ΤΕΙ ΠΕΙΡΑΙΑ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΗΛΕΚΤΡΟΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Σύστημα Android – Arduino για τον προγραμματισμό λειτουργίας των πριζών οικίας

Δημήτρης Σαντοριναίος

Εισηγητής: Ιωάννης Έλληνας, Καθηγητής

ΑΙΓΑΛΕΩ
ΜΑΙΟΣ 2014

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Σύστημα Android – Arduino για τον προγραμματισμό λειτουργίας των πριζών οικίας

Δημήτρης Σαντοριναίος

Α.Μ. 38028

Εισηγητής:
Ιωάννης Έλληνας, Καθηγητής

Εξεταστική Επιτροπή:

Ημερομηνία εξέτασης:

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον εισηγητή της πτυχιακής μου εργασίας κ. Ιωάννη Έλληνα για την βοήθεια και τις συμβουλές που έδωσε για την ολοκλήρωση αυτής της εργασίας.
Επίσης θα ήθελα να ευχαριστήσω την οικογένεια μου που με βοήθησε στην προσπάθεια μου.

Περίληψη

Η παρούσα πτυχιακή εργασία αποτελείται από μια εφαρμογή για κινητό τηλέφωνο Android και την κατασκευή και προγραμματισμό μιας συσκευής που βασίζεται στο Arduino Uno για τον έλεγχο πριζών οικίας.

Γίνεται περιγραφή και ανάλυση των βασικών χαρακτηριστικών του λειτουργικού συστήματος Android, της πλακέτας Arduino Uno και του περιβάλλοντος ανάπτυξης για τον προγραμματισμό της. Ανάλυση του περιβάλλοντος ανάπτυξης Android εφαρμογών Eclipse. Επίσης γίνεται ανάλυση του Amarino toolkit απαραίτητο για την λειτουργία της εφαρμογής και της συσκευής.

Η Android εφαρμογή είναι το μέσο ελέγχου της συσκευής με την πρίζα. Παρουσιάζονται οι λειτουργίες της εφαρμογής αλλά και ο κώδικας που εκτελείται. Η εφαρμογή δίνει την δυνατότητα στον χρήστη να κάνει αναζήτηση συσκευών Bluetooth και να συνδεθεί με μια. Υπάρχουν πολλαπλοί τρόποι ελέγχου της πρίζας στην συσκευή αλλά και ενημέρωση για την κατάσταση και τους ελέγχους που εκτελεί η συσκευή.

Εκτός από την Android εφαρμογή παρουσιάζεται και η συσκευή για τον έλεγχο της πρίζας, τα εξαρτήματα που έχουν χρησιμοποιηθεί και το πρόγραμμα που έχουμε φορτώσει στο Arduino.

Περιεχόμενα

1. Εισαγωγή 9
1.1 Περιγραφή του αντικειμένου της πτυχιακής εργασίας 9
1.2 Δομή της εργασίας 9

2. Android 11
2.1 Τι είναι το Android 11
2.2 Αρχιτεκτονική Android 11
2.2.1 Πυρήνας Linux (Linux Kernel) 12
2.2.2 Βιβλιοθήκες (Libraries) 12
2.2.3 Χρόνος εκτέλεσης (Android Runtime) 12
2.2.4 Πλαίσιο Εφαρμογής (Application Framework) 12
2.2.5 Εφαρμογές (Applications) 13
2.3 Συνιστώσες Εφαρμογής (Application Components) 13
2.3.1 Δραστηριότητα (Activity) 13
2.3.2 Υπηρεσία (Service) 15
2.3.3 Πάροχος Περιεχομένου (Content Provider) 16
2.3.4 Δέκτης Εκπομπών (Broadcast Receiver) 16
2.4 Προθέσεις (Intent) 16
2.5 Application Manifest 17
2.6 Πόροι Εφαρμογής (Application Resources) 18

3. Arduino 19
3.1 Τι είναι το Arduino 19
3.2 Χαρακτηριστικά του Arduino 19
3.3 Η πλακέτα Arduino 20
3.4 Arduino IDE 22
3.4.1 Βιβλιοθήκες (Libraries) 24

4. Amarino toolkit 27
4.1 Τι είναι το Amarino toolkit 27
4.2 Η Android εφαρμογή Amarino 27
4.3 Η βιβλιοθήκη MeetAndroid 29
4.4 Amarino API 30

5. Eclipse IDE 35
5.1 Δημιουργία Android project 35
5.2 Δημιουργία δραστηριότητας 36
5.3 Δημιουργία γραφικού περιβάλλοντος 37
5.4 Έλεγχος της εφαρμογής 39
5.5 Δημοσίευση της εφαρμογής 43

6. Η Android εφαρμογή BluePuc 47
6.1 Εγκατάσταση της εφαρμογής 47
6.2 Αναζήτηση συσκευών και σύνδεση 48
6.3 Έλεγχος της συσκευής 49

6.4 Ρυθμίσεις....52
6.5 Ανάλυση λειτουργίας και σχεδιασμού της εφαρμογής....52
6.5.1 Layouts....53
6.5.2 Δραστηριότητα Αρχικής οθόνης....55
6.5.3 Δραστηριότητα οθόνης ελέγχου....60
6.5.4 Δραστηριότητα οθόνης ρυθμίσεων....65

7. Υλοποίηση και προγραμματισμός συσκευής....71
7.1 Το σχηματικό του κυκλώματος....71
7.2 Το σχεδιάγραμμα για το Arduino....74
7.2.1 Αρχικοποιήσεις....74
7.2.2 Η συνάρτηση setup()....75
7.2.3 Συναρτήσεις λήψης δεδομένων από το κινητό....77
7.2.4 Η συνάρτηση loop()....77
7.3 Κατασκευή της συσκευής....91
7.4 Περιπτώσεις χρήσης....93

Βιβλιογραφία....95

Κεφάλαιο 1

Εισαγωγή

Τα κινητά τηλέφωνα αποκτούν ολοένα και περισσότερες δυνατότητες επιτρέποντας στον χρήστη να κάνει σχεδόν ότι θα έκανε και σε έναν προσωπικό υπολογιστή. Μπορεί να έχει πρόσβαση στο Διαδίκτυο, να αποθηκεύει, να επεξεργάζεται και να μοιράζεται αρχεία, ακόμα και να ελέγχει μια συσκευή όπως την τηλεόραση. Έτσι λόγω του μικρού μεγέθους και των δυνατοτήτων τους χρησιμοποιούνται περισσότερο στην καθημερινότητας μας.

1.1 Περιγραφή του αντικειμένου της πτυχιακής εργασίας

Αντικείμενο της παρούσας πτυχιακής εργασίας είναι η ανάπτυξη μιας εφαρμογής για κινητό τηλέφωνο Android, η κατασκευή μιας συσκευής με πρίζα, που βασίζεται στην πλακέτα Arduino Uno και ο προγραμματισμός της.

Ο χρήστης έχει την δυνατότητα να ελέγξει την πρίζα στην συσκευή από την εφαρμογή στο κινητό τηλέφωνο. Για να γίνει ο έλεγχος της, πρέπει να υπάρχει σύνδεση μέσω Bluetooth. Έτσι ο χρήστης μπορεί από το κινητό να κάνει αναζήτηση συσκευών Bluetooth και να συνδεθεί με εκείνη που θέλει να ελέγχει. Με το πάτημα ενός κουμπιού μπορεί να ελέγξει την πρίζα, μπορεί όμως και να επιλέξει να γίνονται έλεγχοι από την συσκευή για τον έλεγχο της πρίζας αυτόματα. Οι έλεγχοι που μπορεί να επιλέξει ο χρήστης είναι: θερμοκρασίας, φωτεινότητας, χρονοδιακόπτης και χρονική περίοδος. Μπορεί να επιλέξει να γίνεται ένας ή και περισσότεροι έλεγχοι ταυτόχρονα και να ρυθμίσεις τις τιμές τους. Εκτός από τον έλεγχο της συσκευής ο χρήστης μπορεί να βλέπει την κατάσταση της, αν περνάει ρεύμα από την πρίζα και τις μετρήσεις από τα δυο αισθητήρια.

Για να μπορέσει ο χρήστης να ελέγξει την πρίζα από το κινητό χρειάζεται και η συσκευή που αποτελείται από μια πλακέτα Arduino Uno, μια συσκευή Bluetooth, μια οθόνη LCD και το κύκλωμα για τον έλεγχο της πρίζας. Στο Arduino θα πρέπει να φορτωθεί το πρόγραμμα που θα επιτρέπει την επικοινωνία με το κινητό για τον έλεγχο της πρίζας και να συνδεθούν όλα τα εξαρτήματα που θα χρειαστούν. Η συσκευή Bluetooth επιτρέπει στο Arduino να συνδεθεί με το κινητό τηλέφωνο. Η LCD οθόνη χρησιμοποιείται για την εμφάνιση κάποιων πληροφοριών απευθείας, χωρίς να χρειάζεται η χρήση της εφαρμογής στο κινητό τηλέφωνο. Το κύκλωμα για τον έλεγχο της πρίζας επιτρέπει να περνάει ή όχι ρεύμα από την πρίζα, έτσι μπορεί ο χρήστης να συνδέσει μια ηλεκτρική συσκευή που θέλει να ελέγχει όπως μια λάμπα, έναν ανεμιστήρα κλπ.

1.2 Δομή της εργασίας

Στην εργασία αυτή αρχικά γίνεται παρουσίαση του λειτουργικού συστήματος που πρέπει να έχει το κινητό τηλέφωνο, της πλακέτας που θα χρησιμοποιήσουμε αλλά και τον τρόπο προγραμματισμού της, του Amarino

toolkit και του περιβάλλοντος ανάπτυξης εφαρμογών Eclipse. Μετά γίνεται παρουσίαση της εφαρμογής για το κινητό τηλέφωνο και η υλοποίηση και ο προγραμματισμός της συσκευής για τον έλεγχο της πρίζας. Ποιο συγκεκριμένα στο κεφάλαιο 2 γίνεται ανάλυση του λειτουργικού συστήματος Android, της αρχιτεκτονικής του και των συνιστωσών μιας εφαρμογής. Στο κεφάλαιο 3 γίνεται ανάλυση του Arduino, της πλακέτας που θα χρησιμοποιήσουμε για την εργασία αυτή, του περιβάλλοντος ανάπτυξης και τρόπου προγραμματισμού της. Στο κεφάλαιο 4 γίνεται ανάλυση του Amarino toolkit, το οποίο αποτελείται από μια Android εφαρμογή και μια βιβλιοθήκη για το Arduino. Το Amarino είναι απαραίτητο για την ανταλλαγή δεδομένων μέσω Bluetooth ανάμεσα στο κινητό Android και το Arduino. Στο κεφάλαιο 5 γίνεται παρουσίαση του Eclipse, του προγράμματος που θα χρησιμοποιήσουμε για την δημιουργία και τον έλεγχο της Android εφαρμογής. Στο κεφάλαιο 6 γίνεται παρουσίαση της Android εφαρμογής BluePuc για τον έλεγχο της πρίζας, ανάλυση των λειτουργιών και του κώδικα της. Στο κεφάλαιο 7 γίνεται παρουσίαση της συσκευής που πρέπει να φτιάξουμε για τον έλεγχο της πρίζας και του προγράμματος που πρέπει να φορτώσουμε στο Arduino για να μπορέσει η συσκευή να λειτουργεί.

Κεφάλαιο 2

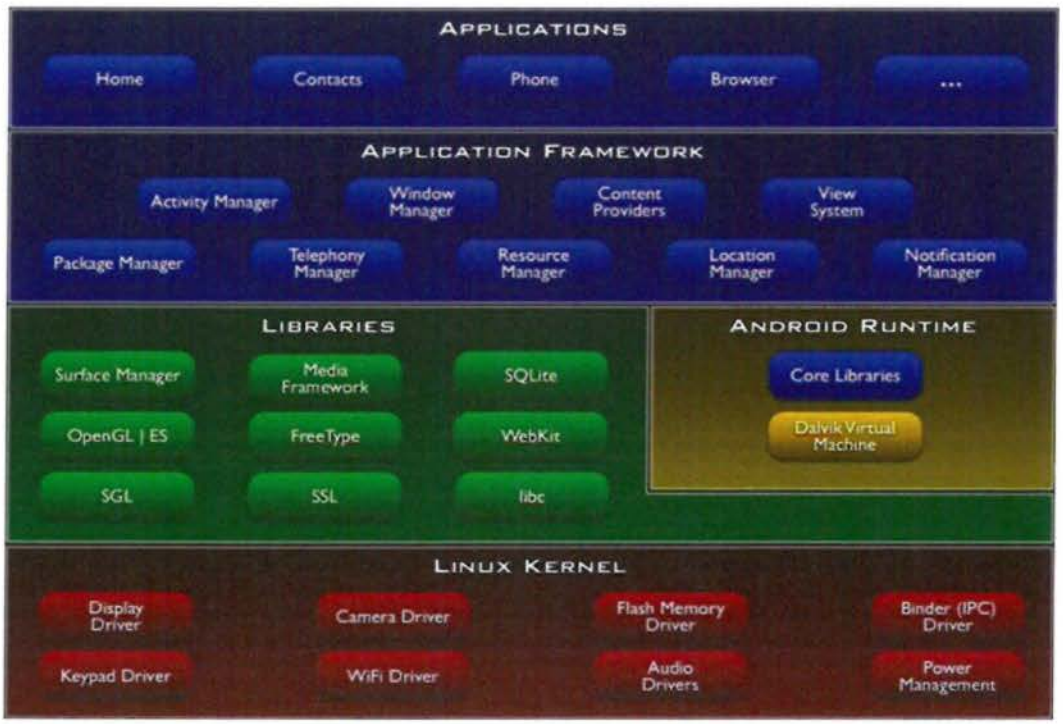
Android

2.1 Τι είναι το Android

Το Android είναι ένα λειτουργικό σύστημα σχεδιασμένο για φορητές συσκευές όπως smartphones και tablets, είναι βασισμένο στο Linux και όπως το Linux είναι ανοιχτού κώδικα. Αρχικά αναπτύχθηκε από την Android, Inc μια μικρή εταιρία την οποία η Google εξαγόρασε το 2005 με σκοπό την είσοδο της στην αγορά κινητής τηλεφωνίας. Η πρώτη παρουσίαση της πλατφόρμας Android έγινε στις 5 Νοεμβρίου 2007 μαζί με την ανακοίνωση της ίδρυσης του οργανισμού Open Handset Alliance, μιας κοινοπραξίας τηλεπικοινωνιακών εταιριών, εταιριών λογισμικού και κατασκευής υλικού. Το πρώτο κινητό τηλέφωνο με Android έγινε διαθέσιμο στις 22 Οκτωβρίου 2008, το HTC-Dream γνωστό και σαν T-Mobile G1.

2.2 Αρχιτεκτονική Android

Το Android είναι μια στοίβα λογισμικού που αποτελείτε από 4 επίπεδα όπως φαίνετε στο σχήμα 2.1. Το κάθε επίπεδο αποτελείτε από διάφορα προγράμματα για διαφορετικές λειτουργίες του λειτουργικού συστήματος. Ξεκινώντας από το πάτο της στοίβας έχουμε το πυρήνα Linux , τις βιβλιοθήκες, το χρόνο εκτέλεσης, το πλαίσιο εφαρμογής και στην κορυφή τις εφαρμογές.



Σχήμα 2.1: Η Αρχιτεκτονική του Android

2.2.1 Πυρήνας Linux (Linux Kernel)

Η Google για την δημιουργία του πυρήνα Android βασίστηκε στο Linux 2.6 (για νεότερες εκδόσεις του Android όπως το 4.4 KitKat βασίστηκε στο Linux 3.4 ή νεότερες εκδόσεις) κάνοντας μερικές αλλαγές. Στο επίπεδο αυτό έχουμε βασικές λειτουργίες συστήματος όπως διαχείριση μνήμης, διαχείριση διεργασιών, την δικτύωση, διαχείριση υλικού (οδηγούς συσκευών - drivers).
Ο Οδηγός συσκευής είναι το πρόγραμμα που επιτρέπει τον έλεγχο της συσκευής, για παράδειγμα την κάμερα του κινητού τηλεφώνου.

2.2.2 Βιβλιοθήκες (Libraries)

Πάνω από το επίπεδο του πυρήνα έχουμε τις βιβλιοθήκες, οι οποίες είναι γραμμένες σε γλώσσα προγραμματισμού C/C++ και μεταγλωττίστηκαν για την αρχιτεκτονική του υλικού που χρησιμοποιείτε από τον τηλέφωνο. Οι βιβλιοθήκες δεν είναι εφαρμογές που μπορούν να σταθούν από μόνες τους αλλά μια ομάδα οδηγιών για το πώς μια συσκευή θα διαχειριστεί διαφορετικά είδη δεδομένων και μπορούν να κληθούν από προγράμματα υψηλότερου επιπέδου. Μερικές από αυτές είναι η βιβλιοθήκη Media Framework η οποία υποστηρίζει αναπαραγωγή και εγγραφή διαφόρων τύπων ήχου, εικόνας και βίντεο, η βιβλιοθήκη SQLite για αποθήκευση πληροφοριών και άλλες.

2.2.3 Χρόνος εκτέλεσης (Android Runtime)

Στο ίδιο επίπεδο με τις βιβλιοθήκες υπάρχει και ο χρόνος εκτέλεσης Android. Εδώ υπάρχουν βασικές βιβλιοθήκες Java και η εικονική μηχανή(VM – Virtual Machine) Dalvik. Η Dalvik είναι μια εικονική μηχανή παρόμοια με την Java VM ειδικά σχεδιασμένη και βελτιστοποιημένη για το Android. Η Dalvik δεν εκτελεί .class αρχεία της Java αλλά .dex αρχεία (Dalvik Executable Files) βελτιστοποιημένα και ποιο συμπαγή αρχεία. Η Dalvik VM είναι σχεδιασμένη για χαμηλές απαιτήσεις μνήμης και δίνει την δυνατότητα σε κάθε εφαρμογή να εκτελείτε στην δικιά του εικονική μηχανή. Αυτό έχει το πλεονέκτημα ότι η κάθε εφαρμογή είναι ανεξάρτητη από τις άλλες, σε περίπτωση που μια εφαρμογή παύει να εκτελείτε (crash) τότε δεν επηρεάζει τις υπόλοιπες που εκτελούνται και απλουστεύει την διαχείριση μνήμης.
Οι βασικές βιβλιοθήκες (υποσύνολο βιβλιοθηκών από την Java Standard Edition και Java Mobile Edition και συγκεκριμένες για το Android) δίνουν την δυνατότητα στους προγραμματιστές να δημιουργήσουν εφαρμογές γραμμένες στη γλώσσα προγραμματισμού Java οι οποίες μεταγλωττίζονται σε .class αρχεία και ακολούθως σε .dex

2.2.4 Πλαίσιο Εφαρμογής (Application Framework)

Το πλαίσιο εφαρμογής βρίσκεται πάνω από το επίπεδο των βιβλιοθηκών και του χρόνου εκτέλεσης Android και δίνει την δυνατότητα στους προγραμματιστές να δημιουργήσουν τις δικές του εφαρμογές, πρόσβαση στο υλικό του κινητού μέσω των κλάσεων που δίνονται, όπως και την δυνατότητα να προσθέσουν δικά τους API's (Application Programming Interface – Διεπαφή Προγραμματισμού Εφαρμογής). Στο επίπεδο αυτό υπάρχουν

διαχειριστές υπεύθυνοι για το κύκλο ζωής μιας εφαρμογής, πρόσβαση σε πόρους όπως εικονίδια, αρχεία xml κλπ, διαχείριση ενημερώσεων και άλλες παρόμοιες υπηρεσίες.

2.2.5 Εφαρμογές (Applications)

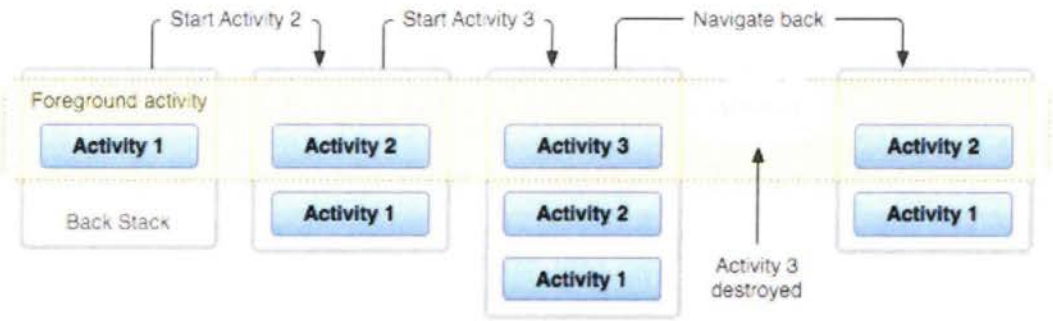
Στην κορυφή της στοίβας βρίσκετε το επίπεδο των εφαρμογών. Εδώ βρίσκονται όλες οι εφαρμογές που είναι προ-εγκατεστημένες στο κινητό, εφαρμογές που μπορεί να κατεβάσει ο χρήστης από το Google Play Store ή από αλλού και εκείνες που μπορεί να δημιουργήσει ο ίδιος (εάν είναι και προγραμματιστής). Σε κάθε κινητό είναι εγκατεστημένες εφαρμογές για τις βασικές λειτουργίες του κινητού όπως κλήσεις, επαφές, μηνύματα, ημερολόγιο, πρόγραμμα περιήγησης και άλλες.

2.3 Συνιστώσες Εφαρμογής (Application Components)

Μια εφαρμογή Android είναι μια δέσμη συνιστωσών εφαρμογής, δεδομένων και πόρων πακεταρισμένα σε ένα Android πακέτο (αρχείο .apk). Υπάρχουν τέσσερις συνιστώσες η κάθε μια με ξεχωριστό σκοπό και το δικό της κύκλο ζωής. Μια συνιστώσα μπορεί να εξαρτάται από κάποια άλλη, όμως η κάθε μια είναι ένα διαφορετικό δομικό μέρος της εφαρμογής. Οι 4 συνιστώσες είναι οι Δραστηριότητες, οι Υπηρεσίες, οι Πάροχοι Περιεχομένου και οι Δέκτες Εκπομπών.

2.3.1 Δραστηριότητα (Activity)

Η ποιο σημαντική συνιστώσα μιας εφαρμογής είναι η δραστηριότητα. Η δραστηριότητα προσφέρει ένα γραφικό περιβάλλον (UI – User Interface) για αλληλεπίδραση με το χρήστη π.χ να κάνει μια κλήση, να βγάλει μια φωτογραφία, να γράψει ένα κείμενο κλπ. Σε κάθε δραστηριότητα αντιστοιχεί ένα παράθυρο με τα στοιχεία του UI του, όπως ένα κουμπί, Checkbox κλπ. Το παράθυρο αυτό συνήθως καλύπτει ολόκληρη την οθόνη, υπάρχουν όμως περιπτώσεις που μπορεί να καλύπτει μέρος της οθόνης.
Μια εφαρμογή αποτελείται από μια ή περισσότερες δραστηριότητες που έχουν χαλαρή δέσμευση μεταξύ τους. Μια δραστηριότητα της εφαρμογής χαρακτηρίζεται σαν *main*, είναι δηλαδή εκείνη που θα εμφανιστεί πρώτη όταν ξεκινήσει για πρώτη φορά η εφαρμογή. Κάθε δραστηριότητα μπορεί να ξεκινήσει μια άλλη δραστηριότητα για την εξυπηρέτηση διαφορετικού σκοπού. Μια δραστηριότητα ξεκινά μια άλλη που ανήκει στην ίδια εφαρμογή συνήθως, όμως μπορεί να ξεκινήσει και δραστηριότητα από κάποια άλλη εφαρμογή αν εκείνη το επιτρέπει. Όταν μια δραστηριότητα ξεκινάει θα εμφανιστεί στην οθόνη δίνοντας την δυνατότητα αλληλεπίδρασης ενώ η προηγούμενη σταματάει και διατηρείται σε μια στοίβα - *back stack*. Ομοίως κάθε νέα δραστηριότητα που ξεκινάει θα βρίσκεται στην κορυφή της στοίβας - προσκήνιο. Η *back stack* χρησιμοποιεί το μηχανισμό *LIFO - Last in First Out*, οπότε αν ο χρήστης πατήσει το Πίσω κουμπί τότε η δραστηριότητα που εμφανίζεται στην οθόνη αφαιρείται από την κορυφή της στοίβας και καταστρέφεται, η προηγούμενη δραστηριότητα παίρνει τώρα την θέση στην κορυφή και ξεκινάει πάλι.

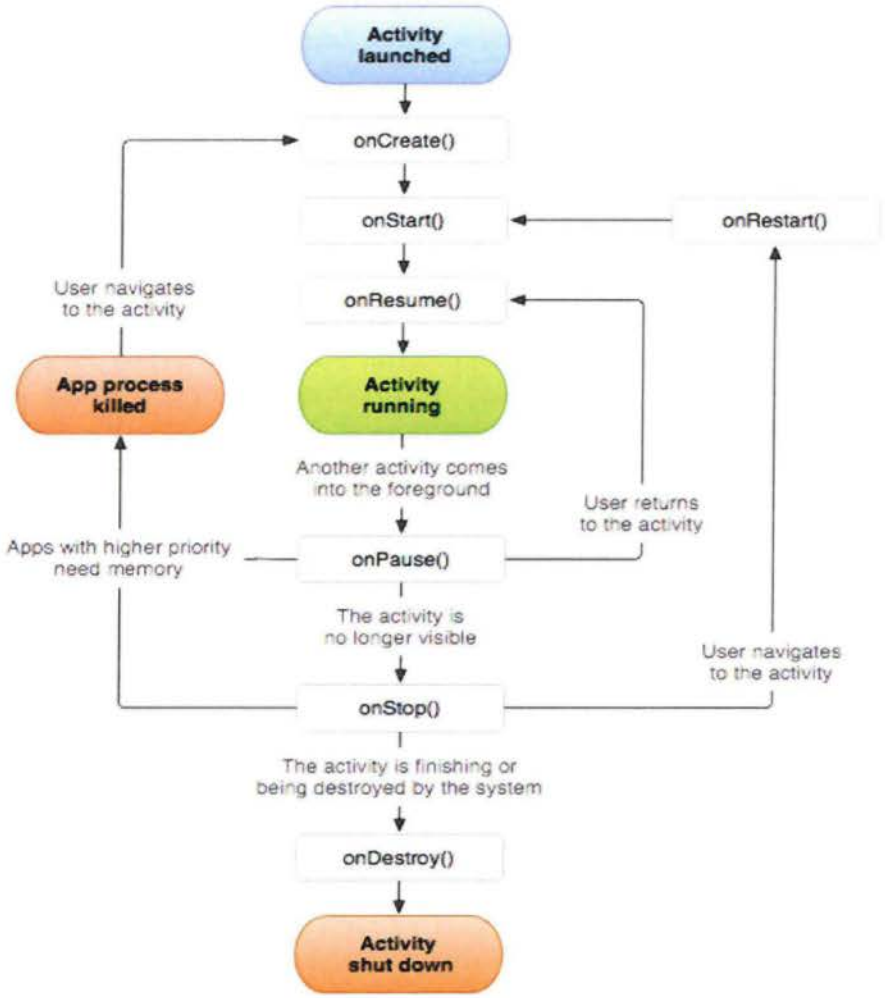


Σχήμα 2.2: Ο τρόπος λειτουργίας της back stack

Η δραστηριότητα μπορεί να είναι σε μια από τις τρεις καταστάσεις *active-resumed, paused* ή *stopped*. Όπως είδαμε και στον τρόπο λειτουργίας της *back stack* μονάχα μια δραστηριότητα μπορεί να είναι *active,* να εμφανίζετε στην οθόνη και με την οποία ο χρήστης αλληλεπιδρά. Μια δραστηριότητα είναι *paused* όταν μια άλλη δραστηριότητα εμφανίζετε στην οθόνη και είναι μερικώς διάφανη ή δεν καλύπτει ολόκληρη την οθόνη. Μια *paused* δραστηριότητα είναι ακόμα στην μνήμη και μπορεί να επιστρέψει σε κατάσταση *active* ή σε κατάσταση *stopped*. Σε ακραίες περιπτώσεις χαμηλής μνήμης η δραστηριότητα μπορεί να διαγραφτεί από το σύστημα. Μια δραστηριότητα είναι σε κατάσταση *stopped* όταν δεν εμφανίζεται ποια στην οθόνη, είναι ακόμα στην μνήμη αν υπάρχει χώρος, όμως το σύστημα μπορεί να την διαγράψει αν χρειαστεί (περιπτώσεις multitasking).

Στο σχήμα 2.3 φαίνετε ο κύκλος ζωής μιας δραστηριότητας. Όταν μια δραστηριότητα ξεκινάει η πρώτη μέθοδος που καλείται είναι η *onCreate*. Εδώ γίνονται όλες οι αρχικοποιήσεις, όπως δημιουργία οθονών, αρχικοποίηση μεταβλητών κλπ. Η αμέσως επόμενη μέθοδος που καλείται είναι η *onStart* και καλείται λίγο πριν γίνει ορατή στον χρήστη. Η επόμενη μέθοδος που καλείται είναι η *onResume,* λίγο πριν η δραστηριότητα γίνει ορατή δίνοντας την δυνατότητα για αλληλεπίδραση. Στο σημείο αυτό η δραστηριότητα βρίσκεται στην κορυφή της στοίβας-*back stack,* είναι στο προσκήνιο και ο χρήστης μπορεί να αλληλεπιδράσει με τη δραστηριότητα.
Η κλήση της μεθόδου *onPause* γίνεται όταν μια δραστηριότητα είναι ενεργή και κάποια άλλη θα εμφανιστεί στην οθόνη, θα γίνει δηλαδή ενεργή. Αυτή η μέθοδος χρησιμοποιείται για να αποθηκεύσουμε πληροφορίες ή δεδομένα όπως π.χ η πρόοδος που έχει κάνει ο χρήστης σε ένα παιχνίδι, να σταματήσουν animations και οτιδήποτε απαιτεί χρήση της CPU (Central Processing Unit – Κεντρική Μονάδα Επεξεργασίας). Ότι εντολές βρίσκονται μέσα στην μέθοδο αυτή πρέπει να ολοκληρωθούν σύντομα για να μην δημιουργήσουν ανεπιθύμητες καθυστερήσεις - *lag* κατά την μετάβαση από τη μία δραστηριότητα στην άλλη. Από εδώ μπορεί να επιστρέψει στο προσκήνιο (*onResume*) ή να σταματήσει εάν δεν είναι ποια ορατή στον χρήστη (*onStop*).
Μια δραστηριότητα μπορεί να σταματήσει την λειτουργία της είτε γιατί καταστρέφεται από το σύστημα, είτε γιατί κάποια άλλη δραστηριότητα γίνεται ενεργή καλύπτοντας την. Στο σημείο αυτό αν ο χρήστης επιστρέψει στην δραστηριότητα καλείται η *onRestart* και μετά *onStart* και *onResume,* ενώ αν φύγει από την δραστηριότητα η *onDestroy*. Η μέθοδος *onDestroy* καλείται

όταν η δραστηριότητα έχει ολοκληρώσει την λειτουργία της και πρέπει να τερματιστεί, έχει γίνει δηλαδή κλήση της *finish*, είτε όταν διαγράφεται από το σύστημα για εξοικονόμηση μνήμης.



Σχήμα 2.3:Κύκλος ζωής δραστηριότητας

2.3.2 Υπηρεσία (Service)

Η υπηρεσία είναι μια συνιστώσα που τρέχει στο παρασκήνιο, για μεγάλο χρονικό διάστημα χωρίς να προσφέρει γραφικό περιβάλλον (UI). Μια υπηρεσία χρησιμοποιείται για να παίξει μουσική, ραδιόφωνο, να κατεβάζει δεδομένα από το δίκτυο και γενικά λειτουργίες που δεν χρειάζεται η συμμετοχή του χρήστη. Μια άλλη συνιστώσα όπως η δραστηριότητα μπορεί να ξεκινήσει μια υπηρεσία και να την αφήσει να τρέχει στο παρασκήνιο έπ' αόριστον ακόμα και όταν η δραστηριότητα έχει καταστραφεί, είναι δηλαδή σε κατάσταση *started*. Μια συνιστώσα της ίδιας εφαρμογής μπορεί να συνδεθεί με την υπηρεσία έτσι ώστε να αλληλεπιδρά με αυτήν, είναι σε κατάσταση *bound*.

Μια υπηρεσία που τρέχει μπορεί να ειδοποιήσει τον χρήστη με ειδοποιήσεις *Toast* ή στην γραμμή κατάστασης – *Status bar*. Η ειδοποίηση *Toast* είναι ένα κείμενο που εμφανίζεται στην οθόνη για σύντομο χρονικό διάστημα και μετά εξαφανίζεται, ενώ η ειδοποίηση στην *Status bar* αποτελείται από ένα εικονίδιο και ένα κείμενο.
Όπως και στην δραστηριότητα έτσι και εδώ υπάρχουν μέθοδοι για το κύκλο ζωής μια υπηρεσίας: *onCreate, onStartCommand* και *onDestroy.* Σε περίπτωση που η υπηρεσία επιτρέπει την σύνδεση συνιστωσών με αυτήν τότε γίνετε χρήση των μεθόδων: *onBind, onUnbind* και *onRebind.*

2.3.3 Πάροχος Περιεχομένου (Content Provider)

Μια εφαρμογή μπορεί να σώσει τα δεδομένα της σε αρχεία, σε μια βάση δεδομένων SQLite, στο δίκτυο ή σε κάποιο μόνιμο αποθηκευτικό χώρο που μπορεί να έχει πρόσβαση. Στα δεδομένα αυτά μπορεί να έχει πρόσβαση μόνο η ίδια η εφαρμογή, υπάρχουν όμως και περιπτώσεις που θέλουμε να έχουν πρόσβαση σε αυτά και άλλες εφαρμογές. Για αυτές τις περιπτώσεις χρησιμοποιούμε το πάροχο περιεχομένου. Μέσω του παρόχου άλλες εφαρμογές μπορούν να έχουν πρόσβαση στα δεδομένα ή ακόμα και να τα τροποποιήσουν αν επιτρέπεται. Τα δεδομένα αυτά αποθηκεύονται συνήθως σε μια βάση δεδομένων SQLite. Το Android προσφέρει μερικούς παρόχους περιεχομένου δίνοντας την δυνατότητα σε εφαρμογές να έχουν πρόσβαση σε επαφές, εικόνες, βίντεο ή άλλα δεδομένα που μπορεί να μοιράζεται. Ένας πάροχος περιεχομένου μπορεί να χρησιμοποιηθεί από μια εφαρμογή για πρόσβαση στα δεδομένα της, χωρίς αυτά να μπορούν να μοιραστούν.

2.3.4 Δέκτης Εκπομπών (Broadcast Receiver)

Ο δέκτης εκπομπών είναι μια συνιστώσα που λαμβάνει εκπεμπόμενα μηνύματα. Αυτά τα μηνύματα μπορεί να προέρχονται από το ίδιο το σύστημα, όπως ένα μήνυμα ότι η μπαταρία έχει χαμηλή στάθμη, μια φωτογραφία έχει τραβηχτεί, το κινητό έχει ολοκληρώσει την εκκίνηση κλπ. Οι εφαρμογές στο κινητό μπορούν και αυτές να εκπέμπουν μηνύματα, όπως για παράδειγμα να ενημερώσουν άλλες εφαρμογές ότι έχουν κατέβει δεδομένα στην συσκευή και μπορούν να τα χρησιμοποιήσουν. Αν και ο δέκτης εκπομπών δεν προσφέρει γραφικό περιβάλλον μπορεί να ενημερώσει τον χρήστη με ειδοποίηση στην γραμμή κατάστασης, όπως όταν έχει λάβει ένα μήνυμα – SMS. Αν μια εφαρμογή ενδιαφέρεται να λαμβάνει ειδοποιήσεις για κάποιο γεγονός τότε πρέπει να κάνει επέκταση της κλάσης *BroadcastReceiver* και να εγγράψει το δέκτη είτε κατά το χρόνο εκτέλεσης ή στο αρχείο AndroidManifest.xml.

2.4 Προθέσεις (Intent)

Ένα intent είναι ένα αντικείμενο που μεταφέρει ένα μήνυμα από μια συνιστώσα σε μια άλλη της ίδιας εφαρμογής ή άλλης εφαρμογής. Ένα intent είναι ένα ασύγχρονο μήνυμα που δίνει την δυνατότητα ενεργοποίησης μια συνιστώσας. Μπορεί να μεταφέρει μηνύματα για τις 3 συνιστώσες: δραστηριότητες, υπηρεσίες και δέκτες εκπομπών. Υπάρχουν 2 είδη intents, τα Explicit intents (σαφείς προθέσεις) όπου πρέπει να προσδιορίσουμε ποια συνιστώσα θα πρέπει να ξεκινήσει, για παράδειγμα μια δραστηριότητα μέσα

στην εφαρμογή και τα Implicit intents (ασαφείς προθέσεις) όπου δηλώνουν μια ενέργεια που θέλουν να πραγματοποιηθεί επιτρέποντας σε άλλες εφαρμογές να χειριστούν το αίτημα. Για παράδειγμα αν έχουμε μια δραστηριότητα που χρειάζεται να ανοίξει ένα σύνδεσμο στο πρόγραμμα περιήγησης στο Web (Web Browser) τότε θα στείλει ένα intent ACTION_WEB_SEARCH μαζί με το URL. Αν υπάρχουν περισσότερες από μία συνιστώσες που έχουν εγγραφεί για την ίδια ενέργεια τότε το λειτουργικό θα ανοίξει ένα παράθυρο διαλόγου ρωτώντας τον χρήστη ποια εφαρμογή θα πρέπει να χειριστεί το αίτημα.

2.5 Application Manifest

Κάθε εφαρμογή θα πρέπει να έχει ένα αρχείο AndroidManifest.xml το οποίο παρουσιάζει σημαντικές πληροφορίες της εφαρμογής στο σύστημα. Μερικές από τις λειτουργίες του αρχείου είναι η δήλωση όλων των συνιστωσών της εφαρμογής - δραστηριότητες, υπηρεσίες, πάροχοι περιεχομένου, δέκτες εκπομπών. Δήλωση των αδειών που χρειάζεται η εφαρμογή όπως πρόσβαση στο Διαδίκτυο, δήλωση του ελάχιστου επιπέδου API, δηλαδή την ελάχιστη έκδοση Android που μπορεί η εφαρμογή να τρέξει, δήλωση αδειών για χρήση υλικού όπως η κάμερα, Bluetooth κλπ. Παρακάτω φαίνεται το αρχείο Manifest της εφαρμογής BluePuc.

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3      package="com.santos.bluepuc"
4      android:versionCode="1"
5      android:versionName="1.0" >
6
7      <uses-sdk
8          android:minSdkVersion="8"
9          android:targetSdkVersion="19" />
10         <uses-permission android:name="android.permission.BLUETOOTH" />
11         <uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
12
13     <application
14         android:allowBackup="true"
15         android:icon="@drawable/ic_launcher"
16         android:label="@string/app_name"
17         android:theme="@style/AppTheme" >
18         <activity
19             android:name="com.santos.bluepuc.Device_Scan"
20             android:label="@string/app_name"
21             android:screenOrientation="portrait" >
22             <intent-filter>
23                 <action android:name="android.intent.action.MAIN" />
24
25                 <category android:name="android.intent.category.LAUNCHER" />
26             </intent-filter>
27         </activity>
28         <activity
29             android:name="com.santos.bluepuc.Control_page"
30             android:label="@string/title_activity_control_page"
31             android:screenOrientation="portrait" >
32         </activity>
33         <activity
34             android:name="com.santos.bluepuc.Settings_page"
35             android:label="@string/title_activity_settings_page"
36             android:screenOrientation="portrait" >
37         </activity>
38     </application>
39
40 </manifest>
```

Μέσα στο στοιχείο manifest υπάρχει το όνομα του πακέτου της εφαρμογής που λειτουργεί σαν μοναδικό αναγνωριστικό, η έκδοση του κώδικα και το όνομα της έκδοσης που θα εμφανίζεται στους χρήστες.
Στο στοιχείο uses-sdk δηλώνεται η ελάχιστη έκδοση API που χρειάζεται για να τρέξει η εφαρμογή, η τιμή 8 αντιστοιχεί στην έκδοση Android 2.2 – Froyo και την έκδοση για την οποία γράφτηκε, στην περίπτωση μας API level 19 δηλαδή Android 4.4 KitKat.
Στις επόμενες δυο γραμμές γίνεται αίτηση για άδεια χρήσης Bluetooth, των βασικών λειτουργιών όπως μεταφορά δεδομένων και η αίτηση BLUETOOTH_ADMIN για εύρεση συσκευών.
Στο στοιχείο application γίνεται δήλωση του εικονιδίου και του ονόματος της εφαρμογής που θα εμφανίζεται στο μενού. Μετά έχουμε τις τρεις δραστηριότητες της εφαρμογής. Το intent-filter καθορίζει ότι η πρώτη δραστηριότητα που θα ενεργοποιηθεί μόλις ο χρήστης πατήσει το εικονίδιο της εφαρμογής είναι η Device_Scan (γραμμή 23) και ότι το εικονίδιο της εφαρμογής θα εμφανίζεται στο μενού. Μετά έχουμε τις άλλες δυο δραστηριότητες οι οποίες δεν έχουν κάποιο intent-filter οπότε μπορούν να ενεργοποιηθούν από συνιστώσες της ίδιας εφαρμογής μόνο. Και οι τρεις δραστηριότητες δηλώνουν ότι λειτουργούν σε portrait mode, δεν περιστρέφεται δηλαδή η οθόνη.

2.6 Πόροι Εφαρμογής (Application Resources)

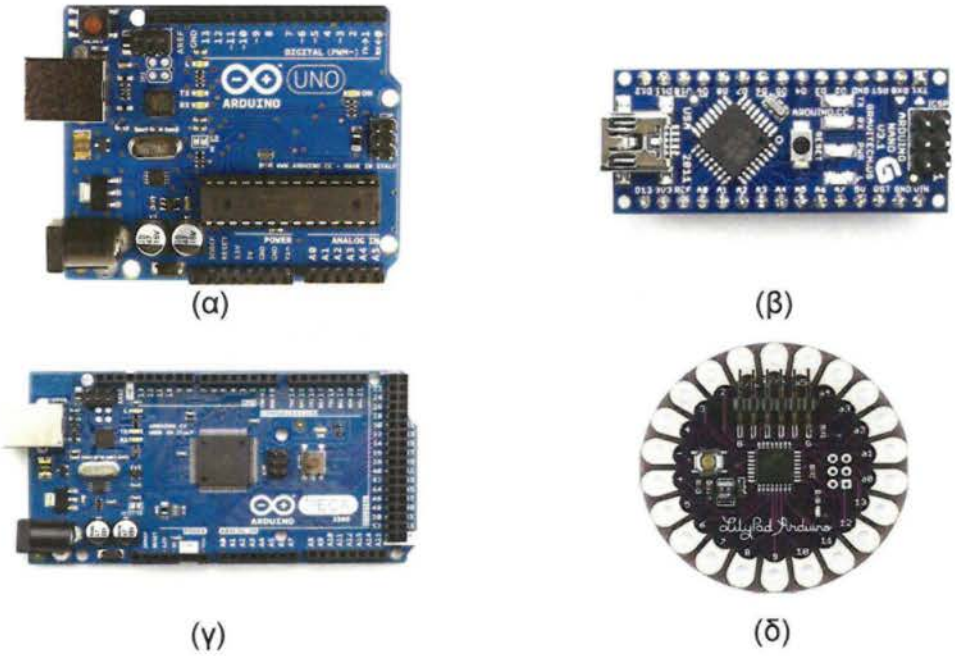
Μια εφαρμογή δεν αποτελείται μονάχα από κώδικα αλλά και από πόρους όπως εικόνες, ήχους και οτιδήποτε σχετικό με την παρουσίαση της εφαρμογής. Υπάρχει ένας φάκελος στο project με όνομα *res* που μπορούμε να αποθηκεύσουμε τους πόρους αυτούς. Μέσα στο φάκελο αυτό οι πόροι είναι οργανωμένοι σε φακέλους. Μερικοί από αυτούς είναι ο *anim* που περιέχει αρχεία xml για animations, ο φάκελος *drawable* που έχει αχρεία Bitmap (.png, .9.png, .jpg, .gif), ο φάκελος *layout* έχει αρχεία xml που προσδιορίζουν το γραφικό περιβάλλον μια δραστηριότητας ή των στοιχείων της (π.χ κουμπιά), ο φάκελος *values* έχει αρχεία xml με τιμές όπως ονόματα, χρώματα κλπ. Οι πόροι δίνουν την δυνατότητα στους προγραμματιστές να αναβαθμίσουν διάφορα χαρακτηριστικά της εφαρμογής, να κάνουν την εφαρμογή διαθέσιμη σε παραπάνω από μια γλώσσες, να βελτιστοποιήσουν την εφαρμογή ώστε να εκμεταλλεύεται τα διαφορετικά χαρακτηριστικά συσκευών (πχ tablets) και άλλες παρόμοιες δυνατότητες.

Κεφάλαιο 3

Arduino

3.1 Τι είναι το Arduino

Το Arduino είναι ένα εργαλείο για να κάνουμε τους υπολογιστές να αντιλαμβάνονται και να ελέγχουν το φυσικό χώρο γύρω τους. Αποτελείται από μια υπολογιστική πλατφόρμα βασισμένη σε μια μητρική πλακέτα ανοιχτού κώδικα με ενσωματωμένο μικροελεγκτή και ένα ολοκληρωμένο περιβάλλον ανάπτυξης (IDE – Integrated Development Environment) που επιτρέπει το προγραμματισμό της πλακέτας.



(α) (β)

(γ) (δ)

Σχήμα 3.1: (α) Arduino Uno, (β) Arduino Nano, (γ) Arduino Mega 2560, (δ) Arduino LilyPad

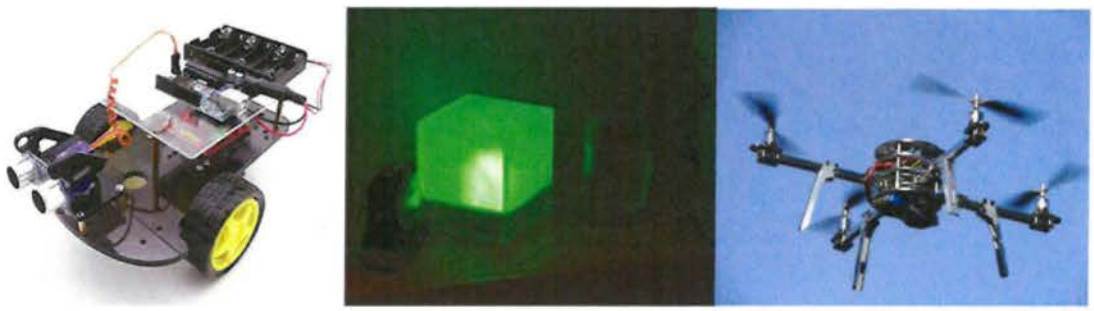
3.2 Χαρακτηριστικά του Arduino

Το Arduino σχεδιάστηκε με σκοπό την ευκολία υλοποίησης ιδεών, για παράδειγμα ένα αρχάριος μπορεί εύκολα και γρήγορα να μάθει να προγραμματίζει ένα Arduino για να μετράει την θερμοκρασία στο χώρο. Έτσι μπορεί να χρησιμοποιείται από μαθητές, καθηγητές ή ερασιτέχνες για απλά project ώστε να μάθουν πως χρησιμοποιείται ένας μικροελεγκτής ή και ποιο περίπλοκα. Μερικά από τα πλεονεκτήματα του Arduino είναι:

- Μια πλακέτα Arduino είναι φθηνή, με κόστος περίπου στα 20 ευρώ.
- Το Arduino τρέχει στα λειτουργικά συστήματα Windows, Macintosh OSX και Linux.

- Το περιβάλλον ανάπτυξης του είναι απλό και εύκολο στη χρήση.
- Το λογισμικό του είναι ανοιχτού κώδικα δίνοντας την δυνατότητα επέκτασης του.
- Ομοίως και στο υλικό δίνεται η δυνατότητα σε όποιον θέλει να φτιάξει τη δικιά του έκδοση της πλακέτας.

Η κοινότητα του Arduino ολοένα και αυξάνεται, αρχικά σχεδιάστηκε για να χρησιμοποιείται από μαθητές τώρα όμως χρησιμοποιείται και από ερασιτέχνες, επαγγελματίες, καλλιτέχνες κλπ. Έτσι υπάρχουν οδηγίες για την κατασκευή και προγραμματισμό ενός κυκλώματος (σχηματικό και κώδικας που χρειάζεται), επίσης υπάρχει και forum που μπορούν οι χρήστες να συζητάνε , να σχολιάζουν και να προτείνουν λύσεις για διάφορα προβλήματα.



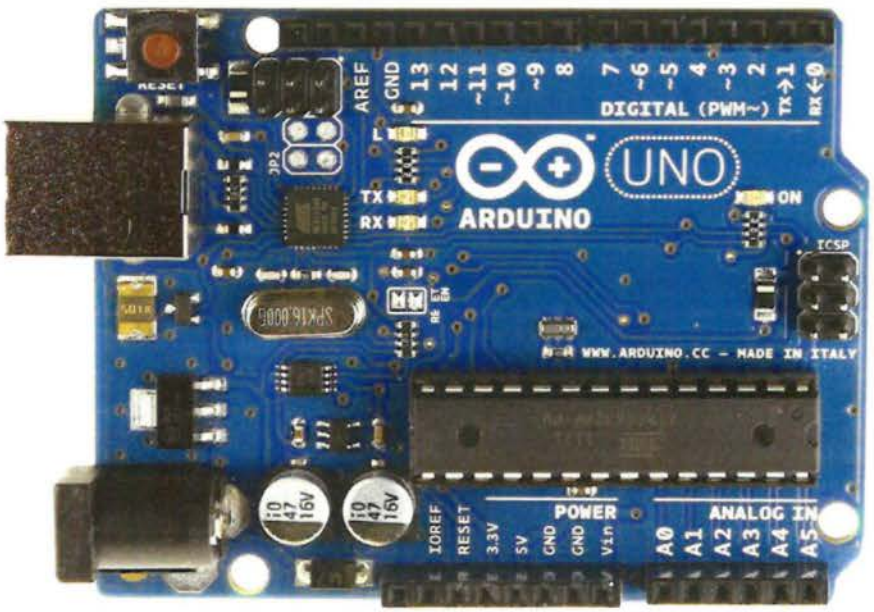
Σχήμα 3.2: Διάφορα project με Arduino

3.3 Η πλακέτα Arduino

Όλες οι πλακέτες Arduino (σχήμα 3.1) αποτελούνται από ένα μικροελεγκτή και απαραίτητα ηλεκτρονικά μέρη για να μπορέσουν να λειτουργήσουν. Υπάρχουν αρκετές πλακέτες Arduino με διαφορετικές δυνατότητες και τιμές, *shields* τυπωμένες πλακέτες δηλαδή που μπορούν να συνδεθούν μέσω των header-pins. Οι πλακέτες αυτές είναι επεκτάσεις και προσφέρουν δυνατότητες που δεν έχουν οι πλακέτες Arduino όπως σύνδεση στο Internet μέσω WiFi, χρήση GSM δικτύου, σύνδεση στο Internet μέσω Ethernet καλωδίου κλπ.

Σχήμα 3.3: Ethernet Shield

Το Arduino Uno είναι το ποιο κοινό χρησιμοποιούμενο προϊόν. Χρησιμοποιεί τον μικροελεγκτή ATmega328, έχει 14 ψηφιακές εισόδους/ εξόδους όπου 6 από αυτές μπορούν να χρησιμοποιηθούν σαν *PWM (Pulse-width modulation)* έξοδοι, 6 αναλογικές εισόδους, *ICSP* header *(In-Circuit-Serial-Programming)* , το ρολόι του μικροελεγκτή είναι στα 16 MHz, έχει ένα πλήκτρο reset, έχει μια θύρα USB, power jack, λειτουργεί στα 5Volt, έχει 32 KB *Flash Memory* - ο χώρος που αποθηκεύεται το πρόγραμμα που του περνάμε, έχει 2 KB *SRAM (Static Random Access Memory)* ο χώρος που το πρόγραμμα δημιουργεί και διαχειρίζεται τις μεταβλητές όταν τρέχει και 1 KB *EEPROM* μνήμη στην οποία μπορούμε να αποθηκεύσουμε δεδομένα που δεν θα χαθούν ακόμα και όταν κλείσει η συσκευή.



Σχήμα 3.4: Arduino Uno

Το Arduino Uno μπορεί να τροφοδοτηθεί από την θύρα USB (σχήμα 3.4 κάτω από το πλήκτρο reset), από μετασχηματιστή AC/DC στο power jack (σχήμα 3.4 κάτω αριστερά), ή με μπαταρία χρησιμοποιώντας τους ακροδέκτες Vin και GND. Η τροφοδοσία πρέπει να είναι 7 με 12Volts αλλιώς υπάρχει περίπτωση να μην λειτουργεί σωστά ή να προκληθεί ζημιά.
Οι ψηφιακοί ακροδέκτες εκτός από είσοδοι/ έξοδοι προσφέρουν και άλλες λειτουργίες. Οι ακροδέκτες 0 (RX) και 1 (TX) χρησιμοποιούνται για σειριακή επικοινωνία *TTL - Transistor Transistor Logic*. Οι ακροδέκτες 2 και 3 χρησιμοποιούνται για σήματα διακοπής. Οι ακροδέκτες 3, 5, 6, 9, 10, και 11 χρησιμοποιούνται για σήματα PWM. Οι ακροδέκτες 10, 11, 12 και 13 χρησιμοποιούνται για *SPI – Serial Peripheral Interface* επικοινωνία. Στον ακροδέκτη 13 είναι συνδεδεμένο ένα LED.
Το Arduino Uno έχει 6 αναλογικές εισόδους A0 ως A5 με ανάλυση 10bits, μπορούν όμως να χρησιμοποιηθούν και για άλλες λειτουργίες. Οι ακροδέκτες A4 και A5 χρησιμοποιούνται για επικοινωνία με *I2C / TWI (Inter-Integrated Circuit)* συσκευές.
Ο ακροδέκτης AREF είναι για την τάση αναφοράς των αναλογικών εισόδων.
Το Arduino Uno έχει ένα bootloader επιτρέποντας έτσι την φόρτωση κώδικα απευθείας από το καλώδιο USB χωρίς να χρειάζεται εξωτερικός προγραμματιστής.
Τέλος δεν χρειάζεται να πατηθεί το πλήκτρο reset προτού του φορτώσουμε κώδικα καθώς είναι σχεδιασμένο έτσι ώστε να κάνει reset από το λογισμικό που τρέχει στον υπολογιστή, μόλις πατηθεί το κουμπί φόρτωση.

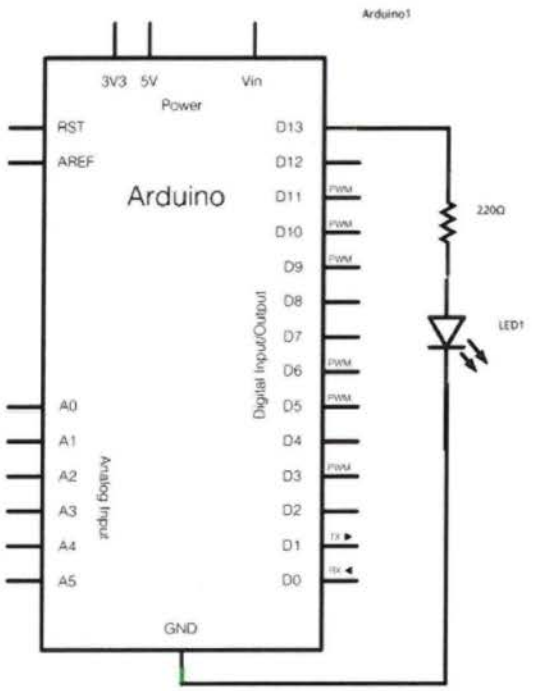
3.4 Arduino IDE

Το περιβάλλον ανάπτυξης του Arduino αποτελείται από ένα κειμενογράφο για συγγραφή του κώδικα, μια περιοχή μηνυμάτων, μια κονσόλα κειμένου, μια μπάρα με κουμπιά για τις βασικές ενέργειες και μια σειρά μενού.

Τα προγράμματα που γράφουμε στο Arduino ονομάζονται σχεδιαγράμματα (sketches) και γράφονται στο κειμενογράφο. Τα σχεδιαγράμματα αποθηκεύονται με την κατάληξη .ino. Η περιοχή μηνυμάτων δίνει πληροφορίες όπως αν αποθηκεύτηκε το σχεδιάγραμμα, αν έγινε η φόρτωση στο Arduino με επιτυχία και εμφανίζει μηνύματα σφαλμάτων. Στην κονσόλα εμφανίζονται μηνύματα για το περιβάλλον ανάπτυξης όπως ολοκληρωμένα μηνύματα σφάλματος. Στο κάτω μέρος του παραθύρου εμφανίζεται η πλακέτα που έχουμε επιλέξει και η θύρα. Τέλος στην μπάρα κάτω από τα μενού μπορούμε να βρούμε τα κουμπιά για βασικές λειτουργίες όπως έλεγχο του κώδικα για λάθη, μεταγλώττιση του κώδικα και φόρτωση στο Arduino, δημιουργία νέου σχεδιαγράμματος, εμφάνιση ενός μενού με όλα τα σχεδιαγράμματα που έχουμε αποθηκεύσει, πατώντας σε κάποιο θα το εμφανίσει στο τρέχον παράθυρο, αποθήκευση του σχεδιαγράμματος και άνοιγμα της σειριακής οθόνης. Η σειριακή οθόνη χρησιμοποιείται για να εμφανίσουμε δεδομένα στον υπολογιστή, που λαμβάνουμε από το Arduino μέσω USB καλωδίου. Μπορούμε επίσης να στείλουμε δεδομένα.

Σχήμα 3.5: Arduino IDE

Παρακάτω θα δούμε ένα απλό παράδειγμα χρήσης του Arduino, πώς να αναβοσβήνουμε ένα LED (Blink LED).



Σχήμα 3.6: Σχηματικό κυκλώματος Blink LED

```
// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
int led = 13;

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  digitalWrite(led, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);               // wait for a second
  digitalWrite(led, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);               // wait for a second
}
```

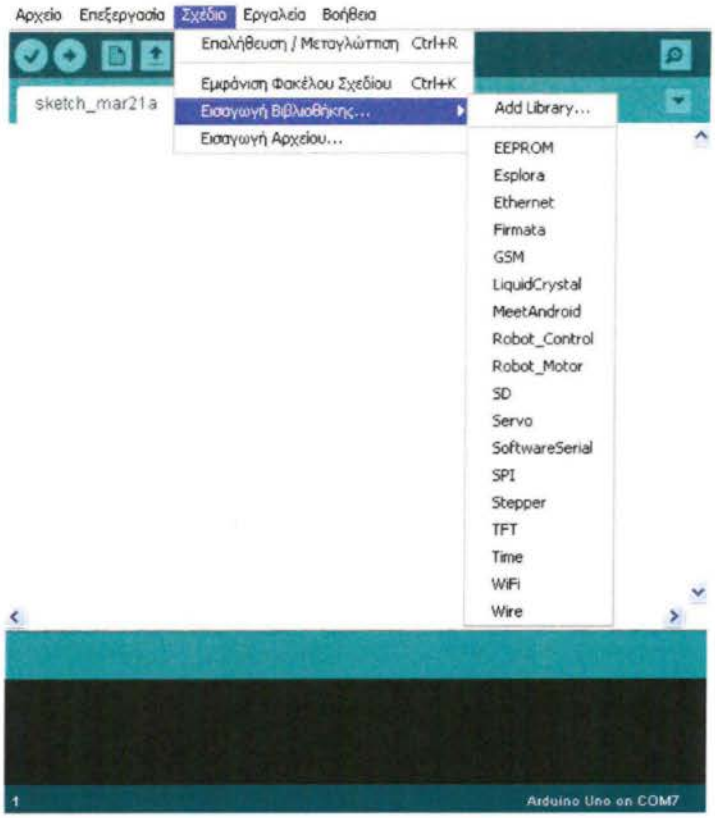
Αρχικά δίνουμε ένα όνομα για τον ακροδέκτη του μικροελεγκτή που θα χρησιμοποιήσουμε. Η συνάρτηση setup() καλείται κατά την έναρξη του προγράμματος ή μετά από reset. Εκτελείται μόνο μια φορά και χρησιμοποιείται για αρχικοποίηση μεταβλητών, λειτουργίας των χρησιμοποιούμενων ακροδεκτών, ρύθμιση του baud rate κλπ. Με την εντολή pinMode() προγραμματίζουμε έναν ακροδέκτη για είσοδο ή έξοδο. Η συνάρτηση loop() εκτελείται μετά την setup() και ότι υπάρχει μέσα στην συνάρτηση αυτή θα εκτελείται συνέχεια, είναι δηλαδή το κυρίως πρόγραμμα. Με την εντολή digitalWrite() ο μικροελεγκτής δίνει 5V – HIGH ή 0V – LOW. Με την εντολή delay() σταματάει η εκτέλεση του προγράμματος για το χρόνο που έχει δοθεί, ο χρόνος είναι σε χιλιοστά του δευτερολέπτου. Άρα το πρόγραμμα ανάβει για 1 δευτερόλεπτο το LED μετά το σβήνει για 1 δευτερόλεπτο και επαναλαμβάνει αυτή τη διαδικασία.

3.4.1 Βιβλιοθήκες (Libraries)

Εκτός από τις βασικές λειτουργίες που προσφέρει το Arduino μπορούμε να χρησιμοποιήσουμε και βιβλιοθήκες για ποιο περίπλοκες περιπτώσεις όπως έλεγχο οθόνης *LCD – Liquid Crystal Display*, βηματικού κινητήρα, σύνδεση στο Internet μέσω WiFi κλπ. Κάποιες από τις βιβλιοθήκες που χρησιμοποιούνται συχνά είναι προ-εγκατεστημένες στο Arduino, μπορούμε όμως να προσθέσουμε ή να φτιάξουμε τις δικές μας βιβλιοθήκες.

Οι βιβλιοθήκες είναι διαθέσιμες συνήθως σε ένα αρχείο .zip ή ένας φάκελος. Ο φάκελος έχει το όνομα της βιβλιοθήκης και μέσα στο φάκελο πρέπει να υπάρχει ένα αρχείο .cpp ένα αρχείο .h, ένα αρχείο keywords.txt για τις μεθόδους και συναρτήσεις της βιβλιοθήκης, ένας φάκελος examples με παραδείγματα χρήσης της βιβλιοθήκης και άλλα αρχεία που χρειάζονται.
Για να εγκαταστήσουμε μια βιβλιοθήκη υπάρχουν δυο τρόποι. Μπορούμε να αντιγράψουμε το φάκελο της βιβλιοθήκης στον υποκατάλογο libraries του φακέλου του Arduino ή μέσα από το πρόγραμμα του Arduino στο μενού επιλέγουμε Σχέδιο → Εισαγωγή Βιβλιοθήκης → Add Library... στο παράθυρο που θα εμφανιστεί επιλέγουμε το αρχείο .zip με την βιβλιοθήκη και πατάμε Open. Μετά αν επιστρέψουμε στο μενού θα δούμε ότι έχει προστεθεί η νέα

βιβλιοθήκη. Για να δούμε τα παραδείγματα της βιβλιοθήκης επιλέγουμε από το μενού Αρχείο → Παραδείγματα και μετά την βιβλιοθήκη που θέλουμε και θα εμφανίσει όλα τα διαθέσιμα παραδείγματα.



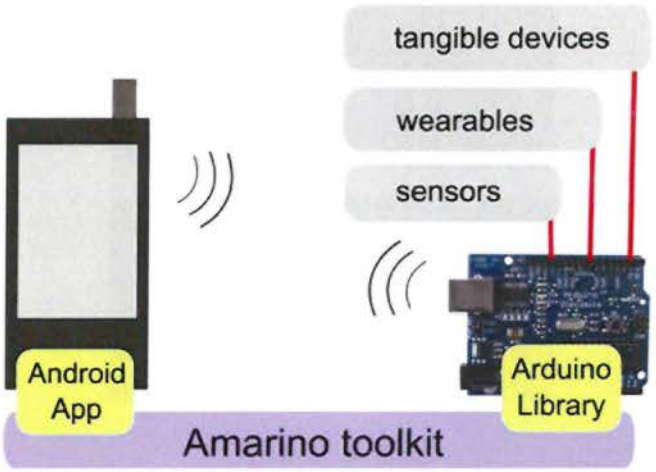
Σχήμα 3.7: Οι βιβλιοθήκες στο Arduino IDE

Κεφάλαιο 4

Amarino toolkit

4.1 Τι είναι το Amarino toolkit

Το Amarino toolkit αποτελείται από δυο συστατικά στοιχεία: Την Android εφαρμογή και μια βιβλιοθήκη για το Arduino. Το Amarino προσφέρει επίσης ένα API και online documentation. Το Amarino λειτουργεί σαν ένα κανάλι επικοινωνίας που επιτρέπει την αποστολή και λήψη δεδομένων μέσω Bluetooth ανάμεσα σε ένα κινητό τηλέφωνο Android και ένα μικροελεγκτή Arduino με σκοπό την επικοινωνία και έλεγχο συσκευών που είναι συνδεμένες στο Arduino. Το Amarino δημιουργεί μια αόρατη στο χρήστη ανταλλαγή δεδομένων η οποία βασίζεται στη δημιουργία γεγονότων – *events*. Το Arduino έχει ένα μηχανισμό κλήσης μεθόδων μόλις συμβεί ένα γεγονός (λήψη δεδομένων) και μεθόδους για αποστολή δεδομένων από τον μικροελεγκτή στο κινητό.



Σχήμα 4.1: Τρόπος λειτουργίας του Amarino

4.2 Η Android εφαρμογή Amarino

Η εφαρμογή Amarino έχει τρία μέρη που χειρίζονται τις συσκευές Bluetooth.

- Τον χειριστή συσκευών Bluetooth
- Τον χειριστή γεγονότων
- Την παρακολούθηση σύνδεσης

Η εφαρμογή στην κεντρική σελίδα έχει ένα κουμπί για ανίχνευση των συσκευών(Add BT Device). Σε περίπτωση που δεν ήταν ενεργοποιημένο το Bluetooth προτού ξεκινήσει η αναζήτηση το ενεργοποιεί. Μόλις πατηθεί το κουμπί θα εμφανιστεί μια λίστα με όλες τις διαθέσιμες συσκευές και ο χρήστης μπορεί να προσθέσει μία. Αφού έχει επιλέξει κάποια συσκευή μπορεί να συνδεθεί ή να αποσυνδεθεί με αυτήν.

Μια σημαντική ιδιότητα του Amarino είναι ότι κρατάει τις ενεργές συνδέσεις ακόμα και όταν έχει κλίσει η εφαρμογή. Η διαχείριση των συνδέσεων γίνεται από την υπηρεσία του Amarino, μια υπηρεσία όπως είπαμε και στο κεφάλαιο 2.3.2 μπορεί να τρέχει ακόμα και αν η εφαρμογή έχει τερματιστεί. Έτσι μπορεί ο χρήστης να ανοίξει κάποια άλλη εφαρμογή χωρίς να χαθεί η σύνδεση. Όπως θα δούμε και παρακάτω μια εφαρμογή που χρησιμοποιεί το Amarino API βασίζεται σε αυτήν την υπηρεσία. Για εξοικονόμηση μπαταρίας και μνήμης η υπηρεσία τρέχει μόνο αν υπάρχουν ενεργές συνδέσεις αλλιώς τερματίζεται μαζί με την εφαρμογή. Η υπηρεσία του Amarino ενημερώνει τον χρήστη για την κατάσταση της σύνδεσης με ειδοποιήσεις στην γραμμή κατάστασης, ακόμα και όταν η εφαρμογή εκτελείται.

Σχήμα 4.2: Ειδοποιήσεις υπηρεσίας Amarino

Στο σχήμα 4.2 βλέπουμε τις 3 περιπτώσεις ειδοποιήσεων, στο επάνω μέρος όταν γίνεται η σύνδεση με την συσκευή, στο μεσαίο όταν έχει γίνει η σύνδεση και στο κάτω όταν γίνει αποσύνδεση.
Ο χειριστής γεγονότων χρησιμοποιείται όταν ο χρήστης θέλει να στείλει δεδομένα από το κινητό στο Arduino (για να είναι δυνατή η λήψη και αποστολή δεδομένων στο Arduino θα πρέπει να γραφτεί και φορτωθεί πρόγραμμα στο Arduino). Αφού επιλέξει ο χρήστης ένα γεγονός τότε το Arduino μπορεί να λάβει αυτά τα δεδομένα εάν υπάρχει σύνδεση μέσω Bluetooth. Μερικά από τα γεγονότα που προσφέρει το Amarino είναι: Light Sensor για τις μεταβολές του φωτός, Battery Level για το επίπεδο της μπαταρίας στο κινητό κλπ. Οι προγραμματιστές μπορούν να εμπλουτίσουν τη λίστα με τα γεγονότα. Ο χρήστης μπορεί να δει σε πραγματικό χρόνο τα δεδομένα από τα ενεργά γεγονότα για έλεγχο και αποσφαλμάτωση – *debugging* του προγράμματος του.
Από την αρχική οθόνη ο χρήστης μπορεί να πατήσει το κουμπί Monitoring για να εμφανιστεί μια σελίδα που δείχνει πληροφορίες όπως δεδομένα που

λαμβάνει ή στέλνει το Amarino, μηνύματα λαθών και debugging. Μπορεί να στείλει δεδομένα σε ένα συνδεμένο Arduino επιλέγοντας μια σημαία και εισάγοντας ένα μήνυμα.

4.3 Η βιβλιοθήκη MeetAndroid

Η βιβλιοθήκη MeetAndroid δίνει την δυνατότητα στο Arduino να επικοινωνήσει με το κινητό τηλέφωνο Android. Προσφέρει τις κατάλληλες μεθόδους για αποστολή και λήψη δεδομένων μέσω Bluetooth και έχει παραδείγματα χρήσης της βιβλιοθήκης.

Στο παρακάτω παράδειγμα θα δούμε πως μπορούμε να λάβουμε και να στείλουμε έναν ακέραιο αριθμό.

Αρχικά εισάγουμε την βιβλιοθήκη με την εντολή #include <MeetAndroid.h> , μετά δηλώνουμε και κατασκευάζουμε το αντικείμενο με την MeetAndroid meetAndroid; .
Μέσα στην setup() ρυθμίζουμε το baud rate για την ανταλλαγή δεδομένων μέσω Bluetooth και εγγράφουμε μια συνάρτηση η οποία καλείται όταν συμβεί ένα γεγονός. Η πρώτη παράμετρος είναι το όνομα της συνάρτησης, intValue και η δεύτερη είναι η σημαία του γεγονότος, ID = 'c'. Το ID χρησιμοποιείται για να μπορέσει να ξεχωρίσει τα δεδομένα που λαμβάνει και θα πρέπει να είναι ίδιο με αυτό στην εφαρμογή στο κινητό. Τα ID με κεφαλαία αντιστοιχούν στα γεγονότα που έχει προ-εγκατεστημένα το Amarino ενώ με πεζά για γεγονότα που έχουμε προσθέσει.
Μέσα στην συνάρτηση loop() που εκτελείται συνέχεια έχουμε την εντολή meetAndroid.receive(). Η μέθοδος receive() ελέγχει αν υπάρχουν διαθέσιμα δεδομένα, σε περίπτωση που υπάρχουν τα αναλύει και καλεί την συνάρτηση που έχουμε εγγράψει για το γεγονός που έλαβε. Με την μέθοδο getInt() λαμβάνουμε τον ακέραιο αριθμό. Εκτός από την μέθοδο αυτή υπάρχουν και άλλες για διάφορους τύπους δεδομένων (σχήμα 4.3). Με την μέθοδο send(v) στέλνουμε στο κινητό τον ακέραιο που λάβαμε.

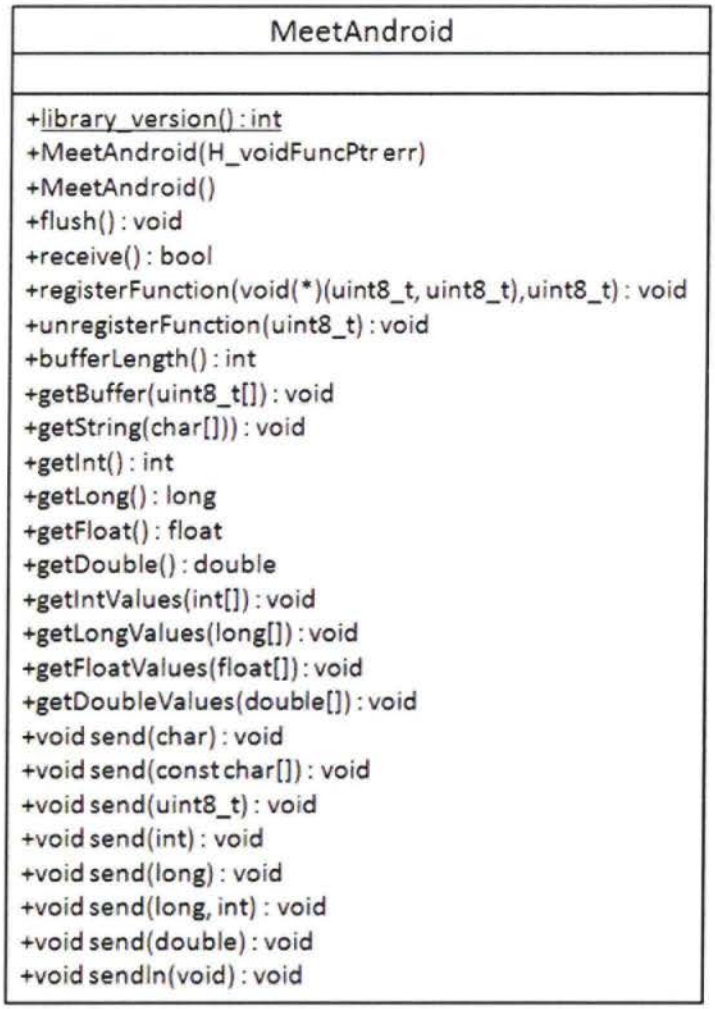
```
#include <MeetAndroid.h>

MeetAndroid meetAndroid;

void setup()
{
   Serial.begin(57600);
   meetAndroid.registerFunction(intValue, 'c');
}

void loop()
{
  meetAndroid.receive();
}

void intValue(byte flag, byte numOfValues)
{
  int v = meetAndroid.getInt();
  meetAndroid.send(v);
}
```



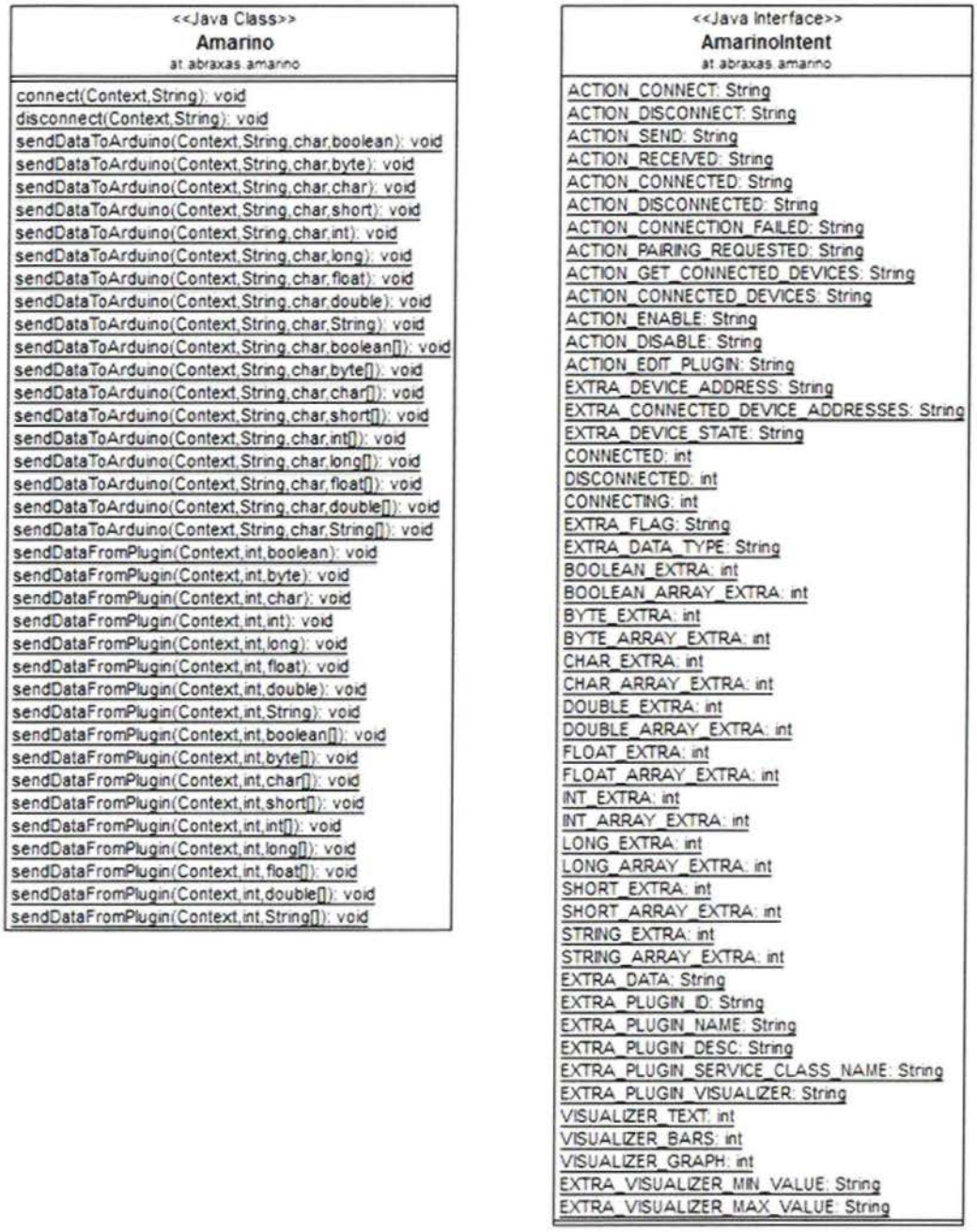
Σχήμα 4.3: Διαθέσιμες συναρτήσεις της βιβλιοθήκης MeetAndroid

4.4 Amarino API

Σε περίπτωση που θέλουμε να δούμε τα δεδομένα που λαμβάνουμε από το Arduino ή να έχουμε αλληλεπίδραση με την οθόνη του κινητού τότε θα πρέπει να φτιάξουμε μια ξεχωριστή εφαρμογή. Για το λόγο αυτό το Amarino προσφέρει ένα API – *Application Programming Interface* με τις κατάλληλες μεθόδους για επικοινωνία με το Arduino.
Η επικοινωνία με την συσκευή Bluetooth του Arduino περιλαμβάνει τέσσερις ενέργειες:

- Σύνδεση με το Arduino
- Αποσύνδεση από το Arduino
- Αποστολή δεδομένων στο Arduino
- Λήψη δεδομένων από το Arduino

Στο σχήμα 4.4 βλέπουμε τις δυο συνιστώσες του Amarino API, μια κλάση Amarino με στατικές μεθόδους και το AmarinoIntent, μια συλλογή προθέσεων σταθερών μηνυμάτων.



Σχήμα 4.4: Amarino API

Σύνδεση και αποσύνδεση:

Για την σύνδεση ή αποσύνδεση με μια Bluetooth συσκευή καλούμε τις μεθόδους connect() ή disconnect() αντίστοιχα και περνάμε το τρέχον αντικείμενο εφαρμογής (this) και την διεύθυνση της συσκευής (MAC Address). Αν θέλουμε να ξέρουμε αν έγινε η σύνδεση ή αποσύνδεση προτού συνεχίσουμε σε άλλες ενέργειες τότε μπορούμε να χρησιμοποιήσουμε ένα Broadcast Receiver για να λάβουμε τα μηνύματα που εκπέμπει το Amarino σε τρίτους με τις αλλαγές κατάστασης της σύνδεσης.

```
1 private static final String DEVICE_ADDRESS = "00:06:66:03:73:7B";
2
3 @Override
4 protected void onStart() {
5 super.onStart();
6
7 Amarino.connect(this, DEVICE_ADDRESS);
8 }
9
10 @Override
11 protected void onStop() {
12 super.onStop();
13
14 // if you connect in onStart() you must not forget to disconnect in onStop()
15 Amarino.disconnect(this, DEVICE_ADDRESS);
16 }
```

Ενημέρωση για αλλαγές της κατάστασης σύνδεσης:

Όπως είπαμε και προηγουμένως για να ξέρουμε τις αλλαγές στην κατάσταση σύνδεσης πρέπει να χρησιμοποιήσουμε ένα Broadcast Receiver όπως φαίνεται παρακάτω.

```
1 public class ConnectionStateReceiver extends BroadcastReceiver() {
2
3 @Override
4 public void onReceive(Context context, Intent intent) {
5 final String action = intent.getAction();
6
7 // this is how to retrieve the address of the sender
8 final String address = intent.getStringExtra(AmarinoIntent.EXTRA_DEVICE_ADDRESS);
9
10 if (AmarinoIntent.ACTION_CONNECTED.equals(action)){
11 // connection has been established
12 }
13 else if (AmarinoIntent.ACTION_DISCONNECTED.equals(action)){
14 // disconnected from a device
15 }
16 else if (AmarinoIntent.ACTION_CONNECTION_FAILED.equals(action)){
17 // connection attempt was not successful
18 }
19 else if (AmarinoIntent.ACTION_PAIRING_REQUESTED.equals(action)){
20 // a notification message to pair the device has popped up
21 }
22 }
23 }
```

Αποστολή δεδομένων στο Arduino:

Εφόσον έχει γίνει η σύνδεση τότε μπορούμε να στείλουμε δεδομένα όπως φαίνεται παρακάτω.

```
1 final char flag = 'a';
2 final String message = "Amarino rocks!";
3
4 Amarino.sendDataToArduino(this, DEVICE_ADDRESS, flag, message);
```

Η μέθοδος sendDataToArduino() παίρνει σαν παραμέτρους το τρέχον αντικείμενο, την διεύθυνση της συσκευής, την σημαία του γεγονότος και τα δεδομένα που θέλουμε να στείλουμε.

Λήψη δεδομένων από το Arduino:

Για να λάβουμε δεδομένα από το Arduino χρειαζόμαστε ένα Broadcast Receiver για να λάβει το μήνυμα ACTION_RECEIVED που εκπέμπει το Amarino. Το Amarino μόλις λάβει τα δεδομένα από το Arduino τα επισυνάπτει σε ένα intent και το εκπέμπει για να μπορέσουν οι ενδιαφερόμενες εφαρμογές να το λάβουν και να εξάγουν τα δεδομένα.

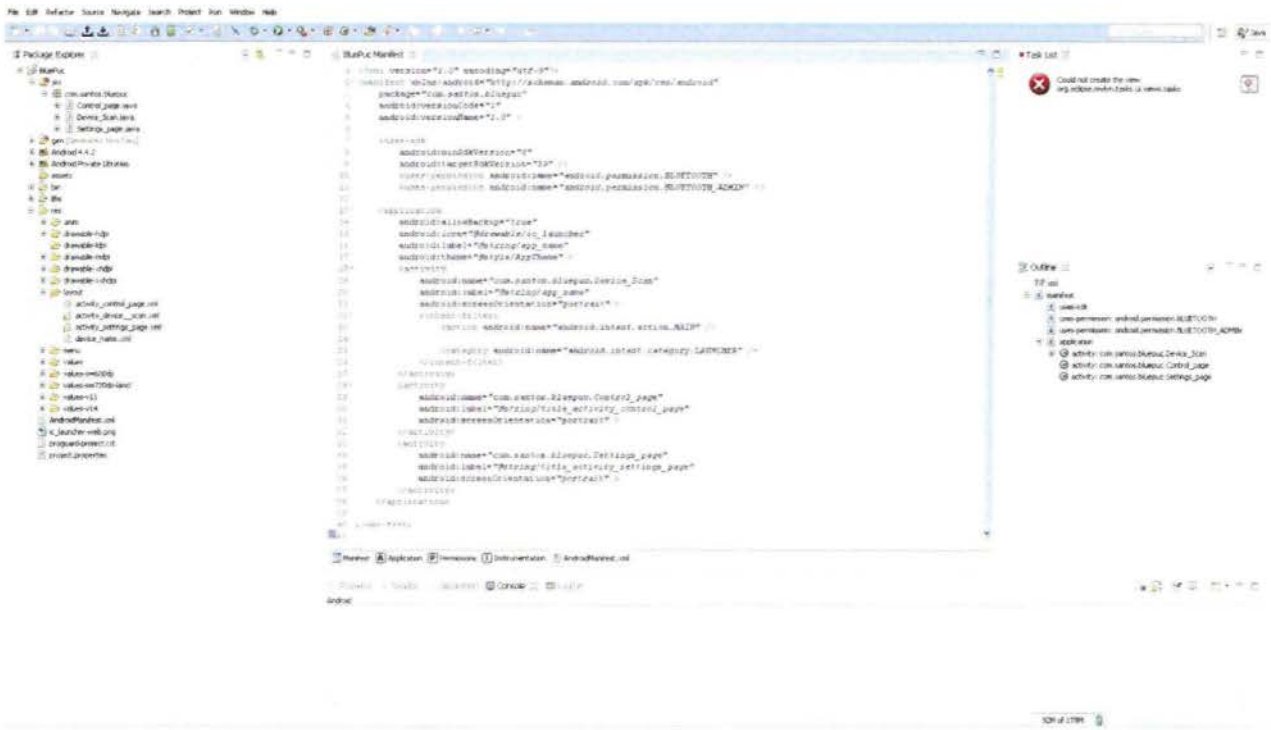
```
1 public class DataReceiver extends BroadcastReceiver {
2
3 @Override
4 public void onReceive(Context context, Intent intent) {
5
6 if (AmarinoIntent.ACTION_RECEIVED.equals(intent.getAction())){
7
8 // the device address from which the data was sent,
9 // we don't need it here but to demonstrate how you retrieve it
10 final String address = intent.getStringExtra(AmarinoIntent.EXTRA_DEVICE_ADDRESS);
11
12 // the type of data which is added to the intent
13 final int dataType = intent.getIntExtra(AmarinoIntent.EXTRA_DATA_TYPE, -1);
14
15 switch (dataType){
16
17 case AmarinoIntent.STRING_EXTRA:
18 String data = intent.getStringExtra(AmarinoIntent.EXTRA_DATA);
19 // do something useful with the received data
20 break;
21 }
22 }
23 }
24 }
```

Μέσα στην onReceive() η οποία καλείται μόλις λάβει ένα intent αποθηκεύουμε το τύπο των δεδομένων (dataType – γραμμή 13) και μετά σε μια μεταβλητή του ίδιου τύπου αποθηκεύουμε τα δεδομένα που λάβαμε (data – γραμμή 18).

Κεφάλαιο 5

Eclipse IDE

Για να μπορέσουμε να δημιουργήσουμε μια Android εφαρμογή θα χρησιμοποιήσουμε το ολοκληρωμένο περιβάλλον ανάπτυξης Eclipse με ενσωματωμένο το ADT (Android Developer Tools). Η Google προσφέρει σε ένα αρχείο .zip το Android SDK (Software Developers Kit) και το Eclipse με το ADT plug-in, απαραίτητα για την δημιουργία, τον έλεγχο και την αποσφαλμάτωση των εφαρμογών. Για να μπορέσει να λειτουργήσει το Eclipse χρειάζεται να κατεβάσουμε και το JDK (Java Development Kit).

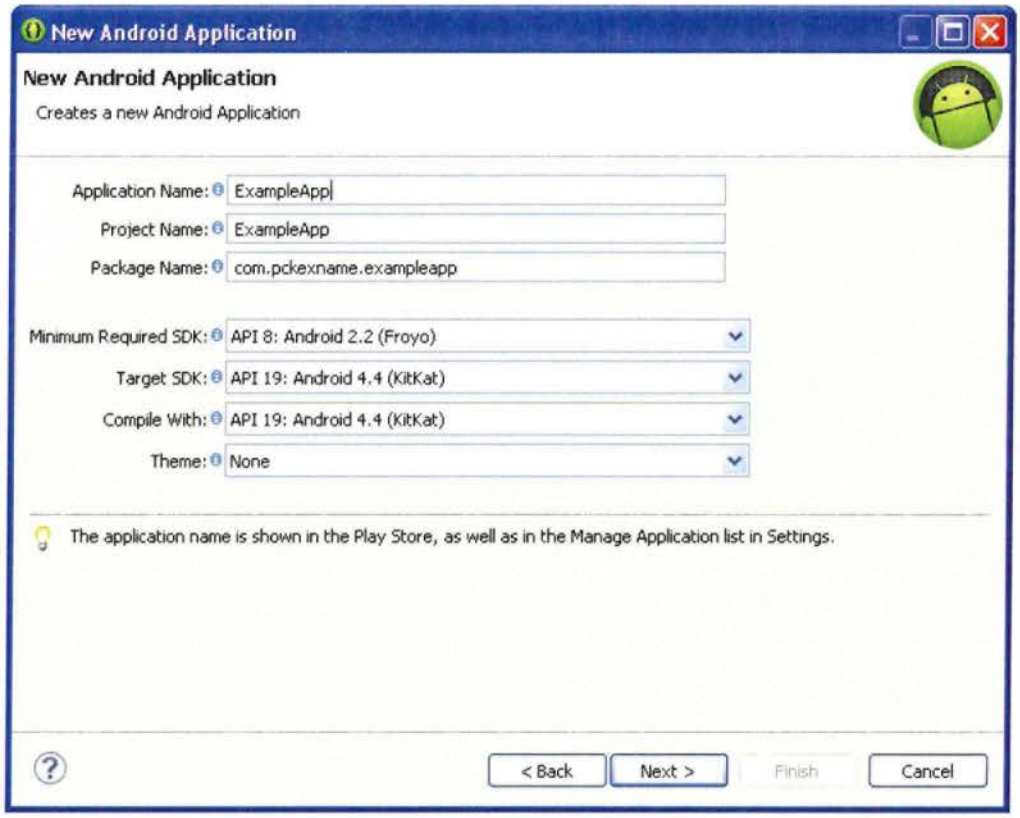


Σχήμα 5.1: Eclipse IDE

5.1 Δημιουργία Android project

Για την δημιουργία μιας εφαρμογής ξεκινάμε με την δημιουργία ενός project. Από το μενού επιλέγουμε File → New → Project. Στο παράθυρο που εμφανίζεται επιλέγουμε από τον φάκελο Android το Android Application Project και πατάμε Next. Θα εμφανιστεί ένα παράθυρο όπως φαίνεται στο σχήμα 5.2 για να συμπληρώσουμε τα πεδία. Το όνομα της εφαρμογής είναι αυτό που θα φαίνετε στους χρήστες, το όνομα του Project είναι το όνομα του καταλόγου με όλα τα αρχεία της εφαρμογής, το όνομα του πακέτου θα πρέπει να είναι μοναδικό στην συσκευή Android. Αν θέλουμε να δημοσιεύσουμε την εφαρμογή που θα φτιάξουμε δεν γίνεται να χρησιμοποιήσουμε το προεπιλεγμένο. Οι Android συσκευές δεν έχουν όλες την ίδια έκδοση του λογισμικού, οπότε πρέπει να δηλώσουμε την ελάχιστη έκδοση του λογισμικού

που θα υποστηρίζει η εφαρμογή, την μέγιστη έκδοση με την οποία έχουμε ελέγξει την εφαρμογή και την έκδοση με την οποία έχουμε μεταγλωττίσει την εφαρμογή. Μόλις συμπληρώσουμε όλα τα πεδία πατάμε Next, στο επόμενο παράθυρο που εμφανίζεται αφαιρούμε την επιλογή Create activity, θα δημιουργήσουμε αργότερα την δραστηριότητα και πατάμε Next. Στο παράθυρο αυτό μπορούμε να επιλέξουμε το εικονίδιο της εφαρμογής, μόλις έχουμε βρει το κατάλληλο πατάμε Finish. Στο αριστερό μέρος του Eclipse, στο Package Explorer μπορούμε να βρούμε τον φάκελο του Project που μόλις φτιάξαμε.

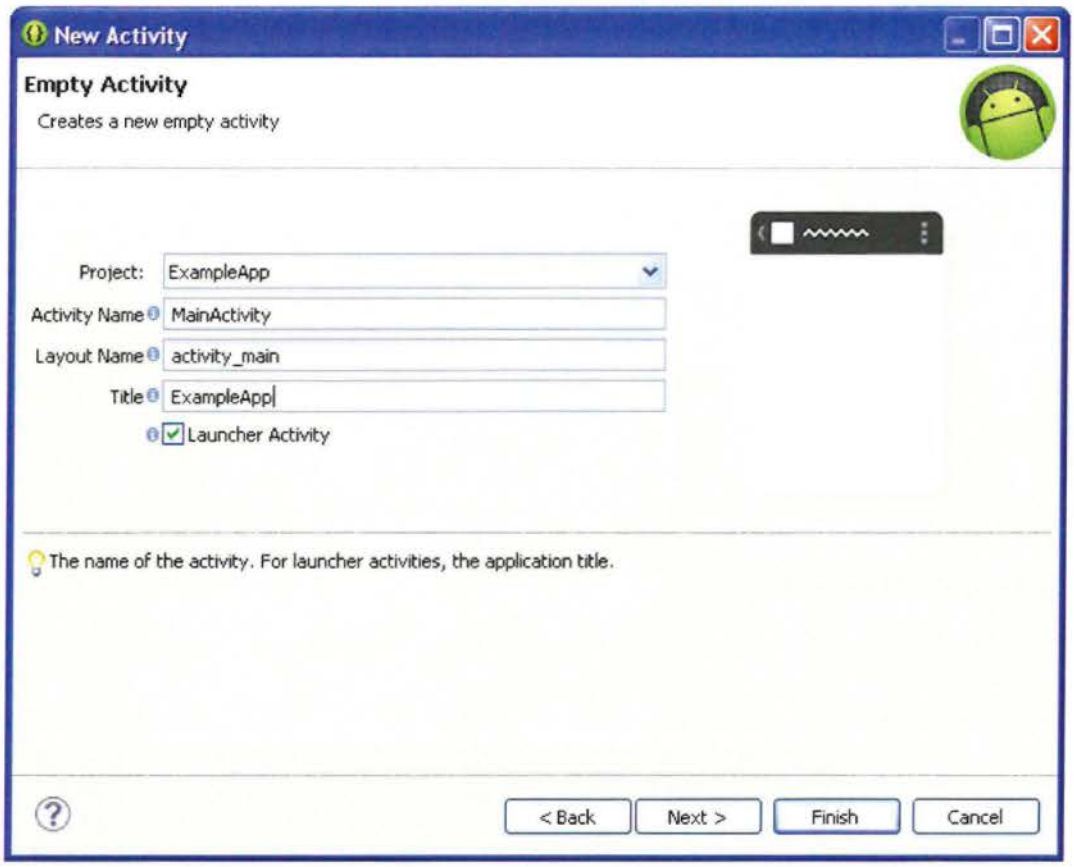


Σχήμα 5.2: New Android Application Project Wizard

5.2 Δημιουργία δραστηριότητας

Για να μπορέσει ο χρήστης να αλληλεπιδρά με την εφαρμογή πρέπει να φτιάξουμε μια δραστηριότητα. Πατάμε το New από την μπάρα κάτω από το μενού. Στο παράθυρο που εμφανίζεται επιλέγουμε από τον φάκελο Android το Android Activity και πατάμε Next. Στο επόμενο παράθυρο επιλέγουμε το είδος της δραστηριότητας που θέλουμε να φτιάξουμε, εδώ επιλέγουμε το Empty Activity και πατάμε Next. Στο επόμενο παράθυρο συμπληρώνουμε τα πεδία όπως φαίνεται στο σχήμα 5.3. Πρώτα επιλέγουμε το project, σε ποια εφαρμογή δηλαδή θα είναι η δραστηριότητα. Το όνομα της δραστηριότητας είναι το όνομα της κλάσης, που μπορούμε να συμπληρώσουμε τον κώδικα για την λειτουργία της εφαρμογής. Το όνομα του Layout είναι το όνομα του αρχείου xml με το γραφικό περιβάλλον της δραστηριότητας. Η Δραστηριότητα αυτή θα πρέπει να είναι η πρώτη που θα

εμφανίζεται μόλις ο χρήστης πατήσει από το μενού το εικονίδιο της εφαρμογής, για το λόγο αυτό επιλέγουμε την επιλογή Launcher Activity. Εφόσον αυτή είναι η πρώτη δραστηριότητα που θα εμφανίζεται, στο πεδίο Title δηλώνουμε τον τίτλο της εφαρμογής, δηλαδή το όνομα που θέλουμε να φαίνεται μαζί με το εικονίδιο της εφαρμογής. Πατάμε Finish για να γίνουν οι αλλαγές. Αν ανοίξουμε το αρχείο AndroidManifest.xml θα δούμε ότι το Eclipse έχει προσθέσει την δραστηριότητα.

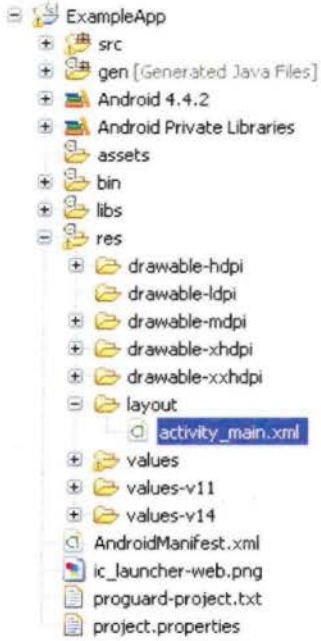


Σχήμα 5.3: Δημιουργία δραστηριότητας

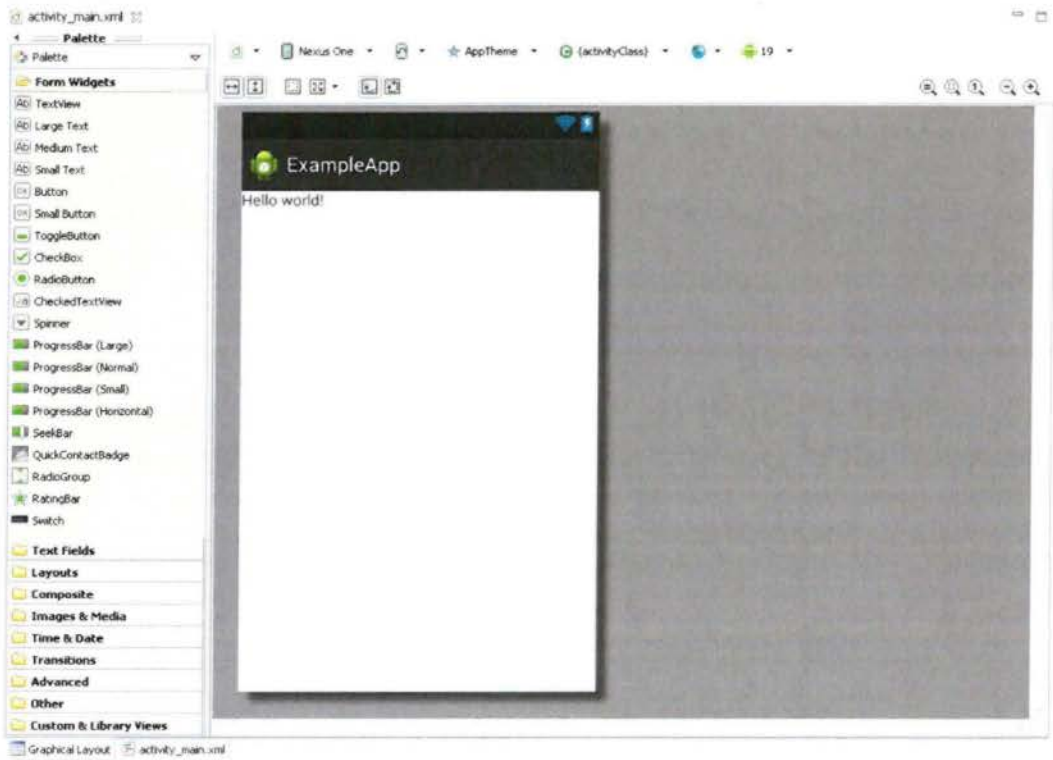
5.3 Δημιουργία γραφικού περιβάλλοντος

Για να μπορέσει ο χρήστης να αλληλεπιδρά με την εφαρμογή θα χρειαστεί να δημιουργήσουμε το γραφικό περιβάλλον. Στο Eclipse στον Package Explorer (σχήμα 5.4) επιλέγουμε τον φάκελο της εφαρμογής που φτιάξαμε. Για την τροποποίηση του γραφικού περιβάλλοντος επιλέγουμε τον φάκελο res → layout και επιλέγουμε το αρχείο activity_main.xml. Στο κεντρικό παράθυρο του Eclipse μπορούμε να επεξεργαστούμε το αρχείο. Στο κάτω μέρος υπάρχουν δυο καρτέλες μια με την γραφική απεικόνιση, όπως δηλαδή θα φαίνεται στην οθόνη του κινητού τηλεφώνου και μια με τον κώδικα του αρχείου. Αν θέλουμε να προσθέσουμε ένα γραφικό στοιχείο μπορούμε να το κάνουμε και στις δυο καρτέλες. Στην καρτέλα με την γραφική απεικόνιση (σχήμα 5.5) υπάρχει μια παλέτα στα αριστερά με όλα τα στοιχεία που μπορούμε να προσθέσουμε με την λειτουργία drag and drop. Το πλεονέκτημα της γραφικής απεικόνισης είναι ότι μπορούμε να δούμε πως θα εμφανίζεται η

εφαρμογή σε διάφορα μεγέθη οθονών χωρίς να χρειάζεται να την μεταγλωττίσουμε και να εκτελείται η εφαρμογή σε κάποια συσκευή. Μπορούμε να κάνουμε αλλαγές στο γραφικό περιβάλλον και κατά τον χρόνο εκτέλεσης της εφαρμογής.



Σχήμα 5.4: Package Explorer



Σχήμα 5.5: Γραφική απεικόνιση Layout

Παρακάτω θα δούμε πως μπορούμε να προσθέσουμε ένα κουμπί το οποίο μόλις πατηθεί θα εμφανίζει ένα μήνυμα στο πεδίο κειμένου με διαφορετικό χρώμα κάθε φορά. Στο αρχείο activity_main.xml θα προσθέσουμε μέσα στο RelativeLayout το παρακάτω κώδικα:

```
<Button
    android:id="@+id/button1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/textView1"
    android:layout_centerHorizontal="true"
    android:text="Button"
    android:onClick="ShowMsg"/>
```

Στο φάκελο src στο αρχείο MainActivity.java συμπληρώνουμε τις εντολές έτσι ώστε να είναι όπως φαίνετε παρακάτω:

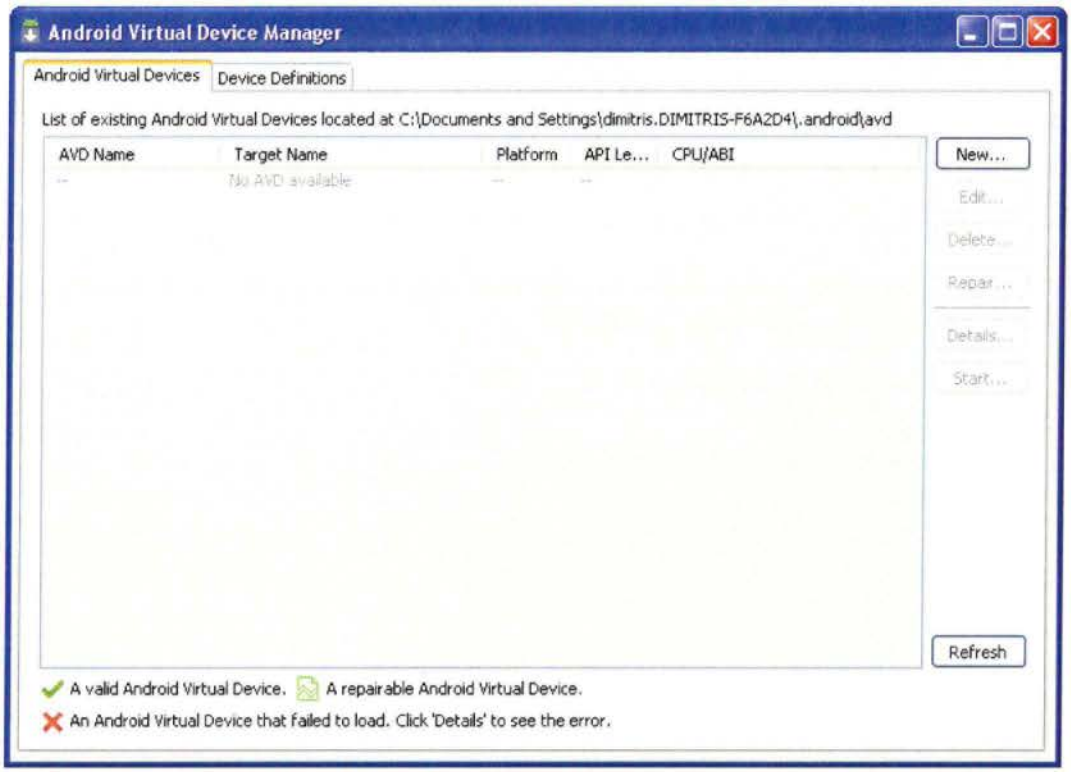
```
1  package com.pckexname.exampleapp;
2+ import java.util.Random;
9
10 public class MainActivity extends Activity {
11
12     TextView textview;
13     int red=0;
14     int green=0;
15     int blue=0;
16
17-    @Override
18     protected void onCreate(Bundle savedInstanceState) {
19         super.onCreate(savedInstanceState);
20         setContentView(R.layout.activity_main);
21
22         textview = (TextView)findViewById(R.id.textView1);
23     }
24
25-    public void ShowMsg(View view){
26
27         Random rnumber= new Random();
28         textview.setText("Hello from Android!");
29         textview.setTextColor(Color.rgb
30                 (rnumber.nextInt(255), rnumber.nextInt(255), rnumber.nextInt(255)));
31     }
32 }
```

Για να μπορέσουμε να αλλάξουμε το κείμενο του πεδίου κειμένου κατά τον χρόνο εκτέλεσης δημιουργούμε μια μεταβλητή τύπου TextView (γραμμή 12). Η μεταβλητή αυτή αντιστοιχεί στο πεδίο κειμένου με id textView1 (γραμμή 22). Το κουμπί μόλις πατηθεί καλεί την μέθοδο ShowMsg στην οποία δημιουργούμε πρώτα ένα αντικείμενο της κλάσης Random για να μπορέσουμε να παράγουμε τυχαίους αριθμούς. Μετά ορίζουμε το κείμενο που θα εμφανιστεί και το χρώμα, 3 τυχαίοι αριθμοί από το 0 έως το 255.

5.4 Έλεγχος της εφαρμογής

Προτού δημιουργήσουμε την τελική έκδοση της εφαρμογής πρέπει να την ελέγξουμε σε φυσική ή εικονική μηχανή. Ο έλεγχος αυτός είναι απαραίτητος για να διορθώσουμε τυχόν προβλήματα ή λάθη και να εξασφαλίσουμε την σωστή λειτουργία της εφαρμογής. Σε περίπτωση που θέλουμε να ελέγξουμε την εφαρμογή σε συσκευές με διαφορετικά χαρακτηριστικά, όπως το μέγεθος

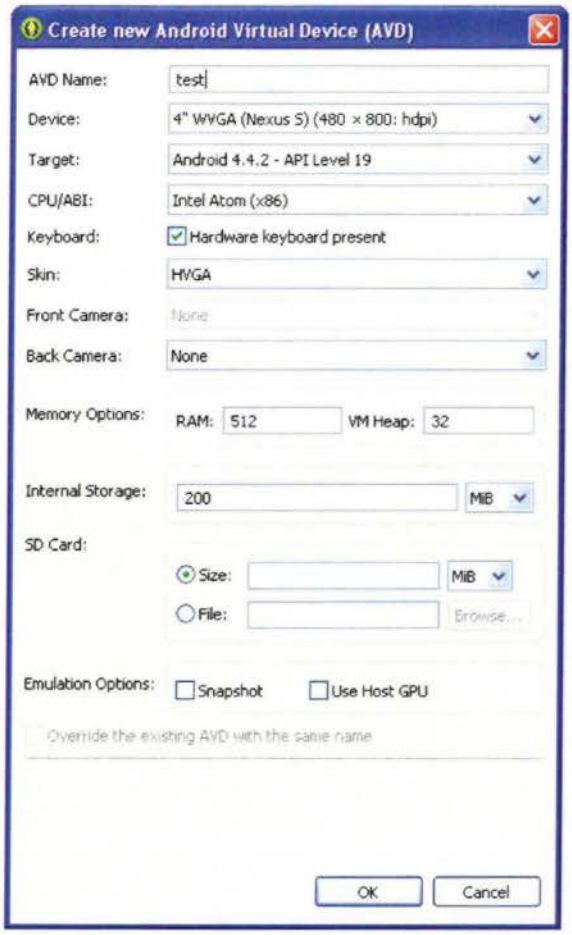
της οθόνης αλλά δεν διαθέτουμε τις συσκευές αυτές τότε μπορούμε να δημιουργήσουμε εικονικές μηχανές με το Android Virtual Device Manager του Eclipse. Πατάμε το από την toolbar και εμφανίζεται το παράθυρο όπως φαίνεται στο σχήμα 5.6 με τις εικονικές μηχανές και τα χαρακτηριστικά τους.



Σχήμα 5.6: Android Virtual Device Manager

Για να δημιουργήσουμε νέα εικονική μηχανή πατάμε το New... και θα εμφανιστεί ένα παράθυρο όπως φαίνεται στο σχήμα 5.7 για να συμπληρώσουμε τα χαρακτηριστικά που θα έχει. Στο πεδίο AVD Name συμπληρώνουμε το όνομα της συσκευής. Τα ονόματα των εικονικών μηχανών που έχουμε φτιάξει θα φαίνονται στο παράθυρο του AVD Manager. Στο Device επιλέγουμε μια πραγματική συσκευή για εξομοίωση, στο Target επιλέγουμε την έκδοση του Android που θα έχει. Επιλέγουμε το είδος του πληκτρολογίου που θα έχει η εικονική μηχανή, τις διαστάσεις και την εμφάνιση της οθόνης, αν θα έχει πίσω κάμερα. Μπορούμε να επιλέξουμε το μέγεθος της μνήμης RAM, το μέγεθος του αποθηκευτικού χώρου και εάν θα έχει κάρτα μνήμης SD για την αποθήκευση δεδομένων. Μπορούμε να δημιουργήσουμε όσες εικονικές μηχανές θέλουμε έτσι ώστε να ελέγξουμε την εφαρμογή. Για να εκτελέσουμε την εφαρμογή στην εικονική μηχανή κάνουμε δεξί κλικ στον φάκελο του project στον Package Explorer → Run As → Android Application.
Μπορούμε να ελέγξουμε την εφαρμογή και σε φυσική συσκευή εάν διαθέτουμε μία. Πρώτα πηγαίνουμε στις ρυθμίσεις της συσκευής → Επιλογές για προγραμματιστές και ενεργοποιούμε τον Εντοπισμό σφαλμάτων USB. Συνδέουμε την συσκευή με τον υπολογιστή με το καλώδιο USB και κάνουμε δεξί κλικ στον φάκελο του project στον Package Explorer → Run As →

Android Application. Θα εμφανιστεί ένα παράθυρο για να διαλέξουμε σε ποια συσκευή θα γίνει η εκτέλεση της εφαρμογής και επιλέγουμε την συσκευή και πατάμε OK. Η εφαρμογή έχει φορτωθεί στην συσκευή και μπορούμε να την βρούμε στο μενού.



Σχήμα 5.7: Δημιουργία νέας εικονικής μηχανής

Το Android διαθέτει ένα μηχανισμό καταγραφής και εμφάνισης μηνυμάτων αποσφαλμάτωσης. Τα μηνύματα αυτά μπορεί να προέρχονται από διάφορες εφαρμογές, το λειτουργικό σύστημα ή και μηνύματα που έχει φτιάξει ο προγραμματιστής της εφαρμογής. Τα μηνύματα αυτά είναι χρήσιμα για τον έλεγχο της σωστής λειτουργίας της εφαρμογής.
Για παράδειγμα αν έχει γίνει λογικό λάθος κατά την συγγραφή του κώδικα της εφαρμογής τότε το πρόβλημα θα εμφανιστεί κατά τον χρόνο εκτέλεσης. Στο παραπάνω παράδειγμα με το κουμπί που εμφανίζει ένα μήνυμα με διαφορετικό χρώμα εάν αφαιρέσουμε την εντολή της γραμμής 22, το πρόγραμμα θα μπορέσει να μεταγλωττιστεί και η εφαρμογή θα ξεκινήσει κανονικά, όταν όμως πατήσουμε το κουμπί τότε θα έχουμε διακοπή της λειτουργίας της εφαρμογής και θα εμφανιστεί ένα μήνυμα ειδοποίησης. Στο Eclipse στο παράθυρο LogCat θα εμφανιστούν τα μηνύματα (με κόκκινο χρώμα και το E στο πεδίο Level) με τις πληροφορίες για το τι προκάλεσε την διακοπή της εφαρμογής.

Σχήμα 5.8: Παράθυρο LogCat στο Eclipse

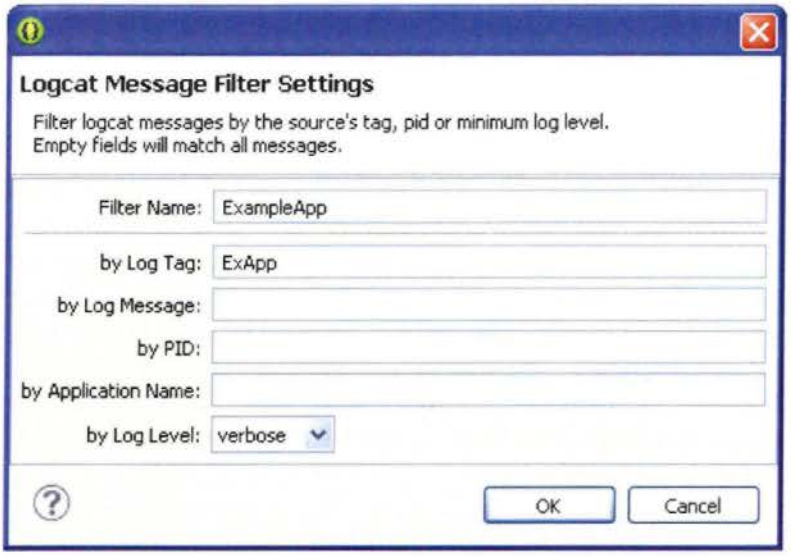
Ο προγραμματιστής έχει την δυνατότητα να δημιουργήσει δικά του μηνύματα για τον έλεγχο της λειτουργίας της εφαρμογής χρησιμοποιώντας την κλάση Log. Στο παραπάνω παράδειγμα εάν θέλουμε να γνωρίζουμε πόσες φορές έχει πατήσει ο χρήστης το κουμπί μπορούμε να κάνουμε τις παρακάτω αλλαγές στον κώδικα:

```
27-    public void ShowMsg(View view){
28
29         Random rnumber= new Random();
30         textview.setText("Hello from Android!");
31         textview.setTextColor(Color.rgb
32                 (rnumber.nextInt(255), rnumber.nextInt(255), rnumber.nextInt(255)));
33
34         counter++;
35         Log.d("ExApp", "Button counter:"+counter);
36     }
```

Έχουμε αρχικοποιήσει την μεταβλητή counter και κάθε φορά που ο χρήστης πατάει το κουμπί θα προσθέτει ένα στην τιμή της. Με την εντολή Log.d δηλώνουμε ότι θα στείλουμε ένα μήνυμα debug και η πρώτη παράμετρος είναι μια ετικέτα ενώ η δεύτερη το μήνυμα. Αν εκτελέσουμε τώρα την εφαρμογή θα δούμε ότι εμφανίζονται όλα τα μηνύματα και δεν μπορούμε να βρούμε με ευκολία το μήνυμα που φτιάξαμε να εμφανίζεται με το πάτημα του κουμπιού. Για να λύσουμε αυτό το πρόβλημα μπορούμε να φτιάξουμε ένα φίλτρο ώστε να εμφανίζει μόνο συγκεκριμένα μηνύματα. Πατώντας το κουμπί συν από τον κατάλογο φίλτρων θα εμφανιστεί ένα παράθυρο για την δημιουργία νέου φίλτρου.

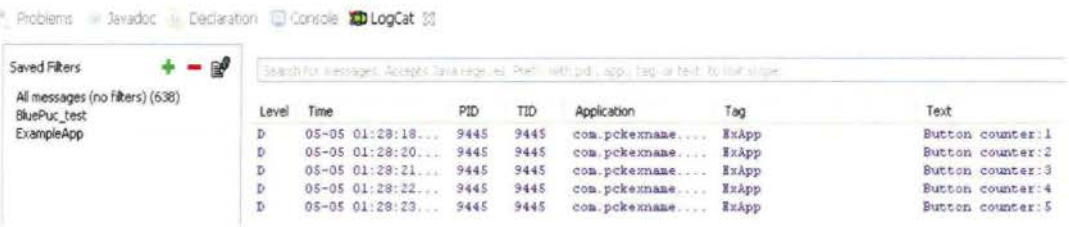
Σχήμα 5.9: Κατάλογος LogCat φίλτρων

Για να εμφανίζει πόσες φορές έχει πατήσει το κουμπί ο χρήστης συμπληρώνουμε τα πεδία όπως φαίνεται στο σχήμα 5.10, δίνουμε ένα όνομα για το φίλτρο και την ετικέτά του μηνύματος.



Σχήμα 5.10: Δημιουργία νέου LogCat φίλτρου

Αν εκτελέσουμε την εφαρμογή, επιλέξουμε το φίλτρο που φτιάξαμε και πατήσουμε το κουμπί πέντε φορές θα δούμε το παρακάτω αποτέλεσμα



Σχήμα 5.11: Εμφάνιση μηνυμάτων με χρήση φίλτρου

Μόλις έχουμε ολοκληρώσει τους ελέγχους της εφαρμογής πρέπει να αφαιρέσουμε τις εντολές που κάνουν κλήση της κλάσης Log για να προχωρήσουμε στην δημοσίευση της.

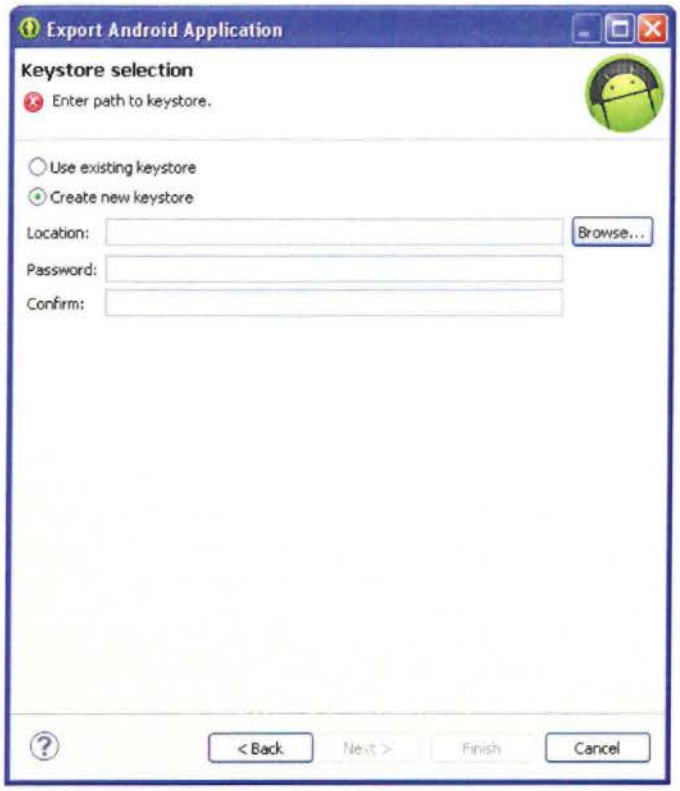
5.5 Δημοσίευση της εφαρμογής

Το τελευταίο βήμα δημιουργίας μιας εφαρμογής είναι η δημοσίευση της στο κοινό. Αφού έχουμε ολοκληρώσει τον έλεγχο της εφαρμογής και τον καθαρισμό του κώδικα από εντολές και πόρους που δεν χρησιμοποιούνται μπορούμε να προχωρήσουμε στην δημιουργία του τελικού αρχείου που θα μπορεί να μοιραστεί στο κοινό για την εγκατάσταση της εφαρμογής στις Android συσκευές τους. Όλες οι εγκατεστημένες εφαρμογές σε μια Android συσκευή πρέπει να είναι ψηφιακά υπογεγραμμένες με ένα πιστοποιητικό του οποίου το ιδιωτικό κλειδί έχει ο προγραμματιστής της εφαρμογής. Το σύστημα Android χρησιμοποιεί τα πιστοποιητικά για να μπορεί να προσδιορίσει τον

συγγραφέα της εφαρμογής. Το σύστημα Android δεν θα επιτρέψει την εγκατάσταση ή εκτέλεση μιας εφαρμογής σε πραγματική συσκευή ή τον εξομοιωτή εάν δεν είναι σωστά υπογεγραμμένη.

Κατά την διάρκεια του ελέγχου της εφαρμογής το Eclipse αναλαμβάνει την υπογραφή της εφαρμογής, είναι δηλαδή σε debug mode η εφαρμογή. Η εφαρμογή θα εγκατασταθεί και μπορεί να εκτελείται στην συσκευή που επιλέξαμε αλλά δεν γίνεται να μοιράσουμε την εφαρμογή σε τρίτους.
Για να μπορέσουμε να δημοσιεύσουμε την εφαρμογή π.χ στο Google Play, σε άλλο κατάστημα ή με άλλο τρόπο θα πρέπει να είναι σε release mode. Για να γίνει η υπογραφή της εφαρμογής χρειάζεται ένα κλειδί το οποίο μόνο ο προγραμματιστής θα πρέπει να έχει. Το κλειδί αυτό είναι απαραίτητο για την υπογραφή μελλοντικών εκδόσεων της εφαρμογής. Αν κάποιος άλλος αποκτήσει πρόσβαση στο κλειδί χωρίς την γνώση του προγραμματιστή τότε υπάρχει κίνδυνος να τροποποιήσει την εφαρμογή και να την δημοσιεύσει προκαλώντας προβλήματα. Αν το κλειδί χαθεί τότε δεν είναι δυνατή η δημοσίευση ενημερώσεων για την εφαρμογή. Δεν γίνεται να δημιουργήσουμε ξανά ένα υπάρχων κλειδί.

Για να δημιουργήσουμε την τελική έκδοση της εφαρμογής με το Eclipse κάνουμε δεξί κλικ στον φάκελο του project στον Package Explorer → Export. Στο παράθυρο που εμφανίζεται επιλέγουμε από τον φάκελο Android το Export Android Application και πατάμε Next. Στο επόμενο παράθυρο γίνεται η επιλογή του project, εδώ είναι ήδη επιλεγμένο οπότε πατάμε Next. Στο παράθυρο όπως φαίνεται στο σχήμα 5.12 μπορούμε να επιλέξουμε έναν κατάλογο με το κλειδί εάν διαθέτουμε ή να δημιουργήσουμε έναν. Επιλέγουμε το Create new keystore και συμπληρώνουμε τα πεδία. Στο πεδίο Location πατάμε το Browse και επιλέγουμε σε ποιο κατάλογο θα αποθηκευτεί το κλειδί. Εδώ χρειάζεται προσοχή όπως περιγράψαμε παραπάνω καθώς αν το κλειδί χαθεί δεν θα μπορούμε να δημοσιεύσουμε την εφαρμογή. Μόλις δημιουργήσουμε και τον κωδικό πατάμε Next. Στο επόμενο παράθυρο όπως φαίνεται στο σχήμα 5.13 δημιουργούμε το κλειδί. Δίνουμε ένα όνομα και κωδικό (ο κωδικός δεν χρειάζεται να είναι ίδιος με τον κωδικό keystore), δηλώνουμε την διάρκεια ζωής της εφαρμογής. Εδώ βάζουμε συνήθως μια τιμή αρκετά μεγάλη ώστε να μπορούν οι χρήστες να χρησιμοποιούν την εφαρμογή για μεγάλο χρονικό διάστημα. Αν θέλουμε να δημοσιεύσουμε την εφαρμογή στο Google Play τότε θα πρέπει η διάρκεια ζωής της εφαρμογής να λήγει μετά τις 22 Οκτωβρίου 2033. Μετά μπορούμε να συμπληρώσουμε τα προσωπικά στοιχεία όπως ονοματεπώνυμο, χώρα κλπ. Μόλις συμπληρώσουμε τα πεδία πατάμε Next και εμφανίζεται το τελευταίο παράθυρο στο οποίο επιλέγουμε σε ποιο κατάλογο θα αποθηκευτεί το αρχείο με κατάληξη .apk και πατάμε Finish. Η τελική έκδοση της εφαρμογής είναι έτοιμη και μπορεί να δημοσιευθεί.



Σχήμα 5.12: Επιλογή ή δημιουργία keystore

Σχήμα 5.13: Δημιουργία κλειδιού

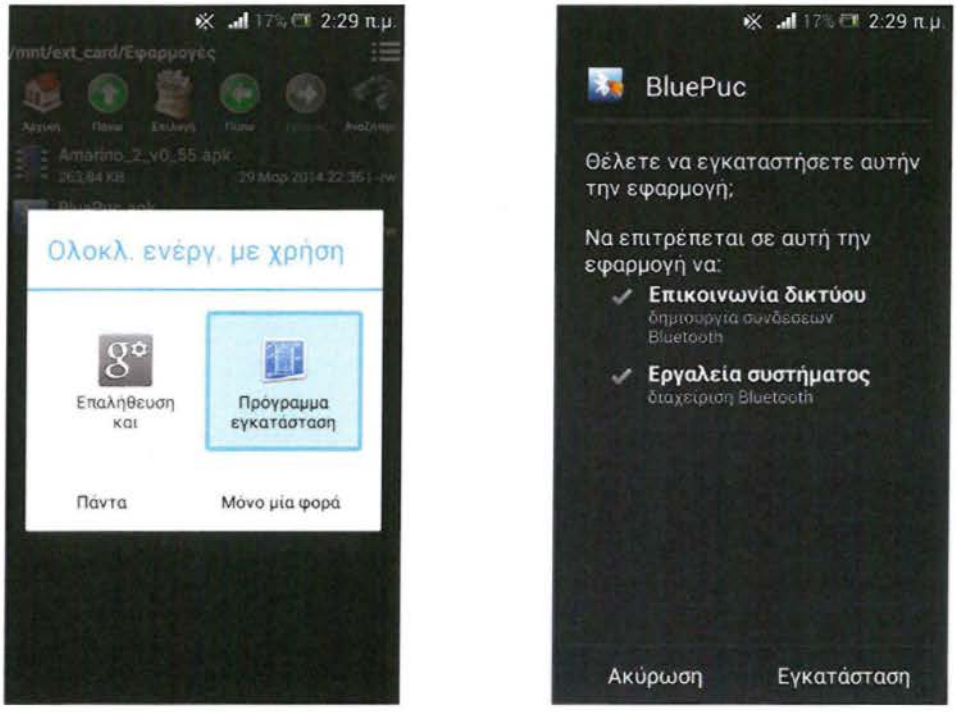
Κεφάλαιο 6

Η Android εφαρμογή BluePuc

Η εφαρμογή BluePuc δίνει την δυνατότητα στον χρήστη να ελέγξει μέσω του κινητού τηλεφώνου του την συσκευή που συνοδεύει την εφαρμογή. Η συσκευή αυτή παρουσιάζει κάποιες πληροφορίες και έχει μια πρίζα στην οποία μπορούμε να συνδέσουμε μια ηλεκτρική συσκευή που θέλουμε να ελέγχουμε. Η εφαρμογή αποτελείται από τρεις οθόνες μια για αναζήτηση συσκευών και σύνδεση, μια οθόνη ελέγχου της συσκευής και μια σελίδα ρυθμίσεων. Παρακάτω θα δούμε τον τρόπο λειτουργίας της εφαρμογής από την μεριά ενός χρήστη αλλά και τον κώδικα της εφαρμογής.

6.1 Εγκατάσταση της εφαρμογής

Το πρώτο πράγμα που πρέπει να κάνουμε είναι να εγκαταστήσουμε την εφαρμογή BluePuc και Amarino στο κινητό τηλέφωνο. Για να μπορέσουμε να εγκαταστήσουμε εφαρμογές που δεν προέρχονται από το Google Play Store πηγαίνουμε στις Ρυθμίσεις → Ασφάλεια και επιλέγουμε το Άγνωστες πηγές. Περνάμε τα δυο αρχεία με κατάληξη .apk σε ένα φάκελο και πατάμε πάνω στο εικονίδιο για εγκατάσταση όπως φαίνεται στο σχήμα 6.1.

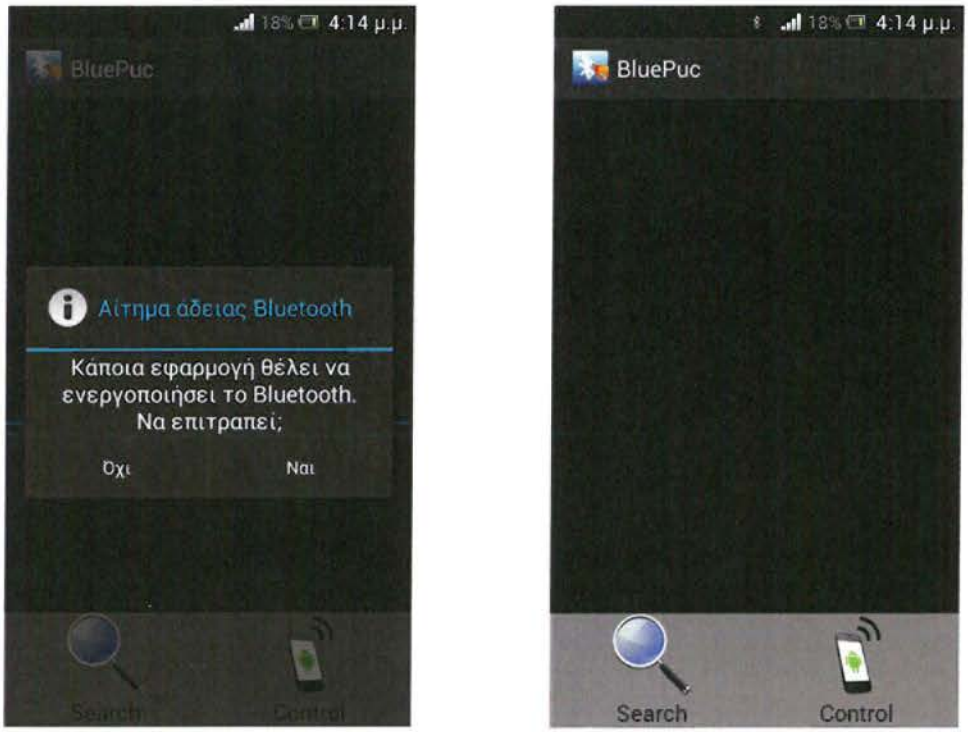


Σχήμα 6.1: Εγκατάσταση εφαρμογής

Μετά την εγκατάσταση της εφαρμογής βλέπουμε το εικονίδιο της στο μενού του τηλεφώνου, πατώντας στο εικονίδιο της εφαρμογής από το μενού εμφανίζεται η αρχική οθόνη.

6.2 Αναζήτηση συσκευών και σύνδεση

Μόλις ξεκινήσει η εφαρμογή θα εμφανιστεί η αρχική οθόνη. Η εφαρμογή χρειάζεται το Bluetooth ενεργό για να μπορέσουμε να συνεχίσουμε, σε αυτή την περίπτωση θα εμφανιστεί ένα παράθυρο (σχήμα 6.2-α) για την ενεργοποίηση του Bluetooth. Στο σημείο αυτό αν δεν επιλέξουμε την ενεργοποίηση τότε θα εμφανιστεί ένα παράθυρο ειδοποίησης (Alert Dialog) που ενημερώνει ότι η εφαρμογή χρειάζεται το Bluetooth και θα τερματίσει την εφαρμογή. Μόλις είναι ενεργό το Bluetooth θα εμφανιστεί η οθόνη όπως φαίνεται στο σχήμα 6.2-β.



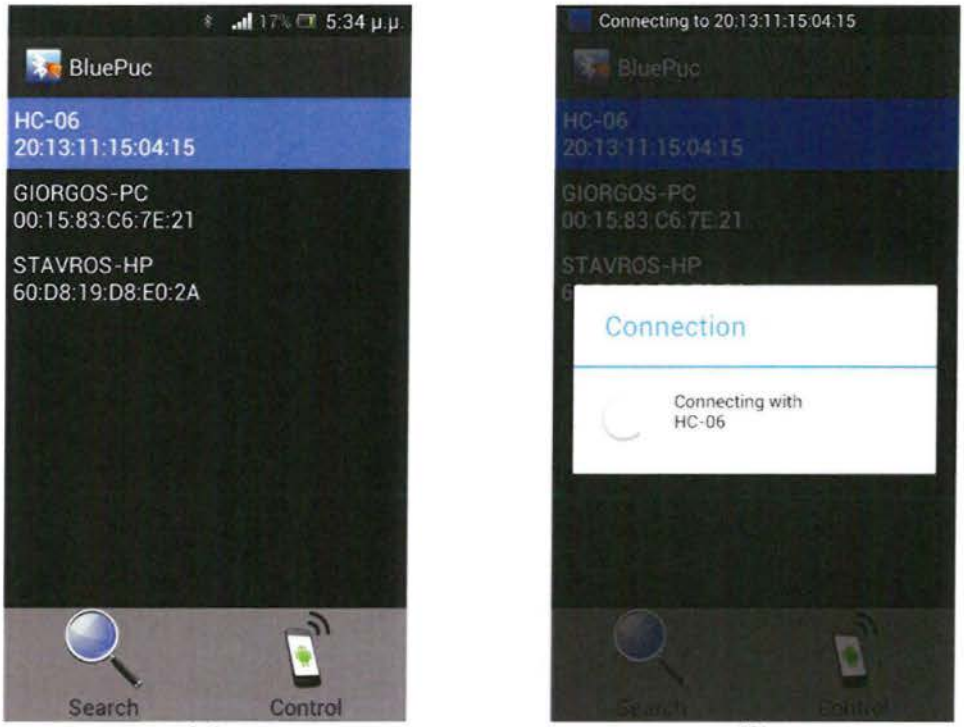
(α) (β)

Σχήμα 6.2: Αρχική οθόνη

Η αρχική οθόνη αποτελείται από δυο μέρη, το επάνω μέρος (η μαύρη περιοχή) είναι μια λίστα που εμφανίζονται οι συσκευές με ενεργό και ανιχνεύσιμο το Bluetooth. Στο κάτω μέρος βρίσκονται δυο κουμπιά, ένα για αναζήτηση συσκευών και ένα για την σύνδεση με την συσκευή που έχουμε επιλέξει.

Πατάμε το κουμπί Search για να εμφανίσει τις διαθέσιμες συσκευές κοντά στο κινητό (σχήμα 6.3-α). Για να μπορέσουμε να συνδεθούμε με μια συσκευή θα πρέπει να έχει γίνει σύζευξη, αν δεν το έχουμε κάνει μέσω των ρυθμίσεων του κινητού μπορούμε και μέσω της εφαρμογής.
Επιλέγουμε την συσκευή που θέλουμε να συνδεθούμε και πατάμε το κουμπί Control. Η πρώτη προσπάθεια θα αποτύχει καθώς δεν έχει γίνει ακόμα η σύζευξη, έτσι θα εμφανιστεί ένα μήνυμα Toast στο κάτω μέρος της οθόνης ενημερώνοντας ότι η συσκευή δεν είναι σε σύζευξη με το κινητό και θα εμφανιστεί ένα παράθυρο που θα ζητηθεί ο κωδικός. Μόλις δώσουμε το

κωδικό και πατήσουμε OK μπορούμε να συνδεθούμε. Η διαδικασία αυτή για την συγκεκριμένη συσκευή δεν χρειάζεται να επαναληφθεί καθώς το κινητό έχει αποθηκεύσει τον κωδικό.
Πατώντας το κουμπί Control θα ξεκινήσει η σύνδεση με την συσκευή, θα εμφανιστεί ένα παράθυρο προόδου (Progress Dialog) μέχρι να ολοκληρωθεί η σύνδεση και θα εμφανιστεί η επόμενη οθόνη (σχήμα 6.3-β). Στην γραμμή κατάστασης φαίνεται η ειδοποίηση της υπηρεσίας Amarino όπως είδαμε και στο κεφάλαιο 4.2.



(α) (β)

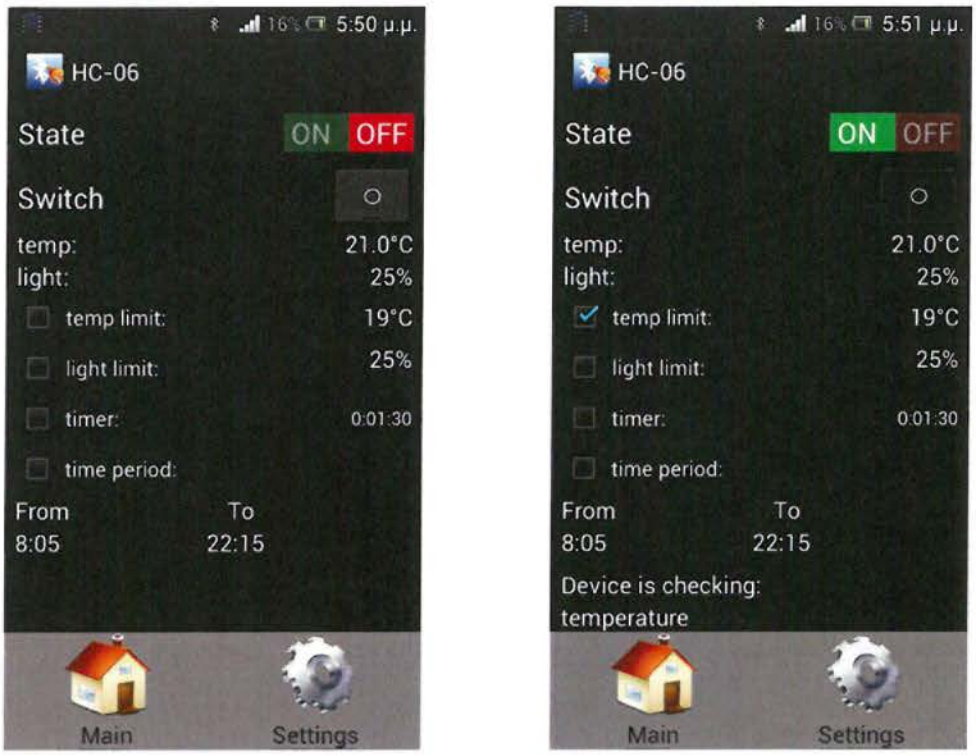
Σχήμα 6.3: Αναζήτηση συσκευών και σύνδεση

6.3 Έλεγχος της συσκευής

Μόλις ολοκληρωθεί η σύνδεση με την συσκευή εμφανίζεται η οθόνη ελέγχου. Στην οθόνη αυτή εμφανίζονται όλες οι πληροφορίες που λαμβάνουμε από την συσκευή και μπορούμε να ελέγξουμε πότε θα δίνει ρεύμα στην πρίζα.
Όπως η αρχική οθόνη έτσι και εδώ η οθόνη ελέγχου αποτελείται από δυο μέρη, στο επάνω μέρος εμφανίζονται οι πληροφορίες και τα κουμπιά ελέγχου της συσκευής, ενώ στο κάτω μέρος έχουμε δύο κουμπιά, ένα για επιστροφή στην αρχική οθόνη και ένα για μετάβαση στις ρυθμίσεις.

Στο πάνω μέρος της οθόνης βλέπουμε ότι ο τίτλος της δραστηριότητας είναι το όνομα της συσκευής που έχουμε συνδεθεί. Από κάτω βλέπουμε μια ένδειξη για την κατάσταση της συσκευής με δυο περιπτώσεις ON / OFF. Όταν η συσκευή δίνει ρεύμα στην πρίζα (κατάσταση ON) τότε το πράσινο εικονίδιο φαίνεται έντονα ενώ το κόκκινο είναι ποιο σκοτεινό, αντίθετα για την κατάσταση OFF το κόκκινο εικονίδιο φαίνεται έντονα και το πράσινο όχι.

Μετά βρίσκεται ένα TextView και ένα Toggle Button για τον διακόπτη. Από εδώ μπορούμε να ελέγξουμε την συσκευή, για να δίνει ή όχι ρεύμα στην πρίζα.



Σχήμα 6.4: Οθόνη ελέγχου

Κάτω από το διακόπτη εμφανίζονται οι πληροφορίες από τα αισθητήρια της συσκευής, η θερμοκρασία σε βαθμούς Κελσίου και η φωτεινότητα.
Κάτω από τις πληροφορίες αυτές είναι τα Checkbox ελέγχου για όριο θερμοκρασίας, όριο φωτεινότητας, χρονοδιακόπτη και χρονική περίοδο. Στα αριστερά είναι το Checkbox του ελέγχου ενώ στα δεξιά το όριο ή η τιμή που έχουμε θέσει. Τα όρια θερμοκρασίας και φωτεινότητας μπορούν να είναι μέγιστα ή ελάχιστα όπως θα δούμε και παρακάτω. Μπορούμε να επιλέξουμε ένα ενεργό έλεγχο κάθε φορά ή και συνδυασμό αυτών όπως φαίνεται και στο παρακάτω πίνακα.

α/α	Όριο θερμοκρασίας	Όριο φωτεινότητας	Χρονοδιακόπτης	Χρονική περίοδος
1	✓	–	–	–
2	–	✓	–	–
3	–	–	✓	–
4	–	–	–	✓
5	✓	✓	–	–
6	✓	–	✓	–
7	–	✓	✓	–

8	✓	✓	✓	–
9	✓	–	–	✓
10	–	✓	–	✓
11	✓	✓	–	✓

Πίνακας 6.1: Συνδυασμός επιλογών

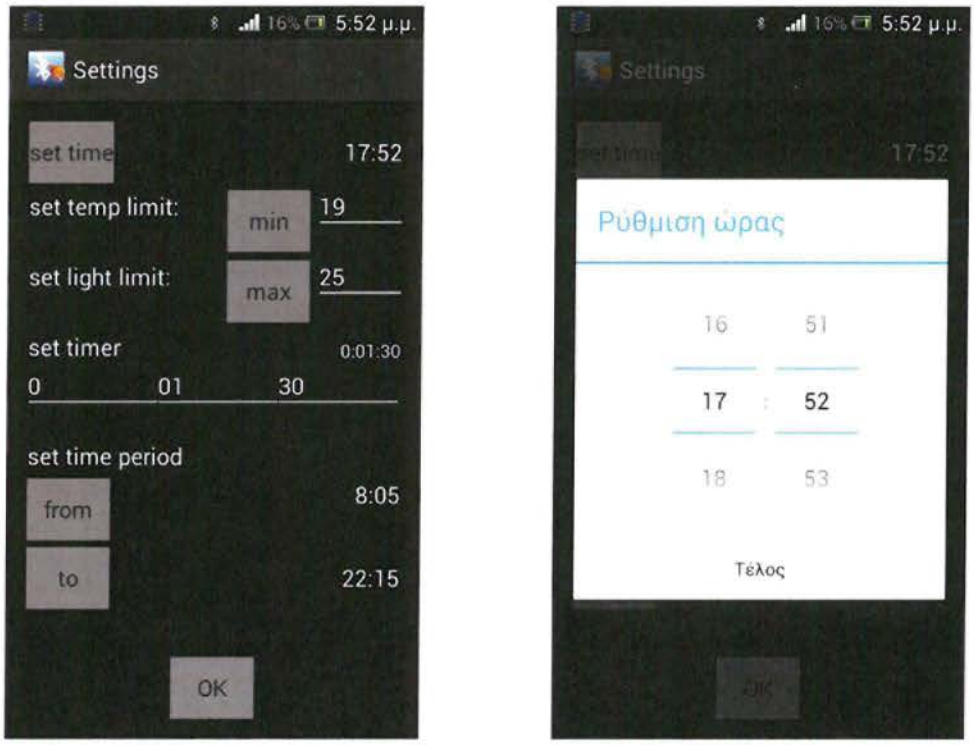
Όπως βλέπουμε δεν μπορούμε να επιλέξουμε ταυτόχρονα το χρονοδιακόπτη και την χρονική περίοδο. Όταν έχουμε επιλέξει ένα Checkbox τότε το Toggle Button απενεργοποιείται.
Αν έχουμε επιλέξει κάποιο από τους παραπάνω ελέγχους τότε στο μαύρο μέρος κάτω από τα Checkboxes θα εμφανιστεί ένα μήνυμα ενημερώνοντας μας για τους ελέγχους που κάνει η συσκευή (σχήμα 6.4 δεξιά). Αυτό είναι χρήσιμο σε περίπτωση που τερματίσουμε την εφαρμογή και επιστρέψουμε σε αυτήν να ξέρουμε τι ελέγχους είχαμε επιλέξει για την συσκευή. Έτσι αν έχουμε περισσότερες από μια συσκευές δεν χρειάζεται να θυμόμαστε τι είχαμε επιλέξει σε κάθε μια.

Στο κάτω μέρος της οθόνης βρίσκονται δυο κουμπιά, πατώντας το κουμπί Main θα αποσυνδεθούμε από την συσκευή και θα επιστρέψουμε στην αρχική σελίδα, ενώ με το κουμπί Settings θα εμφανιστεί η οθόνη με τις ρυθμίσεις.
Σε περίπτωση που η συσκευή πάψει να δουλεύει (π.χ έχει πέσει το ρεύμα) τότε θα εμφανιστεί ένα παράθυρο ειδοποίησης να ελέγξουμε την συσκευή.

Σχήμα 6.5: Ειδοποίηση προβλήματος συσκευής

6.4 Ρυθμίσεις

Στην οθόνη ρυθμίσεων ο χρήστης μπορεί να επιλέξει τις τιμές για τους ελέγχους. Μπορεί να ρυθμίσει την ώρα για την συσκευή, πατώντας το κουμπί set time θα εμφανιστεί ένα παράθυρο διαλόγου για ώρα (Time Picker Dialog). Τα όρια θερμοκρασίας και φωτεινότητας μπορούν να πάρουν τιμές από 0 ως 100 και να είναι μέγιστα ή ελάχιστα. Αν ένα όριο είναι μέγιστο τότε σημαίνει ότι η συσκευή θα δίνει ρεύμα στην πρίζα για τιμές από 0 έως την τιμή που δώσαμε, αντίθετα όταν το όριο είναι ελάχιστο θα δίνει ρεύμα για τιμές από την τιμή που δώσαμε έως τη μέγιστη.
Για τον χρονοδιακόπτη μπορούμε να δώσουμε τιμές για ώρα, λεπτά και δευτερόλεπτα. Η μέγιστη τιμή που μπορούμε να δώσουμε είναι 168 ώρες, ή αλλιώς 7 ημέρες. Τέλος μπορούμε να επιλέξουμε την αρχική και τελική ώρα για την χρονική περίοδο. Πατώντας το κουμπί OK θα επιστρέψουμε στην οθόνη ελέγχου. Οι τιμές που έχουμε δώσει δεν χάνονται μόλις τερματιστεί η εφαρμογή.



Σχήμα 6.6: Ρυθμίσεις

6.5 Ανάλυση λειτουργίας και σχεδιασμού της εφαρμογής

Παρακάτω θα δούμε ένα μέρος από το κώδικα των κλάσεων Java και τα Layouts. Στα αρχεία Java βρίσκονται οι εντολές που εκτελούνται κατά το χρόνο εκτέλεσης και καθορίζουν τον τρόπο λειτουργίας της εφαρμογής, ενώ τα αρχεία xml (Layouts) έχουν τα χαρακτηριστικά του γραφικού περιβάλλοντος που εμφανίζονται στην οθόνη του κινητού.

6.5.1 Layouts

Σε ένα Layout μπορούμε να τοποθετήσουμε αντικείμενα View όπως κουμπιά, πεδία κειμένου κλπ. Τα αντικείμενα αυτά μπορούν να συμπεριλαμβάνονται μέσα σε ένα αντικείμενο ViewGroup το οποίο καθορίζει τον τρόπο που θα εμφανιστούν αυτά. Δυο ViewGroup που χρησιμοποιούνται συνήθως είναι το Linear Layout το οποίο ευθυγραμμίζει τα αντικείμενα προς μια κατεύθυνση οριζόντια ή κάθετα και το Relative Layout στο οποίο προσδιορίζουμε την θέση ενός αντικειμένου σε σχέση με κάποιο άλλο (π.χ το αντικείμενο Α δεξιά του Β). Ένα ViewGroup μπορεί να είναι εμφωλευμένο σε άλλο διευκολύνοντας έτσι την σχεδίαση της οθόνης. Όλα τα αντικείμενα View/ViewGroup θα πρέπει να είναι εμφωλευμένα σε ένα ViewGroup που καταλαμβάνει όλη την οθόνη. Τα αντικείμενα View/ViewGroup έχουν ιδιότητες για την εμφάνιση και συμπεριφορά τους. Κάποιες από τις ιδιότητες αυτές είναι κοινές για όλα τα αντικείμενα, π.χ android:layout_height που καθορίζει το ύψος, android:layout_width το πλάτος, android:id ένας μοναδικός ακέραιος αριθμός που προσδιορίζει το αντικείμενο, ενώ κάποιες άλλες χρησιμοποιούνται μόνο από κάποιο αντικείμενο.
Στο σχήμα 6.7 φαίνεται το σχεδιάγραμμα της αρχικής σελίδας το οποίο αποτελείται από ένα Linear Layout, μέσα σε αυτό υπάρχει ένα ListView και ένα Linear Layout με δυο κουμπιά.

Σχήμα 6.7: Σχεδιάγραμμα αρχικής οθόνης

Ο κώδικας του αρχείου xml είναι:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/black"
    android:orientation="vertical"
    android:padding="0dp"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".Device_Scan" >

    <ListView
        android:id="@+id/listView1"
        android:layout_width="match_parent"
        android:layout_height="355dp"
        android:layout_weight="0.55" >
    </ListView>

```

```
21      <LinearLayout
22          android:layout_width="fill_parent"
23          android:layout_height="wrap_content"
24          android:orientation="horizontal"
25          android:weightSum="100" >
26
27          <Button
28              android:id="@+id/btnSearch"
29              android:layout_width="wrap_content"
30              android:layout_height="wrap_content"
31              android:layout_weight="50"
32              android:background="@color/gray"
33              android:drawableTop="@drawable/search_devices"
34              android:text="Search" />
35
36          <Button
37              android:id="@+id/btnControl"
38              android:layout_width="wrap_content"
39              android:layout_height="wrap_content"
40              android:layout_weight="50"
41              android:background="@color/gray"
42              android:drawableTop="@drawable/control_page"
43              android:text="Control" />
44      </LinearLayout>
45
46  </LinearLayout>
```

Για τις ιδιότητες ύψους και πλάτους μπορούμε να δώσουμε τιμές όπως match_parent / fill_parent που δηλώνουν ότι πρέπει να έχουν ίδιες διαστάσεις με το αντικείμενο-πατέρα, wrap_content δηλαδή το αντικείμενο θα έχει διαστάσεις όσο και το περιεχόμενο του, αριθμό pixel κλπ. Στα Linear Layouts η ιδιότητα android:orientation δηλώνει την κατεύθυνση που θα ευθυγραμμιστούν τα αντικείμενα (οριζόντια ή κάθετα). Η ιδιότητα android:layout_weight δηλώνει το χώρο που θα καταλάβει ένα αντικείμενο, αν έχει τιμή 0 τότε θα καταλάβει ακριβώς όσο και το περιεχόμενο του. Η ιδιότητα android:weightSum δηλώνει το συνολικό χώρο διαθέσιμο για τα αντικείμενα-παιδιά, στο Linear Layout με τα δυο κουμπιά έχει τιμή 100 και μετά το κάθε κουμπί έχει την ιδιότητα android:layout_weight ίση με 50, καταλαμβάνουν δηλαδή το χώρο ισάξια. Εκτός από αυτές τις ιδιότητες υπάρχουν και άλλες όπως για την απόσταση ανάμεσα στο περιεχόμενο και τα άκρα του αντικειμένου – android:layout_padding, για την απόσταση από τα άκρα ενός αντικειμένου με άλλα γύρω του – android:layout_margin, για το χρώμα ενός αντικειμένου – android:background, για το κείμενο που θα εμφανίζεται – android:text κλπ.

Οι οθόνες ελέγχου και ρυθμίσεων έχουν παρόμοια δομή, με την διαφορά ότι αντί για ListView έχουν ScrollView στο επάνω μέρος και στο κάτω μέρος της οθόνης τα αντίστοιχα κουμπιά. Το ScrollView έχει το πλεονέκτημα ότι αν το περιεχόμενο του δεν μπορεί να εμφανιστεί όλο στην οθόνη του κινητού τότε ο χρήστης μπορεί να κυλήσει προς κάποια κατεύθυνση (π.χ πάνω/ κάτω) για να δει και το υπόλοιπο.

Τα αντικείμενα ενός Layout μπορούν να τροποποιηθούν και μέσω εντολών στα αρχεία Java, κατά το χρόνο εκτέλεσης έτσι ώστε το Layout να μην είναι στατικό αλλά να μπορεί να μεταβληθεί μετά από κάποια ενέργεια.

6.5.2 Δραστηριότητα Αρχικής οθόνης

Φόρτωση Layout:

Το πρώτο πράγμα που πρέπει να κάνουμε σε κάθε δραστηριότητα είναι να φορτώσουμε το Layout που τις αντιστοιχεί με την μέθοδο setContentView() μέσα στην onCreate():

```
62      @Override
63      protected void onCreate(Bundle savedInstanceState) {
64          super.onCreate(savedInstanceState);
65          setContentView(R.layout.activity_device__scan);
```

Ενεργοποίηση του Bluetooth:

Η εφαρμογή όπως είδαμε χρειάζεται το Bluetooth για να λειτουργήσει, οπότε θα πρέπει να ελέγξουμε πρώτα αν η συσκευή διαθέτει Bluetooth και εάν έχει τότε να το ενεργοποιήσουμε. Η κλάση BluetoothAdapter μας επιτρέπει να κάνουμε τις βασικές λειτουργίες όπως αναζήτηση συσκευών, εμφάνιση των συσκευών με σύζευξη και σύνδεση με συσκευή. Η αρχικοποίηση του BluetoothAdapter είναι:

```
50      // the local bluetooth adapter
51      BluetoothAdapter btadapter;
68          btadapter = BluetoothAdapter.getDefaultAdapter();
```

Αν η τιμή που επιστρέψει η getDefaultAdapter() είναι μηδέν τότε το κινητό δεν υποστηρίζει Bluetooth, εμφανίζουμε ένα μήνυμα Toast και τερματίζουμε την εφαρμογή. Σε αντίθετη περίπτωση ελέγχουμε αν είναι ενεργοποιημένο. Αν δεν είναι ενεργό τότε καλούμε την startActivityForResult() με Intent το ACTION_REQUEST_ENABLE. Η σταθερά REQUEST_BLUETOOTH_ENABLE είναι ένα ακέραιος αριθμός μεγαλύτερος του 0.

```
69          if (btadapter == null) {
70              //  no bluetooth module found!
71              Toast.makeText(this, "Sorry, no bluetooth module found!",Toast.LENGTH_SHORT).show();
72              finish();
73          } else {
74              if (!btadapter.isEnabled()) {
75                  startActivityForResult(new Intent(
76                          BluetoothAdapter.ACTION_REQUEST_ENABLE),REQUEST_BLUETOOTH_ENABLE);
77              }
78          }
```

Το αποτέλεσμα θα είναι να εμφανιστεί ένα παράθυρο διαλόγου (σχήμα 6.2-α).

Εμφάνιση συσκευών στο ListView:

Όπως είδαμε και πριν στο επάνω μέρος της οθόνης βρίσκεται ένα ListView στο οποίο η κάθε γραμμή αντιστοιχεί σε ένα αντικείμενο View, εδώ ένα μόνο TextView. Για να εμφανίσουμε τις συσκευές στο ListView χρειαζόμαστε ένα ArrayAdapter. Θα χρειαστούμε επίσης μια λίστα – List με αντικείμενα BluetoothDevice.Η δήλωση τους είναι:

```
55      // view adapter for the list of devices
56      ArrayAdapter<String> devicesArrayAdapter;
57
58      List<BluetoothDevice> tmpBtChecker = new ArrayList<BluetoothDevice>();
```

Θα δημιουργήσουμε τον ArrayAdapter ο οποίος θα φορτώσει την πληροφορία σε ένα αρχείο xml με ένα μόνο TextView. Μετά θα φορτώσουμε σε ένα αντικείμενο τύπου ListView (το devicesListView) το ListView από το layout και μετά θέτουμε τα δεδομένα για την ListView με τον ArrayAdapter.

```
84          // create the list view adapter
85          devicesArrayAdapter = new ArrayAdapter<String>(this,R.layout.device_name);
86
87          devicesListView = (ListView) findViewById(R.id.listView1);
88          // set the adapter
89          devicesListView.setAdapter(devicesArrayAdapter);
```

Αναζήτηση συσκευών:

Για να ξεκινήσει η αναζήτηση συσκευών πρέπει να πατήσει ο χρήστης το κουμπί Search. Για να ξέρουμε πότε έχει πατηθεί το κουμπί πρέπει πρώτα να έχουμε φορτώσει σε ένα αντικείμενο τύπου Button (startDiscoveryButton) το View του από το Layout και μετά να του θέσουμε έναν OnClickListener με την μέθοδο OnClick η οποία καλείται μόλις πατηθεί το κουμπί. Μέσα στην μέθοδο OnClick καθαρίζουμε από τον ArrayAdapter και την προσωρινή λίστα ότι δεδομένα είχανε, ξεκινάμε την αναζήτηση συσκευών, απενεργοποιούμε το κουμπί και αλλάζουμε το όνομα του σε Searching για όσο διαρκεί η αναζήτηση.

```
119         startDiscoveryButton = (Button) findViewById(R.id.btnSearch);
120         startDiscoveryButton.setOnClickListener(new OnClickListener() {
121
122             @Override
123             public void onClick(View arg0) {
124                 startDiscovery();
125             }
126         });
```

```
206     public void startDiscovery() {
207
208         //Clear any existing data
209         devicesArrayAdapter.clear();
210         tmpBtChecker.clear();
211
212         //Start the device discovery process
213         btadapter.startDiscovery();
214
215         startDiscoveryButton.setEnabled(false);
216         startDiscoveryButton.setText("Searching");
217     }
```

Λήψη μηνυμάτων με Broadcast Receiver:

Μέχρι στιγμής είδαμε πως ξεκινάμε την αναζήτηση και πως θα εμφανίσουμε τα αποτελέσματα. Για να ξέρουμε πότε βρέθηκε νέα συσκευή, πότε ολοκληρώθηκε η αναζήτηση κλπ θα χρειαστούμε έναν Broadcast Receiver. Ο Broadcast Receiver λαμβάνει εκπεμπόμενα μηνύματα από το σύστημα και από την εφαρμογή Amarino, οπότε πρέπει να δηλώσουμε ποια μηνύματα θέλουμε να λαμβάνει. Ποιο συγκεκριμένα μας ενδιαφέρουν οι ενέργειες: εύρεση συσκευής, τέλος αναζήτησης, σύνδεση με την συσκευή και αποτυχία σύνδεσης με την συσκευή. Μέσα στην μέθοδο onStart() της δραστηριότητας θα εγγράψουμε τον Broadcast Receiver.

```
190-    private void setup() {
191
192         IntentFilter filter = new IntentFilter(BluetoothDevice.ACTION_FOUND);
193         this.registerReceiver(discoveryReceiver, filter);
194
195         filter = new IntentFilter(BluetoothAdapter.ACTION_DISCOVERY_FINISHED);
196         this.registerReceiver(discoveryReceiver, filter);
197
198         filter = new IntentFilter(AmarinoIntent.ACTION_CONNECTED);
199         this.registerReceiver(discoveryReceiver, filter);
200
201
202         filter = new IntentFilter(AmarinoIntent.ACTION_CONNECTION_FAILED);
203         this.registerReceiver(discoveryReceiver, filter);
204
205         check_reg=true; //if we registered check_reg->true
206     }
```

Δημιουργούμε τον Broadcast Receiver με την μέθοδο onReceive() που καλείται μόλις λάβει ένα intent (μήνυμα), αποθηκεύουμε σε μια μεταβλητή τύπου String την ενέργεια του μηνύματος.

```
309-    BroadcastReceiver discoveryReceiver = new BroadcastReceiver() {
310-        @Override
311         public void onReceive(Context context, Intent intent) {
312
313             String action = intent.getAction();
```

Στην περίπτωση που βρέθηκε μια συσκευή κατά την αναζήτηση, ελέγχουμε αν η ενέργεια αντιστοιχεί στην ενέργεια ACTION_FOUND. Αν ναι αποθηκεύουμε σε ένα αντικείμενο τύπου BluetoothDevice (device) τις πληροφορίες της συσκευής. Η κλάση BluetoothDevice αντιπροσωπεύει μια συσκευή Bluetooth και παρέχει πληροφορίες όπως το όνομα της συσκευής, την διεύθυνση της, την κατάσταση σύζευξης. Τη λίστα που είχαμε δημιουργήσει (tmpBtChecker) θα την χρησιμοποιήσουμε για δυο λόγους, εδώ ελέγχουμε αν η λίστα έχει την συσκευή που βρέθηκε, αν δεν υπάρχει στην λίστα τότε προσθέτουμε την συσκευή στην λίστα και στον ArrayAdapter το όνομα και την διεύθυνση της. Η λίστα χρησιμοποιείται για έλεγχο προτού περάσουμε την πληροφορία στον ArrayAdapter καθώς σε μερικές συσκευές ο Broadcast Receiver λαμβάνει παραπάνω από μια φορές το intent με ACTION_FOUND για την ίδια συσκευή.

```
315             if (BluetoothDevice.ACTION_FOUND.equals(action)) {
316                 device = intent.getParcelableExtra(BluetoothDevice.EXTRA_DEVICE);
317
318                 if (!tmpBtChecker.contains(device)) {
319                     tmpBtChecker.add(device);
320                     devicesArrayAdapter.add(device.getName() + "\n"
321                             + device.getAddress());
322                 }
323
324             }
```

Όπως είδαμε και προηγουμένως για να εμφανιστεί η οθόνη ελέγχου πρέπει πρώτα να έχει γίνει η σύνδεση με την συσκευή. Για να ελέγξουμε πότε έχει γίνει αυτό ελέγχουμε αν η ενέργεια του intent είναι η ACTION_CONNECTED, κλείνουμε το παράθυρο προόδου (σχήμα 6.3-β) και καλούμε την μέθοδο connectionOk(). Στην μέθοδο αυτή όπως θα δούμε και παρακάτω ξεκινάμε την δραστηριότητα της οθόνης ελέγχου και περνάμε σε αυτήν χρήσιμες πληροφορίες.

```
337             // if the connection was made
338             if(AmarinoIntent.ACTION_CONNECTED.equals(action)){
339                 connectionDialog.dismiss();
340                 connectionOk();
341             }
```

Εφόσον κάναμε register τον Broadcast Receiver στην onStart() δεν πρέπει να ξεχάσουμε να κάνουμε unregister στην onStop(). Σταματάμε επίσης την αναζήτηση συσκευών.

```
158     @Override
159     protected void onStop() {
160         super.onStop();
161         btadapter.cancelDiscovery();
162         if (check_reg==true){
163             this.unregisterReceiver(discoveryReceiver);
164         }
165
166     }
```

Επιλογή συσκευής για σύνδεση:

Μόλις τελειώσει η αναζήτηση και εμφανιστούν στην οθόνη οι συσκευές Bluetooth ο χρήστης πρέπει να επιλέξει μια για να συνδεθεί. Για να ξέρουμε ποια συσκευή έχει επιλέξει θα χρειαστούμε έναν OnItemClickListener με την μέθοδο onItemClick(). Το πρώτο πράγμα που κάνουμε μόλις ο χρήστης επιλέξει μια συσκευή είναι να σταματήσουμε την διαδικασία αναζήτησης. Μαρκάρουμε την συσκευή που επέλεξε για διευκόλυνση. Μετατρέπουμε το κείμενο από το TextView και το αποθηκεύουμε σε μια μεταβλητή τύπου String (info) με το όνομα και την MAC διεύθυνση της συσκευής και μετά σε άλλα δυο ξεχωριστά τις πληροφορίες αυτές. Κρατάμε σε μια μεταβλητή και την θέση του View για να μπορέσουμε να ελέγξουμε αργότερα αν η συσκευή αυτή είναι σε σύζευξη ή όχι.

```
276     @Override
277     public void onItemClick(AdapterView<?> arg0, View arg1, int arg2, long arg3) {
278         // Cancel discovery
279         btadapter.cancelDiscovery();
280
281
282             if(prevposition==arg2&&arg2==0){
283                 arg1.setBackgroundColor(Color.rgb(30,144,255));
284             }
285
286
287             if (arg2!=prevposition){
288                 arg1.setBackgroundColor(Color.rgb(30,144,255));
289                 View notChecked = devicesListView.getChildAt(prevposition);
290                 notChecked.setBackgroundColor(Color.BLACK);
291                 prevposition=arg2;
292             }
293
294                 // Get the device MAC address, which is the last 17 chars in the View
295                 String info = ((TextView) arg1).getText().toString();
296                 address = info.substring(info.length() - 17);
297
298
299                 // Get the device name
300                 int telosName = info.length()-17;
301                 deviceName = info.substring(0, telosName);
302
303                 //Get the position of the device in the adapter
304                 position=arg2;
305
306     }
```

Σύνδεση με την συσκευή:

Μετά την επιλογή της συσκευής ο χρήστης πρέπει να πατήσει το κουμπί Control. Η θέση της συσκευής στον AdapterView είναι ίδια με την θέση στην λίστα tmpBtChecker, έτσι μπορούμε να ελέγξουμε αν η συσκευή αυτή είναι σε σύζευξη με το κινητό. Αν δεν είναι τότε η σύνδεση θα αποτύχει, ενημερώνουμε τον χρήστη και θα εμφανιστεί ένα παράθυρο από το σύστημα που θα πρέπει να δώσουμε τον κωδικό σύζευξης. Αν όλα είναι εντάξει τότε μόλις πατήσουμε το κουμπί Control θα ξεκινήσει η σύνδεση και θα εμφανιστεί στην οθόνη ένα παράθυρο προόδου μέχρι να ολοκληρωθεί η σύνδεση.

```
221     public void  ToControlPage(){
222
223         if (address!=null){
224
225             deviceCheck = tmpBtChecker.get(position);
226             //Check if the device we selected is paired
227             if (deviceCheck.getBondState()==BluetoothDevice.BOND_NONE){
228                 Amarino.connect(this,address);
229                 Toast.makeText(getBaseContext(),"The device "+deviceName+" is not paired",Toast.LENGTH_LONG).show();
230
231             }else {
232                 Amarino.connect(this,address);
233                 connectionDialog.setTitle("Connection");
234                 connectionDialog.setMessage("Connecting with\n"+deviceName);
235                 connectionDialog.setProgressStyle(ProgressDialog.STYLE_SPINNER);
236                 connectionDialog.setIndeterminate(true);
237                 connectionDialog.show();
238             }
```

Έναρξη επόμενης δραστηριότητας:

Μόλις ολοκληρωθεί η σύνδεση ο Broadcast Receiver θα λάβει το μήνυμα από το Amarino. Στο σημείο αυτό θα πρέπει να περάσουμε το όνομα και την MAC διεύθυνση της συσκευής που έχουμε συνδεθεί στην δραστηριότητα της οθόνης ελέγχου. Για να γίνει αυτό δημιουργούμε ένα Bundle, ένα 'πακέτο' που μια μεταβλητή αντιστοιχεί σε ένα κλειδί. Προσθέτουμε τις μεταβλητές με τα αντίστοιχα κλειδιά τους. Δημιουργούμε το intent που θα ξεκινήσει την δραστηριότητα της οθόνης ελέγχου και θα μεταφέρει τις πληροφορίες. Προσθέτουμε στο intent το Bundle και ξεκινάμε την δραστηριότητα.

```
260-    public void connectionOk(){
261
262         Bundle device_info = new Bundle();
263
264         //add the data to Bundle
265         device_info.putString("mac_dieuthinsi", address);
266         device_info.putString("dev_name", deviceName);
267
268         Intent toControlPage = new Intent(this, Control_page.class);
269
270         //add data to the intent
271         toControlPage.putExtras(device_info);
272
273         startActivity(toControlPage);
274     }
```

6.5.3 Δραστηριότητα οθόνης ελέγχου

Λήψη δεδομένων από δραστηριότητα:

Στην δραστηριότητα της αρχικής οθόνης βάλαμε τις πληροφορίες που θέλαμε να μεταφέρουμε σε ένα Bundle και τις στείλαμε μέσω του intent. Στην δραστηριότητα αυτής της οθόνης για να παραλάβουμε τις πληροφορίες δημιουργούμε ένα Bundle και βάζουμε σε αυτό τα δεδομένα που μετέφερε το intent. Οι μεταβλητές που στείλαμε ήτανε τύπου String οπότε για να τις παραλάβουμε χρησιμοποιούμε την μέθοδο getString() με παράμετρο το κλειδί της κάθε μεταβλητής.

```
566             //the information to receive from the intent
567             Bundle getDevInfo = getIntent().getExtras();
568
569             if (getDevInfo!=null){
570                 //the name of the device that is connected
571                 got_dev_name= getDevInfo.getString("dev_name");
572                 //and the MAC address
573                 got_mac_a= getDevInfo.getString("mac_dieuthinsi");
574             }
```

Αλλάζουμε τον τίτλο της δραστηριότητας, να έχει το όνομα της συσκευής που έχουμε συνδεθεί, για διευκόλυνση του χρήστη. Σε περίπτωση που υπάρχουν περισσότερες από μια συσκευές να μην χρειάζεται να επιστρέψει στην αρχική οθόνη για να δει με ποια έχει συνδεθεί.

```
83          //change the title to the device name
84          setTitle(got_dev_name);
```

Έλεγχος της συσκευής:

Για να ελέγξουμε την συσκευή μπορούμε να επιλέξουμε κάποιον έλεγχο με τα Checkboxes ή το Toggle Button. Στην αρχική οθόνη είδαμε πως ένα κουμπί έχει έναν Listener σε αυτήν την οθόνη θα δούμε πως μπορούμε να έχουμε έναν Listener για περισσότερα κουμπιά. Για τον έλεγχο των Checkboxes και του Toggle Button χρειαζόμαστε έναν OnCheckedChangedListener με την μέθοδο onCheckedChanged(). Αν ο χρήστης επιλέξει τον έλεγχο θερμοκρασίας στέλνουμε πρώτα το όριο και μετά έναν πίνακα με τις καταστάσεις των Checkboxes. Η τιμή 1 αντιστοιχεί σε επιλεγμένο ενώ το 0 το αντίθετο. Με βάση αυτές τις τιμές το Arduino θα μπορέσει να ξεχωρίσει ποιους ελέγχους πρέπει να κάνει.

```
161     @Override
162     public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
163         switch(buttonView.getId()){
164         case R.id.checkBoxTempLimit:
165             if (isChecked){
166                 chkbTempLimit.setButtonDrawable(R.drawable.ic_check_on);
167
168                 //send the limit
169                 Amarino.sendDataToArduino(this, got_mac_a, 't', tempLimit);
170
171                 chkb_state[0]=1;
172                 Amarino.sendDataToArduino(this,got_mac_a,'d',chkb_state);
173
174             }else{
175                 chkbTempLimit.setButtonDrawable(R.drawable.btn_check_off_normal_holo_dark);
176
177                 chkb_state[0]=0;
178                 Amarino.sendDataToArduino(this,got_mac_a,'d',chkb_state);
179
180             }
181         break;
```

Σε κάθε περίπτωση αλλάζουμε και το εικονίδιο για την αντίστοιχη περίπτωση. Στην περίπτωση του Toggle Button στέλνουμε μια τιμή, 1 για ON και 0 για OFF.

```
249         case R.id.toggleButton1:
250             if (isChecked){
251                 state_on_off=1;
252                 Amarino.sendDataToArduino(this,got_mac_a,'i',state_on_off);
253
254             }else{
255                 state_on_off=0;
256                 Amarino.sendDataToArduino(this,got_mac_a,'i',state_on_off);
257             }
258         break;
```

Ο χρήστης δεν μπορεί να επιλέξει ταυτόχρονα το Checkbox του χρονοδιακόπτη και της χρονικής περιόδου, για να γίνει αυτό ελέγχουμε μόλις πατηθεί ένα από τα δυο και αφαιρούμε την επιλογή από το άλλο. Ο έλεγχος αυτός γίνεται με την μέθοδο onClick() του OnClickListener .

```
121         case R.id.checkBoxTimer:
122             if (chkbTimer.isChecked()){
123                 chkbTimePeriod.setChecked(false);
124             }
125             break;
126         case R.id.checkBoxTimePeriod:
127             if (chkbTimePeriod.isChecked()){
128                 chkbTimer.setChecked(false);
129             }
130             break;
```

Λήψη μηνυμάτων από την συσκευή:

Για να λαμβάνουμε τις πληροφορίες που στέλνει η συσκευή, όπως θερμοκρασία, φωτεινότητα κλπ θα χρειαστεί να εγγράψουμε έναν Broadcast Receiver για μηνύματα με ενέργεια ACTION_RECEIVED μέσα στην μέθοδο onStart(). Αφού δημιουργήσουμε τον Broadcast Receiver μέσα στην μέθοδο onReceive() μηδενίζουμε μια μεταβλητή που χρησιμοποιούμε σε έναν χρονοδιακόπτη ελέγχου. Για τον χρονοδιακόπτη αυτόν θα δούμε παρακάτω τον τρόπο λειτουργίας του. Αποθηκεύουμε σε μια μεταβλητή (dataType) τον τύπο των δεδομένων που λάβαμε. Μέχρι στιγμής το Amarino υποστηρίζει μόνο δεδομένα τύπου String, οπότε αποθηκεύουμε σε μια μεταβλητή τύπου String (data) τα δεδομένα από το μήνυμα με την μέθοδο getStringExtra().

Το Arduino όπως θα δούμε και παρακάτω στέλνει ποιους ελέγχους κάνει και αν είναι σε κατάσταση ON ή OFF. Με βάση τον πίνακα 6.1 για κάθε περίπτωση το Arduino στέλνει ένα μήνυμα 'scase' και τον αριθμό περίπτωσης. Αν είναι σε κατάσταση ON τότε στέλνει ένα μήνυμα 'son' αλλιώς 'soff'.

Όταν ο Broadcast Receiver λάβει ένα μήνυμα θα πρέπει να ελέγξουμε ποια περίπτωση είναι. Για παράδειγμα για την περίπτωση 1 απενεργοποιούμε το Toggle Button και εμφανίζουμε ένα μήνυμα στην οθόνη ενημερώνοντας τον χρήστη ότι η συσκευή ελέγχει την θερμοκρασία.

```
357             if (AmarinoIntent.ACTION_RECEIVED.equals(intent.getAction())){
358                  data = null;
359
360                 // the type of data which is added to the intent
361                 final int dataType = intent.getIntExtra(AmarinoIntent.EXTRA_DATA_TYPE, -1);
362
363
364                  try{
365                  if (dataType == AmarinoIntent.STRING_EXTRA){
366                         data = intent.getStringExtra(AmarinoIntent.EXTRA_DATA);
367                         if (data != null){
368                             if(data.equals("scase1")){
369                                 switch_btn.setChecked(false);
370                                 switch_btn.setEnabled(false);
371                                 displayMsg.setText("Device is checking:");
372                                 displayMsgChecked.setText("temperature");
```

Στο επάνω μέρος της οθόνης ελέγχου υπάρχει η ένδειξη για την κατάσταση της συσκευής. Για να αλλάξουμε το χρώμα ενός TextView χρησιμοποιούμε την μέθοδο setBackgroundColor() και για να αλλάξουμε το χρώμα του κειμένου την μέθοδο setTextColor().

```
425                         else if(data.equals("son")){  // if in status on
426                             displayStateON.setBackgroundColor(Color.rgb(0,190 ,0 )); //green
427                             displayStateON.setTextColor(Color.rgb(255,255 ,255 )); // white
428                             displayStateOFF.setBackgroundColor(Color.rgb(102,0 ,0 )); // dark red
429                             displayStateOFF.setTextColor(Color.rgb(160,160 ,160 )); // / light gray
430                         }else if (data.equals("soff")){// if in status off
431                             displayStateOFF.setBackgroundColor(Color.rgb(255,0 ,0 )); // red
432                             displayStateOFF.setTextColor(Color.rgb(255,255,255)); // white
433                             displayStateON.setBackgroundColor(Color.rgb(0,70 ,0 )); //dark green
434                             displayStateON.setTextColor(Color.rgb(160,160 ,160 )); //light gray
```

Χρονοδιακόπτης ελέγχου:

Αν ο χρήστης δεν βρίσκεται κοντά στην συσκευή και αυτή σταματήσει να δουλεύει για κάποιο λόγο, π.χ πέσει το ρεύμα, τότε πρέπει να ενημερωθεί. Για το λόγο αυτό χρησιμοποιούμε έναν χρονοδιακόπτη για έλεγχο. Ο χρονοδιακόπτης αυτός τρέχει σε διαφορετικό νήμα, οπότε αν θέλουμε να εμφανίσουμε κάτι στο UI νήμα τότε θα πρέπει να βάλουμε τον κώδικα που αντιστοιχεί για το παράθυρο ειδοποίησης μέσα στην μέθοδο runOnUiThread().

```
459-    public void startTimer(){
460-        last_received.scheduleAtFixedRate(new TimerTask(){
461
462-             @Override
463              public void run() {
464-                 runOnUiThread(new Runnable() {
465
466-                     @Override
467                      public void run() {
```

Μόλις ξεκινήσει ο χρονοδιακόπτης θα αυξάνει την τιμή μια μεταβλητής (counter) κατά 1 κάθε δευτερόλεπτο. Αν ο Broadcast Receiver έλαβε ένα μήνυμα από το Arduino τότε μηδενίζει την τιμή αυτή. Αν η δραστηριότητα σταματήσει τότε ο Broadcast Receiver δεν λαμβάνει μηνύματα, άρα σε αυτή την περίπτωση δεν αυξάνουμε την τιμή του χρονοδιακόπτη αλλά την μηδενίζουμε (γραμμή 490).

Αν περάσουν 3 δευτερόλεπτα και δεν έχουμε λάβει κάτι τότε σταματάμε τον χρονοδιακόπτη, δημιουργούμε και εμφανίζουμε το παράθυρο ειδοποίησης. Στο παράθυρο αυτό υπάρχει ένα κουμπί OK, πατώντας το θα τερματίσει την δραστηριότητα και θα επιστρέψουμε στην αρχική οθόνη.

```
                        if (counter_stoped==false){

                            counter++;

                             if (counter>2){
                              last_received.cancel();
                              AlertDialog.Builder device_problem = new AlertDialog.Builder(Control_page.this
                              device_problem.setTitle("Attention!")
                                     .setMessage("Check the device please")
                                     .setIcon(R.drawable.alert_gray)
                                     .setCancelable(false)
                                     .setPositiveButton("OK",new DialogInterface.OnClickListener() {
                                        @Override
                                        public void onClick(DialogInterface dialog, int which) {
                                            dialog.cancel();
                                            disconnectFromAdruino();
                                            Control_page.this.finish();
                                        }
                                    });
                              device_problem.show();
                             }
                        }else{
                            counter=0;
                        }
                    }
                });
            }
        },0,1000);
```

Έναρξη δραστηριότητας για αποτελέσματα:

Αν ο χρήστης πατήσει το κουμπί Settings τότε θα εμφανιστεί η οθόνη ρυθμίσεων. Στην δραστηριότητα της οθόνης ρυθμίσεων θα χρειαστούμε την MAC διεύθυνση της συσκευής που έχουμε συνδεθεί για να μπορέσει να ενημερώσει την συσκευή με τις νέες τιμές. Σε αυτή την περίπτωση για να ξεκινήσουμε την δραστηριότητα θα χρησιμοποιήσουμε την μέθοδο startActivityForResult() με παραμέτρους το intent που θα ξεκινήσει την δραστηριότητα της οθόνης ελέγχου και μια σταθερά η οποία πρέπει να έχει τιμή μεγαλύτερη ή ίση του 0.

```
Bundle mac_address = new Bundle();
mac_address.putString("mac_address", got_mac_a);
Intent toSettingsPage = new Intent(this,Settings_page.class);
toSettingsPage.putExtras(mac_address);
startActivityForResult(toSettingsPage, SETTINGS_RESULT);
this.overridePendingTransition(R.anim.slide_in_left_to_right,R.anim.slide_out_right_to_left );
```

Λήψη αποτελεσμάτων:

Όταν τελειώσουμε την επιλογή τιμών στην οθόνη ρυθμίσεων και επιστρέψουμε στην οθόνη ελέγχου τότε η μέθοδος onActivityResult() καλείται για να μπορέσουμε να παραλάβουμε τις πληροφορίες που έστειλε η οθόνη ρυθμίσεων. Αν το resultCode είναι ίδιο με αυτό που θέσαμε στην οθόνη ρυθμίσεων και το requestCode ίδιο με αυτό που θέσαμε στην μέθοδο startActivityForResult() για την έναρξη της οθόνης ρυθμίσεων τότε μπορούμε να παραλάβουμε τα δεδομένα από το intent. Με την μέθοδο displayValues() εμφανίζουμε στα TextViews τις καινούργιες τιμές.

```
137-    @Override
138     protected void onActivityResult(int requestCode, int resultCode, Intent data) {
139
140         super.onActivityResult(requestCode, resultCode, data);
141         if (resultCode == RESULT_OK && requestCode==SETTINGS_RESULT){
142
143                 tempLimit = data.getExtras().getInt("tempVal");
144                 lightLimit=data.getExtras().getInt("lightVal");
145
146                 timer_h = data.getExtras().getInt("timerHval");
147                 timer_m = data.getExtras().getInt("timerMval");
148                 timer_s = data.getExtras().getInt("timerSval");
149
150                 from_h = data.getExtras().getInt("fromHval");
151                 from_m = data.getExtras().getInt("fromMval");
152
153                 to_h = data.getExtras().getInt("toHval");
154                 to_m  = data.getExtras().getInt("toMval");
155
156                 displayValues();
157
158         }
159     }
```

6.5.4 Δραστηριότητα οθόνης ρυθμίσεων

Επιλογή ώρας:

Στην οθόνη αυτή ο χρήστης μπορεί να δώσει τιμές για τους ελέγχους και να ενημερώσει το Arduino για να έχει την τρέχουσα ώρα. Για την επιλογή ώρας θα χρησιμοποιήσουμε ένα παράθυρο διαλόγου, το Time Picker Dialog. Μόλις πατήσει ο χρήστης το κουμπί set time καλείται η μέθοδος showDialog() με παράμετρο μια σταθερά για να ξεχωρίσουμε ποιο παράθυρο διαλόγου θα δημιουργήσουμε. Για την δημιουργία του Dialog καλείται η μέθοδος onCreateDialog() με παράμετρο έναν ακέραιο αριθμό που αντιστοιχεί στο Dialog που θέλουμε να φτιάξουμε. Δημιουργούμε το Time Picker Dialog με παραμέτρους το τρέχον αντικείμενο, έναν OnTimeSetListener για να γνωρίζουμε πότε έχει τελειώσει με την επιλογή ώρας, την ώρα, τα λεπτά και τον τρόπο εμφάνισης της ώρας.

```
507-    @Override
508     protected Dialog onCreateDialog(int id)
509     {
510     switch (id) {
511     case TIME_DIALOG_ID:
512         return new TimePickerDialog(
513                 this, hTimeSetListener, hour, minute, true);
```

Για διευκόλυνση του χρήστη το Time Picker Dialog εμφανίζεται με την τρέχουσα ώρα, για να γίνει αυτό χρησιμοποιούμε την κλάση Calendar. Δημιουργούμε ένα αντικείμενο της και με την μέθοδο get() φορτώνουμε στις αντίστοιχες μεταβλητές την τρέχουσα ώρα και λεπτά.

```
36      final Calendar c = Calendar.getInstance();
37      int hour = c.get(Calendar.HOUR_OF_DAY);
38      int minute = c.get(Calendar.MINUTE);
```

Για να αποθηκεύσουμε την επιλογή του χρήστη χρησιμοποιούμε την μέθοδο onTimeSet(). Αποθηκεύουμε σε έναν πίνακα τις τιμές για την ώρα και τα λεπτά, αυτός ο πίνακας είναι που θα στείλουμε στο Arduino και εμφανίζουμε στο Text View ώρας την επιλογή του χρήστη.

```
528     // for time
529     private TimePickerDialog.OnTimeSetListener hTimeSetListener = new TimePickerDialog.OnTimeSetListener(){
530         public void onTimeSet(TimePicker view, int hourOfDay, int minuteOfHour){
531             hour = hourOfDay;
532             minute = minuteOfHour;
533
534             hm_array[0]=hour;
535             hm_array[1]=minute;
536
537
538             if (minute<=9){
539                 displayTime.setText(""+hour+":"+"0"+minute);
540             }else{
541                 displayTime.setText(""+hour+":"+minute);
542             }
543
544         }
545     };
```

Έλεγχος πεδίων κειμένου:

Ο χρήστης μπορεί να δώσει τις τιμές για τα όρια θερμοκρασίας, φωτεινότητας και χρονοδιακόπτη σε πεδία κειμένου, τα Edit Text.
Για να ελέγξουμε αν οι τιμές που δίνει ο χρήστης είναι εντός των ορίων θα χρειαστούμε έναν Text Watcher και την μέθοδο afterTextChanged() για να γίνετε ο έλεγχος όταν το κείμενο έχει αλλάξει, δηλαδή μόλις πληκτρολογήσει κάτι ο χρήστης.
Μέσα στη μέθοδο αυτή καλούμε την μέθοδο check_the_input() με παράμετρο έναν αριθμό που αντιστοιχεί στον έλεγχο κάθε ορίου.

```
146                 //for temp
147                 etTempLimit.addTextChangedListener(new TextWatcher(){
148
149                     @Override
150                     public void afterTextChanged(Editable s) {
151                         check_the_input(4);
152                     }
153                     @Override
154                     public void beforeTextChanged(CharSequence s, int start, int count,
155                             int after) {
156                     }
157                     @Override
158                     public void onTextChanged(CharSequence s, int start, int before,
159                             int count) {
160                     }
161                 });
```

Στην μέθοδο check_the_input() ελέγχουμε για κάθε Edit Text αν η τιμή που έχει δώσει ο χρήστης είναι εντός των ορίων. Αρχικοποιούμε δυο μεταβλητές για την μέγιστη και την ελάχιστη τιμή. Η ελάχιστη τιμή είναι σε όλες τις περιπτώσεις το 0 ενώ η μέγιστη είναι διαφορετική σε κάποιες περιπτώσεις.

```
272     //edit text check
273     public void check_the_input(int whichcase){
274         int minVal=0;
275         int maxVal;
276
277      switch(whichcase){
```

Η περίπτωση 4 αντιστοιχεί στον έλεγχο του Edit Text θερμοκρασίας, άρα η μέγιστη τιμή είναι το 100. Για κάθε Edit Text υπάρχουν τρεις περιπτώσεις, να μην έχει δώσει ο χρήστης κάποια τιμή (μηδενικό πλήθος χαρακτήρων) οπότε και βάζουμε σαν είσοδο την τιμή μηδέν, να έχει δώσει μια τιμή και να είναι εκτός των ορίων. Για να το ελέγξουμε αυτό μετατρέπουμε το κείμενο σε έναν ακέραιο αριθμό και ελέγχουμε αν είναι μεγαλύτερος από την μέγιστη επιτρεπτή τιμή ή μικρότερος από την μικρότερη.
Η τρίτη περίπτωση είναι να έχει δώσει μια τιμή η οποία είναι εντός των ορίων, ομοίως με πριν μετατρέπουμε το κείμενο σε ακέραιο αριθμό και τον αποθηκεύουμε.

```
415         case 4: //4->temp
416             maxVal=100; // until 100 degrees C
417
418             if (etTempLimit.getText().toString().length() <= 0) {
419                 inputETTemp=0;
420             }
421             else if (Integer.parseInt(etTempLimit.getText().toString())<minVal
422                     || Integer.parseInt(etTempLimit.getText().toString())>maxVal) {
423
424                 etTempLimit.setText("");
425                 inputETTemp=0;
426                  Toast.makeText(getBaseContext(),"Temperature limit is 100°C ",Toast.LENGTH_SHORT).show();
427             } else {
428                 inputETTemp = Integer.parseInt(etTempLimit.getText().toString());
429             }
430         break;
```

Αποστολή δεδομένων στην οθόνη ρυθμίσεων:

Στην δραστηριότητα της οθόνης ελέγχου είδαμε πως μπορούμε να παραλάβουμε τα δεδομένα από μια δραστηριότητα που δημιούργησε (εδώ η δραστηριότητα της οθόνης ρυθμίσεων) με την μέθοδο onActivityResult(). Στην δραστηριότητα της οθόνης ρυθμίσεων μόλις ο χρήστης πατήσει το κουμπί OK ή το πίσω κουμπί καλείται η μέθοδος sendChoices() για να στείλουμε τα δεδομένα. Δημιουργούμε ένα intent και χρησιμοποιούμε την μέθοδο putExtra() για να προσθέσουμε μια μεταβλητή μαζί με το κλειδί της. Το κλειδί αυτό θα χρησιμοποιηθεί από την δραστηριότητα πατέρα για να λάβει τα δεδομένα από το intent. Μόλις η δραστηριότητα της οθόνης ρυθμίσεων τερματίσει καλεί την μέθοδο setResult() για επιστρέψει τα δεδομένα. Η μέθοδος αυτή έχει δυο παραμέτρους, το result Code και το intent που μεταφέρει τα δεδομένα.

```
232    public void sendChoices(){
233
234        //passage of values to control page
235          Intent toControlPage = new Intent();
236          toControlPage.putExtra("tempVal", inputETTemp);
237          toControlPage.putExtra("lightVal",inputETLight );
238
239          toControlPage.putExtra("timerHval", inputETH);
240          toControlPage.putExtra("timerMval", inputETM);
241          toControlPage.putExtra("timerSval", inputETS);
242
243          toControlPage.putExtra("fromHval", from_hour);
244          toControlPage.putExtra("fromMval", from_minute);
245
246          toControlPage.putExtra("toHval", to_hour);
247          toControlPage.putExtra("toMval", to_minute);
248
249          setResult(RESULT_OK,toControlPage );
250
251    }
```

Κύλιση οθόνης κατά την είσοδο / έξοδο:

Για την μετάβαση ανάμεσα στις δυο δραστηριότητες, της οθόνης ελέγχου και της οθόνης ρυθμίσεων χρησιμοποιούμε animation. Για την δημιουργία του animation κύλισης έχουμε φτιάξει δυο αρχεία xml για κάθε δραστηριότητα. Μετά την κλήση της startActivity() ή finish() χρησιμοποιούμε την μέθοδο overridePendingTransition() με παραμέτρους το ID του animation για την εισερχόμενη δραστηριότητα και το ID του animation για την εξερχόμενη.
Από την οθόνη ρυθμίσεων αν επιστρέψουμε στην οθόνη ελέγχου τότε η οθόνη (ρυθμίσεων) θα κυλήσει προς τα δεξιά. Το xml αρχείο του animation είναι:

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <translate xmlns:android="http://schemas.android.com/apk/res/android"
3  android:fromXDelta="0"
4  android:toXDelta="100%p"
5  android:duration="1000"/>
```

Το 0 αντιστοιχεί στο αριστερό άκρο της οθόνης ενώ το 100 στο δεξί, η διάρκεια είναι 1 δευτερόλεπτο.

Η εντολή για την εμφάνιση του animation είναι:

this.overridePendingTransition(R.anim.*slide_in_left_to_right*, R.anim.*slide_out_left_to_right*);

Αποθήκευση και επανάκτηση δεδομένων:

Η εφαρμογή αποθηκεύει τις επιλογές του χρήστη για διευκόλυνση του. Για να γίνει αυτό χρησιμοποιούμε την κλάση sharedPreferences που δίνει την δυνατότητα αποθήκευσης και ανάκτησης δεδομένων. Η αποθήκευση των δεδομένων γίνεται στην μέθοδο onStop() που είναι ένδειξη ότι ο χρήστης φεύγει από την δραστηριότητα. Για την δημιουργία ενός αντικειμένου της κλάσης sharedPreferences θα χρησιμοποιήσουμε την μέθοδο getSharedPreferences() με παραμέτρους το όνομα ενός αρχείου xml που θα αποθηκευτούν τα δεδομένα και ο τρόπος λειτουργίας, η τιμή 0 σημαίνει ότι

μόνο η εφαρμογή που έφτιαξε το αρχείο xml έχει πρόσβαση σε αυτό. Στο αρχείο που έχουμε φτιάξει με την μέθοδο getSharedPreferences() έχουν πρόσβαση οι δραστηριότητες της οθόνης ρυθμίσεων και οθόνης ελέγχου. Για να γράψουμε κάτι στο αρχείο αυτό δημιουργούμε ένα αντικείμενο Editor και χρησιμοποιούμε την μέθοδο edit(), μετά προσθέτουμε τις μεταβλητές που θέλουμε να αποθηκεύσουμε με ένα κλειδί και χρησιμοποιούμε την μέθοδο commit() για να κάνουμε τις αλλαγές στο αρχείο.

```
589         SharedPreferences saveChoices = getSharedPreferences("MyPrefsFile", 0);
590         SharedPreferences.Editor editor = saveChoices.edit();
591         editor.putInt("savedTempVal", inputETTemp);
592         editor.putInt("savedLightVal", inputETLight);
593
594         editor.putInt("savedHourVal",inputETH );
595         editor.putInt("savedMinVal",inputETM );
596         editor.putInt("savedSecVal",inputETS );
597
598         editor.putInt("savedFromHourVal", from_hour);
599         editor.putInt("savedFromMinVal",from_minute );
600
601         editor.putInt("savedToHourVal", to_hour);
602         editor.putInt("savedToMinVal", to_minute);
603
604         editor.putInt("saved_minmaxTemp", minmaxTemp);
605         editor.putInt("saved_minmaxLight", minmaxLight);
606
607         editor.commit();
```

Μόλις επιστρέψουμε στην εφαρμογή θα εμφανιστούν τα δεδομένα που είχαμε αποθηκεύσει. Με την μέθοδο getSharedPreferences() και ίδιες παραμέτρους όπως πριν έχουμε πρόσβαση στο αρχείο που φτιάξαμε και μπορούμε να αποθηκεύσουμε σε μεταβλητές τα δεδομένα που είχαμε αποθηκεύσει δίνοντας το αντίστοιχο κλειδί τους.

```
630         SharedPreferences saveChoices = getSharedPreferences("MyPrefsFile", 0);
631         inputETTemp = saveChoices.getInt("savedTempVal", 0);
632         inputETLight = saveChoices.getInt("savedLightVal", 0);
633
634         inputETH = saveChoices.getInt("savedHourVal", 0);
635         inputETM = saveChoices.getInt("savedMinVal", 0);
636         inputETS = saveChoices.getInt("savedSecVal", 0);
637
638         from_hour = saveChoices.getInt("savedFromHourVal", 0);
639         from_minute = saveChoices.getInt("savedFromMinVal", 0);
640
641         to_hour = saveChoices.getInt("savedToHourVal", 0);
642         to_minute = saveChoices.getInt("savedToMinVal", 0);
643
644         minmaxTemp = saveChoices.getInt("saved_minmaxTemp", 0);
645         minmaxLight = saveChoices.getInt("saved_minmaxLight", 0);
```

Κεφάλαιο 7

Υλοποίηση και προγραμματισμός συσκευής

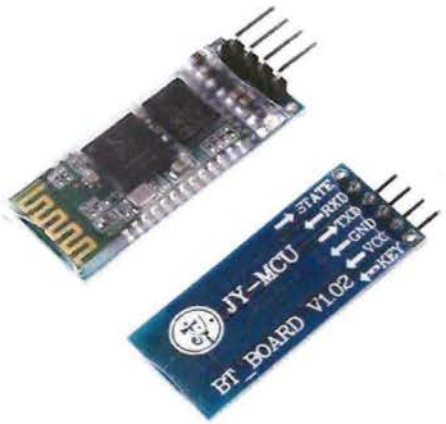
Στο προηγούμενο κεφάλαιο είδαμε τον τρόπο λειτουργίας της Android εφαρμογής για να ελέγχουμε την συσκευή, στο κεφάλαιο αυτό θα δούμε τον τρόπο λειτουργίας της συσκευής, το κύκλωμα που θα χρειαστεί να φτιάξουμε για να μπορέσει να λειτουργήσει και το πρόγραμμα που επιτρέπει την επικοινωνία της συσκευής με το κινητό τηλέφωνο.

7.1 Το σχηματικό του κυκλώματος

Στο σχήμα 7.2 φαίνεται το σχηματικό του κυκλώματος. Στο Arduino θα συνδέσουμε όλα τα υλικά (Bluetooth Module, οθόνη LCD κλπ) που είναι απαραίτητα για την λειτουργία της συσκευής.

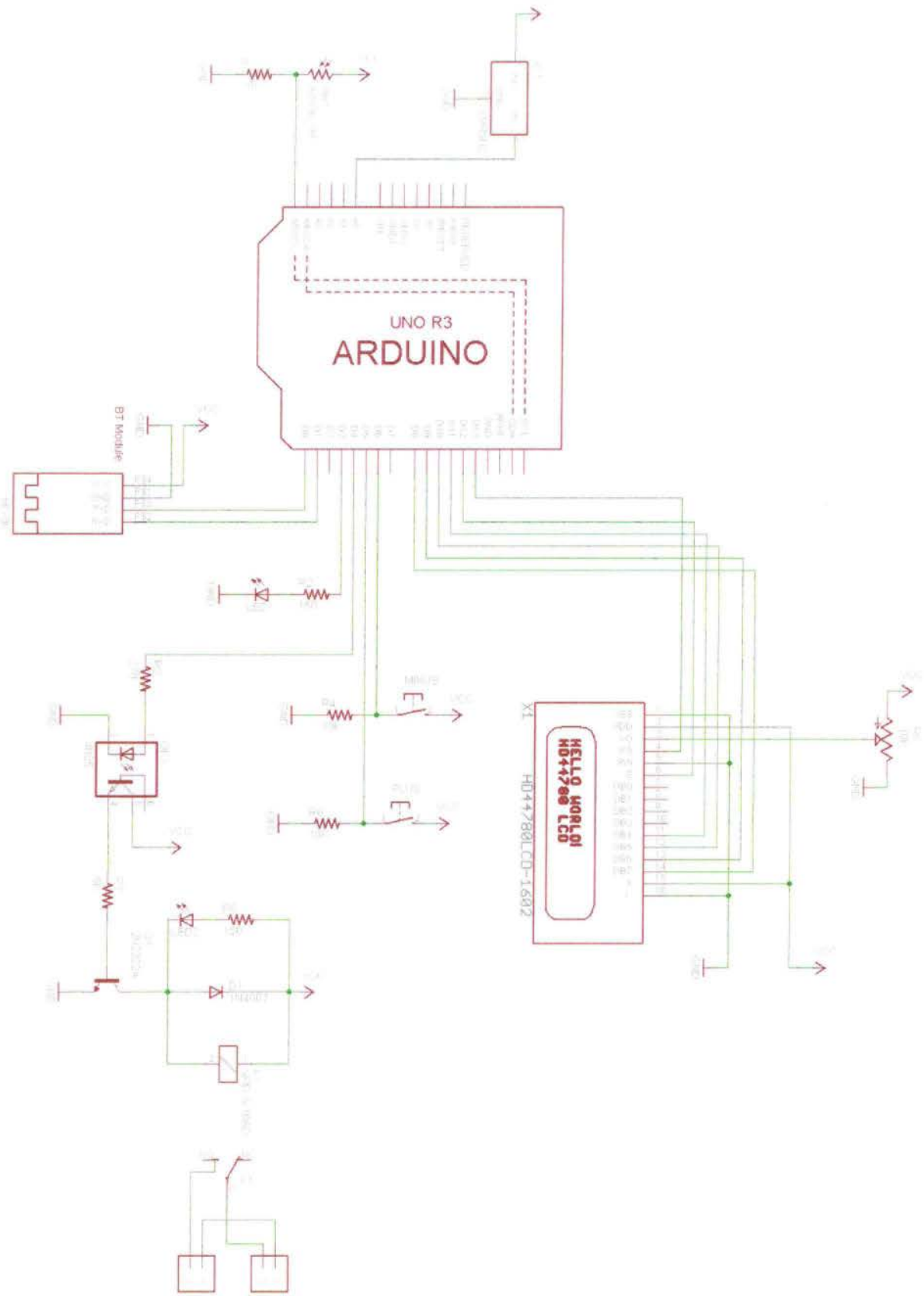
Συσκευή Bluetooth:

Για να μπορέσει το Arduino να επικοινωνήσει με το κινητό θα χρειαστούμε μια συσκευή Bluetooth, εδώ χρησιμοποιούμε το Bluetooth Module HC-06.



Σχήμα 7.1: Bluetooth Module

Ο ακροδέκτης Vcc είναι για την τροφοδοσία της συσκευής Bluetooth και δέχεται από 3.3V έως 5V. Ο ακροδέκτης GND για το ground, ο TXD για αποστολή δεδομένων και RXD για λήψη δεδομένων. Ο ακροδέκτης STATE συνδέεται με το LED ένδειξης κατάστασης, αν είναι συνδεδεμένη η συσκευή τότε είναι αναμμένο ενώ αν είναι έτοιμη για σύνδεση τότε αναβοσβήνει. Ο ακροδέκτης KEY είναι για τον προγραμματισμό (κατάσταση HIGH) ή λειτουργία της συσκευής (κατάσταση LOW).



Σχήμα 7.2: Σχηματικό κυκλώματος

Για την ανταλλαγή δεδομένων κινητού και Arduino συνδέουμε το TXD της συσκευής Bluetooth στο RXD του Arduino (Digital pin 0) και το RXD της συσκευής Bluetooth στο TXD του Arduino (Digital pin 1).

Σε περίπτωση που θέλουμε να αλλάξουμε το όνομα της συσκευής (το όνομα αυτό είναι που θα εμφανίζεται στο κινητό όταν κάνουμε αναζήτηση συσκευών), το κωδικό για σύζευξη ή το baud rate θα χρειαστεί να προγραμματίσουμε την συσκευή. Βγάζουμε από την πλακέτα το μικροελεγκτή ATmega328, συνδέουμε την τροφοδοσία της συσκευής (5V και GND), στον ακροδέκτη KEY συνδέουμε μια αντίσταση 10KΩ προς τα 5V, έτσι μπορούμε τώρα να προγραμματίσουμε την συσκευή. Συνδέουμε το Arduino με το PC μέσω του καλωδίου USB.

Ανοίγουμε το περιβάλλον Arduino IDE πατάμε το κουμπί Σειριακής Οθόνης (Serial Monitor) και θέτουμε 'No line ending' και '9600' για το baud rate. Εισάγουμε AT και πατάμε Send, η συσκευή απαντά με OK, ένδειξη καλής επικοινωνίας. Για να αλλάξουμε όνομα της συσκευής εισάγουμε AT+NAMEnewname και πατάμε Send, για να αλλάξουμε τον κωδικό εισάγουμε AT+PINpinnumber και πατάμε Send και για να αλλάξουμε το baud rate εισάγουμε AT+BAUDn (n=1,2,3,4,5,6,7,8) και πατάμε Send. Οι τιμές του n αντιστοιχούν σε τιμές για το baud rate δηλαδή 1200, 2400, 4800, 9600, 19200, 38400, 57600,115200. Για να επικοινωνήσει η συσκευή Bluetooth με το κινητό χρειάζεται baud rate 57600.
Αφαιρούμε την αντίσταση από τον ακροδέκτη, κάνουμε reset στην συσκευή Bluetooth και είναι έτοιμη για λειτουργία με τα νέα χαρακτηριστικά.

Έλεγχος της πρίζας:

Στο Digital pin 4 έχουμε συνδέσει τα υλικά για τον έλεγχο της πρίζας. Χρησιμοποιούμαι ένα φωτοτρανζίστορ (optocoupler) για ασφάλεια του μικροελεγκτή, καθώς το φωτοτρανζίστορ προσφέρει λογική σύνδεση του κυκλώματος αλλά όχι ηλεκτρική. Το φωτοτρανζίστορ δεν μπορεί να δώσει αρκετό ρεύμα για να διεγείρει το relay οπότε χρησιμοποιούμε ένα τρανζίστορ. Το relay χρησιμοποιείται σαν διακόπτης, όταν είναι διεγερμένο περνάει ρεύμα στην πρίζα. Παράλληλα με το relay τοποθετούμε μια δίοδο καθώς κατά το άνοιγμα ή το κλείσιμο του relay δημιουργούνται ανάστροφες υπερτάσεις από το επαγωγικό τύλιγμα του relay, οι υπερτάσεις αυτές καταναλώνονται πάνω στην ωμική αντίσταση της διόδου. Το LED είναι και αυτό συνδεδεμένο παράλληλα με το relay για να βλέπουμε πότε είναι ενεργό το relay. Στην μια κλέμα (terminal block) έχουμε συνδέσει τον ακροδέκτη COM, εδώ θα βάλουμε το καλώδιο με την φάση και στην άλλη κλέμα έχουμε συνδέσει τον ακροδέκτη NO που συνδέουμε το καλώδιο με την φάση για την έξοδο στην πρίζα. Βραχυκυκλώνουμε το ελεύθερο ακροδέκτη κάθε κλέμας και συνδέουμε σε αυτούς το καλώδιο με την επιστροφή.

Πλήκτρα για έλεγχο ώρας:

Όπως θα δούμε και παρακάτω έχουμε προγραμματίσει το Arduino για να δουλεύει και σαν ρολόι. Στο Digital pin 5 και 6 έχουμε συνδέσει δυο πλήκτρα για τον έλεγχο της ώρας, πρόσθεση ή αφαίρεση λεπτών. Σε κάθε πλήκτρο

είναι συνδεδεμένη μια αντίσταση 10ΚΩ με τη γείωση. Όταν το πλήκτρο δεν είναι πατημένο η τιμή είναι σε κατάσταση LOW.

Αισθητήρια:

Στα Analog pin 0 και 5 έχουμε συνδέσει τα αισθητήρια θερμοκρασίας και φωτεινότητας. Για την μέτρηση της θερμοκρασίας χρησιμοποιούμε το LM35DZ που έχει 3 ακροδέκτες ένα για τα 5V, ένα για την τάση εξόδου που συνδέουμε με το Arduino και ένα για GND. Για την μέτρηση της φωτεινότητας χρησιμοποιούμε μια φωτοαντίσταση μαζί με μια αντίσταση συνδεδεμένη στη γείωση.

Οθόνη LCD:

Η οθόνη χρησιμοποιείται για να εμφανίσουμε κάποιες πληροφορίες χωρίς να χρειάζεται να συνδεθούμε με την συσκευή από το κινητό. Η οθόνη έχει 16 στήλες και 2 γραμμές, λειτουργεί με τάση 5V και έχει οπίσθιο φωτισμό για να μπορούμε να βλέπουμε ακόμα και σε σκοτάδι. Για να ελέγξουμε την αντίθεση της οθόνης χρειαζόμαστε ένα ποτενσιόμετρο 10ΚΩ.
Στην επάνω γραμμή εμφανίζεται η ώρα ενώ στην κάτω η θερμοκρασία και η φωτεινότητα.

7.2 Το σχεδιάγραμμα για το Arduino

Παρακάτω θα δούμε μερικά μέρη από το πρόγραμμα που πρέπει να φορτώσουμε στο Arduino για να μπορέσει να λειτουργήσει η συσκευή.

7.2.1 Αρχικοποιήσεις

Για να μπορέσει η συσκευή να επικοινωνήσει με το κινητό τηλέφωνο μέσω Bluetooth και να ελέγχει την οθόνη LCD για να εμφανίζει πληροφορίες χρειάζεται να εισάγουμε τις δυο βιβλιοθήκες MeetAndroid.h και LiquidCrystal.h.
Δημιουργούμε μια μεταβλητή τύπου MeetAndroid για να μπορέσουμε αργότερα να χρησιμοποιήσουμε τις μεθόδους για αποστολή και λήψη δεδομένων. Ομοίως για την οθόνη δημιουργούμε μια μεταβλητή τύπου LiquidCrystal με παραμέτρους του ακροδέκτες που θα χρησιμοποιήσουμε. Για να ελέγξουμε την οθόνη μπορούμε να χρησιμοποιήσουμε τους 4 ή 8 ακροδέκτες δεδομένων, εδώ χρησιμοποιούμε τους 4.
Η πρώτη παράμετρος είναι ο ακροδέκτης του Arduino που είναι συνδεδεμένος με το ακροδέκτη RS της οθόνης, με HIGH επιλέγεται ο καταχωρητής δεδομένων ο οποίος περιέχει ότι εμφανίζεται στην οθόνη ενώ με LOW επιλέγεται ο καταχωρητής εντολών μέσω του οποίου ρυθμίζεται η λειτουργία της οθόνης. Η δεύτερη παράμετρος αντιστοιχεί στο ακροδέκτη συνδεδεμένο με το Enable, για την εγγραφή των καταχωρητών. Οι επόμενοι 4 παράμετροι αντιστοιχούν στους ακροδέκτες D4 έως D7 για την μεταφορά των δεδομένων.

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

// initialize the LiquidCrystal library with the numbers of the interface pins
LiquidCrystal lcd(13, 12, 11, 10, 9, 8);

// declare MeetAndroid so that we can call functions with it
MeetAndroid meetAndroid;
```

Μετά αρχικοποιούμε όλες τις μεταβλητές που θα χρησιμοποιήσουμε στο πρόγραμμα.
Στην οθόνη όταν εμφανίζουμε την θερμοκρασία χρησιμοποιούμε και τον χαρακτήρα '°', για να γίνει αυτό πρέπει να δημιουργήσουμε έναν πίνακα των 8 bytes ένα για κάθε γραμμή όπως φαίνετε παρακάτω.

```
byte degree[8] = {
  B00110,
  B01001,
  B01001,
  B00110,
  B00000,
  B00000,
  B00000,
};
```

7.2.2 Η συνάρτηση setup()

Μέσα στην setup() ρυθμίζουμε το baud rate για την ανταλλαγή δεδομένων μέσω Bluetooth. Προγραμματίζουμε τους δυο ακροδέκτες που είναι συνδεδεμένοι με τα πλήκτρα σαν εισόδους ενώ τους άλλους δυο για τον έλεγχο της πρίζας και εμφάνιση κατάστασης σαν εξόδους. Δίνουμε LOW (λογικό 0) στον ακροδέκτη που είναι συνδεδεμένα τα εξαρτήματα για τον έλεγχο της πρίζας, έτσι δεν περνάει ρεύμα στην πρίζα και HIGH στον ακροδέκτη που είναι συνδεδεμένο το LED ένδειξης ότι δεν περνάει ρεύμα από την πρίζα.
Προτού καλέσουμε κάποια μέθοδο από την βιβλιοθήκη LiquidCrystal και εμφανίσουμε κάτι στην οθόνη πρέπει να δηλώσουμε τις στήλες και γραμμές που έχει με την μέθοδο begin(), η οθόνη που χρησιμοποιούμε έχει 16 στήλες και 2 γραμμές.
Με την μέθοδο createChar() δημιουργούμε τον χαρακτήρα '°' με βάση τον πίνακα degree που δημιουργήσαμε προηγουμένως. Μπορούμε να δημιουργήσουμε μέχρι 8 χαρακτήρες. Η πρώτη παράμετρος της μεθόδου είναι ο αριθμός του χαρακτήρα και η δεύτερη παράμετρος ο πίνακας με τα pixels που θα σχηματίσουν το χαρακτήρα.
Με την μέθοδο clear() καθαρίζουμε την οθόνη από τυχόν χαρακτήρες και τοποθετούμε τον κέρσορα στην επάνω αριστερή γωνία.

```
void setup(){

  //set the baud rate
  Serial.begin(57600);

  //Configure pin behavior (input/output)
  pinMode(switchPinPlus, INPUT);
  pinMode(switchPinMin, INPUT);
  pinMode(relayPin, OUTPUT);
  pinMode(offrelay, OUTPUT);

  digitalWrite(relayPin,LOW);
  digitalWrite(offrelay,HIGH);

  //set up the LCD's number of columns and rows
  lcd.begin(16, 2);

  //Make degree character
  lcd.createChar(0, degree);

  lcd.clear();
```

Εγγράφουμε τις συναρτήσεις που θα καλούνται μόλις συμβεί το αντίστοιχο γεγονός, για παράδειγμα μόλις σταλούν από το κινητό οι τιμές που έχει επιλέξει ο χρήστης για τα όρια θερμοκρασίας, φωτεινότητας κλπ θα κληθούν οι αντίστοιχες συναρτήσεις για να παραλάβει το Arduino τις τιμές αυτές. Η εγγραφή μιας συνάρτησης γίνετε με την μέθοδο registerFunction() με παραμέτρους το όνομα της συνάρτησης και τη σημαία του γεγονότος. Η σημαία του γεγονότος πρέπει να είναι ίδια με αυτή που δηλώσαμε την εφαρμογή στο κινητό.

```
  //---------- register callback functions ------------------

  //state ON/OFF
  meetAndroid.registerFunction(state_on_off_Android, 'i');

  //Control page - checkboxes
  meetAndroid.registerFunction(all_chk_Android, 'd');


  //hour and minute from android
  meetAndroid.registerFunction(hm_array_Android, 'b');
  //temp limit from android
  meetAndroid.registerFunction(temp_Android, 't');
  //light limit from android
  meetAndroid.registerFunction(light_Android, 'l');
  //timer(hour/min/sec) from android
  meetAndroid.registerFunction(timer_array_Android, 'x');
  //time period -from- from android
  meetAndroid.registerFunction(from_array_Android, 's');
  //time period -to- from android
  meetAndroid.registerFunction(to_array_Android, 'e');

  //min or max temp limit from android
  meetAndroid.registerFunction(minmaxTemp_Android, 'o');
  //min or max light limit from android
  meetAndroid.registerFunction(minmaxLight_Android, 'p');
}
```

7.2.3 Συναρτήσεις λήψης δεδομένων από το κινητό

Τα δεδομένα που στέλνουμε από την εφαρμογή BluePuc από το κινητό είναι ακέραιοι αριθμοί ή πίνακες με ακέραιους αριθμούς. Για να λάβουμε έναν ακέραιο αριθμό όπως το όριο θερμοκρασίας χρησιμοποιούμε την μέθοδο getInt() και αποθηκεύουμε την τιμή σε μια μεταβλητή.

```
//function for temp limit from Android
void temp_Android(byte flag,byte Val){
  tempLimit = meetAndroid.getInt();
}
```

Οι επιλογές του χρήστη για τους ελέγχους που πρέπει να κάνει η συσκευή στέλνονται όλες μαζί με έναν πίνακα με ακέραιους αριθμούς. Σε αυτήν την περίπτωση για να λάβουμε τον πίνακα χρησιμοποιούμε την μέθοδο getIntValues() και αποθηκεύουμε τις τιμές σε έναν πίνακα ίσου μεγέθους. Για ευκολία στον προγραμματισμό αποθηκεύουμε τα περιεχόμενα του πίνακα σε τέσσερις διαφορετικές μεταβλητές που αντιστοιχούν στις 4 επιλογές ελέγχου. Οι τιμές αυτές αν είναι 1 σημαίνει πως έχει επιλεχτεί ο αντίστοιχος έλεγχος ενώ η τιμή 0 ότι δεν έχει επιλεχτεί. Παρακάτω θα δούμε πως μπορούμε να ξεχωρίσουμε ποιους ελέγχους πρέπει να κάνει η συσκευή.

```
//***************** function for the check boxes ****************
void all_chk_Android(byte flag,byte Val){

  int check_box_Android[Val];
  meetAndroid.getIntValues(check_box_Android);
  temp_chk = check_box_Android[0];
  light_chk = check_box_Android[1];
  timer_chk = check_box_Android[2];
  time_period_chk = check_box_Android[3];
}
```

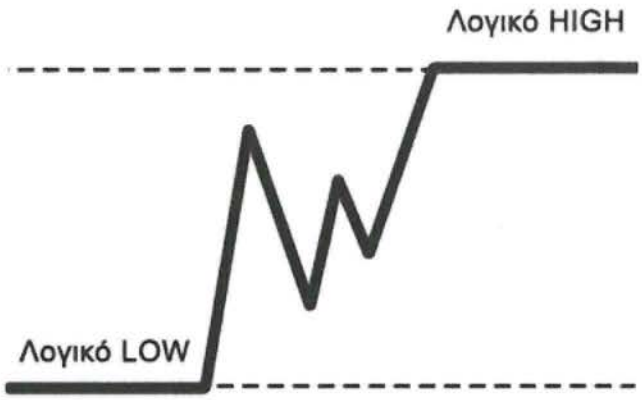
7.2.4 Η συνάρτηση loop()

Λήψη δεδομένων με την receive():

Μέσα στην συνάρτηση loop() θα βάλουμε ότι θέλουμε να εκτελείται συνέχεια. Για να λάβουμε τα δεδομένα που στέλνει το κινητό πρέπει να χρησιμοποιήσουμε την μέθοδο receive(), η μέθοδος θα αναλύσει τα δεδομένα και θα καλέσει την συνάρτηση που έχουμε εγγράψει για το γεγονός που έλαβε.

Συναρτήσεις debounce:

Στην ανάγνωση ενός πλήκτρου υπάρχει η έννοια του 'debouncing', η μετάβαση από την μια κατάσταση στην άλλη μπορεί να μην είναι ομαλή με αποτέλεσμα να υπάρξει λανθασμένη ανάγνωση. Για να λύσουμε αυτό το πρόβλημα αρχικά διαβάζουμε την κατάσταση του πλήκτρου, συγκρίνουμε την τιμή της τρέχουσας ανάγνωσης με μια βοηθητική μεταβλητή τύπου Boolean με λογικό LOW. Αν αυτές οι δυο τιμές δεν είναι ίσες τότε κάνουμε μια καθυστέρηση 5ms και διαβάζουμε ξανά την κατάσταση του πλήκτρου. Ο χρόνος αυτός είναι αρκετός για να σταθεροποιηθεί η κατάσταση του.



Σχήμα 7.3: Αναπηδήσεις τάσης κατά την αλαλαγή κατάστασης

Για ευκολία στον προγραμματισμό η παραπάνω διαδικασία βρίσκεται μέσα σε μια συνάρτηση η οποία επιστρέφει το αποτέλεσμα της ανάγνωσης και με βάση αυτό μπορούμε να διακρίνουμε ποια πλήκτρα είναι πατημένα κάθε φορά.

```
currentButtonPlus = debounceP(lastButton);
currentButtonMin = debounceM(lastButton);
```

Η συνάρτηση για το πλήκτρο συν:

```
//-------------debounce functions-----------------------
//for button +
boolean debounceP(boolean last)
{
  boolean currentP = digitalRead(switchPinPlus);
  if (last != currentP)
  {
    delay(5);
    currentP = digitalRead(switchPinPlus);
  }
  return currentP;
}
```

Η συνάρτηση για το πλήκτρο μείον είναι παρόμοια, με διαφορά την μεταβλητή για την τρέχουσα ανάγνωση.

Ώρα στην συσκευή:

Η συσκευή είναι προγραμματισμένη για να δουλεύει και σαν ρολόι. Για να γίνει αυτό θα χρησιμοποιήσουμε 3 μεταβλητές, μια για τα δευτερόλεπτα, μια για τα λεπτά, άλλη μια για τις ώρες και την μέθοδο delay().

Για να υπολογίσουμε την ώρα κάνουμε μια καθυστέρηση ενός δευτερολέπτου και προσθέτουμε ένα στην μεταβλητή των δευτερολέπτων. Μετά από κάθε δευτερόλεπτο που έχει περάσει κάνουμε έλεγχο, αν τα δευτερόλεπτα είναι 60 τότε τα μηδενίζουμε και προσθέτουμε ένα στην μεταβλητή των λεπτών. Ομοίως αν τα λεπτά είναι 60 τότε μηδενίζουμε τα λεπτά και προσθέτουμε ένα στην μεταβλητή για τις ώρες. Αν οι ώρες είναι 24 τότε μηδενίζουμε την μεταβλητή.

```
delay (1000);

seconds = seconds +1;

if (seconds == 60){
  seconds = 0;
  minutes = minutes +1;
}
if (minutes == 60){
  minutes = 0;
  hours = hours +1;
}
if (hours == 24){
  hours = 0;
}
```

Λειτουργία πλήκτρων συσκευής:

Η ώρα όπως είδαμε και στο κεφάλαιο 6.5.4 μπορεί να ρυθμιστεί από την εφαρμογή στο κινητό τηλέφωνο. Μπορεί όμως να ρυθμιστεί και απευθείας χωρίς να χρειάζεται η σύνδεση με την συσκευή μέσω Bluetooth, με τα πλήκτρα συν και πλειν που βρίσκονται πάνω στην συσκευή. Υπάρχουν 3 περιπτώσεις χρήσης των πλήκτρων , να είναι πατημένο το πλήκτρο συν, να είναι πατημένο το πλήκτρο πλειν ή να είναι πατημένα ταυτόχρονα και τα δυο.
Τα δυο πλήκτρα είναι προγραμματισμένα να προσθέτουν ή να αφαιρούν λεπτά για όσο χρονικό διάστημα είναι πατημένα. Παρακάτω φαίνετε ο τρόπος λειτουργίας του πλήκτρου συν.
Χρησιμοποιούμε δυο βοηθητικές μεταβλητές. Η μεταβλητή bima έχει αρχική τιμή 1 και όσο διάστημα το πλήκτρο είναι πατημένο η μεταβλητή αυτή θα αυξάνετε κατά ένα. Η μεταβλητή checkbima έχει το άθροισμα των λεπτών και της μεταβλητής bima (της τιμής δηλαδή που θα προστεθεί στα λεπτά). Αν τα λεπτά είναι μικρότερα ή ίσα με 59 και η μεταβλητή checkbima έχει τιμή μέχρι 60 τότε προσθέτουμε στα λεπτά την τιμή της μεταβλητής bima αλλιώς αν η checkbima έχει ξεπεράσει την τιμή 60 τότε επαναφέρουμε το βήμα στο 1. Κάνουμε έλεγχο για τις τιμές των λεπτών και ώρας ώστε να είναι μέσα στα όρια του 24ώρου.

```
//check for button +
if (lastButton == LOW && currentButtonPlus == HIGH){

  checkbima = minutes + bima;

  if (minutes <= 59 && checkbima <=60){
    minutes = minutes +bima;
  }
  else if(checkbima > 60){
    bima = 1;
  }
  if (minutes == 60){
    minutes = 0;
    hours = hours +1;
  }
  if (hours == 24){
    hours = 0;
  }
  bima = bima + 1;

}
```

Έτσι αν το πλήκτρο συν είναι πατημένο για σύντομο χρονικό διάστημα τότε θα αυξήσει τα λεπτά με αργό ρυθμό , πχ αν θέλουμε να κάνουμε μικρές αλλαγές, ενώ αν το πλήκτρο είναι πατημένο για αρκετή ώρα μπορούμε να αλλάξουμε τα λεπτά και την ώρα με ευκολία.

Πατώντας ταυτόχρονα και τα δυο πλήκτρα θα επαναφέρει τις μεταβλητές για την ώρα και τα λεπτά στις αρχικές του τιμές το μηδέν, ενώ αν δεν είναι πατημένο κανένα πλήκτρο επαναφέρουμε την τιμή της μεταβλητής bima στην αρχική της τιμή.

```
//check if both buttons are pressed
if (lastButton == LOW &&  currentButtonMin == HIGH && currentButtonPlus == HIGH){
  minutes = 0;
  hours = 0;
}
if (lastButton == LOW &&  currentButtonMin == LOW && currentButtonPlus == LOW){
  bima = 1;
}
```

Μετρήσεις θερμοκρασίας και φωτεινότητας:

Στην συσκευή έχουμε συνδέσει δυο αισθητήρια, ένα για θερμοκρασία και ένα για φωτεινότητα. Για την ανάγνωση της τιμής από έναν αναλογικό ακροδέκτη θα χρησιμοποιήσουμε την μέθοδο analogRead(). Η μέθοδος αυτή επιστρέφει έναν ακέραιο αριθμό από 0 έως 1023.

Παρακάτω φαίνεται η ανάγνωση της θερμοκρασίας από το αντίστοιχο αισθητήριο LM35DZ. Πρώτα κάνουμε μια ανάγνωση χωρίς να αποθηκεύσουμε κάπου την τιμή, μετά κάνουμε μια καθυστέρηση 10ms και κάνουμε ξανά ανάγνωση αποθηκεύοντας την τιμή αυτή την φορά σε μια μεταβλητή. Ο λόγος που κάνουμε έτσι την ανάγνωση είναι ότι το Arduino έχει ένα μόνο ADC (Analog/Digital Converter) και πρέπει να κάνουμε δυο μετρήσεις (μια για θερμοκρασία και μια για φωτεινότητα). Για να ολοκληρωθεί

μια ανάγνωση από έναν ακροδέκτη χρειάζεται ένα μικρό χρονικό διάστημα προτού κάνουμε νέα ανάγνωση. Έτσι για την απαλοιφή του θορύβου κάνουμε την πρώτη μέτρηση και την καθυστέρηση ενώ η δεύτερη μέτρηση θα είναι η σωστή. Χρησιμοποιούμε την αντίστοιχη εξίσωση για την μετατροπή της τιμής της ανάγνωσης σε δεκαδικό αριθμό που αντιστοιχεί σε βαθμούς Κελσίου.

```
//getting the voltage reading from the temperature sensor
analogRead(sensorTemp);
delay(10);
int reading = analogRead(sensorTemp);

//make the temperature float
float voltage = reading * 5.0 / 1024.0;
temperatureC = (voltage) * 100.0 ;
```

Για την ανάγνωση της φωτεινότητας από την φωτοαντίσταση χρησιμοποιούμε τον ίδιο τρόπο με πριν για να έχουμε σωστή ανάγνωση χωρίς θόρυβο. Η φωτεινότητα μπορεί να πάρει κάποια τιμή από το 0 έως 100%. Οπότε η τιμή που επιστρέφει η analogRead() πρέπει να αντικατασταθεί με μια σε αυτό το εύρος τιμών. Για να γίνει αυτό χρησιμοποιούμε την μέθοδο map() με παραμέτρους τον αριθμό που πρέπει να αντικαταστήσουμε, την ελάχιστη τιμή από το αρχικό εύρος τιμών, την μέγιστη τιμή από το αρχικό εύρος τιμών, την ελάχιστη τιμή από το τελικό εύρος τιμών και την μέγιστη τιμή από το τελικό εύρος τιμών.

```
//get the voltage from light sensor
analogRead(sensorLight);
delay(10);
int  lightVal=analogRead(sensorLight);

//make the % value
lightLvl= map(lightVal,0,1023,0,100);
```

<u>Εμφάνιση πληροφοριών στην οθόνη της συσκευής:</u>

Η οθόνη χρησιμοποιείται για να εμφανίσουμε βασικές πληροφορίες χωρίς να χρειάζεται να συνδεθούμε με την συσκευή από το κινητό. Αποτελείται από 16 στήλες και 2 γραμμές. Στην επάνω γραμμή εμφανίζεται η ώρα ενώ στην κάτω η θερμοκρασία και η φωτεινότητα. Ο διαθέσιμος χώρος δεν είναι αρκετός για να εμφανίσουμε ταυτόχρονα την φωτεινότητα και την θερμοκρασία οπότε η κάθε τιμή εμφανίζεται για 3 δευτερόλεπτα, για ευκολία στην ανάγνωση.

Για να εμφανίσουμε την ώρα ελέγχουμε την τιμή της αντίστοιχης μεταβλητής, αν η τιμή είναι μικρότερη του 10 τότε στην πρώτη στήλη από αριστερά δεν εμφανίζουμε κάτι και ξεκινάμε από την επόμενη στήλη. Αντίθετα αν η ώρα είναι μεγαλύτερη ή ίση του 10 τότε εμφανίζουμε την ώρα από την πρώτη στήλη της οθόνης. Με την μέθοδο setCursor() δηλώνουμε σε ποια στήλη και γραμμή της οθόνης θα εμφανιστεί ότι δώσουμε με την μέθοδο print().

```
//----display time on LCD----
if (hours<10){
  lcd.setCursor(0,0);
  lcd.print(" ");
  lcd.setCursor(1,0);
  lcd.print(hours);
}
else{
  lcd.setCursor(0,0);
  lcd.print(hours);
}
```

Για την εμφάνιση των λεπτών και δευτερολέπτων χρησιμοποιούμε τον ίδιο τρόπο, ελέγχουμε την τιμή και εμφανίζουμε στην κατάλληλη στήλη την τιμή.

Για να μπορέσουμε να κάνουμε την εναλλαγή στο κάτω μέρος της οθόνης θα χρειαστούμε δυο μεταβλητές για χρονοδιακόπτες. Η μεταβλητή για τον χρονοδιακόπτη της θερμοκρασίας έχει αρχική τιμή 3 ενώ η μεταβλητή για την φωτεινότητα 0. Κάθε δευτερόλεπτο κάνουμε 3 ελέγχους για τους 2 αυτούς χρονοδιακόπτες. Αν η τιμή του χρονοδιακόπτη θερμοκρασίας είναι μεγαλύτερη του 0 τότε αφαιρούμε ένα από αυτή. Μόλις ο χρονοδιακόπτης θερμοκρασίας μηδενιστεί παραμένει στο μηδέν και ξεκινάμε τον χρονοδιακόπτη για την φωτεινότητα, μετά από κάθε δευτερόλεπτο η τιμή του αυξάνετε κατά ένα. Μετά από 3 δευτερόλεπτα επαναφέρουμε και τους δυο στις αρχικές τους τιμές και ξεκινάει η διαδικασία από την αρχή.

```
//timers for temp/light switch on LCD
if (lcdTimerT>0){
  lcdTimerT= lcdTimerT -1;
}
if (lcdTimerT==0){
  lcdTimerL=lcdTimerL+1;
}
if (lcdTimerL==4){
  lcdTimerL=0;
  lcdTimerT=3;
}
```

Προτού εμφανίσουμε την θερμοκρασία ή την φωτεινότητα στην οθόνη ελέγχουμε τις τιμές των δυο χρονοδιακοπτών, η μία από τις δυο θα έχει τιμή μεγαλύτερη του 0 ενώ η άλλη ίση του 0. Για παράδειγμα όταν είναι να εμφανιστεί η θερμοκρασία ο χρονοδιακόπτης της θα έχει τιμή μεγαλύτερη του μηδενός. Για να εμφανίσουμε τον χαρακτήρα '°' χρησιμοποιούμε την μέθοδο write().

```
//----display temp/light on LCD----
if (lcdTimerT!=0 && lcdTimerL==0){
  lcd.setCursor(0,1);
  lcd.print("temp: ");
  lcd.print(temperatureC);
  lcd.write((byte)0);
  lcd.print("C");
  lcd.print("   ");
}
```

Αποστολή δεδομένων στο κινητό:

Μετά την ανάγνωση των τιμών από τα αισθητήρια θερμοκρασίας και φωτεινότητας στέλνουμε τις τιμές αυτές στο κινητό με την χρήση της μεθόδου send(). Οι εντολές για την αποστολή των δεδομένων βρίσκονται μέσα στην συνάρτηση loop() που έχει διάρκεια ενός δευτερολέπτου, έτσι το κινητό ενημερώνεται κάθε δευτερόλεπτο.

```
//send the data to android
meetAndroid.send(temperatureC); // temp
meetAndroid.send(lightLvl);     //light
```

Περιπτώσεις ελέγχου:

Προηγουμένως είδαμε πως μπορούμε να λάβουμε τις επιλογές για ελέγχους που επέλεξε ο χρήστης από την εφαρμογή στο κινητό τηλέφωνο. Μέσα στην συνάρτηση loop() πρέπει να ελέγξουμε τις τιμές των μεταβλητών των επιλογών για να κάνουμε τους αντίστοιχους ελέγχους. Ο συνδυασμός των περιπτώσεων ελέγχου φαίνετε στον πίνακα 6.1.

Για την περίπτωση που ο χρήστης έχει επιλέξει να γίνετε έλεγχος για θερμοκρασία μόνο, τότε η τιμή της μεταβλητής temp_chk θα έχει τιμή 1 ενώ οι άλλες 0. Μέσα στην συνθήκη ελέγχου κάθε περίπτωσης η συσκευή στέλνει ένα μήνυμα στο κινητό τηλέφωνο με την περίπτωση ελέγχου που εκτελεί.
Έτσι αν έχουμε παραπάνω από μια συσκευές δεν χρειάζεται να θυμάται ο χρήστης τι επιλογές είχε κάνει για κάθε συσκευή, αντίθετα θα ενημερώνεται από την συσκευή. Ομοίως αν τερματίσει την Android εφαρμογή BluePuc και την εκκινήσει πάλι θα γνωρίζει τι ελέγχους κάνει η συσκευή.

Τα όρια θερμοκρασίας και φωτεινότητας μπορούν να είναι μέγιστα ή ελάχιστα. Ελέγχουμε την τιμή της αντίστοιχης μεταβλητής (minmaxTemp), αν η τιμή είναι 0 δηλαδή το όριο είναι ελάχιστο και για θερμοκρασίες μικρότερες από το όριο δεν περνάει ρεύμα από την πρίζα (έχουμε δώσει λογικό LOW στο κύκλωμα για τον έλεγχο της πρίζας). Το πράσινο LED είναι σβησμένο ενώ δίνουμε λογικό HIGH για το κόκκινο LED που είναι για ένδειξη ότι δεν περνάει ρεύμα από την πρίζα και στέλνουμε το μήνυμα soff στο κινητό για να ενημερωθεί για την κατάσταση.

Αντίθετα αν η θερμοκρασία έχει τιμή ίση ή μεγαλύτερη από το όριο τότε δίνουμε λογικό HIGH στο κύκλωμα που ελέγχει την πρίζα επιτρέποντας να περάσει ρεύμα. Το πράσινο LED που είναι η ένδειξη ότι περνάει ρεύμα είναι αναμμένο ενώ το κόκκινο όχι και στέλνουμε το μήνυμα son στο κινητό.

Αν η τιμή της μεταβλητής minmaxTemp έχει τιμή 1, το όριο δηλαδή είναι μέγιστο τότε για να περνάει ρεύμα από την πρίζα θα πρέπει η θερμοκρασία να έχει τιμή μικρότερη ή ίση με το όριο.

```
//case 1.
if(temp_chk==1 && light_chk==0 && timer_chk==0 && time_period_chk==0){

  meetAndroid.send("scase1");

  if (minmaxTemp==0){
    if(temperatureC<tempLimit){
      digitalWrite(relayPin,LOW);
      digitalWrite(offrelay,HIGH);
      meetAndroid.send("soff");
    }
    else{
      digitalWrite(relayPin,HIGH);
      digitalWrite(offrelay,LOW);
      meetAndroid.send("son");
    }
  }
  else{
    if(temperatureC>tempLimit){
      digitalWrite(relayPin,LOW);
      digitalWrite(offrelay,HIGH);
      meetAndroid.send("soff");
    }
    else{
      digitalWrite(relayPin,HIGH);
      digitalWrite(offrelay,LOW);
      meetAndroid.send("son");
    }
  }
```

Για την περίπτωση που ο χρήστης έχει επιλέξει τον χρονοδιακόπτη μόνο, τότε η τιμή της μεταβλητής timer_chk θα έχει τιμή 1 ενώ οι άλλες 0. Οι επιλογές του χρήστη για ώρα, λεπτά και δευτερόλεπτα του χρονοδιακόπτη στέλνονται στην συσκευή σε έναν πίνακα, μετά για ευκολία αποθηκεύουμε τις τιμές αυτές σε τρεις μεταβλητές.

Εφόσον έχουμε προγραμματίσει την συσκευή να δουλεύει σαν ρολόι εκτελώντας όλες τις εντολές κάθε δευτερόλεπτο μπορούμε να την προγραμματίσουμε να δουλεύει και σαν χρονοδιακόπτης αν μειώνουμε κατά ένα την τιμή που έχει δώσει ο χρήστης.

Μέσα στην συνθήκη ελέγχου στέλνουμε πρώτα την περίπτωση ελέγχου στο κινητό και μετά ελέγχουμε τις 3 μεταβλητές του χρονοδιακόπτη. Πρώτα ελέγχουμε τα δευτερόλεπτα, αν έχουν τιμή μεγαλύτερη του 0 τότε αφαιρούμε ένα από την μεταβλητή. Ο χρονοδιακόπτης δεν έχει μηδενιστεί άρα περνάει ρεύμα από την πρίζα. Αν τα δευτερόλεπτα μηδενιστούν και τα λεπτά έχουν τιμή μεγαλύτερη ή ίση του 1, τότε αλλάζουμε την τιμή των δευτερολέπτων στην μέγιστη, δηλαδή 59 και αφαιρούμε ένα από την μεταβλητή των λεπτών. Αν τα δευτερόλεπτα και τα λεπτά έχουν τιμή 0 και οι ώρες έχουν τιμή μεγαλύτερη ή ίση του 1, τότε επαναφέρουμε τα δευτερόλεπτα και τα λεπτά στις μέγιστες τιμές τους (59) και αφαιρούμε ένα από την μεταβλητή των ωρών.

Αν και οι τρεις μεταβλητές μηδενιστούν, έχει τελειώσει δηλαδή ο χρονοδιακόπτης τότε δεν περνάει ρεύμα από την πρίζα.

```
//case 3.
if(timer_chk==1 && temp_chk==0 && light_chk==0 && time_period_chk==0){

  meetAndroid.send("scase3");
  if(timer_s>0){
    timer_s--;
    digitalWrite(relayPin,HIGH);
    digitalWrite(offrelay,LOW);
    meetAndroid.send("son");
  }
  if(timer_s==0 && timer_m>=1){
    timer_s=59;
    timer_m--;
    digitalWrite(relayPin,HIGH);
    digitalWrite(offrelay,LOW);
    meetAndroid.send("son");
  }
  if(timer_s==0 && timer_m==0 && timer_h>=1){
    timer_s=59;
    timer_m=59;
    timer_h--;
    digitalWrite(relayPin,HIGH);
    digitalWrite(offrelay,LOW);
    meetAndroid.send("son");
  }
  if (timer_s==0 && timer_m==0 && timer_h==0){
    digitalWrite(relayPin,LOW);
    digitalWrite(offrelay,HIGH);
    meetAndroid.send("soff");
  }
}
```

Για την περίπτωση που ο χρήστης έχει επιλέξει την χρονική περίοδο μόνο, τότε η τιμή της μεταβλητής time_period_chk θα έχει τιμή 1 ενώ οι άλλες 0. Όπως και πριν στέλνουμε στο κινητό την περίπτωση ελέγχου που εκτελεί.

Με την επιλογή της χρονικής περιόδου σημαίνει πως θα πρέπει να περνάει ρεύμα από την πρίζα για το διάστημα από την αρχική ώρα που έχει επιλέξει ο χρήστης μέχρι την τελική, ενώ για το υπόλοιπο διάστημα δεν θα περνάει ρεύμα από την πρίζα. Για να ελέγξουμε αν η ώρα είναι εντός αυτού του διαστήματος θα πρέπει να την συγκρίνουμε με την αρχική και τελική ώρα. Η τρέχουσα ώρα, η αρχική και η τελική ώρα που επέλεξε ο χρήστης αποτελούνται από δυο ακέραιους αριθμούς. Για να μπορέσουμε να κάνουμε τον έλεγχο πρέπει πρώτα να μετατρέψουμε αυτούς τους αριθμούς σε έναν για κάθε περίπτωση.

Για να μετατρέψουμε την αρχική ώρα από δυο ακέραιους σε έναν τετραψήφιο πολλαπλασιάζουμε την ώρα με το 100 και μετά προσθέτουμε τα λεπτά. Μετά ελέγχουμε την τελική ώρα που έχει επιλέξει ο χρήστης, αν είναι μεγαλύτερη ή ίση με την αρχική τότε μετατρέπουμε με τον ίδιο τρόπο την τελική ώρα και την τρέχουσα ώρα. Αν η τρέχουσα ώρα είναι μεγαλύτερη ή ίση με την αρχική και μικρότερη ή ίση με την τελική τότε περνάει ρεύμα από την πρίζα ενώ για το υπόλοιπο διάστημα όχι. Ένα παράδειγμα για αυτήν την περίπτωση είναι να είχε επιλέξει αρχική ώρα 14:25, τελική ώρα 22:00 και η τρέχουσα ώρα να είναι 18:30. Οι 3 ώρες θα μετατραπούν στους τετραψήφιους αριθμούς 1425, 2200

και 1830. Το 1830 είναι μεγαλύτερο του 1425 και μικρότερο του 2200 άρα περνάει ρεύμα.

```
//case 4.
if(time_period_chk==1 && temp_chk==0  && light_chk==0 && timer_chk==0){

  meetAndroid.send("scase4");

  //convert time-from
  temp_from_h=from_h*100;
  new_from_hm = temp_from_h + from_m;

  //first we compare end-time to start-time
  if (to_h>=from_h){


    //convert time-to
    temp_to_h = to_h*100;
    new_to_hm = temp_to_h + to_m;

    //convert current time
    temp_hours = hours*100;
    temp_hoursmin = temp_hours + minutes;


    if(temp_hoursmin>=new_from_hm && temp_hoursmin<=new_to_hm){
      digitalWrite(relayPin,HIGH);
      digitalWrite(offrelay,LOW);
      meetAndroid.send("son");
    }
    else{
      digitalWrite(relayPin,LOW);
      digitalWrite(offrelay,HIGH);
      meetAndroid.send("soff");
    }
  }
```

Η άλλη περίπτωση είναι η τελική ώρα που έχει επιλέξει ο χρήστης να είναι μικρότερη της αρχικής. Σε αυτή την περίπτωση προσθέτουμε το 24 στην τελική ώρα, το πολλαπλασιάζουμε με το 100 και προσθέτουμε μετά τα λεπτά.

Προτού μετατρέψουμε την τρέχουσα ώρα ελέγχουμε την τιμή της, αν η ώρα είναι μεγαλύτερη ή ίση του 0 και μικρότερη ή ίση με την τελική ώρα που επέλεξε ο χρήστης τότε προσθέτουμε το 24 στην τρέχουσα ώρα, το πολλαπλασιάζουμε με το 100 και προσθέτουμε μετά τα λεπτά. Αντίθετα αν η τρέχουσα ώρα είναι μεγαλύτερη από την τελική ώρα που επέλεξε ο χρήστης τότε πολλαπλασιάζουμε την ώρα με το 100 και μετά προσθέτουμε τα λεπτά.

Ελέγχουμε την τρέχουσα ώρα και αν είναι εντός του διαστήματος τότε περνάει ρεύμα από την πρίζα. Ένα παράδειγμα για αυτήν την περίπτωση είναι να έχει επιλέξει ο χρήστης αρχική ώρα 22:15, τελική ώρα 1:00 και η τρέχουσα ώρα είναι 0:27. Οι 3 ώρες θα μετατραπούν στους τετραψήφιους αριθμούς 2215, 2500 ((24+1)*100+0) και 2427. Το 2427 είναι μεγαλύτερο του 2215 και μικρότερο του 2500, άρα θα περνάει ρεύμα από την πρίζα. Αν η τρέχουσα ώρα γίνει 8:55 τότε δεν θα προσθέσουμε σε αυτήν το 24 καθώς το 8 είναι μεγαλύτερο του 1, θα γίνει δηλαδή 855 όπου ο αριθμός αυτός είναι εκτός του διαστήματος άρα δεν θα περνάει ρεύμα.

```
  if (to_h<from_h){
    //change time-to (+24)
    help_to_h = (to_h+24)*100;
    help_to_hm = help_to_h + to_m;

    if (hours>=0 && hours<=to_h){
      help_hours = (hours+24)*100;
      help_hoursmin = help_hours + minutes;  //help_hoursmin -> current time
    }
    else{
      help_hours = hours*100;
      help_hoursmin = help_hours + minutes; //help_hoursmin -> current time
    }

    if (help_hoursmin>=new_from_hm && help_hoursmin<=help_to_hm){
      digitalWrite(relayPin,HIGH);
      digitalWrite(offrelay,LOW);
      meetAndroid.send("son");
    }
    else{
      digitalWrite(relayPin,LOW);
      digitalWrite(offrelay,HIGH);
      meetAndroid.send("soff");
    }

  }


}//end - case 4.
```

Όπως είδαμε και στον πίνακα 6.1 ο χρήστης μπορεί να επιλέξει να γίνονται μέχρι και τρεις έλεγχοι ταυτόχρονα. Για την περίπτωση που ο χρήστης έχει επιλέξει να γίνετε έλεγχος για θερμοκρασία και φωτεινότητα πρέπει να ελέγξουμε τα δυο όρια αν είναι μέγιστα ή ελάχιστα.

Υπάρχουν τέσσερις περιπτώσεις: 1) τα δυο όρια να είναι ελάχιστα, 2) το όριο θερμοκρασίας να είναι ελάχιστο και το όριο φωτεινότητας μέγιστο, 3) το όριο θερμοκρασίας να είναι μέγιστο και το όριο φωτεινότητας ελάχιστο, 4) τα δυο όρια να είναι μέγιστα.

Οπότε και για τις τέσσερις υποπεριπτώσεις για να περνάει ρεύμα από την πρίζα θα πρέπει να ισχύουν ταυτόχρονα και οι δυο έλεγχοι των ορίων θερμοκρασίας και φωτεινότητας. Για παράδειγμα για την $1^{η}$ υποπερίπτωση που τα δυο όρια είναι ελάχιστα θα πρέπει η θερμοκρασία και η φωτεινότητα να έχουν τιμές μεγαλύτερες ή ίσες των ορίων για να περνάει ρεύμα από την πρίζα. Αντίθετα αν έστω και ένας από τους δυο ελέγχους των ορίων δεν ισχύει τότε δεν θα περνάει ρεύμα από την πρίζα.

```
//case 5.
if(temp_chk==1 && light_chk==1 && timer_chk==0 && time_period_chk==0){

  meetAndroid.send("scase5");

  //sub-case1:
  if (minmaxTemp==0 && minmaxLight==0){
    if(temperatureC<tempLimit || lightLvl<lightLimit){
      digitalWrite(relayPin,LOW);
      digitalWrite(offrelay,HIGH);
      meetAndroid.send("soff");
    }
    else if (temperatureC>=tempLimit && lightLvl>=lightLimit){
      digitalWrite(relayPin,HIGH);
      digitalWrite(offrelay,LOW);
      meetAndroid.send("son");
    }
  }//sub-case2:
  else if(minmaxTemp==0 && minmaxLight==1){
    if(temperatureC<tempLimit || lightLvl>lightLimit){
      digitalWrite(relayPin,LOW);
      digitalWrite(offrelay,HIGH);
      meetAndroid.send("soff");
    }
    else if (temperatureC>=tempLimit && lightLvl<=lightLimit){
      digitalWrite(relayPin,HIGH);
      digitalWrite(offrelay,LOW);
      meetAndroid.send("son");
    }
  }//sub-case3:
  else if (minmaxTemp==1 && minmaxLight==0){
    if(temperatureC>tempLimit || lightLvl<lightLimit){
      digitalWrite(relayPin,LOW);
      digitalWrite(offrelay,HIGH);
      meetAndroid.send("soff");
    }
    else if (temperatureC<=tempLimit && lightLvl>=lightLimit){
      digitalWrite(relayPin,HIGH);
      digitalWrite(offrelay,LOW);
      meetAndroid.send("son");
    }
  }//sub-case4(both max):
  else {
    if(temperatureC>tempLimit || lightLvl>lightLimit){
      digitalWrite(relayPin,LOW);
      digitalWrite(offrelay,HIGH);
      meetAndroid.send("soff");
    }
    else if (temperatureC<=tempLimit && lightLvl<=lightLimit){
      digitalWrite(relayPin,HIGH);
      digitalWrite(offrelay,LOW);
      meetAndroid.send("son");
    }
  }


}//end - case 5.
```

Για την περίπτωση που ο χρήστης έχει επιλέξει τον χρονοδιακόπτη μαζί με τον έλεγχο θερμοκρασίας θα συνδυάσουμε τους ελέγχους από τη περίπτωση 1 και 3. Μέχρι να μηδενιστεί ο χρονοδιακόπτης ελέγχουμε πρώτα το όριο εάν είναι μέγιστο ή ελάχιστο και μετά ελέγχουμε την τιμή της θερμοκρασίας. Αν ο χρονοδιακόπτης μηδενιστεί τότε δεν χρειάζεται να ελέγχουμε άλλο την θερμοκρασία.

```
//case 6
if(temp_chk==1 && light_chk==0 && timer_chk==1 && time_period_chk==0){

  meetAndroid.send("scase6");

  if(timer_s>0){
    timer_s--;

    if (minmaxTemp==0){
      if(temperatureC<tempLimit){
        digitalWrite(relayPin,LOW);
        digitalWrite(offrelay,HIGH);
        meetAndroid.send("soff");
      }
      else{
        digitalWrite(relayPin,HIGH);
        digitalWrite(offrelay,LOW);
        meetAndroid.send("son");
      }
    }
    else{
      if(temperatureC>tempLimit){
        digitalWrite(relayPin,LOW);
        digitalWrite(offrelay,HIGH);
        meetAndroid.send("soff");
      }
      else{
        digitalWrite(relayPin,HIGH);
        digitalWrite(offrelay,LOW);
        meetAndroid.send("son");
      }
    }

  }
  if(timer_s==0 && timer_m>=1){
    timer_s=59;
    timer_m--;
    if (minmaxTemp==0){
      if(temperatureC<tempLimit){
        digitalWrite(relayPin,LOW);
        digitalWrite(offrelay,HIGH);
        meetAndroid.send("soff");
      }
      else{
        digitalWrite(relayPin,HIGH);
        digitalWrite(offrelay,LOW);
        meetAndroid.send("son");
      }
    }
```

```
    else{
      if(temperatureC>tempLimit){
        digitalWrite(relayPin,LOW);
        digitalWrite(offrelay,HIGH);
        meetAndroid.send("soff");
      }
      else{
        digitalWrite(relayPin,HIGH);
        digitalWrite(offrelay,LOW);
        meetAndroid.send("son");
      }
    }
  }
  if(timer_s==0 && timer_m==0 && timer_h>=1){
    timer_s=59;
    timer_m=59;
    timer_h--;
    if (minmaxTemp==0){
      if(temperatureC<tempLimit){
        digitalWrite(relayPin,LOW);
        digitalWrite(offrelay,HIGH);
        meetAndroid.send("soff");
      }
      else{
        digitalWrite(relayPin,HIGH);
        digitalWrite(offrelay,LOW);
        meetAndroid.send("son");
      }
    }
    else{
      if(temperatureC>tempLimit){
        digitalWrite(relayPin,LOW);
        digitalWrite(offrelay,HIGH);
        meetAndroid.send("soff");
      }
      else{
        digitalWrite(relayPin,HIGH);
        digitalWrite(offrelay,LOW);
        meetAndroid.send("son");
      }
    }
  }
  if (timer_s==0 && timer_m==0 && timer_h==0){
    digitalWrite(relayPin,LOW);
    digitalWrite(offrelay,HIGH);
    meetAndroid.send("soff");
  }


}//end - case 6.
```

Ομοίως και για τις άλλες δυο περιπτώσεις ελέγχου θερμοκρασίας ή και φωτεινότητας μαζί με τον χρονοδιακόπτη, ελέγχουμε πρώτα αν ο χρονοδιακόπτης έχει μηδενιστεί και αν δεν έχει μηδενιστεί τότε συνεχίζουμε και στους υπόλοιπους ελέγχους.

Για τις περιπτώσεις που ο χρήστης έχει επιλέξει το χρονικό διάστημα μαζί με κάποιον έλεγχο θερμοκρασίας ή και φωτεινότητας τότε ελέγχουμε πρώτα αν η ώρα στο Arduino είναι εντός του χρονικού διαστήματος, όπως στην περίπτωση 4 που περιγράψαμε ποιο πάνω και μετά συνεχίζουμε στους άλλους ελέγχους. Αν η ώρα δεν είναι εντός του χρονικού διαστήματος τότε δεν θα περνάει ρεύμα από την πρίζα.

7.3 Κατασκευή της συσκευής

Για την κατασκευή της συσκευής χρησιμοποιήθηκαν τα παρακάτω υλικά:

- Arduino Uno
- Bluetooth Module HC-06
- Οθόνη LCD 16x2
- Διάτρητη πλακέτα
- Φωτοτρανζίστορ 4n25
- Τρανζίστορ 2n2222a
- Δίοδος 1n4007
- Relay 5V DC
- 2 Κλέμες
- LED Κόκκινο
- LED Πράσινο
- 2 Πλήκτρα
- Ποτενσιόμετρο 10KΩ
- Αντίσταση 270Ω
- Αντίσταση 5KΩ
- 2 Αντιστάσεις 150Ω
- 3 Αντιστάσεις 10KΩ
- Φωτοαντίσταση
- LM35DZ
- Header pins
- Τροφοδοτικό DC 7-12V
- Ξύλο κόντρα πλακέ
- Ξύλο MDF
- Πολύπριζο
- Πρίζα επίτοιχη
- Προέκταση καλωδίου (φις σούκο)
- Αποστάτες
- Βίδες / παξιμάδια
- Καλώδια

Εκτός από τα παραπάνω υλικά χρησιμοποιήθηκαν και τα απαραίτητα εργαλεία όπως κολλητήρι, καλάι, κόλλα, τρυπάνι, κόφτης και πολύμετρο για έλεγχο.

Για την κατασκευή της συσκευής ακολουθήσαμε την εξής διαδικασία, πρώτα κάνουμε όλες τις τρύπες στις πλευρές της συσκευής για να μπορέσουμε να τις ενώσουμε αλλά και για να ενώσουμε τα εξαρτήματα

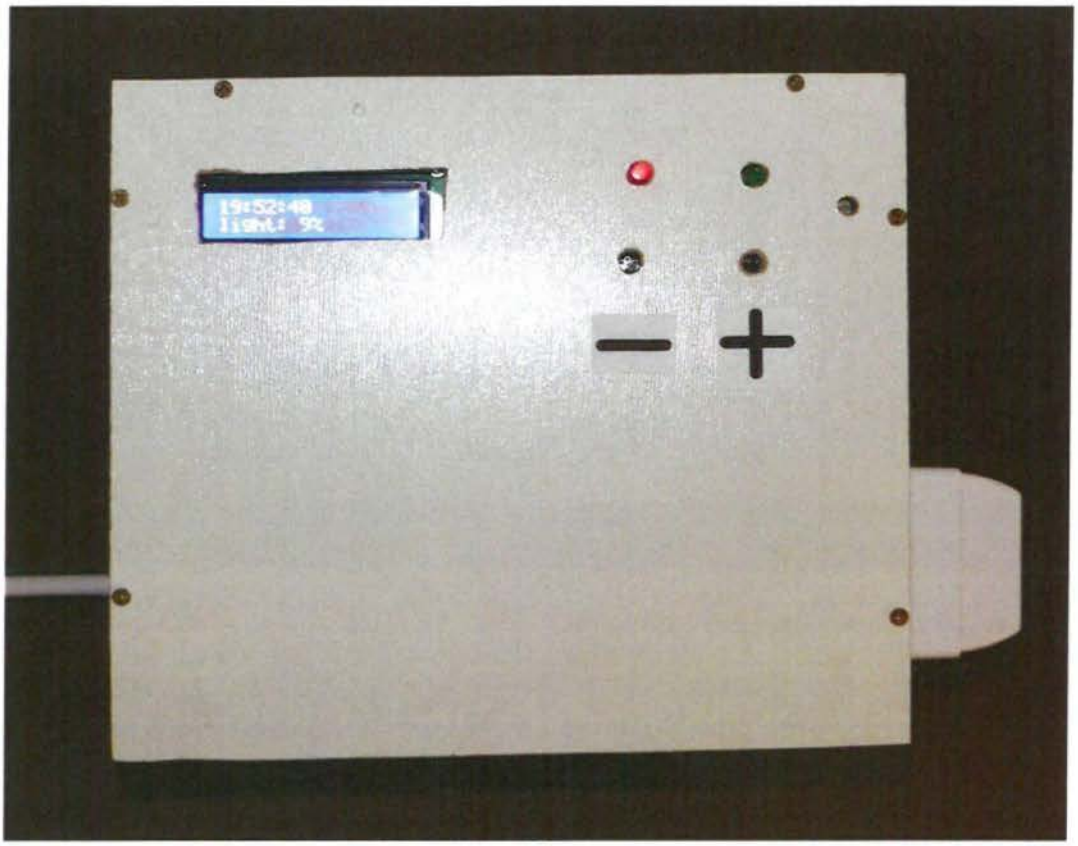
όπως το πολύπριζο και την πρίζα που θα ελέγχουμε. Τοποθετούμε στο πάτο της συσκευής (στο κόντρα πλακέ) το Arduino Uno, την πλακέτα για τον έλεγχο της πρίζας και το πολύπριζο. Ενώνουμε τις 3 πλευρές της συσκευής δημιουργώντας ένα Π και τοποθετούμε τις βάσεις για τις πλακέτες στο επάνω μέρος της συσκευής. Συνδέουμε τις τρεις πλευρές με τον πάτο της συσκευής και μετά βιδώνουμε την επίτοιχη πρίζα στο εξωτερικό μέρος της συσκευής. Τοποθετούμε στις δυο βάσεις τις πλακέτες, μια που έχει την οθόνη και την άλλη με τα LED ένδειξης κατάστασης της πρίζας, τα 2 πλήκτρα και την φωτοαντίσταση. Συνδέουμε στο πολύπριζο το τροφοδοτικό και την προέκταση καλωδίου. Ενώνουμε όλες τις πλακέτες με τα καλώδια που χρειάζονται και τα καλώδια της προέκτασης στην μια κλέμα (IN). Ενώνουμε την πρίζα με τα καλώδια στην άλλη κλέμα (OUT). Τέλος βιδώνουμε την μια πλευρά και την κορυφή της συσκευής.

Σχήμα 7.4: Το εσωτερικό της συσκευής

Η συσκευή είναι φτιαγμένη έτσι ώστε να μπορούν να γίνουν εύκολα τροποποιήσεις, για παράδειγμα οι δυο πλακέτες που βρίσκονται στο επάνω μέρος της συσκευής μπορούν να αφαιρεθούν. Η οθόνη μπορεί να αφαιρεθεί και να αντικατασταθεί με κάποια άλλη. Το Bluetooth Module είναι συνδεδεμένο με header pins επιτρέποντας έτσι να το αφαιρέσουμε αν θέλουμε για να το προγραμματίσουμε ή να το αντικαταστήσουμε με κάποιο άλλο και μετά να το συνδέσουμε πάλι. Έτσι ο χρήστης μπορεί αν θέλει να κάνει τροποποιήσεις στην συσκευή ανάλογα με τις ανάγκες του.

7.4 Περιπτώσεις χρήσης

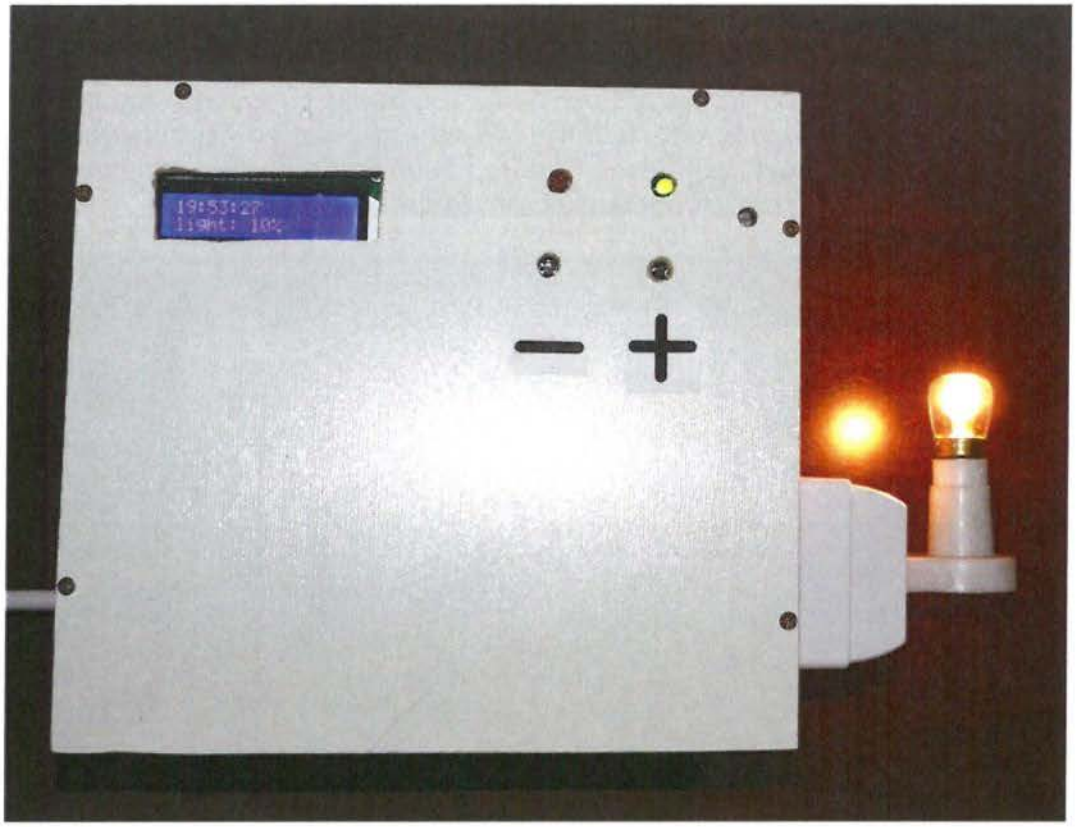
Για να μπορέσουμε να ελέγξουμε την πρίζα της συσκευής συνδέουμε πρώτα το καλώδιο της συσκευής σε μια πρίζα για να μπορέσει να λειτουργήσει και μετά μέσω της Android εφαρμογής BluePuc συνδεόμαστε μέσω Bluetooth με την συσκευή και από εκεί μπορούμε να ελέγξουμε την πρίζα.



Σχήμα 7.5: Η συσκευή σε λειτουργία

Στην πρίζα της συσκευής μπορούμε να συνδέσουμε κάποια συσκευή που θέλουμε να ελέγχουμε απομακρυσμένα. Όπως είδαμε και στο κεφάλαιο 6.3 μπορούμε να ελέγξουμε την συσκευή με ένα απλό διακόπτη ON / OFF ή να επιλέξουμε να γίνονται ένας ή και περισσότεροι έλεγχοι ώστε να λειτουργεί αυτόματα η συσκευή.

Παρακάτω αναλύονται μερικές περιπτώσεις χρήσης της συσκευής. Μπορούμε να ελέγχουμε μέσω της συσκευής μικρές οικιακές συσκευές. Για παράδειγμα το καλοκαίρι μπορούμε να συνδέσουμε έναν ανεμιστήρα στην συσκευή και να ορίσουμε το ελάχιστο όριο θερμοκρασίας, έτσι μόλις η θερμοκρασία ξεπεράσει το όριο που έχουμε δώσει ο ανεμιστήρας θα ξεκινήσει να δουλεύει και μόλις η θερμοκρασία πέσει κάτω από το όριο τότε θα σταματήσει. Μπορούμε να ελέγχουμε τα φώτα στο σπίτι με την συσκευή αν ορίσουμε το όριο φωτεινότητας να είναι μέγιστο. Έτσι αν η φωτεινότητα είναι πάνω από το όριο που έχουμε δώσει τα φώτα είναι σβηστά, αν όμως η φωτεινότητα πέσει κάτω από το όριο (π.χ νυχτώνει) τότε θα φώτα θα ανάψουν αυτόματα.



Σχήμα 7.6: Η συσκευή μαζί με λάμπα

Όπως είδαμε και προηγουμένως η συσκευή μπορεί να δουλεύει σαν χρονοδιακόπτης, έτσι μπορούμε να επιλέξουμε πόση ώρα θα δουλεύει η συσκευή που έχουμε συνδέσει στην πρίζα. Μπορούμε να συνδέσουμε ένα ραδιόφωνο και να επιλέξουμε πόσες ώρες, λεπτά και δευτερόλεπτα θα δουλεύει. Αν έχουμε συνδέσει κάποια συσκευή και θέλουμε να δουλεύει για ένα συγκεκριμένο χρονικό διάστημα μόνο τότε μπορούμε να θέσουμε από το κινητό την αρχική και τελική ώρα λειτουργίας, για παράδειγμα μπορούμε να συνδέσουμε μια καφετιέρα για να ετοιμάζει καφέ κάθε πρωί και μετά να σβήνει.
Εκτός από αυτές τις περιπτώσεις χρήσης υπάρχουν και άλλες πολλές επιτρέποντας μας να ελέγχουμε διάφορες συσκευές από το κινητό τηλέφωνο.

Βιβλιογραφία

[1] Mark L. Murphy, "Android Programming Tutorials", CommonsWare, LLC, USA, 2011

[2] Lauren Darcey & Shane Conder, "Sams Teach Yourself Android Application Development in 24 Hours", Sams Publishing, Indianapolis, Indiana, 2010

[3] Wei-Meng Lee, "Beginning Android Application Development", Wiley Publishing, Inc., Indianapolis, Indiana, 2011

[4] Marko Gargenta, "Learning Android", O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472, USA, 2011

[5] Reto Meier, "Professional Android Application Development", Wiley Publishing, Inc., Indianapolis, Indiana, 2009

[6] Brian W. Evans "Arduino Programming Notebook", 2008

[7] Mike McRoberts, "Arduino Starters Kit Manual – A Complete Beginners Guide to the Arduino", Earthshine Design, 2010

[8] Tom Igoe, "Making Things Talk", O'Reilly Media, Inc. 1005 Gravenstein Highway North, Sebastopol, CA 95472, Canada, 2007

[9] Mike Riley, "Programming Your Home - Automate with Arduino, Android, and Your Computer", The Pragmatic Programmers, LLC, 2012

[10] Bonifaz Kaufmann, "Design and Implementation of a Toolkit for the Rapid Prototyping of Mobile Ubiquitous Computing", Μεταπτυχιακή Εργασία, Σχολή Τεχνικών Επιστημών, Πανεπιστήμιο του Klagenfurt, 2010

[11] Λήψη του Android SDK και του Eclipse IDE με το ADT plug-in:
http://developer.android.com/sdk/index.html

[12] Λήψη του JDK:
http://www.oracle.com/technetwork/java/javase/downloads/index.html

[13] Λήψη του Arduino IDE:
http://arduino.cc/en/Main/Software

[14] Λήψη του EAGLE Layout Editor:
http://www.cadsoftusa.com/download-eagle/

[15] Λήψη της βιβλιοθήκης Adafruit Eagle library:
https://github.com/adafruit/Adafruit-Eagle-Library

[16] Λήψη του Amarino toolkit:
http://www.amarino-toolkit.net/index.php/download.html

[17] Πληροφορίες για το Android:
http://developer.android.com/about/index.html

http://en.wikipedia.org/wiki/Android_%28operating_system%29

[18] Αρχιτεκτονική του Android:
http://electronics.howstuffworks.com/google-phone2.htm

http://www.tutorialspoint.com/android/android_architecture.htm

[19] Συνιστώσες μιας εφαρμογής:
http://developer.android.com/guide/components/fundamentals.html

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

[20] Το Manifest αρχείο:
http://developer.android.com/guide/topics/manifest/manifest-intro.html

[21] Οι πόροι μιας εφαρμογής:
http://developer.android.com/guide/topics/resources/index.html

[22] Πληροφορίες για το Arduino:
http://www.arduino.cc

[23] Πλακέτα Arduino Uno:
http://arduino.cc/en/Main/ArduinoBoardUno

[24] Arduino βιβλιοθήκες:
http://arduino.cc/en/Reference/Libraries

[25] Πληροφορίες για το Amarino toolkit:
http://www.amarino-toolkit.net/index.php/docs.html

[26] Δημιουργία Android Project:
http://developer.android.com/training/basics/firstapp/creating-project.html

[27] Διαχείριση εικονικής μηχανής Android:
http://developer.android.com/tools/devices/index.html

[28] Πληροφορίες για το εργαλείο LogCat:
http://developer.android.com/tools/help/logcat.html

[29] Δημοσίευση εφαρμογής:
http://developer.android.com/tools/publishing/publishing_overview.html

[30] Πληροφορίες για Layout:
http://developer.android.com/guide/topics/ui/declaring-layout.html

[31] Πληροφορίες για ενέργειες με Bluetooth:
http://developer.android.com/guide/topics/connectivity/bluetooth.html

[32] Πληροφορίες για κατασκευή κυκλώματος και προγραμματισμός του Arduino:
http://playground.arduino.cc

[33] Forum προγραμματιστών:
http://stackoverflow.com

[34] Forum Arduino:
http://forum.arduino.cc

[35] Forum Amarino toolkit:
https://groups.google.com/forum/#!forum/amarino-toolkit