παρέχουν οι βιβλιοθήκες και του χρόνου εκτέλεσης της εικονικής μηχανής Dalvik τις λειτουργίες τις οποίας, περιγράψαμε παραπάνω.

3.2.5 Πλαίσιο Εφαρμογής (Application Framework)

Το Android παρέχει στους developers μια ανοιχτού κώδικα πλατφόρμα ανάπτυξης και τη δυνατότητα να αναπτύξουν με αυτή ιδιαίτερα καινοτόμες και πλούσιες σε υλικό, εφαρμογές. Οι developers έχουν στην διάθεση τους τη δυνατότητα ελέγχου του υλικού της συσκευής και μέσω αυτής μπορούν να αποκτήσουν πρόσβαση σε υπηρεσίες εντοπισμού, εκτέλεση διεργασιών παρασκηνίου, και πάρα πολλές ακόμη δυνατότητες οι οποίες βασίζονται στα APIs που είναι διαθέσιμα.

Στο επόμενο επίπεδο της αρχιτεκτονικής του Android λοιπόν, συναντάμε το πλαίσιο των εφαρμογών. Οι developers έχουν πρόσβαση σε όλα τα APIs μεταξύ αυτών και στα κύρια APIs που χρησιμοποιούν οι ενσωματωμένες εφαρμογές. Η δομή των εφαρμογών είναι τέτοια που ευνοείται η επαναχρησιμοποίηση δομικών συστατικών, και επίσης επιτρέπεται η χρήση των δυνατοτήτων τις μίας εφαρμογής από άλλες εφαρμογές, βέβαια κάτω από τις προδιαγραφές ασφάλειας του Android. Τα σημαντικότερα δομικά στοιχεία του πλαισίου εφαρμογών είναι:

• Σύστημα προβολών (View System) – αποτελεί ένα εκτενές σύνολο από αντικείμενα GUI τα οποία μπορούν να χρησιμοποιηθούν κατά το σχεδιασμό μιας εφαρμογής. Παραδείγματα προβολών είναι οι λίστες (listView), το πλέγμα (GridView), πεδία εισαγωγής κειμένου, κουμπιά, κλπ

• Πάροχος Περιεχομένου (Content Provider) – δίνει τη δυνατότητα στις εφαρμογές να μοιράζονται ή να ανταλλάσσουν δεδομένα μιας συγκεκριμένης μορφής η οποία ορίζεται από τον πάροχο. Παραδείγματα δεδομένων, είναι οι επαφές χρήστη και οι βάσεις δεδομένων των εφαρμογών.

• Διαχειριστής Πόρων (Resource Manager) – παρέχει πρόσβαση σε υλικό το οποίο δεν είναι σε μορφή κώδικα όπως πχ, εικόνες, αρχεία xml, πίνακες χαρακτήρων, κλπ

• Διαχειριστής Ειδοποιήσεων (Notification Manager) – δίνει στις εφαρμογές πρόσβαση στις υπηρεσίες ειδοποιήσεων χρήστη. Τέτοιες είναι οι ειδοποιήσεις στη notification bar,

τα toast μηνύματα στο κάτω μέρος της οθόνης, η δόνηση του κινητού και η ενεργοποίηση της οθόνης, κλπ

• Διαχειριστής Δραστηριοτήτων (Activity Manager) – διαχειρίζεται τον κύκλο ζωής των δραστηριοτήτων και παρέχει δυνατότητα πλοήγησης από δραστηριότητα σε δραστηριότητα κρατώντας αποθηκευμένη στη μνήμη τη σειρά εκτέλεσης αυτών. Στο σχεδιάγραμμα (Εικόνα 1.12) φαίνεται λεπτομερώς ο κύκλος ζωής κάθε δραστηριότητας.

3.3 Περιβάλλον ανάπτυξης εφαρμογών Android

Για να δημιουργήσουμε κάποια εφαρμογή για το λειτουργικό σύστημα Android θα πρέπει να έχουμε ένα κατάλληλο περιβάλλον ανάπτυξης το οποίο αποτελείται από τρία βασικά προγράμματα που πρέπει να εγκατασταθούν με την ακόλουθη σειρά:

• JAVA (link:http://www.java.com/en/download/index.jsp)

• Eclipse IDE (link:https://www.eclipse.org/downloads/)

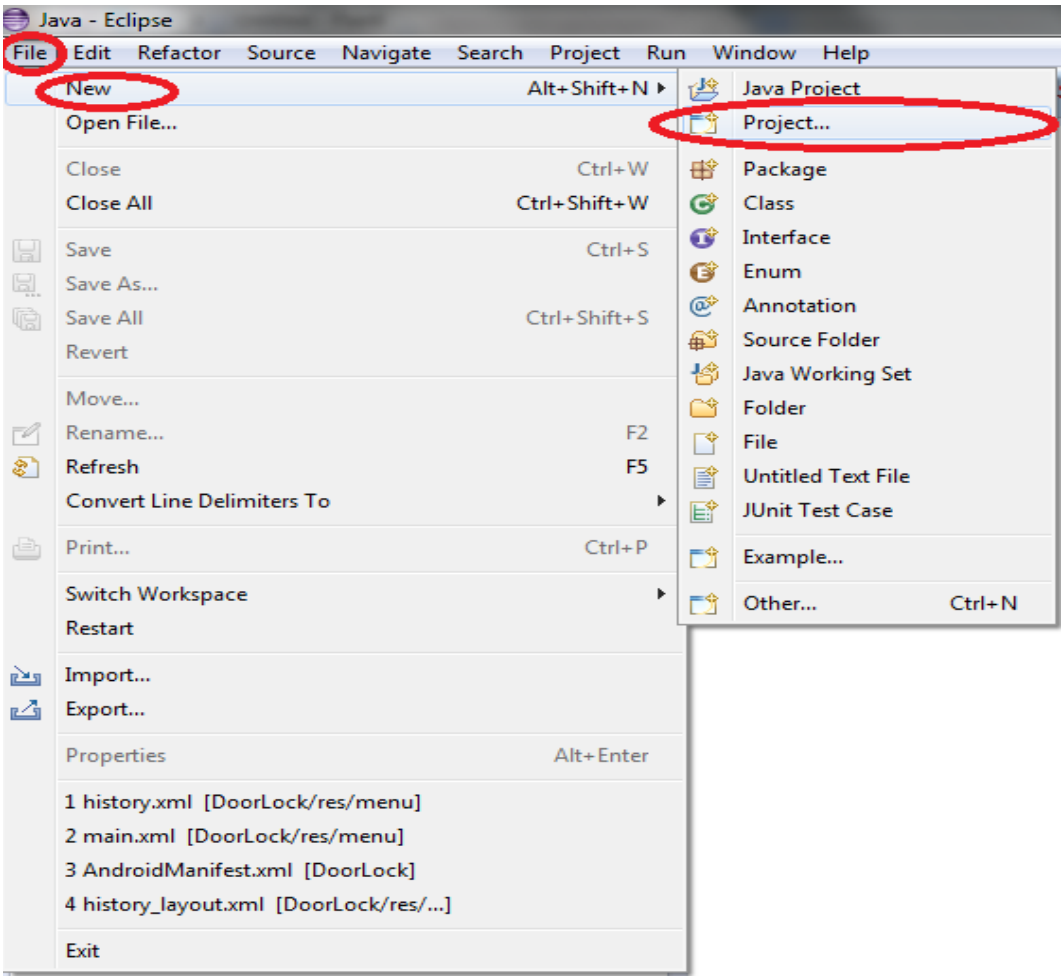
• Android SDK (link:http://developer.android.com/sdk/index.html?hl=sk)

Η σειρά πρέπει να τηρηθεί γιατί το καθένα έχει σαν προϋπόθεση ότι το προηγούμενο ήδη έχει εγκατασταθεί στο σύστημα. Το θετικό σημείο για το συγκεκριμένο περιβάλλον είναι η δωρεάν διανομή των προαναφερθέντων προγραμμάτων, βέβαια πρέπει να υπάρχουν κάποιες βασικές γνώσεις στην γλώσσα προγραμματισμού Java για την καλύτερη και πιο εύκολη δημιουργία εφαρμογών.

Προτού ξεκινήσουμε την δημιουργία ενός καινούριου project πρέπει να κατανοήσουμε την λειτουργία και την φιλοσοφία των προγραμμάτων. Την δουλειά για την εκτέλεση της εφαρμογής έπειτα από την σωστή διατύπωση του κώδικα αναλαμβάνει είτε ένας εξομοιωτής (emulator) και συγκεκριμένα ο Dalvik Virtual Machine (DVM) ο οποίος (όπως αναφέραμε και στο κεφάλαιο 3.2.3) προσομοιώνει μια Android κινητή συσκευή, είτε μια συσκευή με λειτουργικό σύστημα Android το οποίο είναι κατάλληλα συνδεδεμένο με τον υπολογιστή. Από την πλευρά του προγραμματιστή, η χρήση του εξομοιωτή είναι ένα πολύ καλό και θετικό στοιχείο γιατί μπορεί να αναπτύσσει Android εφαρμογές χωρίς να διαθέτει Android συσκευή αλλά ταυτόχρονα να δοκιμάζει τις εφαρμογές αν τρέχουν σωστά σε Android περιβάλλον.

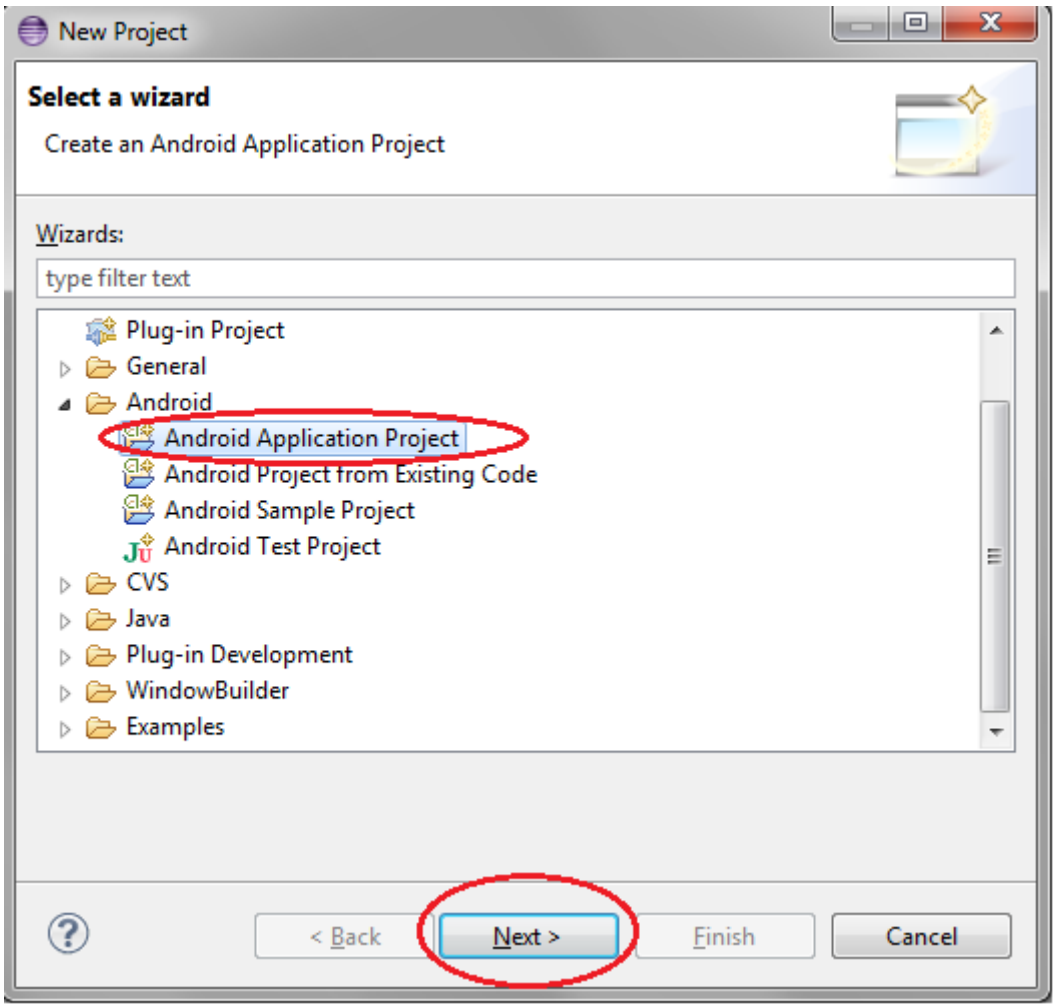
Έχοντας προβεί στις κατάλληλες εγκαταστάσεις και ρυθμίσεις των απαραίτητων εργαλείων, είμαστε έτοιμοι να δημιουργήσουμε και να τρέξουμε το πρώτο μας Android Project. Παρακάτω θα ακολουθήσουν βήματα με τα οποία θα πετύχουμε την σωστή εκτέλεση μιας Android εφαρμογής.

Αρχικά ξεκινάμε το πρόγραμμα eclipse και το πρώτο βήμα που πρέπει να κάνουμε είναι να δημιουργήσουμε ένα νέο project. Οπότε πατάμε File->New->Project όπως βλέπουμε στην εικόνα 3.12



Εικόνα 3.12: Επιλογή καινούριου Project

Έχοντας πατήσει το project θα εμφανιστεί το παρακάτω παράθυρο (Εικόνα 3.13) στο οποίο κάνουμε κλικ στον Android φάκελο ώστε να μας ανοίξουν τα περιεχόμενα του και επιλέγουμε το Android Project. Στην συνέχεια πατάμε NEXT.



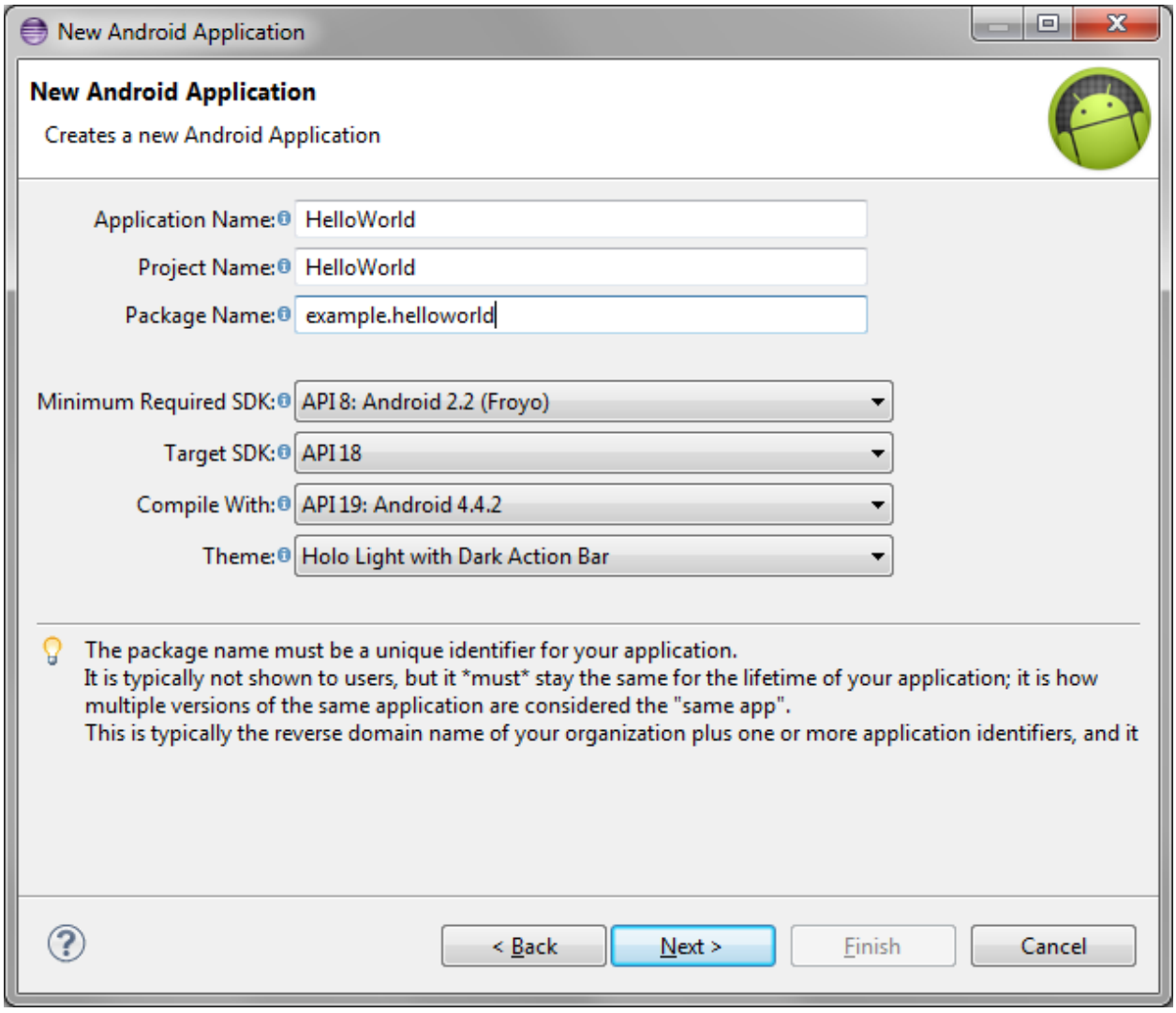
Εικόνα 3.13: Επιλογή τύπου του Project

Έπειτα θα μας εμφανιστεί το παρακάτω παράθυρο (Εικόνα 3.14) το οποίο είναι και το πιο σημαντικό για τη σωστή δημιουργία ενός καινούριου project. Πρέπει να κατανοήσουμε την σημασία του κάθε πεδίου πριν ξεκινήσουμε να τα συμπληρώνουμε. Τα πιο σημαντικά πεδία είναι :

- Το Application Name όπου γράφουμε το όνομα που θέλουμε να δώσουμε στην εφαρμογή μας. Είναι το ίδιο όνομα που θα εμφανίζεται και στην Android συσκευή όταν εγκαταστήσουμε την εφαρμογή μας. Οπότε προσέχουμε με την ονομασία του.
- Το Project Name στο οποίο καθορίζουμε το όνομα της εφαρμογής. Είναι το όνομα το οποίο θα έχει η εφαρμογή και θα βλέπουν οι χρήστες της.
- Το Package Name στο οποίο όπως γνωρίζουμε από την java πρέπει να βάλουμε όνομα xxx.xxx για να δημιουργήσουμε πακέτο. Η χρήση των πακέτων γίνεται για να

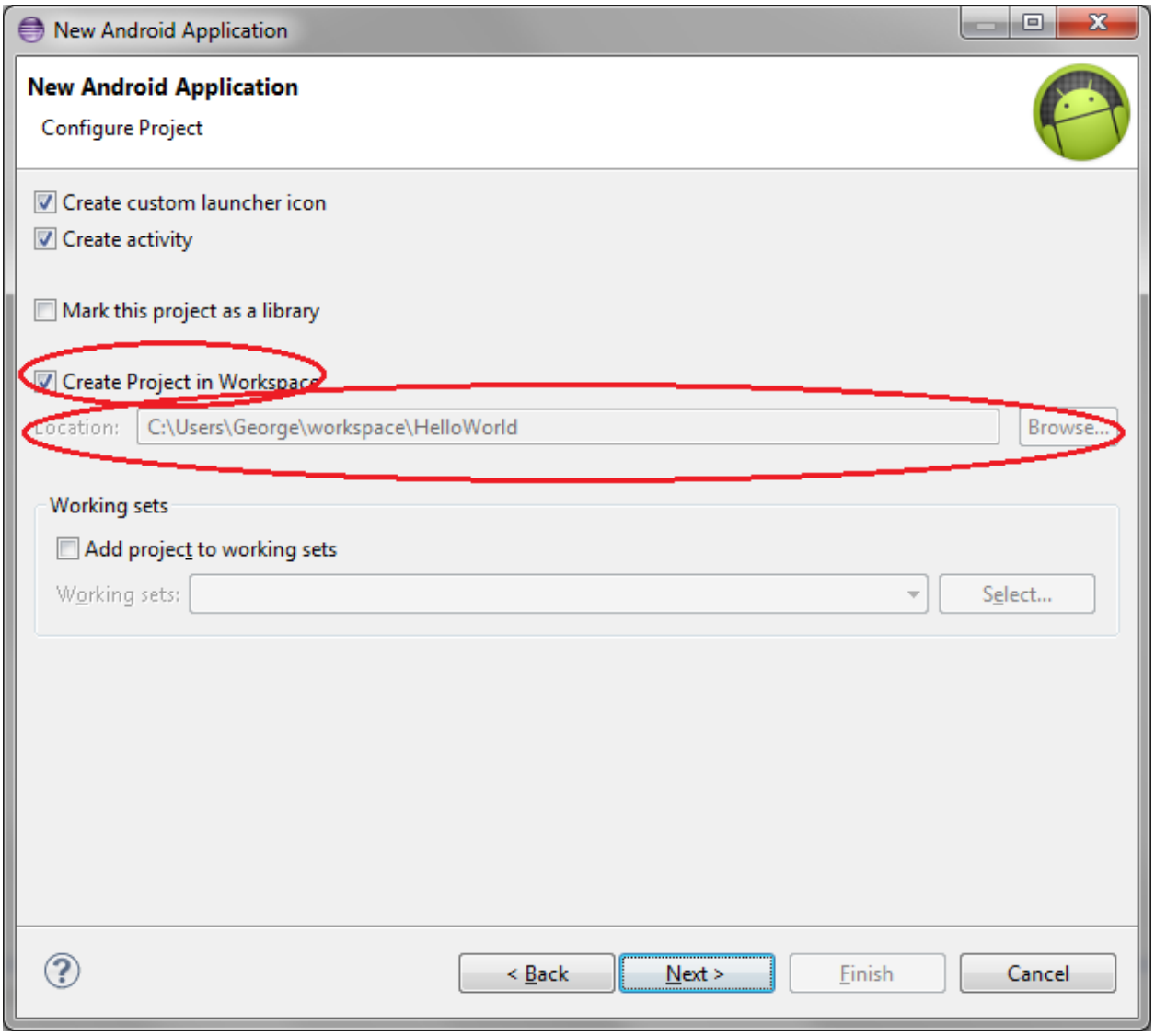
ταξινομηθούν και να ομαδοποιηθούν τα .java αρχεία για ευκολότερη αντιμετώπιση προβλημάτων σε επίπεδο κώδικα αλλά και για να καθορίζουν δικαιώματα πρόσβασης.

- Το Minimum Required SDK όπου επιλέγουμε την μικρότερη έκδοση που θα υποστηρίζει η εφαρμογή μας.
- Το Target SDK στο οποίο επιλέγουμε τη μεγαλύτερη δυνατή έκδοση που θα υποστηρίζει η εφαρμογή μας.
- Το Compile With και το Theme αν δεν υπάρχει κάποιος λόγος τα αφήνουμε ως έχουν



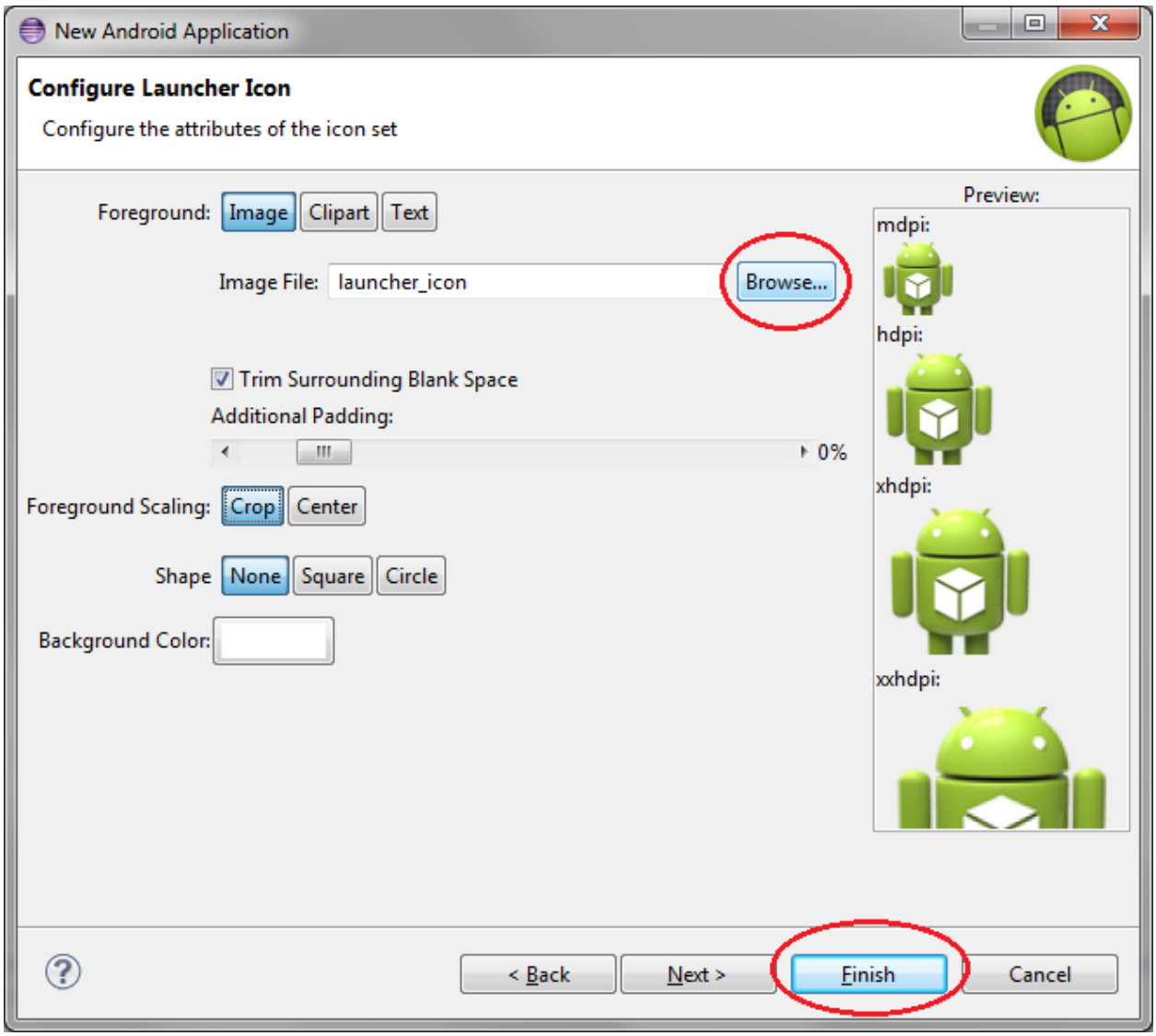
Εικόνα 3.14: Συμπλήρωση στοιχείων

Στη συνέχεια αν πατήσουμε NEXT θα εμφανιστεί το παρακάτω παράθυρο (Εικόνα 3.15) στο οποίο αν επιθυμούμε μπορούμε να αλλάξουμε τη θέση όπου θα αποθηκευτεί στον υπολογιστή μας τα αρχεία του project που δημιουργούμε.



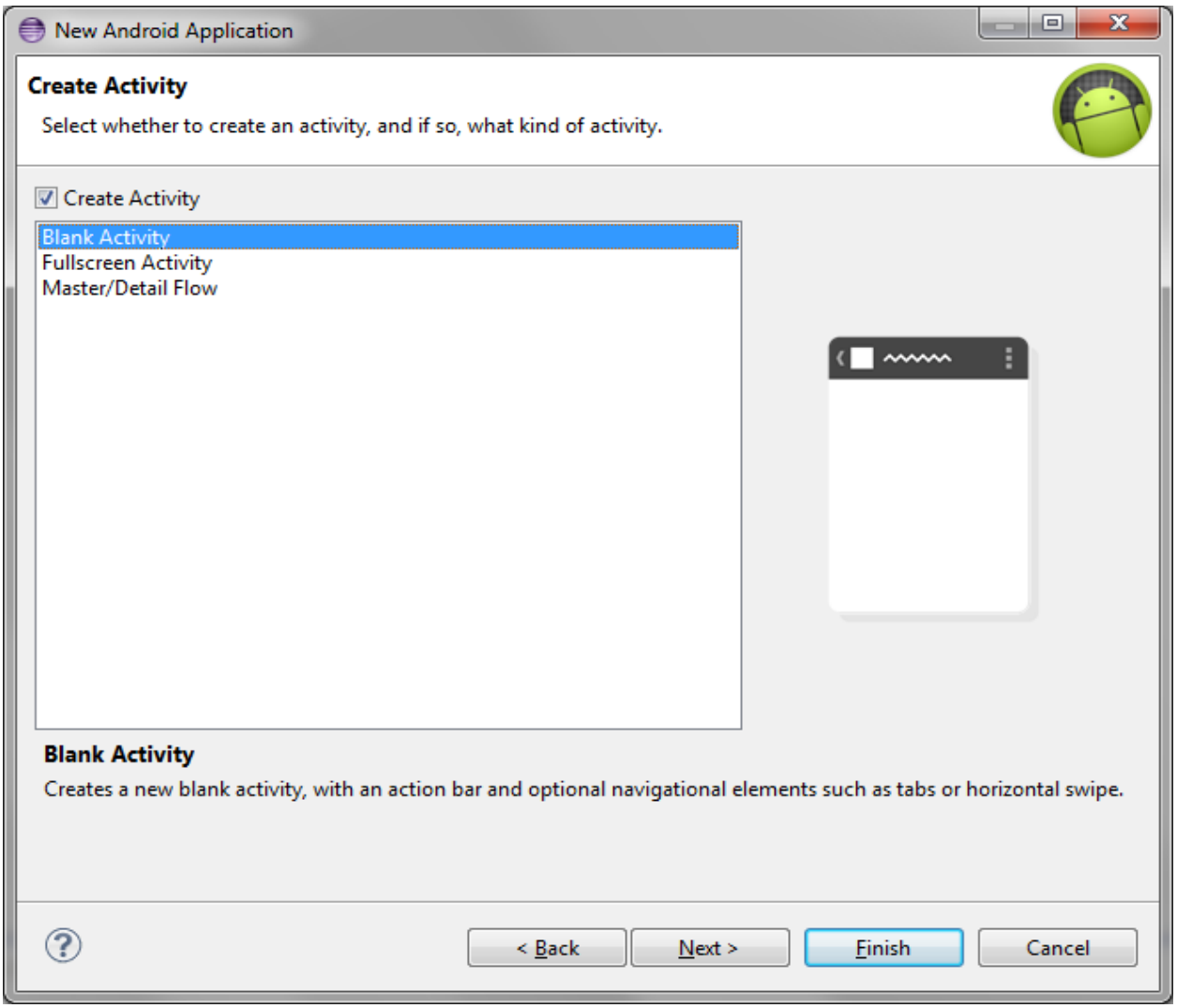
Εικόνα 3.15: Επιλογή Location

΄Υστερα θα εμφανιστεί το παρακάτω παράθυρο (Εικόνα 3.16) όπου επιλέγουμε το εικονίδιο που θα έχει η εφαρμογή μας. Αν θέλουμε να προσθέσουμε δικό μας πατάμε στο "Browse...". Δοκιμάζοντας και τις υπόλοιπες επιλογές που υπάρχουν όπως το κεντράρισμα του εικονιδίου θα φτάσουμε στο επιθυμητό αποτέλεσμα.



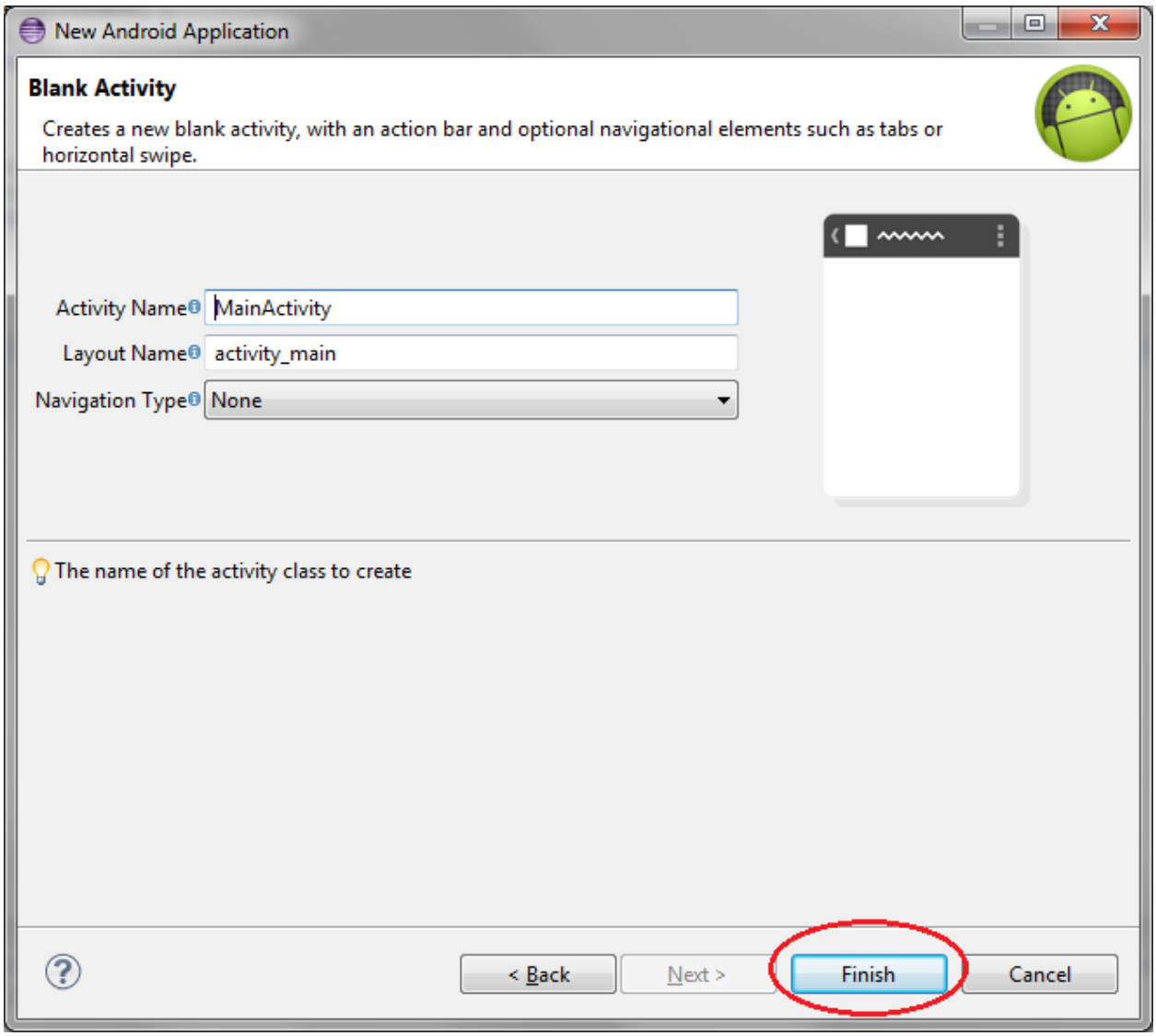
Εικόνα 3.16: Επιλογή Εικονιδίου

Παρατηρούμε ότι μπορούμε να πατήσουμε Finish χωρίς να προβούμε σε επιπλέον ρυθμίσεις. Αν όμως θέλουμε να συνεχίσουμε πατάμε το Next και στο επόμενο βήμα μας δίνεται η δυνατότητα να αλλάξουμε τη μορφή του "action bar" (Εικόνα 3.17)



Εικόνα 3.17: Επιλογή Action Bar

Στο τελευταίο βήμα μπορούμε να αλλάξουμε το όνομα που θα έχει η κλάση και το όνομα του layout της συγκεκριμένης κλάσης (Εικόνα 3.18).



Εικόνα 3.18: Επιλογή στοιχείων κλάσης

Πατώντας Finish θα δημιουργήσουμε με επιτυχία το project μας και παρατηρούμε ότι υπάρχει κώδικας γραμμένος ήδη μέσα στη κλάση.

Τέλος για να τρέξουμε την εφαρμογή μας θα πρέπει να δημιουργήσουμε μια εικονική μηχανή(ή να συνδέσουμε το κινητό όπως έχουμε αναφέρει και πιο πάνω) η οποία να συμβαδίζει με τις επιλογές των εκδόσεων API που κάναμε κατά τη δημιουργία του project.

ΚΕΦΑΛΑΙΟ 4

4 AMARINO

Το Amarino είναι μια εφαρμογή Android που δημιουργεί κανάλι επικοινωνίας μεταξύ ενός εξωτερικού κυκλώματος, όπως το Arduino στην παρούσα εργασία, και ενός κινητού τηλεφώνου μέσω σύνδεσης Bluetooth.

Το Amarino αποτελείται από δύο κύρια στοιχεία: μια Android εφαρμογή για το κινητό και μια βιβλιοθήκη που είναι απαραίτητη για τον Arduino. Η Android εφαρμογή και η βιβλιοθήκη για τον Arduino επικοινωνούν μέσω Bluetooth. Έτσι δίνεται η δυνατότητα στο χρήστη να δημιουργήσει εφαρμογές για το κινητό και τον Arduino εύκολα χρησιμοποιώντας το κανάλι επικοινωνίας του Amarino.

4.1 Συνιστώσες εφαρμογής Android

Μια εφαρμογή Android αποτελεί ένα πακέτο συνιστωσών και αρχείων με δεδομένα όλα μαζί σε ένα αρχείο .apk. Όμως αντί να έχουμε μια μέθοδο main σαν μοναδικό σημείο εισόδου για το ξεκίνημα μιας εφαρμογής, στο Android OS κάθε συνιστώσα μιας εφαρμογής μπορεί να ξεκινάει ανεξάρτητα αν αυτό επιτρέπεται από την εφαρμογή. Έτσι, οι εφαρμογές Android διαθέτουν πολλές τέτοιες συνιστώσες οι οποίες μπορούν να επαναχρησιμοποιηθούν από τις εφαρμογές των χρηστών.

Κάθε εφαρμογή Android διαθέτει από ένα αρχείο xml, το AndroidManifest.xml. Εκεί δηλώνονται οι συνιστώσες της εφαρμογής και η διαθεσιμότητα τους σε άλλη εφαρμογή. Επίσης υπάρχουν και κάποιες ακόμη πληροφορίες όπως εικονίδιο, τίτλος, version της εφαρμογής, κλπ.

Η δραστηριότητα Activity είναι η σημαντικότερη συνιστώσα μιας εφαρμογής. Το User Interface σε μια συσκευή Android παρέχεται με μια επέκταση στην κλάση Activity. Μια εφαρμογή μπορεί να περιέχει μία ή περισσότερες δραστηριότητες. Η δραστηριότητα είναι σε μια από τρεις καταστάσεις: active, pause, stop. Κάθε χρονική στιγμή υπάρχει μόνο μια active εφαρμογή, και είναι ακριβώς αυτή που δείχνει η οθόνη της συσκευής και με την οποία αλληλεπιδρά ο χρήστης. Μια δραστηριότητα είναι σε κατάσταση pause αν ένα μέρος της είναι ακόμα ορατό παρόλο που έχει ενεργοποιηθεί μια άλλη δραστηριότητα. Αυτή η δραστηριότητα είναι ακόμα στη μνήμη και μπορεί να ξανά ενεργοποιηθεί. Μια δραστηριότητα είναι σε κατάσταση stop αν δεν είναι ορατή στον χρήστη. Το λειτουργικό μπορεί κάθε στιγμή να τη διαγράψει αν χρειάζεται τη μνήμη.

Η πρώτη μέθοδος που καλείται όταν ξεκινάει μια δραστηριότητα είναι η onCreate. Σε αυτήν γίνονται όλες οι αρχικοποιήσεις, όπως η δημιουργία των οθονών και η αρχικοποίηση των μεταβλητών. Η μέθοδος onStart καλείται μόλις πριν η δραστηριότητα γίνει ορατή στον χρήστη. Η μέθοδος onResume καλείται αμέσως μετά επιτρέποντας την αλληλεπίδραση του χρήστη. Μετά από αυτές τις τρεις μεθόδους, η δραστηριότητα θεωρείται ενεργή. Ο χρήστης μπορεί να αλληλεπιδράσει με τη δραστηριότητα.

Η κλήση της μεθόδου onPause γίνεται όταν μια άλλη δραστηριότητα ξεκινήσει (start) ή επανέλθει σε λειτουργία (resume). Σε αυτό το σημείο, οι χρήστες τοποθετούν κώδικα για την αποθήκευση ή καθάρισμα διαφόρων πληροφοριών ή δεδομένων. Η δραστηριότητα είναι σε παύση (pause) και μπορεί είτε να επανέλθει στο προσκήνιο (onResume) ή να σταματήσει η λειτουργία της (onStop) σε περίπτωση που πάψει να είναι ορατή στον χρήστη. Μετά τη μέθοδο onStop, η δραστηριότητα μπορεί να παραμείνει στη μνήμη (αν το λειτουργικό δεν χρειάζεται χώρο στη μνήμη) και να ξανά ενεργοποιηθεί με onRestart και onStart. Το λειτουργικό μπορεί να την καταργήσει χωρίς να καλέσει την onDestroy εάν χρειάζεται χώρο στη μνήμη. Τέλος, η onDestroy καλείται όταν ο χρήστης εξέλθει της δραστηριότητας.

4.2 Broadcast Receiver (δέκτης εκπομπής)

Ο Broadcast Receiver είναι μια συνιστώσα η οποία μπορεί να λαμβάνει τα μηνύματα που εκπέμπονται. Το λειτουργικό σύστημα εκπέμπει συνεχώς πολλά μηνύματα για να πληροφορήσει τις εφαρμογές του κινητού σχετικά με διάφορα γεγονότα που συμβαίνουν στο σύστημα, όπως για παράδειγμα πληροφόρηση σχετικά με το επίπεδο της μπαταρίας, εισερχόμενες κλήσεις, μηνύματα, κλπ. Όλες οι εφαρμογές επιτρέπεται να εκπέμπουν μηνύματα και κάθε εφαρμογή που εγγράφεται για να λαμβάνει έναν ειδικό τύπο μηνύματος (action) θα λαμβάνει την ειδοποίηση. Το Amarino στέλνει μηνύματα για τις μεταβολές στην κατάσταση σύνδεσης και κατά τη διάρκεια που λαμβάνει δεδομένα από τον Arduino. Με αυτόν τον τρόπο, όλες οι ενδιαφερόμενες εφαρμογές μπορούν να λαμβάνουν αυτήν την πληροφόρηση αν επεκτείνουν την κλάση BroadcastReceiver και εγγράψουν το δέκτη είτε κατά το runtime ή στο AndroidManifest.xml.

4.3 Amarino toolkit

Το Amarino toolkit έχει δύο κύρια στοιχεία: Μια Android εφαρμογή και μια βιβλιοθήκη για τον Arduino. Το Amarino αποτελεί στην ουσία ένα κανάλι επικοινωνίας το οποίο

δίνει τη δυνατότητα σε ένα κινητό τηλέφωνο και σε μια συσκευή με μικροελεγκτή μέσω του Bluetooth να ανταλλάσουν δεδομένα.

Το Amarino δημιουργεί μια αόρατη στο χρήστη ανταλλαγή δεδομένων η οποία βασίζεται στη δημιουργία γεγονότων (events). Από την άλλη μεριά, ο Arduino με τη βιβλιοθήκη του έχει ένα μηχανισμό κλήσης μεθόδων μόλις υπάρξει είσοδος ενός δεδομένου (callback methods). Ο χρήστης χρησιμοποιεί διαφορετικές μεθόδους ανάλογα με τον τύπο των δεδομένων εισόδου, ενώ συγχρόνως διαθέτει μεθόδους για την αποστολή δεδομένων στο κινητό αλλά και για την εξαγωγή των δεδομένων αφού διαχωρίσει τα γεγονότα εισόδου και τα οδηγήσει στην κατάλληλη συνάρτηση.

Η εφαρμογή Android είναι εφοδιασμένη με ένα GUI που επιτρέπει στο χρήστη να χειριστεί τις διάφορες συνδέσεις Bluetooth, να δημιουργήσει μια σειρά γεγονότων και να στείλει δεδομένα για κάθε γεγονός στον Arduino. Επίσης, έχει τρία μέρη που χειρίζονται τις συσκευές Bluetooth, τον χειριστή των συσκευών Bluetooth, το χειριστή των γεγονότων και την παρακολούθηση σύνδεσης.

4.4 Η βιβλιοθήκη MeetAndroid για τον Arduino

Περιέχει συναρτήσεις για την εύκολη είσοδο ή έξοδο δεδομένων μέσω της συσκευής Bluetooth. Η βιβλιοθήκη MeetAndroid, η οποία τοποθετείται στις βιβλιοθήκες του Arduino, χρησιμοποιεί έναν function pointer (το ID της λειτουργίας που στέλνει το Amarino) για την κλήση των κατάλληλων εγγεγραμμένων μεθόδων που θα αντλήσουν τα δεδομένα. Αυτός ο function pointer είναι ένας ASCII χαρακτήρας από 75 διαφορετικές τιμές (0-9, A-Z, a-z, τιμές από 9-A και τιμές από Z-a).

4.5 Amarino API

Τα γεγονότα που συζητήθηκαν μέχρι τώρα και η διαχείρισή τους επιτρέπουν την αποστολή μηνυμάτων στον Arduino αλλά δεν δείχνουν τα δεδομένα που λαμβάνονται από αυτόν ούτε επιτρέπουν την αλληλεπίδραση του χρήστη με την οθόνη αφής. Επομένως, για τη δημιουργία ενός UI ή για την εμφάνιση των δεδομένων από τον Arduino χρειάζεται η ανάπτυξη μιας ξεχωριστής εφαρμογής. Γι’ αυτόν το λόγο, το Amarino προσφέρει ένα Application Programming Interface (API) παρέχοντας πολλές μεθόδους στο χρήστη για την επικοινωνία με τον Arduino. Η επικοινωνία με τη συσκευή Bluetooth του Arduino περιλαμβάνει κάποιες ενέργειες, τη σύνδεση και την αποσύνδεση με τον Arduino και την αποστολή και λήψη δεδομένων από τον Arduino.

Σύνδεση και αποσύνδεση με Arduino

Για τη σύνδεση και την αποσύνδεση με μια Bluetooth συσκευή καλούνται οι μέθοδοι connect και disconnect αντίστοιχα, της κλάσης Amarino και περνιέται σε αυτήν το αντικείμενο (this) και η διεύθυνση της συσκευής (MAC ADDRESS). Μετά την κλήση των μεθόδων σύνδεσης ή αποσύνδεσης, ο χρήστης μπορεί να καταλάβει από τα εκπεμπόμενα μηνύματα του Amarino (intents), ότι η συσκευή BT έχει συνδεθεί ή αποσυνδεθεί.

```
protected void onStart(){
    super.onStart();
    // to receive broadcasted intents we must register our receiver
    registerReceiver(arduinoReceiver, new IntentFilter(AmarinoIntent.ACTION_RECEIVED));
    Amarino.connect(this,MAC_ADDRESS);
}
```

Εικόνα 4.1: Κομμάτι κώδικα σύνδεσης

```
protected void onDestroy(){ //onDestroy stops the communication of bluetooth with arduino
    super.onDestroy();
    // do not forget to unregister a registered receiver
    unregisterReceiver(arduinoReceiver);
    Amarino.disconnect(this,MAC_ADDRESS);
}
```

Εικόνα 4.2: Κομμάτι κώδικα αποσύνδεσης

Αποστολή και λήψη δεδομένων

Η αποστολή δεδομένων στον Arduino, αφού έχει πραγματοποιηθεί η σύνδεση με την Bluetooth συσκευή, γίνεται με τον παρακάτω κώδικα:

```
private void sendData(char flag,String message){
    Amarino.sendDataToArduino(this,MAC_ADDRESS,flag,message);
}

public void sendData(char flag,int num){
    Amarino.sendDataToArduino(this,MAC_ADDRESS,flag,num);
}
```

Εικόνα 4.3: Κομμάτι κώδικα αποστολής δεδομένων

Η μέθοδος Amarino.sendDataToArduino ζητάει ένα αντικείμενο εφαρμογής, τη διεύθυνση της συσκευής, τη σημαία (ID) της λειτουργίας και το μήνυμα που θα αποσταλεί.

Η λήψη δεδομένων χρειάζεται, όπως και η λήψη μηνυμάτων για την κατάσταση της σύνδεσης, έναν Broadcast Receiver, ο οποίος θα λαμβάνει τα δεδομένα (ACTION_RECEIVED intent). Όταν ο Amarino λάβει δεδομένα από τον Arduino, τα επισυνάπτει σε ένα ACTION_RECEIVED intent το οποίο λαμβάνουν οι ενδιαφερόμενες εφαρμογές και εξάγουν τα δεδομένα όπως στον παρακάτω κώδικα.

```
@Override
public void onReceive(Context context, Intent intent) {
    // TODO Auto-generated method stub
    String data = null;
    String action = intent.getAction();
    final int dataType = intent.getIntExtra(AmarinoIntent.EXTRA_DATA_TYPE, -1); // we only expect String data though,
    //but it is better to check if really string was sent
    // later Amarino will support differnt data types, so far data comes always as string and
    // you have to parse the data to the type you have sent from Arduino, like it is shown below

if(action.equals(AmarinoIntent.ACTION_RECEIVED)){//check that we recieve data
    if (dataType == AmarinoIntent.STRING_EXTRA){
        data = intent.getStringExtra(AmarinoIntent.EXTRA_DATA);
        if (data != null){
```

Εικόνα 4.4: Κομμάτι κώδικα λήψης δεδομένων

ΚΕΦΑΛΑΙΟ 5

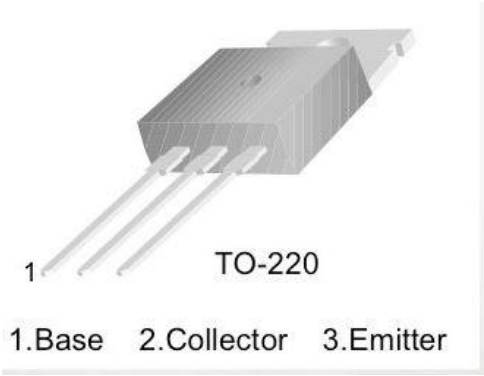
5.1 Υλικό μέρος του συστήματος

Η ανάπτυξη του υλικού του project θα γίνει με τα παρακάτω υλικά

- Έναν Arduino-Uno
- Μια ηλεκτρική κλειδαριά
- Ένα κινητό τηλέφωνο με λειτουργικό σύστημα Android

Και για τη κατασκευή του κυκλώματος για τη λειτουργία του Bluetooth με το Arduino θα γίνει με τα παρακάτω υλικά

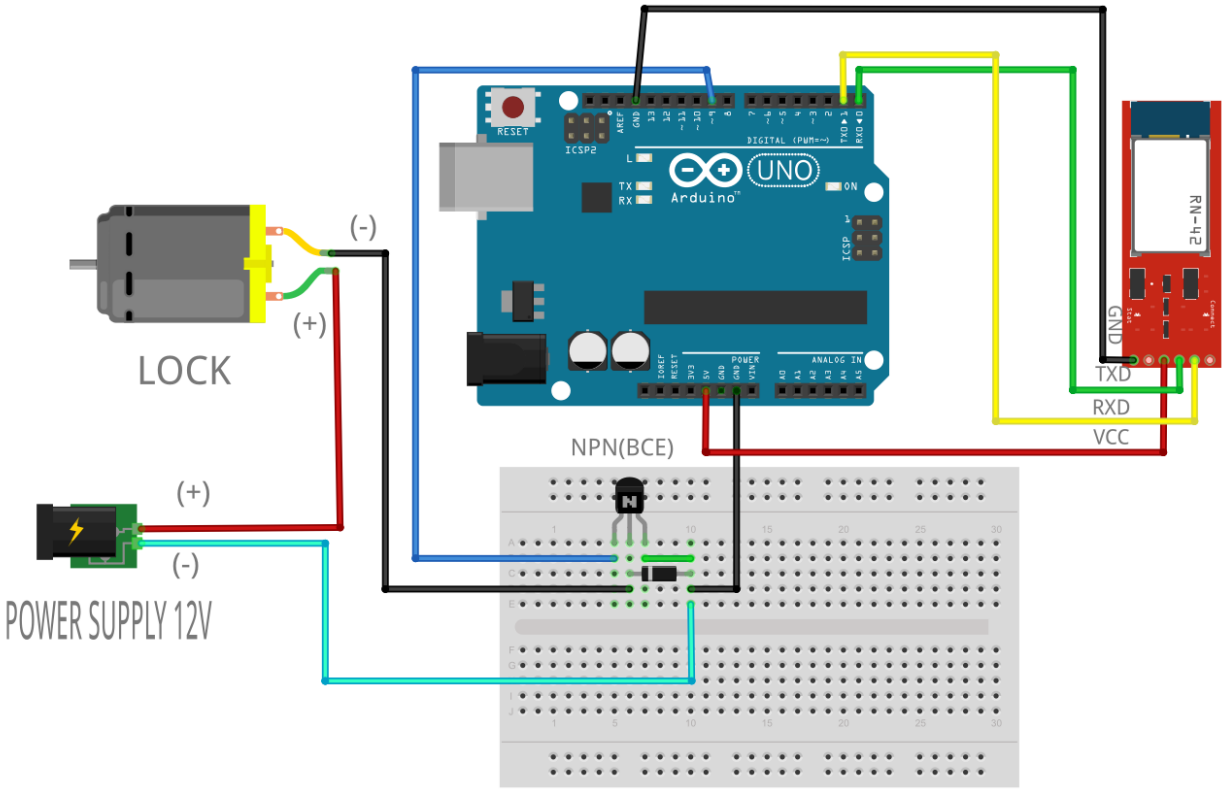
- Μια διάτρητη πλακέτα
- Μια Bluetooth συσκευή
- Ένα transistor TIP120 (NPN)



Εικόνα 5.1: Transistor TIP120

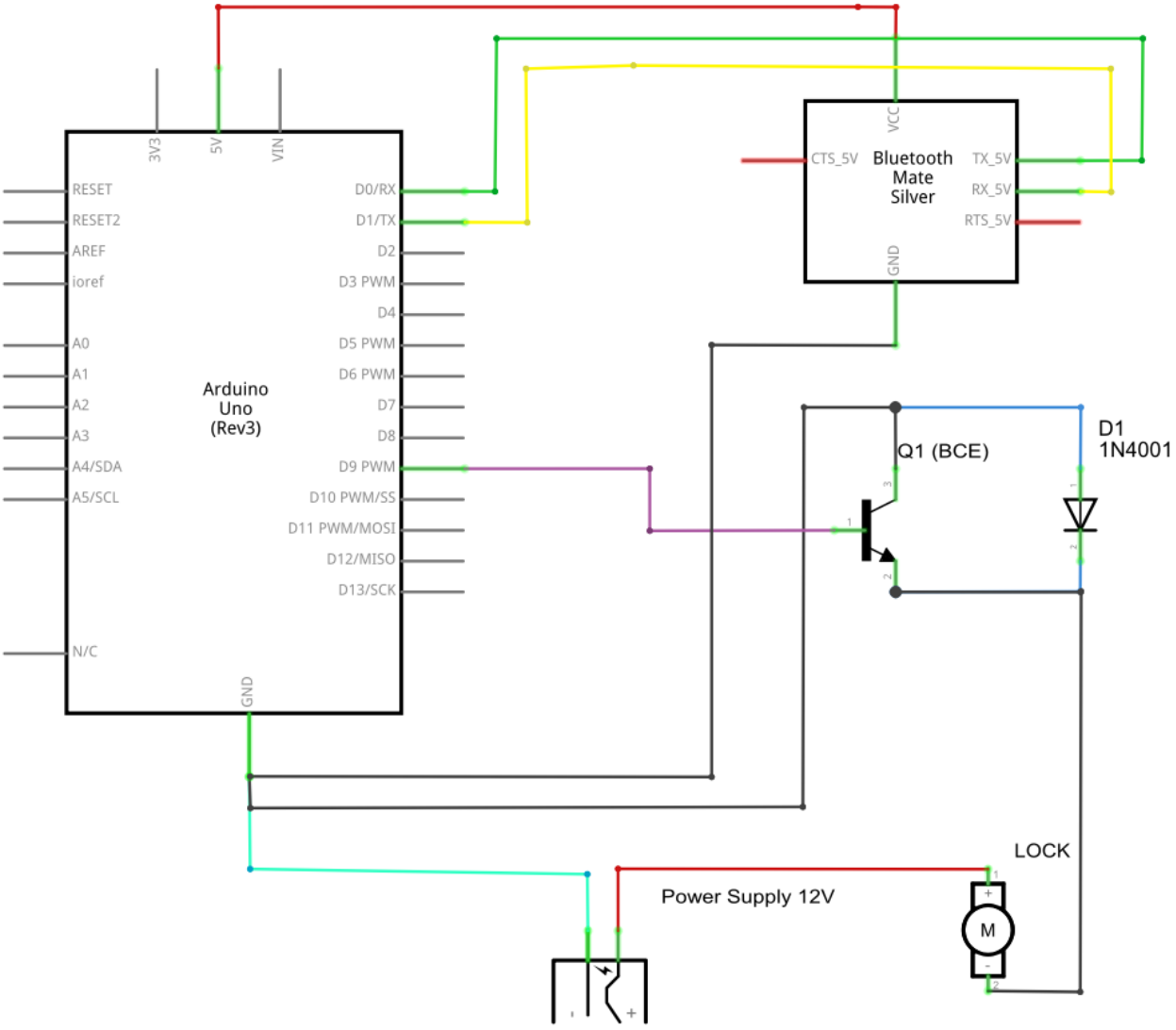
- Μια δίοδος 1N4001
- Καλώδια

5.2 Το κύκλωμα του συστήματος



Εικόνα 5.2: Γραφική αναπαράσταση του κυκλώματος

Στην παραπάνω εικόνα μας παρουσιάζεται μια αναπαράσταση του κυκλώματος η όποια βέβαια είναι φιλική προς τον αναγνώστη αλλά δεν αποτελεί μοναδικό οδηγό προς τη πλήρη κατανόηση και σωστή υλοποίηση του κυκλώματος. Πιθανόν κάποιοι ακροδέκτες να μην είναι ευανάγνωστοι. Για αυτό το λόγο παραθέτουμε το παρακάτω σχεδιάγραμμα.



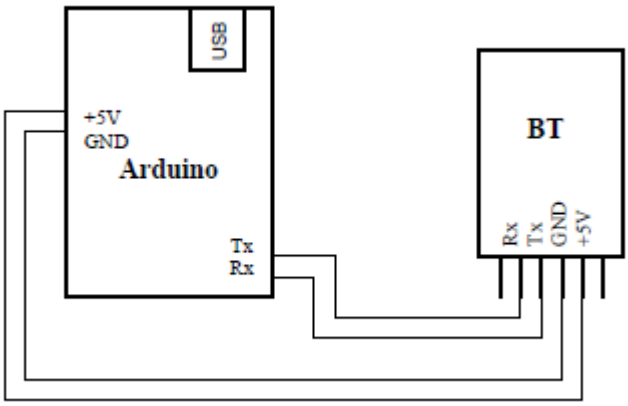
Εικόνα 5.3: Σχέδιο του κυκλώματος

Παρατηρώντας το παραπάνω σχεδιάγραμμα μπορούμε να δούμε όλους τους ακροδέκτες του Arduino στους οποίους έχουμε ήδη αναφερθεί (κεφάλαιο 2.5) κατά κύριο λόγο και την σωστή συνδεσμολογία που ακολουθήθηκε για την υλοποίηση της πτυχιακής εργασίας.

5.2.1 Η σύνδεση της συσκευής Bluetooth

Η συσκευή Bluetooth θα πρέπει να προγραμματιστεί σε ταχύτητα επικοινωνίας 9600bps.

Στο παρακάτω σχήμα παρατηρούμε την ορθή σύνδεση της συσκευής με τον Arduino.

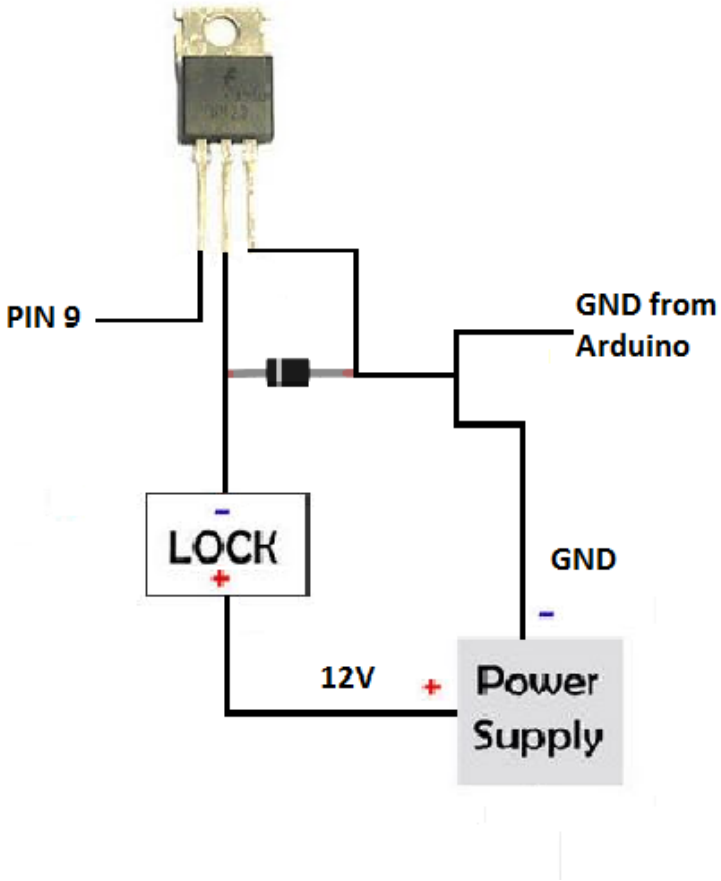


Εικόνα 5.4: Σύνδεση Arduino με Bluetooth

Σημαντικό είναι όταν προγραμματίζουμε τον Arduino να ΜΗΝ υπάρχει σύνδεση με το Bluetooth επειδή τα ίδια pins χρησιμοποιούνται για τον προγραμματισμό. Αν ξεχάσουμε να το αφαιρέσουμε δεν θα προγραμματιστεί καθόλου και θα μας εμφανίσει μήνυμα σφάλματος κατά τη φόρτωση του κώδικα στο Arduino.

5.2.2 Η σύνδεση με την κλειδαριά και τον Arduino

Η βάση του τρανζίστορ οδηγείται από το pin9, ο συλλέκτης συνδέεται με την κλειδαριά ενώ στο άλλο άκρο η κλειδαριά συνδέεται με τροφοδοσία, και το άλλο άκρο του transistor συνδέεται με γείωση. Αυτό που πρέπει να επισημάνουμε είναι οτι στον έναν ακροδέκτη του τρανζίστορ θα έχουμε δύο γειώσεις, μία από το τροφοδοτικό των 12V που χρησιμοποιούμε για την κλειδαριά και μία από την πλακέτα του arduino.



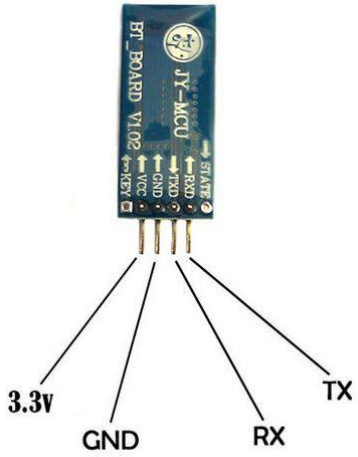
Εικόνα 5.5: Σύνδεση με την κλειδαριά και τον Arduino

5.3 Συσκευή Bluetooth

Τεχνικά χαρακτηριστικά

- Bluetooth με σειριακή επικοινωνία
- Ταχύτητα σειριακής επικοινωνίας 9600 bps
- Όνομα ‘LINVOR’
- Τροφοδοσία 3,3 ως 5 Volt

Τα σήματα του ακροδέκτη



Εικόνα 5.6: Τα σήματα του ακροδέκτη του Bluetooth

PIN	NAME	FUNCTION
1	KEY	HIGH=προγραμματισμός, LOW=λειτουργία
2	VCC	3.3 - 5V
3	GND	Ground
4	TXD	TX transmit pin
5	RXD	RX receive pin
6	STATE	Κατάσταση σύνδεσης

5.4 Οδηγός χρήσης της εφαρμογής DoorLock

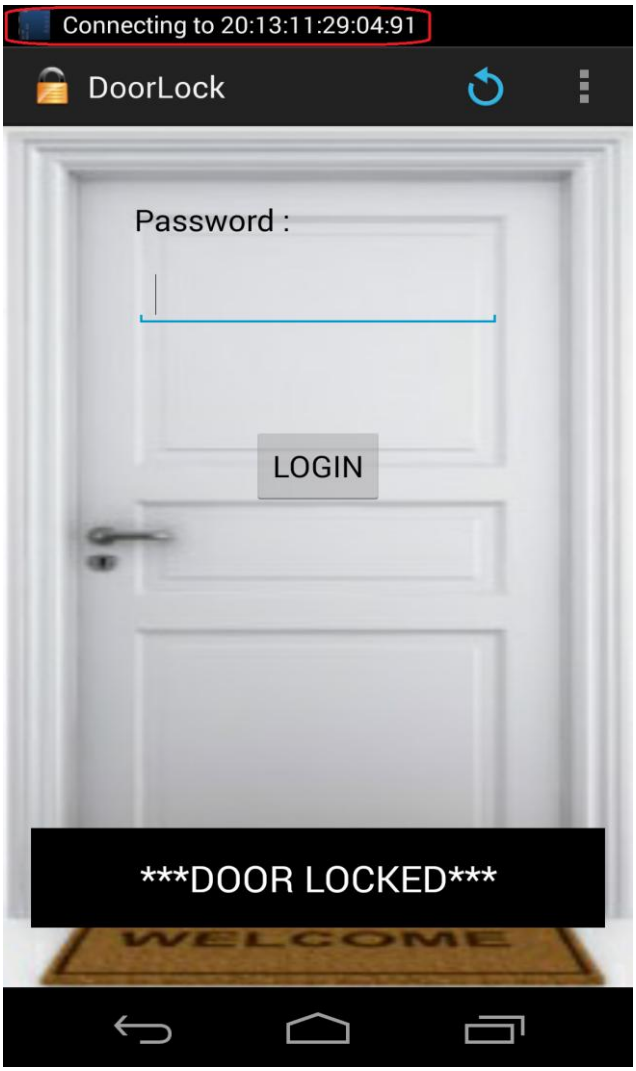
Αρχικά πρέπει να αναφέρουμε ότι η εφαρμογή μας χωρίζεται σε 3 διαφορετικές οθόνες διεπαφής οι οποίες είναι:

- Η Main page(Κύρια οθόνη για το άνοιγμα της κλειδαριάς, εικόνα 5.7)
- Η History (για την εμφάνιση του ιστορικού, εικόνα 5.8)
- Η Change Password (για την αλλαγή του κωδικού, εικόνα 5.9)

Επίσης να σημειώσουμε ότι ο κάθε όροφος θεωρείται ως ένας χρήστης άσχετα με τον αριθμό των ατόμων που μπορούν να χρησιμοποιήσουν τον ίδιο κωδικό, αφού σε ένα διαμέρισμα μπορεί να κατοικούν πολλοί άνθρωποι.

5.4.1 Έναρξη εφαρμογής και εισαγωγής κωδικού

Κατά την έναρξη της εφαρμογής παρατηρούμε στην μπάρα κατάστασης(βλέπε εικόνα 5.7) πως εμφανίζονται μηνύματα που μας υποδεικνύουν ότι προσπαθεί να γίνει αυτόματη σύνδεση με το Bluetooth που έχουμε συνδέσει στην πλατφόρμα Arduino, χρησιμοποιώντας την εφαρμογή "Amarino 2.0". Αν δεν ήταν δυνατή η σύνδεση, μας δίνεται η δυνατότητα μέσω του κουμπιού  που υπάρχει στο πάνω δεξιά μέρος της οθόνης μας να πραγματοποιήσουμε ανανέωση της σύνδεσης μας. Έπειτα από την επιτυχή σύνδεση μπορούμε να εισάγουμε τον κωδικό ώστε να ανοίξουμε την ηλεκτρική κλειδαριά. Αν ο κωδικός είναι εσφαλμένος ή είναι κενός εμφανίζονται κατάλληλα μηνύματα τα όποια ενημερώνουν τον χρήστη. Επίσης στην περίπτωση όπου έγινε προσπάθεια εισαγωγής με λανθασμένο κωδικό εμφανίζεται μήνυμα που αναφέρει πως έχει άλλες 2 προσπάθειες να πληκτρολογήσει σωστά το κωδικό αλλιώς θα κλειδωθεί το κουμπί "Login" για 10 δευτερόλεπτα. Αυτό συμβαίνει για λόγους ασφάλειας. Να μην μπορεί δηλαδή κάποιος κακόβουλος χρήστης να δοκιμάζει συνέχεια διαφορετικούς κωδικούς χωρίς να υπάρχει παρεμπόδιση. Στην συνέχεια αφού εισάγουμε σωστά στοιχεία θα ανοίξει η κλειδαριά και θα εμφανιστεί κατάλληλο μήνυμα στην οθόνη για 2 δευτερόλεπτα. Ακόμα μας δίνεται η δυνατότητα να μεταβούμε σε άλλη οθόνη πατώντας το πλήκτρο μενού  που βρίσκεται στο πάνω δεξιά μέρος και μας εμφανίζει τις επιλογές για το ιστορικό και για την αλλαγή του κωδικού πρόσβασης.



Εικόνα 5.7: Αρχική οθόνη (Main Page)

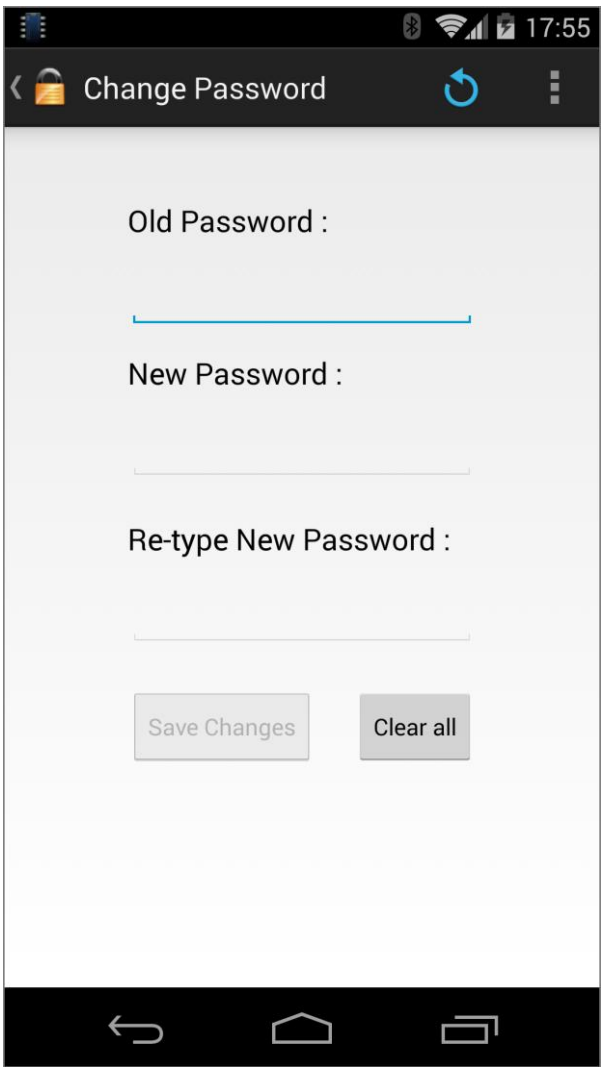
5.4.2 Εμφάνιση ιστορικού εισόδου και λανθασμένου κωδικού

Έχοντας πατήσει την επιλογή "History" θα μεταφερθούμε στην παρακάτω οθόνη (εικόνα 5.8) η οποία μας εμφανίζει την ημερομηνία και ώρα από τη τελευταία είσοδο όπως επίσης και από τη τελευταία λανθασμένη προσπάθεια εισόδου. Βεβαίως, και σε αυτό το σημείο αν δεν υπάρχει σύνδεση ή δεν υπάρχει καταχώρηση σε κάποιο από τα δύο πεδία ενημερώνεται ο χρήστης με μήνυμα. Επιπλέον, έχουμε την επιλογή να επιστρέψουμε στην προηγούμενη οθόνη είτε πατώντας το "back button" είτε πατώντας στο πλήκτρο μενού διαλέγοντας το προορισμό που επιθυμούμε.

Εικόνα 5.8: Ιστορικό χρήστη (History)

5.4.3 Αλλαγή κωδικού πρόσβασης

Αφού έχουμε πραγματοποιήσει την επιλογή "Change Password" θα μεταφερθούμε στην οθόνη όπου μπορούμε να αλλάξουμε το κωδικό πρόσβασης (βλέπε εικόνα 5.9). Στην συνέχεια για να πραγματοποιήσουμε επιτυχής αλλαγή θα πρέπει να συμπληρώσουμε κατάλληλα τα πεδία που μας ζητούνται. Επίσης, είναι απαραίτητο να συμπληρωθούν με την σειρά αφού αν δεν πληκτρολογήσουμε κάποιον αριθμό στο πρώτο πεδίο δεν θα μπορέσουμε να μεταβούμε στο δεύτερο και αντίστοιχα στο τρίτο. Αυτό συμβαίνει για λόγους ασφάλειας και ορθής χρήσης των πεδίων. Φυσικά και σε αυτό το σημείο ενημερώνεται ο χρήστης με κατάλληλα μηνύματα για την επιτυχής αλλαγή ή για να βοηθηθεί ώστε να διορθώσει πιθανόν λάθος.



Εικόνα 5.9: Αλλαγή κωδικού πρόσβασης (Change Password)

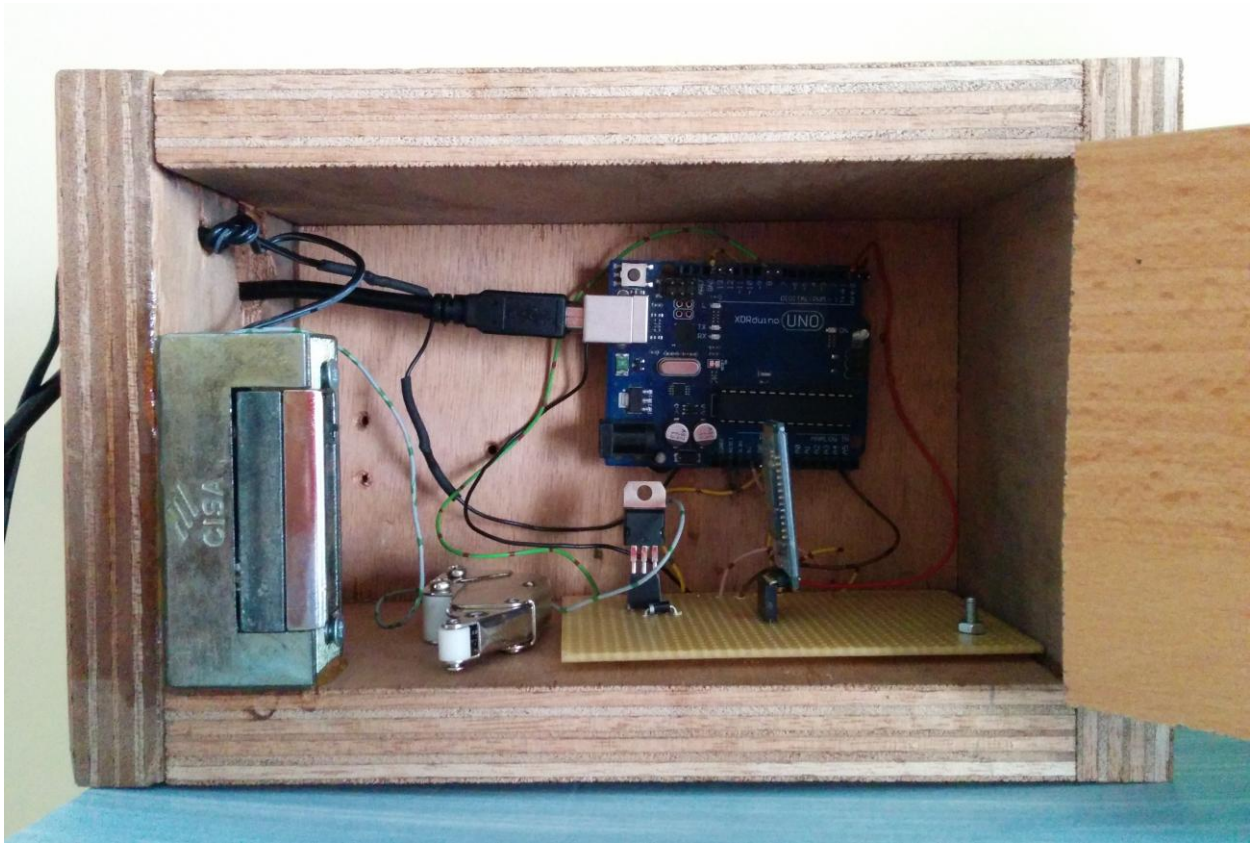
5.4.4 Τερματισμός εφαρμογής

Ένα από τα πιο σημαντικά στοιχεία μιας εφαρμογής είναι ο σωστός τρόπος εξόδου από αυτήν ώστε να μην λειτουργεί στο παρασκήνιο και καταναλώνει πόρους από τη συσκευή. Για να τα το καταφέρουμε αυτό θα πρέπει είτε να πατήσουμε όσες φορές χρειαστεί το πλήκτρο επιστροφής(που υπάρχει στην συσκευή) είτε να μεταφερθούμε στην αρχική σελίδα (Main Page) και έπειτα να πατήσουμε μια φορά το πλήκτρο επιστροφής. Υλοποιώντας μια από τις παραπάνω ενέργειες θα μας εμφανιστεί μήνυμα για την επιβεβαίωση της εξόδους μας. Τέλος πατώντας OK θα τερματίσουμε σωστά την εφαρμογή μας η οποία κλείνει αυτόματα και το Bluetooth της συσκευής μας.

ΚΕΦΑΛΑΙΟ 6

6 Ανάλυση και περιγραφή της λειτουργίας της ηλεκτρικής κλειδαριάς

Σε αυτό το σημείο θα κάνουμε μια συνοπτική περιγραφή χωρίς να χρησιμοποιήσουμε δυσνόητους όρους για τη λειτουργία τόσο της πλατφόρμας Arduino όσο και της εφαρμογής Android που κατασκευάσαμε. Θα αναφέρουμε επίσης τη νοοτροπία την οποία ακολουθήσαμε καθώς και τα σημεία που μας δυσκόλεψαν.



Εικόνα 6.1: Κατασκευή πτυχιακής εργασίας

6.1 Ανάλυση της λειτουργίας της πλατφόρμας Arduino

Εκεί που πρέπει να εστιάσουμε έχοντας ήδη παρατηρήσει το σχεδιάγραμμα (εικόνα 5.3) που βρίσκεται στο προηγούμενο κεφάλαιο είναι στη συνδεσμολογία της κλειδαριάς καθώς και στη χρησιμότητα του τρανζίστορ. Να θυμίσουμε οτι η κλειδαριά χρειάζεται 12V για να λειτουργήσει με αποτέλεσμα να πρέπει να χρησιμοποιήσουμε κάποιο τροφοδοτικό καθώς το Arduino δεν έχει ακροδέκτη που να μας προσφέρει αυτή τη τάση. Έχοντας αυτόν τον περιορισμό σκεφτήκαμε πως πρέπει να χρησιμοποιήσουμε ένα

τρανζίστορ στον αρνητικό πόλο της τροφοδοσίας ώστε να ελέγχουμε τη διέλευση του ρεύματος, δηλαδή σαν να χρησιμοποιούμαι κάποιον διακόπτη. Το πότε θα περνάει ή όχι ρεύμα εξαρτάται από την έξοδο που παίρνουμε μέσω του ακροδέκτη "9" του Arduino. Επιπλέον την κατάσταση αυτού του ακροδέκτη την ελέγχει ο χρήστης καθώς πρέπει να εισάγει σωστό κωδικό στην εφαρμογή ώστε όταν πατήσει είσοδο να μετατρέψει την κατάσταση του από "HIGH"(αρχική κατάσταση) σε "LOW" επιτρέποντας έτσι στην ηλεκτρική κλειδαριά να γειωθεί. Μετά από 2 δευτερόλεπτα βέβαια επιστρέφει στην αρχική κατάσταση. Ας το δούμε τώρα και από τη πλευρά του κώδικα, όπου επικοινωνούμε με το κινητό μέσω της βιβλιοθήκης Amarino. Η αποστολή δεδομένων από το Android στο Arduino είναι εύκολη υπόθεση καθώς στέλνουμε σηματοφόρους δηλαδή τα δεδομένα που θέλουμε μαζί με μια ετικέτα ώστε να τα ξεχωρίσουμε κατά την παραλαβή στο Arduino. Αυτό καθιστά εύκολη και απλή την ανταλλαγή.

6.2 Ανάλυση της λειτουργίας της εφαρμογής Android

Θα αρχίσουμε εξηγώντας την λογική που ακολουθήσαμε για την ανταλλαγή δεδομένων από το Arduino στο Android .Αναφέραμε πιο πάνω ότι η αντίστροφη επικοινωνία ήταν απλή όμως εδώ τα πράγματα περιπλέκονται. Σε αυτό το σημείο δυσκολευτήκαμε καθώς ναι μεν μπορείς να στείλεις μηνύματα τύπου String αλλά χωρίς κάποια ετικέτα μαζί ώστε να τα ξεχωρίζεις όταν τα παραλαμβάνεις στην εφαρμογή Android. Ο λόγος που θέλαμε να στείλουμε δεδομένα κατά αυτή τη κατεύθυνση ήταν για παράδειγμα αφού γίνει έλεγχος στη βάση δεδομένων για το αν υπάρχει ο κωδικός να μας δίνει και απάντηση ώστε να εμφανίζουμε κατάλληλο μήνυμα και να ενημερώνουμε το χρήστη. Για να το πραγματοποιήσουμε αυτό ακολουθήσαμε την νοοτροπία που χρησιμοποιούν σε όλες τις γλώσσες προγραμματισμού τα interrupts(διακοπές), δηλαδή στέλνουμε ακέραιο αριθμό από το Arduino και όταν λαμβάνουμε στο Android τον συγκεκριμένο αριθμό εκτελούμε το αντίστοιχο κομμάτι κώδικα. Αυτό είναι υλοποιήσιμο γιατί γνωρίζουμε από πριν τους αριθμούς που μπορούν να μας έρθουν με αποτέλεσμα να μπορούμε να τα ξεχωρίσουμε και ο καθένας να κάνει τις ενέργειες που θέλουμε. Δηλαδή χρησιμοποιούμαι τους αριθμούς σαν ετικέτες. Με την παραπάνω μέθοδο καταφέρνουμε να ενημερώνουμε και τις 3 διαφορετικές οθόνες διεπαφής της εφαρμογής. Τέλος εκεί που δώσαμε μεγαλύτερη έμφαση είναι στην ασφάλεια και στην αξιοπιστία της εφαρμογής καθώς προσπαθήσαμε να προστατέψουμε το κωδικό αποθηκεύοντας και κάνοντας όλους τους ελέγχους που ζητούνται από το χρήστη στο Arduino.

6.3 Παραδείγματα χρήσης της κατασκευής στη καθημερινότητα

Ενημερώνοντας ή επισκέπτοντας κανείς διάφορες χώρες στο εξωτερικό θα διαπιστώσει ότι έχουν σταματήσει να χρησιμοποιούν τα κοινά κλειδιά για τις εισόδους σε κατοικίες ή και σε επιχειρήσεις αντικαθιστώντας τα με κάρτες rfid ή πλήκτρα εισαγωγής κωδικών. Στην Ελλάδα χρησιμοποιούν αυτό το τρόπο εισόδου κυρίως σε εσωτερικούς χώρους και αποφεύγουν να τα εγκαταστήσουν σε εξωτερικές πόρτες. Όμως αφού η τεχνολογία κάθε μέρα εξελίσσεται και πρέπει να την εκμεταλλευόμαστε θα μπορούσαμε να αντικαταστήσουμε τις παραπάνω μεθόδους ή να συνυπάρξουν (αφού μπορεί να γίνει εγκατάσταση στις υπάρχουσες υποδομές) με την κατασκευή που δημιουργήσαμε. Έτσι το "έξυπνο" κινητό που μέρα με τη μέρα μπαίνει όλο και περισσότερο στη καθημερινότητα μας θα μπορούσαμε να το χρησιμοποιούμαι για να ανοίξουμε ακόμα και την πόρτα του σπιτιού μας. Επίσης είναι δυνατό να το χρησιμοποιήσουμε και μέσα από το σπίτι μας για παράδειγμα αν κάποιος επισκέπτης χτυπήσει το κουδούνι, μας δίνεται η δυνατότητα να μην κουνηθούμε καθόλου και να του ανοίξουμε από τη θέση που βρισκόμαστε καθώς το bluetooth έχει μεγάλη εμβέλεια.

ΠΑΡΑΡΤΗΜΑ

ΚΩΔΙΚΑΣ ARDUINO

```
#include <MeetAndroid.h>
#include <EEPROM.h>

MeetAndroid meetAndroid;
int lock = 9;     //pin 9 on Arduino
int counter=0;    //counts the fault attempts
int counter_old=0;//counts the changing password fault attempts
int user_name=0;  //first table position of each floor
boolean password_status=false;

void setup(){
  pinMode(lock, OUTPUT);
  Serial.begin(9600);
  counter=0;
  meetAndroid.registerFunction(button,'b');
  meetAndroid.registerFunction(history_last_entry,'e');        //for recieving the last entry
to eeprom
  meetAndroid.registerFunction(history_last_false_attempt,'f'); //for recieving the last
false attempt to eeprom
  meetAndroid.registerFunction(history_send,'s');              //for sending information of
history to Android
  meetAndroid.registerFunction(check_old_password,'p');        //for checking the old
password
  meetAndroid.registerFunction(new_password,'n');              //for checking the new
password

    digitalWrite(lock, LOW);  //By default, lock is active(locked)

// DEFAULT PASSWORDS FOR EACH FLOOR
  WritePassToEEPROM(1111,0);
  WritePassToEEPROM(2222,20);
  WritePassToEEPROM(3333,40);
  WritePassToEEPROM(4444,60);
  WritePassToEEPROM(5555,80);
}
void loop(){
  meetAndroid.receive(); //recieve signal form android
  delay(300);
}

void button(byte flag,byte numOfValues){
   int num = meetAndroid.getInt(); //recieve the password that the user gives

  if(checkPass(num)){
    counter=0;
    digitalWrite(lock,HIGH); //unlock the door
    meetAndroid.send(100);  //send message to unlock
```

```
    delay(2000);
    digitalWrite(lock,LOW); //lock the door
    meetAndroid.send(101);  //send message to lock
  }
  else{
    counter++;
    if(counter<3){
      meetAndroid.send(102);//Access denied
    }
    if(counter>=3){
      counter=0;
      meetAndroid.send(103);//send message to start delay
      delay(10000);         //delay 10 sec after 3 attemps
      meetAndroid.send(104);//send message to try again
    }
  }
}
// check the password and give access to user
boolean checkPass(int pass){
  int j;
  boolean test;
  for(j=0;j<=20;j=j+20){// the eeprom space for each floor

    if(pass==ReadPassFromEEPROM(j)){
      user_name=j;
      test=true;
      break;
    }
    else{
      test=false;
    }
  }
  return test;
}

 //check the old password
void check_old_password(byte flag,byte numOfValues){
   int num2 = meetAndroid.getInt();
  if(checkPass(num2)){
    password_status=true;
  }else{
    password_status=false;
  }
}

//recieve and write the new password to eeprom
void  new_password(byte flag,byte numOfValues){
   int new_pass = meetAndroid.getInt();
 if(checkPass(new_pass)){ //if new password already in use then send 304 else write in
eeprom
  meetAndroid.send(304);//send message to choose different new password
```

```
 }else{
  if(password_status){
    counter_old=0;
    WritePassToEEPROM(new_pass,user_name);
    meetAndroid.send(300);//send message for changing password correct
  }else{
    counter_old++;
    if(counter_old<3){
       meetAndroid.send(301);//send message for wrong old password
    }
    if(counter_old>=3){
      counter_old=0;
      meetAndroid.send(302);//send message to start delay
      delay(10000);         //delay 10 sec after 3 attemps
      meetAndroid.send(303);//send message to try again
    }
  }
 }
}
//================================================================
================
void WritePassToEEPROM(unsigned int pass,int i)
{

  EEPROM.write(i,(pass%100));
  EEPROM.write(i+1,(pass/100));
}
//================================================================
================
unsigned int ReadPassFromEEPROM(int i)
{
  return ((EEPROM.read(i+1)*100 + EEPROM.read(i)));
}
//================================================================
================
int counter_history=0;

void history_last_entry(byte flag,byte numOfValues){
  if(counter_history<=6){
     EEPROM.write((user_name+2+counter_history),meetAndroid.getInt());//write to
eeprom the last entry
    //user_name+2 first table element for last entry
    counter_history++;
  }
  if(counter_history==7){//reaload counter_history after put all last entry data
    counter_history=0;
  }
}

int counter_history2=0;
void history_last_false_attempt(byte flag,byte numOfValues){
```

```
  if(counter_history2<=6){
     EEPROM.write((user_name+10+counter_history2),meetAndroid.getInt());//write to
eeprom the last false attempt
    //user_name+10 first table element for last false attempt
    counter_history2++;
  }
  if(counter_history2==7){//reaload counter_history after put all last entry data
    counter_history2=0;
  }
}

//============================================================
===============
int counter_history_send=0;
int counter_history_send2=0;
void history_send(byte flag,byte numOfValues){

     //========LAST ENTRY================
    if(EEPROM.read((user_name+2+counter_history_send))!= 0){ //if eeprom is empty
don't send information
      for(counter_history_send=0;counter_history_send<=6;){
        //user_name+2 first table element for last entry
       meetAndroid.send(EEPROM.read((user_name+2+counter_history_send))); //read
from eeprom the last entry
        counter_history_send++;
      }
      if(counter_history_send==7){
       counter_history_send=0;
      }
    }else{                  //empty last_entry
      meetAndroid.send(200);
    }

    //========LAST FALSE ATTEMPT================
    if(EEPROM.read((user_name+10+counter_history_send2))!= 0){//if eeprom is empty
don't send information
      for(counter_history_send2=0;counter_history_send2<=6;){
        //user_name+2 first table element for last entry
       meetAndroid.send(EEPROM.read((user_name+10+counter_history_send2)));
//read from eeprom the last entry
        counter_history_send2++;
      }
      if(counter_history_send2==7){
       counter_history_send2=0;
      }
    }else{                  //empty last_false_attempt
      meetAndroid.send(201);
    }

}
```

ΚΩΔΙΚΑΣ ANDROID

Main Activity Class

```
package com.example.doorlock;

import android.os.Bundle;
import android.annotation.SuppressLint;
import android.app.Activity;
import android.app.AlertDialog;
import android.bluetooth.BluetoothAdapter;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.IntentFilter;
import android.text.format.Time;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.RelativeLayout;
import android.widget.TextView;
import android.widget.Toast;
import at.abraxas.amarino.Amarino;
import at.abraxas.amarino.AmarinoIntent;

@SuppressLint("NewApi")
public class MainActivity extends Activity{
    private static final String MAC_ADDRESS = "20:13:11:29:04:91";//Arduino's
mac_address
     Button b1; //the button to unlock the door
     TextView textAnswer;
     TextView last_entry_text;
    ////TIME////
    Time today = new Time(Time.getCurrentTimezone());
    int year=0,year1=0,year2=0;
    /////
    TextView false_attempt_text;
    Integer user_password=0;//The password that user inserts
    EditText pass;
    RelativeLayout layout;

    private ArduinoReceiver arduinoReceiver = new ArduinoReceiver();//create object for
recieving


    @Override
    protected void onCreate(Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        layout =(RelativeLayout)findViewById(R.id.relativeLayoutActivity_main);
        layout.setBackgroundResource(R.drawable.close_door2);

        getActionBar().setDisplayHomeAsUpEnabled(false);
        b1=(Button) findViewById(R.id.button1);
        pass= (EditText)findViewById(R.id.pass);
        textAnswer= (TextView)findViewById(R.id.textAnswer);
        initialization();
    }

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

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // TODO Auto-generated method stub
        if(item.getItemId()==(android.R.id.home)){
            setContentView(R.layout.activity_main);
            getActionBar().setDisplayHomeAsUpEnabled(false);

        }
        if(item.getItemId()==(R.id.History)){
            startActivity(new Intent(MainActivity.this,HistoryActivity.class));//calls
HistoryActivity.class ----> view history_layout
            sendData('s',"1");//send flag for history to arduino

        }else if(item.getItemId()==(R.id.Change_Password)){
            startActivity(new Intent(MainActivity.this,ChangePasswordActivity.class));//calls
changepassword--->view change_password_layout

        }else if(item.getItemId()==(R.id.refresh)){//refreshing Bluetooth connection
            Amarino.disconnect(this,MAC_ADDRESS);
            Amarino.connect(this,MAC_ADDRESS);
        }
        return super.onOptionsItemSelected(item);
    }

    private void initialization(){
        b1.setOnClickListener(new OnClickListener(){
            public void onClick(View v)
            {
            if(pass.getText().toString().length()>10){  //check password length
                textAnswer.setText("Max password size 10 numbers");
            }else{
                try{ //check if password text is empty
```

```
                user_password =Integer.parseInt(pass.getText().toString());
            }catch(Exception exc){
                Toast.makeText(MainActivity.this,"INSERT THE
PASSWORD",Toast.LENGTH_LONG).show();
            }
                if(arduinoReceiver.lockReturn_status()==1){ //check if door is open
                    sendData('b',user_password); //send user_password to arduino
                }
            }
        }
    });
  }
  //disable button after 3 false attempts
  public void set_text(String message){
    textAnswer.setText(message);
    if(message.equals("You have to wait 10 sec")){
      b1.setEnabled(false);
    }else{
      b1.setEnabled(true);
    }
  }
  //send string to Arduino using Amarino library
  private void sendData(char flag,String message){
    Amarino.sendDataToArduino(this,MAC_ADDRESS,flag,message);
  }

  //send integer to Arduino using Amarino library
  public void sendData(char flag,int num){
    Amarino.sendDataToArduino(this,MAC_ADDRESS,flag,num);
  }
  protected void onStart(){
    super.onStart();
    // to receive broadcasted intents we must register our receiver
    registerReceiver(arduinoReceiver, new
IntentFilter(AmarinoIntent.ACTION_RECEIVED));
    Amarino.connect(this,MAC_ADDRESS);
  }
  protected void onStop(){
    super.onStop();
    // do not forget to unregister a registered receiver
  }

  protected void onDestroy(){ //We use onDestroy method to keep the connection
between bluetooth and android until the application die
    super.onDestroy();
    //Disable bluetooth, we put admin permission for bluetooth on AndroidManifest.xml
    BluetoothAdapter mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
    if (mBluetoothAdapter.isEnabled()) {
      mBluetoothAdapter.disable();
    }
    // do not forget to unregister a registered receiver
```

```
        unregisterReceiver(arduinoReceiver);
        Amarino.disconnect(this,MAC_ADDRESS);
    }

    @Override
    public void onBackPressed() {//Ask to exit from application when you are at Main
Page
        new AlertDialog.Builder(this)
            .setTitle("Really Exit?")
            .setMessage("Are you sure you want to exit?")
            .setNegativeButton(android.R.string.no, null)
            .setPositiveButton(android.R.string.yes, new DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialog, int which) {
                    finish();//Close application
                }
            }).create().show();
    }

    protected void onResume(){
        super.onResume();
    }

    protected void onPause(){
        super.onPause();
    }

    public class ArduinoReceiver extends BroadcastReceiver {

        private int lock_status=1;//Default lock closed
        private int counter=0;
        private int counter_old=0;
        private int counter_history=0;
        private int counter_history_false=0;
        private boolean status_last_entry=true;
        private boolean status_false_attempt=true;

        @Override
        public void onReceive(Context context, Intent intent) {
            // TODO Auto-generated method stub
            String data = null;
            //final String address =
intent.getStringExtra(AmarinoIntent.EXTRA_DEVICE_ADDRESS); // the type of data
which is added to the intent
            String action = intent.getAction();

            final int dataType = intent.getIntExtra(AmarinoIntent.EXTRA_DATA_TYPE, -1);
// we only expect String data though,
            //but it is better to check if really string was sent
            // later Amarino will support differnt data types, so far data comes always as
string and
```

```
        // you have to parse the data to the type you have sent from Arduino, like it is
shown below

      if(action.equals(AmarinoIntent.ACTION_RECEIVED)){
         if (dataType == AmarinoIntent.STRING_EXTRA){
            data = intent.getStringExtra(AmarinoIntent.EXTRA_DATA);
            if (data != null){
             Integer info =Integer.parseInt(data);//recieve message from arduino
                if(info==100){ //unlock the door
                   counter=0;
                   layout.setBackgroundResource(R.drawable.open_door2);//change
background when door unlocks

                   set_text("***DOOR UNLOCKED***");
                   send_last_entry();//Send last entry time to arduino
                   setLock_status(0); //set status that door is unlocked
                }else if(info==101){ //lock the door
                   layout.setBackgroundResource(R.drawable.close_door2);//change
background when door locks
                   set_text("***DOOR LOCKED***");
                   setLock_status(1); //set status that door is locked
                }else if(info==102){ //Access denied
                   counter++;
                   set_text("ACCESS DENIED \n"+ (3-counter) +" ATTEMPTS LEFT");
                   send_last_false_attempt();
                }else if(info==103){//after 3 attempts 10 sec delay
                   counter=0;
                   set_text("You have to wait 10 sec");
                }else if(info==104){//after 10 sec delay try again
                   counter=0;
                   set_text("***Try again***");
                }else if(info==304){//correct changing password
                   Toast.makeText(MainActivity.this,"Choose different
password!\nPassword already in use!!!",Toast.LENGTH_LONG).show();
                }else if(info==300){//correct changing password
                   Toast.makeText(MainActivity.this,"Password
Changed!!!",Toast.LENGTH_LONG).show();
                }else if(info==301){//wrong old password
                    counter_old++;
                    Toast.makeText(MainActivity.this,"Wrong old password \n"+ (3-
counter_old) +" ATTEMPTS LEFT",Toast.LENGTH_LONG).show();
                    //set_text("ACCESS DENIED \n"+ (3-counter) +" ATTEMPTS LEFT");
                }else if(info==302){//after 3 attempts 10 sec delay
                   counter_old=0;
                   ChangePasswordActivity.save_changes.setEnabled(false);
                   Toast.makeText(MainActivity.this,"You have to wait 10
sec",Toast.LENGTH_LONG).show();
                }else if(info==303){//after 10 sec delay try again
                   counter_old=0;
                   ChangePasswordActivity.save_changes.setEnabled(true);
```

```
                    Toast.makeText(MainActivity.this,"Try
again!!",Toast.LENGTH_LONG).show();

                }else{
                    //=======LAST ENTRY=======
                    if((info!=200)&&(status_last_entry==true)){//print last entry
                        if(counter_history<=6){
                            HistoryActivity.set_last_entry(info);
                            counter_history++;
                            status_false_attempt=false;
                        }

                    }else if(info==200){//don't print last entry, is empty
                        status_last_entry=false;
                    }
                    //=======LAST FALSE ATTEMPT=======
                    if((info!=201)&&(info!=200)&&(status_false_attempt==true)){//print last
false attempt
                        if(counter_history_false<=6){
                            HistoryActivity.set_last_false_attempt(info);
                            counter_history_false++;
                        }
                    }else if(info==201){//don't print last false attempt, is empty
                        counter_history_false=7;
                    }
                    if(counter_history_false==7){//reload the counters after finish last false
attempt
                        counter_history_false=0;
                        counter_history=0;
                        status_last_entry=true;
                        status_false_attempt=true;
                    }
                    if(counter_history==7){//set status because last entry finish
                        status_last_entry=false;
                        status_false_attempt=true;
                    }
                }
            }
        }
    }
    private void send_last_entry(){
        today.setToNow();
        sendData('e',today.monthDay);
        sendData('e',(today.month+1));
        //year///
        year=today.year;
        year1=year%100;
        year2=year/100;
        sendData('e',year1);
        sendData('e',year2);
```

```
            //year////
            sendData('e',today.hour);
            sendData('e',today.minute);
            sendData('e',today.second);
        }

        private void send_last_false_attempt(){
            today.setToNow();
            sendData('f',today.monthDay);
            sendData('f',(today.month+1));
            //year///
            year=today.year;
            year1=year%100;
            year2=year/100;
            sendData('f',year1);
            sendData('f',year2);
            //year////
            sendData('f',today.hour);
            sendData('f',today.minute);
            sendData('f',today.second);
        }
        public int lockReturn_status(){
           return lock_status;
        }
        private void setLock_status(int lock_status){
           this.lock_status=lock_status;
        }
    }
}
```

Change Password Activity Class

```
package com.example.doorlock;

import android.os.Bundle;
import android.app.Activity;
import android.content.Intent;
import android.text.Editable;
import android.text.TextWatcher;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
import at.abraxas.amarino.Amarino;

public class ChangePasswordActivity extends Activity {
    private static final String MAC_ADDRESS = "20:13:11:29:04:91";//Arduino's
mac_address
    static Button save_changes;
    Button clear_all;

    EditText old_pass_text;
    EditText new_pass_text;
    EditText new_confirm_pass_text;

    int old_pass=0;
    int new_pass=0;
    int new_confirm_pass=0;
    MainActivity mainActivity=new MainActivity();

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.change_password_layout);//set change password layout
        getActionBar().setDisplayHomeAsUpEnabled(true);// enable home button at
actionbar
        save_changes=(Button) findViewById(R.id.save_changes);
        clear_all=(Button) findViewById(R.id.clear_all);

        save_changes.setEnabled(false);//unclickable save changes button

        old_pass_text= (EditText)findViewById(R.id.oldPassword);
        new_pass_text= (EditText)findViewById(R.id.newPassword);
        new_confirm_pass_text= (EditText)findViewById(R.id.confirmPassword);

        new_pass_text.setEnabled(false);
        new_confirm_pass_text.setEnabled(false);
```

```
        initialization();
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.change_password, menu);
        return true;
    }
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // TODO Auto-generated method stub
        if(item.getItemId()==(android.R.id.home)){
            onBackPressed();//to go back when we press home button from action bar
        }
        else if(item.getItemId()==(R.id.Main)){
            Intent intent = new Intent(this, MainActivity.class);
            intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);//clear "the path" for
backbutton when we go to main page
            startActivity(intent);//calls MainActivity--->view activity_main_layout
        }
        if(item.getItemId()==(R.id.History)){
            startActivity(new
Intent(ChangePasswordActivity.this,HistoryActivity.class));//calls HistoryActivity.class --
--> view history_layout
            sendData('s',"1");//send flag for history to arduino
        }else if(item.getItemId()==(R.id.refresh)){//refreshing Bluetooth connection
            Amarino.disconnect(this,MAC_ADDRESS);
            Amarino.connect(this,MAC_ADDRESS);
        }
        return super.onOptionsItemSelected(item);
    }

    public void sendData(char flag,int num){//send integer to Arduino using Amarino
library
        Amarino.sendDataToArduino(this,MAC_ADDRESS,flag,num);
    }

    //send string to Arduino using Amarino library
    private void sendData(char flag,String message){
        Amarino.sendDataToArduino(this,MAC_ADDRESS,flag,message);
    }

    private void initialization(){
        save_changes.setOnClickListener(new OnClickListener(){
            public void onClick(View v)
            {
                old_pass=Integer.parseInt(old_pass_text.getText().toString());
                new_pass=Integer.parseInt(new_pass_text.getText().toString());
          new_confirm_pass=Integer.parseInt(new_confirm_pass_text.getText().toString());

                if(new_pass!=new_confirm_pass){
```

```
                Toast.makeText(ChangePasswordActivity.this,"Retype the new
password",Toast.LENGTH_LONG).show();
            }else{
                sendData('p',old_pass);//send old password to arduino
                sendData('n',new_pass);//send new password to arduino
            }
        }
    });

    clear_all.setOnClickListener(new OnClickListener(){
        public void onClick(View v)
        {
            old_pass_text.setText("");
            new_pass_text.setText("");
            new_confirm_pass_text.setText("");
        }
    });

    old_pass_text.addTextChangedListener(new TextWatcher(){
        @Override
        public void afterTextChanged(Editable arg0) {
            // TODO Auto-generated method stub
            if(old_pass_text.getText().toString().equals("")){//check if old_pass_text is
empty
                save_changes.setEnabled(false);
                new_pass_text.setEnabled(false);
                new_confirm_pass_text.setEnabled(false);
            }else{
                new_pass_text.setEnabled(true);//enable new_pass_text if old_pass_text
is not empty
                if(!(new_confirm_pass_text.getText().toString().equals(""))){//check if
new_confrim_pass_text is NOT empty
                    new_confirm_pass_text.setEnabled(true);
                    save_changes.setEnabled(true);
                }
            }
        }
        @Override
        public void beforeTextChanged(CharSequence arg0, int arg1,
                int arg2, int arg3) {
            // TODO Auto-generated method stub
            save_changes.setEnabled(false);//disable save_changes button by default
        }
        @Override
        public void onTextChanged(CharSequence arg0, int arg1, int arg2,
                int arg3) {
            // TODO Auto-generated method stub
        }
    });
    new_pass_text.addTextChangedListener(new TextWatcher(){
        @Override
```

```
        public void afterTextChanged(Editable arg0) {
            // TODO Auto-generated method stub
            if(new_pass_text.getText().toString().equals("")){
                save_changes.setEnabled(false);
                new_confirm_pass_text.setEnabled(false);
            }else{
                if(!(new_confirm_pass_text.getText().toString().equals(""))){
                    save_changes.setEnabled(true);
                }
                new_confirm_pass_text.setEnabled(true);
            }
        }
        @Override
        public void beforeTextChanged(CharSequence arg0, int arg1,
                int arg2, int arg3) {
            // TODO Auto-generated method stub
            save_changes.setEnabled(false);
        }
        @Override
        public void onTextChanged(CharSequence arg0, int arg1, int arg2,
                int arg3) {
            // TODO Auto-generated method stub
        }
    });

    new_confirm_pass_text.addTextChangedListener(new TextWatcher(){
        @Override
        public void afterTextChanged(Editable arg0) {
            // TODO Auto-generated method stub
            if(new_confirm_pass_text.getText().toString().equals("")){
                save_changes.setEnabled(false);
            }else{//after all text is filled enable save changes button
                save_changes.setEnabled(true);
            }
        }
        @Override
        public void beforeTextChanged(CharSequence arg0, int arg1,
                int arg2, int arg3) {
            // TODO Auto-generated method stub
            save_changes.setEnabled(false);
        }
        @Override
        public void onTextChanged(CharSequence s, int start, int before,
                int count) {
            // TODO Auto-generated method stub

        }
    });
  }
}
```

History Activity Class

```
package com.example.doorlock;

import java.text.DecimalFormat;

import android.os.Bundle;
import android.app.Activity;
import android.content.Intent;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.TextView;

public class HistoryActivity extends Activity {

    static TextView last_entry_text;
    static TextView last_false_attempt;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.history_layout);
        getActionBar().setDisplayHomeAsUpEnabled(true);
        last_entry_text= (TextView)findViewById(R.id.textView2);
        last_false_attempt= (TextView)findViewById(R.id.textView4);
    }

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

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // TODO Auto-generated method stub
        if(item.getItemId()==(android.R.id.home)){
            onBackPressed();//go back when we press home button from action bar
        }
        else if(item.getItemId()==(R.id.Main)){
            Intent intent = new Intent(this, MainActivity.class);
            intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);//clear "the path" for
backbutton when we go to main page
            startActivity(intent);//calls MainActivity--->view activity_main_layout
        }
        else if(item.getItemId()==(R.id.Change_Password)){
            startActivity(new
Intent(HistoryActivity.this,ChangePasswordActivity.class));//calls
ChangePasswordActivity--->view change_password_layout
        }
```

```
        return super.onOptionsItemSelected(item);
    }

    static int i=0;
    static int history[] =new int [7];

    public static void set_last_entry(int message){//change last_entry text
        history[i]=message;
        i++;
        if(i==7){
            DecimalFormat df = new DecimalFormat("00");//to show two digits for
minute/second

last_entry_text.setText(history[0]+"/"+history[1]+"/"+((history[3]*100)+history[2])+" at
"+history[4]+":"+df.format(history[5])+":"+df.format(history[6]));
            i=0;
        }
    }

    static int j=0;
    static int history_false[] =new int [7];

    public static void set_last_false_attempt(int message){//change last_false_attempt
text
        history_false[j]=message;
        j++;
        if(j==7){
            DecimalFormat df = new DecimalFormat("00");//to show two digits for
minute/second

last_false_attempt.setText(history_false[0]+"/"+history_false[1]+"/"+((history_false[3]*1
00)+history_false[2])+" at
"+history_false[4]+":"+df.format(history_false[5])+":"+df.format(history_false[6]));
            j=0;
        }
    }
}
```

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Arduino Official. [Online]. www.arduino.cc/en/

[2] Arduino Hardware. [Online]. http://arduino.cc/en/Main/Hardware

[3] Atmel ATmega328. [Online]. http://www.atmel.com/devices/atmega328.aspx

[4] Arduino Versions - IDE. [Online]. http://deltahacker.gr/2009/08/01/arduino-intro

[5] Arduino Hardware.[Online]. http://www.microplanet.gr/tutorials/microcontrollers/arduino

[6] Arduino Windows.[Online]. http://arduino.cc/en/Guide/Windows

[7] Arduino IDE.[Online]. http://arduino.cc/en/Guide/Environment?from=Tutorial.Bootloader

[8] Arduino Language Reference. [Online]. http://arduino.cc/en/Reference/HomePage

[9] Arduino Uno. [Online]. http://arduino.cc/en/Main/ArduinoBoardUno

[10] Arduino Memory. [Online]. http://arduino.cc/en/Tutorial/Memory

[11] Android Official. [Online]. http://el.wikipedia.org/wiki/Android

[12] Android Basics. [Online]. http://developer.android.com/about/dashboards/index.html?utm_source=ausdroid.net

[13] Android Basics. [Online]. http://developer.android.com/guide/basics/what-is-android.html

[14] Android Basics. [Online]. http://www.adds.gr /company/technology/what-is-android/

[15] Android Basics. [Online]. http://developer.android.com/guide/basics/what-is-android.html

[16] Android Versions. [Online]. http://en.wikipedia.org/wiki/Android_version_history

[17] Android Architecture. [Online]. http://www.tutorialspoint.com/android/android_architecture.ht

[18] Amarino Basics. [Online]. http://www.amarino-toolkit.net

[19] Google Code - host for Amarino sources, http://code.google.com/p/amarino/

[20] Google groups - discussion forum for the Amarino software toolkit,

http://groups.google.com/group/amarino-toolkit

[21] Meier R., Professional Android 2 Application Development, Wiley Publishing, Ed. 2010.