ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΣΥΣΤΗΜΑΤΩΝ ΜΕΤΑΔΟΣΗΣ ΠΛΗΡΟΦΟΡΙΑΣ
ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ ΥΛΙΚΩΝ

Ευφυής Οικία για τον αυτοματισμό περιβαλλοντικών συνθηκών μέσω κινητών εφαρμογών Ανοικτού Υλικού και Λογισμικού

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

Αλέξανδρου Ι. Γεωργαντά

Επιβλέπων: Φίλιππος Κωνσταντίνου
Καθηγητής Ε.Μ.Π

Αθήνα, Ιούλιος 2013

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΣΥΣΤΗΜΑΤΩΝ ΜΕΤΑΔΟΣΗΣ ΠΛΗΡΟΦΟΡΙΑΣ
ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ ΥΛΙΚΩΝ

Ευφυής Οικία για τον αυτοματισμό περιβαλλοντικών συνθηκών μέσω κινητών εφαρμογών Ανοικτού Υλικού και Λογισμικού

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

Αλέξανδρου Ι. Γεωργαντά

Επιβλέπων: Φίλιππος Κωνσταντίνου
Καθηγητής Ε.Μ.Π

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την

.....	.....	.....
Φίλιππος Κωνσταντίνου Καθηγητής Ε.Μ.Π	Αθανάσιος Παναγόπουλος Λέκτορας Ε.Μ.Π	Κωνσταντίνα Νικήτα Καθηγήτρια Ε.Μ.Π

Αθήνα, Ιούλιος 2013

.....
Αλέξανδρος Ι. Γεωργαντάς

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

Ευχαριστίες

Πρωτίστως θα ήθελα να ευχαριστήσω την οικογένεια μου για την συμπαράσταση σε όλα μου τα βήματα καθώς και την μεγάλη υποστήριξη και κατανόηση που έδειξαν τα χρόνια των σπουδών μου.

Επίσης θα ήθελα να ευχαριστήσω τον καθηγητή μου Κωνσταντίνου Φίλιππο για την ευκαιρία που μου έδωσε με αυτή την Διπλωματική Εργασία να ανακαλύψω καινούργιους τεχνολογικούς δρόμους και να διευρύνω τις γνώσεις μου ως Μηχανικός.

Τέλος θα ήθελα να ευχαριστήσω πολύ θερμά τον συνεπιβλέποντα μου Καραγιάννη Αλέξανδρο για την αμέριστη βοήθεια και συμπαράσταση που μου πρόσφερε καθ’ όλη την διάρκεια εκπόνησης αυτής της Διπλωματικής Εργασίας.

Περιεχόμενα

Περίληψη

Ο σκοπός αυτής της διπλωματικής εργασίας είναι η ανάπτυξη και η μελέτη μιας διαδραστικής εφαρμογής για την υλοποίηση τηw ιδέας της ευφυούς οικίας. Με τον όρο ευφυής οικία εννοείται ένας χώρος του οποίου οι φυσικές παράμετροι παρακολουθούνται και ελέγχονται είτε με την παρέμβαση ενός χρήστη, είτε αυτόματα, λαμβάνοντας υπόψη επιλογές άλλοτε προσωπικού περιεχομένου και άλλοτε βελτιστοποίησης μεγεθών και συνθηκών.

Η ανάπτυξη της εφαρμογής στηρίχτηκε στις τεχνολογίες ασύρματων αισθητήρων για την παρακολούθηση φυσικών μεγεθών όπως η θερμοκρασία ενός χώρου και η ένταση της φωτεινότητας, καθώς επίσης έγινε χρήση και της πλατφόρμας ανοικτού υλικού Arduino για την ενεργοποίηση φωτιστικών σωμάτων από τον χρήστη.

Στο επίπεδο λογισμικού έγινε χρήση τεχνολογιών διαδικτύου βασισμένων σε ανοικτά πρότυπα. Επίσης χρησιμοποιήθηκε και η πλατφόρμα έξυπνων κινητών τηλεφώνων Android. Με αυτό τον τρόπο υλοποιήθηκε μια εφαρμογή διαδικτύου για χρήση από τερματικά τα οποία είναι συνδεδεμένα στο διαδίκτυο καθώς επίσης υλοποιήθηκε και μια εφαρμογή η οποία εκτελείται σε έξυπνα κινητά εφοδιασμένα με λειτουργικό σύστημα Android.

Στο επίπεδο σχεδιασμού της εφαρμογής υλοποίησης της ευφυούς οικίας, το ασύρματο δίκτυο αισθητήρων παρέχει το σύστημα παρακολούθησης και καταγραφής φυσικών παραμέτρων, ενώ το σύστημα που βασίζεται στην πλατφόρμα Arduino υλοποιεί το ασύρματο δίκτυο που ελέγχει και καθορίζει τις φυσικές παραμέτρους με αυτοματοποιημένο ή χειροκίνητο τρόπο. Οι διεπαφές παρακολούθησης και ελέγχου παρέχονται είτε διαδικτυακά με χρήση της εφαρμογής που αναπτύχθηκε είτε μέσω ευφυούς τηλεφώνου.

Λέξεις κλειδιά: Ασύρματα Δίκτυα Αισθητήρων, διαδίκτυο, έξυπνα κινητά τηλέφωνα, Tmote, Arduino, Android, ZigBee, 802.15.4, TinyOS, PHP, JavaScript, HTML, Java, MySQL, Apache.

Abstract

The purpose of this dissertation is to develop and study an interactive application, which will implement applications that constitute the concept of the “intelligent house”. As an “intelligent house” is considered a wider area in which all physical parameters are being monitored and controlled either via direct intervention of the occupant, or automatically, considering once personally set preferences or another time optimization criteria of specific figures under different conditions.

Application development used wireless sensor technology to monitor physical parameters such as room temperature and luminosity, while Arduino open source hardware enables the user to activate lighting devices.

On software level open access standards were used. Moreover Android smartphone platform was utilized. Thus an Internet application was created that can be accessed from any terminal connected to the Internet, while a separate application offers the same options to Android running smartphones’ users.

The aforementioned steps are crucial to the implementation of the “intelligent house” concept. Wireless sensors provide monitoring and recording of house conditions while Arduino platform provides the network that defines and controls parameters, automatically or manually. Additionally command and control interfaces are accessible either via the Internet or through a specially designed Android application

Keywords: Wireless Sensor Networks, internet, smartphones, Tmote, Arduino, Android, ZigBee, 802.15.4, TinyOS, PHP, JavaScript, HTML, Java, MySQL, Apache.

Εισαγωγή

Την σημερινή εποχή οι γοργοί ρυθμοί εξέλιξης της τεχνολογίας έχουν μεγάλο αντίκτυπο στην καθημερινή ζωή των ανθρώπων σε επίπεδο δυνατοτήτων και ευκολιών. Η ευχέρεια στην πρόσβαση πληροφοριών παντός τύπου και μορφής είναι κάτι που χαρακτηρίζει τις σημερινές μέρες λόγω της ευρείας διάδοσης του διαδικτύου. Ως αποτέλεσμα των ανωτέρω είναι και η συνεχής ανάπτυξη νέων τάσεων και εφαρμογών.

Συγκεκριμένα τα τελευταία χρόνια παρατηρείται η ανάπτυξη ενός κλάδου που αναφέρεται βιβλιογραφικά με τον όρο διαδίκτυο των πραγμάτων (Internet of Things). Αυτός ο κλάδος περιγράφει την διασύνδεση υλικών οντοτήτων και αντικειμένων σε μια μορφή διαδικτύου έτσι ώστε να παρέχουν στον χρήστη πληροφορίες είτε για την κατάσταση της λειτουργία τους, είτε για διάφορα γεγονότα που έχουν αξία, είτε για να παρέχεται η δυνατότητα ενεργειών πάνω σε αυτά ανεξάρτητα από την φυσική τοποθεσία του χρήστη. Σημαντικό κομμάτι αυτού του κλάδου βασίζεται στις τεχνολογίες δικτύων ασύρματων κόμβων αισθητήρων. Τα δίκτυα αυτά αποτελούνται από μικρούς σε διαστάσεις κόμβους, σχεδιασμένους με έμφαση στην χαμηλή κατανάλωση για μεγάλη αυτονομία λειτουργίας, οι οποίοι παρέχουν πληροφορίες σχετικά με διάφορα φυσικά φαινόμενα τα οποία δειγματοληπτούν μέσω μιας μεγάλης ποικιλίας αισθητήρων που έχουν αναπτυχθεί για το σκοπό αυτό.

Ο συνδυασμός αυτών των τεχνολογιών μαζί με την ραγδαία ανάπτυξη των έξυπνων κινητών τηλεφώνων (smartphones) και κυρίως των δυνατοτήτων τους όπως οι πολλαπλές διεπαφές συνδεσιμότητας ανά κινητό τερματικό και η αύξηση της επεξεργαστικής ισχύος αυτών, δίνει στον τελικό χρήστη την ευελιξία ελέγχου και καθορισμού των παραμέτρων απομακρυσμένων αντικείμενων με βάση τις προσωπικές επιθυμίες και προτιμήσεις. Ένα παράδειγμα εφαρμογής των ανωτέρω είναι η δυνατότητα του χρήστη να πληροφορείται για την θερμοκρασία των δωματίων του σπιτιού μέσω του έξυπνου κινητού τηλεφώνου που διαθέτει, και η ρύθμιση της σύμφωνα με τις προτιμήσεις χωρίς την παρουσία του ίδιου στο χώρο.
Στην ανάλυση που ακολουθεί γίνεται η περιγραφή της υλοποίησης μιας τέτοιας εφαρμογής που βασίζεται στις προαναφερθείσες τεχνολογίες. Στα παρακάτω κεφάλαια περιλαμβάνονται:

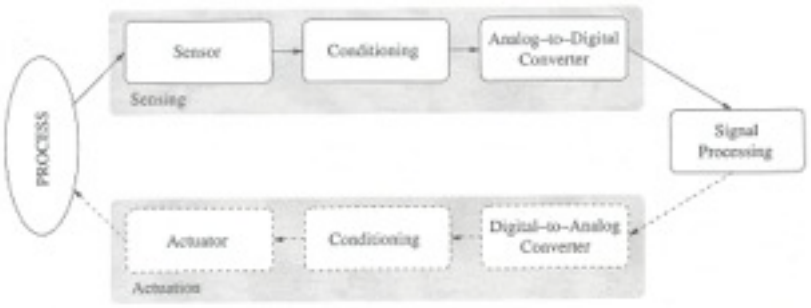
- Η ανάπτυξη του θεωρητικού υποβάθρου που σχετίζεται με την υλοποίηση της εφαρμογής, όπως η έννοιες της αίσθησης, της δειγματοληψίας, των ασύρματων δικτύων και της διαχείρισης δεδομένων.
- Η περιγραφή και ανάλυση των χαρακτηριστικών των υλικών που απαιτούνται για την κατασκευή του δικτύου όπως οι ασύρματοι κόμβοι αισθητήρων Tmote και η πλατφόρμα Arduino.
- Η αρχιτεκτονική της εφαρμογής σε επίπεδο υλοποίησης με έμφαση στον σχεδιασμό του ασύρματου δικτύου αισθητήρων, την συλλογή δεδομένων από φυσικές παραμέτρους, την διασύνδεση σημείου συλλογής των δεδομένων με το διαδίκτυο, και την ανάπτυξη βάσης δεδομένων.
- Η κατασκευή του δικτύου σε λειτουργικό επίπεδο, η ανάπτυξη εφαρμογών για τους κόμβους αισθητήρων Tmote και την πλατφόρμα Arduino, η ανάπτυξη διαδικτυακής εφαρμογής, και η ανάπτυξη εφαρμογής λειτουργικού έξυπνων κινητών τηλεφώνων Android.

ΚΕΦΑΛΑΙΟ 1: Θεωρητικό Υπόβαθρο Δικτύων Ασύρματων Αισθητήρων

Αίσθηση και Αισθητήρες

Αίσθηση είναι η τεχνική που χρησιμοποιείται για να την συλλογή πληροφοριών που σχετίζονται με φυσικά αντικείμενα και φαινόμενα, με διαδικασίες, και με την εμφάνιση-ύπαρξη γεγονότων. Ένα παράδειγμα είναι η συλλογή πληροφοριών για την αλλαγή στην θερμοκρασία ή την πίεση.

Το αντικείμενο που εκτελεί τέτοιου είδους διεργασίες ονομάζεται αισθητήρας. Από τεχνικής άποψης, ο αισθητήρας είναι μια συσκευή που μεταφράζει παραμέτρους ή γεγονότα από τον φυσικό κόσμο σε σήματα τα οποία μπορούν να μετρηθούν και να αναλυθούν. Ένας άλλος συχνά εμφανιζόμενος όρος είναι ο μετατροπέας ή μορφοτροπέας (transducer), που χρησιμοποιείται για να περιγράψει μια συσκευή η οποία μετατρέπει ενέργεια από μια μορφή σε μία άλλη. Ο αισθητήρας λοιπόν είναι ένας μετατροπέας που μετατρέπει ενέργεια που σχετίζεται με μια φυσική παράμετρο σε ηλεκτρική ενέργεια και μπορεί να εισαχθεί σε ένα υπολογιστικό σύστημα ή ελεγκτή. Ένα παράδειγμα για τα βήματα που ακολουθούνται στην διαδικασία της αίσθησης βρίσκεται στην εικόνα 1.



Εικόνα 1: Βήματα διαδικασίας αίσθησης

Φαινόμενα που συμβαίνουν στον φυσικό κόσμο παρακολουθούνται από μια συσκευή αισθητήρα. Συνήθως, τα ηλεκτρικά σήματα που προκύπτουν δεν είναι έτοιμα για άμεση επεξεργασία από τις επόμενες βαθμίδες ενός ηλεκτρονικού συστήματος και απαιτείται ένα στάδιο προετοιμασίας. Σε αυτό το στάδιο συνήθως εφαρμόζεται μια βαθμίδα ενίσχυσης του λαμβανόμενου αναλογικού σήματος, έπειτα στο σήμα εφαρμόζονται κατάλληλες βαθμίδες φίλτρων για την ανάδειξη συγκεκριμένων χαρακτηριστικών, υλοποιείται κβάντιση με βάση τον ρυθμό της δειγματοληψίας και στο τελικό στάδιο υλοποιείται η απομόνωση των ωφέλιμων πληροφοριών καθώς και η βαθμονόμηση του ψηφιακού σήματος. Συγκεκριμένα, τα σήματα χρειάζονται ενίσχυση για να αλλάξει το εύρος πλάτους τους και να ταιριάζει με το εύρος σήματος του επόμενου σταδίου που είναι η μετατροπή της μορφής τους από αναλογική σε ψηφιακή. Επιπλέον τα εφαρμοζόμενα φίλτρα υλοποιούν την αφαίρεση του ανεπιθύμητου θορύβου σε συγκεκριμένο εύρος συχνοτήτων. Η κβάντιση με τον κατάλληλο ρυθμό δειγματοληψίας συντελεί στην υλοποίηση του

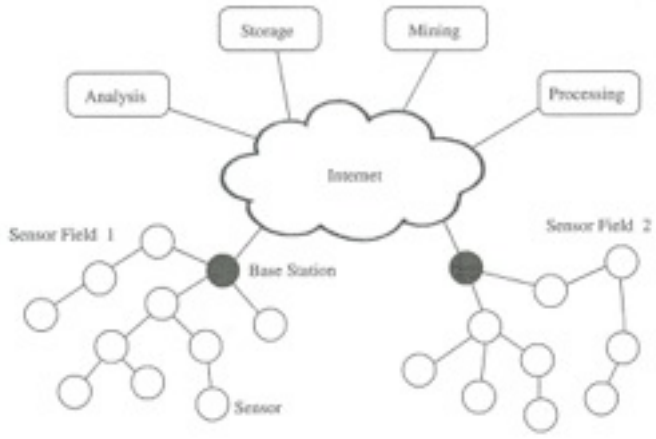
μετατροπέα από αναλογική σε ψηφιακή μορφή (ADC). Τέλος τα στάδια απομόνωσης και βαθμονόμησης ολοκληρώνουν την προετοιμασία του σήματος το οποίο βρίσκεται πλέον σε ψηφιακή μορφή και μπορεί να εφαρμοστεί σε αυτό επιπλέον επεξεργασία, αποθήκευση ή οπτικοποίηση.

Συνήθως τα ασύρματα δίκτυα αισθητήρων περιλαμβάνουν ενεργοποιητές που τους επιτρέπουν να ελέγχουν απευθείας παραμέτρους του φυσικού κόσμου. Για παράδειγμα ένας ενεργοποιητής μπορεί να είναι μια βαλβίδα που να ελέγχει την ροή ζεστού νερού, μια μηχανή που να ανοίγει ή να κλείνει μια πόρτα ή ένα παράθυρο, ή μια αντλία που να ελέγχει τα ποσοστά του καυσίμου σε μια μηχανή. Ένα τέτοιο δίκτυο ασύρματων αισθητήρων και ενεργοποιητών (WSAN) λαμβάνει εντολές από μια μονάδα ελέγχου και μετατρέπει τις εντολές αυτές σε σήματα εισόδου για τους ενεργοποιητές, οι οποίοι επενεργούν με μια φυσική διεργασία όπως φαίνεται στην εικόνα 1.

Προδιαγραφές Αισθητήρων

Η επιλογή του είδους των αισθητήρων καθορίζεται από τις φυσικές παραμέτρους προς μέτρηση/παρακολούθηση, για παράδειγμα τέτοιες παράμετροι είναι η θερμοκρασία, η πίεση, το φώς, η υγρασία. Στον παρακάτω πίνακα φαίνονται κάποιες τυπικές φυσικές παράμετροι και ποιες τεχνολογίες αισθητήρων χρησιμοποιούνται. Εκτός από τις φυσικές παραμέτρους οι προδιαγραφές των αισθητήρων μπορούν να βασιστούν και σε άλλες μεθόδους, για παράδειγμα αν απαιτείται εξωτερική τροφοδοσία. Αν ένας αισθητήρας χρειάζεται εξωτερική τροφοδοσία ονομάζεται ενεργός αισθητήρας. Αυτό συμβαίνει όταν πρέπει να εκπέμψει κάποιου είδους ενέργεια (π.χ. μικροκύματα, φώς, ήχο) για να ενεργοποιήσει μια αντίδραση ή να ανιχνεύσει μια αλλαγή στην ενέργεια το εκπεμπόμενου σήματος. Από την άλλη πλευρά υπάρχουν οι παθητικοί αισθητήρες που ανιχνεύουν την ενέργεια στο περιβάλλον και αντλούν την ισχύ τους από αυτή την ενέργεια εισόδου.

Ασύρματα Δίκτυα Αισθητήρων



Εικόνα 2: Παράδειγμα Δικτύου Ασύρματων Αισθητήρων.

Με τον όρο ασύρματα δίκτυα αισθητήρων νοείται μια χωρική αρχιτεκτονική ασύρματων και αυτόνομων κόμβων που επικοινωνούν μεταξύ τους αλλά και με σταθμούς βάσης έτσι ώστε να ανταλλάσουν πληροφορίες και μετρήσεις. Η ασύρματη επικοινωνία μεταξύ των κόμβων και των σταθμών βάσης διευκολύνει σε εφαρμογές που απαιτούν μεγάλο πλήθος κόμβων αλλά επιπλέον προσδίδει και βαθμούς ελευθερίας στην διάταξη του δικτύου λόγω της φορητότητας των κόμβων και της ευελιξίας των ασύρματων προτύπων επικοινωνιών.

Ένας κόμβος αισθητήρα συνήθως αποτελείται από τρία στοιχεία τα οποία μπορεί να είναι είτε σε μεμονωμένες πλακέτες είτε να είναι ενσωματωμένα σε ένα ενιαίο σύστημα. Τα στοιχεία αυτά είναι:

- **Ασύρματα δομοστοιχεία (wireless modules or motes)**: είναι τα βασικά στοιχεία του δικτύου αισθητήρων καθώς κατέχουν τις δυνατότητες επικοινωνίας και την προγραμματιζόμενη μνήμη στην οποία βρίσκεται ο κώδικας της εφαρμογής. Ένα τέτοιο δομοστοιχείο συνήθως αποτελείται από μικροελεγκτή, πομποδέκτη, πηγή ισχύος, μονάδα μνήμης, και μπορεί να περιέχει μερικούς αισθητήρες.
- **Πλακέτα αισθητήρων (sensor board)**: είναι η πλακέτα που συνδέεται στο ασύρματο δομοστοιχείο η οποία έχει ενσωματωμένους αισθητήρες πολλών τύπων. Η πλακέτα αισθητήρων μπορεί να περιέχει και μια περιοχή προτυποποίησης για να συνδέονται και επιπλέον είδη αισθητήρων. Εναλλακτικά οι αισθητήρες μπορεί να είναι ενσωματωμένοι στο ασύρματο δομοστοιχείο.
- **Πλακέτα προγραμματισμού (programming board) ή πλακέτα πύλης σύζευξης (gateway board)**: είναι η πλακέτα που παρέχει πολλαπλές διεπαφές συμπεριλαμβανομένων των Ethernet, Wi-Fi, USB, ή σειριακών θυρών για την σύνδεση διαφορετικών δομοστοιχείων στο δίκτυο μιας επιχείρησης ή σε ένα βιομηχανικό δίκτυο ή τοπικά σε έναν υπολογιστή. Αυτές οι πλακέτες χρησιμοποιούνται είτε για να τον προγραμματισμό του δομοστοιχείου είτε για να την συλλογή δεδομένων από αυτό. Συγκεκριμένα,

ειδικές διατάξεις πρέπει να είναι συνδεδεμένες στην πλακέτα προγραμματισμού για να υπάρχει δυνατότητα φόρτωσης του κώδικα της εφαρμογής στην προγραμματιζόμενη μνήμη του ασύρματου δομοστοιχείου.

Παρακάτω ακολουθεί πίνακας με τα πιο δημοφιλή ασύρματα δομοστοιχεία που έχουν κατασκευαστεί τα τελευταία χρόνια δίνοντας έμφαση στην ταχύτητα της επεξεργαστικής τους μονάδας, το μέγεθος της προγραμματιζόμενης και αποθηκευτικής τους μνήμης, της συχνότητας λειτουργίας και τους ρυθμούς μετάδοσης τους.

Πίνακας 1: Τεχνολογίες Ασύρματων Κόμβων Αισθητήρων με έμφαση στους μικροελεγκτές, τους πομποδέκτες, την χωρητικότητα μνήμης δεδομένων και την προγραμματιστική γλώσσα.

Sensor Node Name	Microcontroller	Tranceiver	Program+Data Memory	External Memory	Programming
Eyes	MSP430F149	TR1001		8 Mbit	PeerOS Support
EyesIFX v1	MSP430F149	TDA5250 (868 MHz) FSK		8 Mbit	TinyOS Support
EyesIFX v2	MSP430F1611	TDA5250 (868 MHz) FSK		8 Mbit	TinyOS Support
IMote	ARM core 12 MHz	Bluetooth with the range of 30 m	64K SRAM	512K Flash	TinyOS Support
IMote 1.0	ARM 7TDMI 12-48 MHz	Bluetooth with the range of 30 m	64K SRAM	512K Flash	TinyOS Support
IMote 2.0	Marvell PXA271 ARM 11-400 MHz	TI CC2420 802.15.4/ZigBee compliant radio	32 MB SRAM	32 MB Flash	Microsoft .NET Micro, Linux, TinyOS Support
Mica	ATmega 103 4 MHz 8-bit CPU	RFM TR1000 radio 50 kbit/s	128+4K RAM	512K Flash	nesC Programming
Mica2	ATMEGA 128L	Chipcon 868/916 MHz	4K RAM	128K Flash	TinyOS, SOS and MantisOS Support
Mica2Dot	ATMEGA 128		4K RAM	128K Flash	
MicaZ	ATMEGA 128	TI CC2420 802.15.4/ZigBee compliant radio	4K RAM	128K Flash	nesC
Telos	MSP430		2K RAM		
TelosB	Texas Instruments MSP430 microcontroller	250 kbit/s 2.4 GHz IEEE 802.15.4 Chipcon Wireless Transceiver	10k RAM	48k Flash	Contiki, TinyOS, SOS and MantisOS Support
T-Mote Sky	Texas Instruments MSP430 microcontroller	250 kbit/s 2.4 GHz IEEE 802.15.4 Chipcon	10k RAM	48k Flash	Contiki, TinyOS, SOS and MantisOS Support
Waspmote	Atmel ATmega 1281	ZigBee/802.15.4/DigiMesh/RF, 2.4 GHz/868/900 MHz	8K SRAM	128K FLASH ROM, 4K EEPROM, 2 GB SD card	C/Processing

Εικόνα 3: Χρονολογική εξέλιξη Ασύρματων Κόμβων Αισθητήρων.

Πλατφόρμες Χαμηλού Επιπέδου

Οι πλατφόρμες χαμηλού επιπέδου χαρακτηρίζονται από τις περιορισμένες τους δυνατότητες σε σχέση με την επεξεργασία, την μνήμη και την επικοινωνία. Σχεδιάστηκαν για να χρησιμοποιηθούν σε μεγάλους αριθμούς σε Ασύρματα Δίκτυα Αισθητήρων για να παρέχουν τις δυνατότητες παρακολούθησης παραμέτρων (sensing) καθώς και την υποδομή για τις επικοινωνίες.

- **Οικογένεια Mica**: Η οικογένεια αυτή περιλαμβάνει τους κόμβους Mica, Mica2, MicaZ και IRIS. Κάθε κόμβος αποτελείται από 8-bit Atmel AVR μικροελεγκτές με ταχύτητα από 4-16MHz και προγραμματιζόμενη μνήμη flash από 128-256MB. Οι κόμβοι της οικογένειας Mica έχουν μεγάλο εύρος από πομποδέκτες. Ο κόμβος Mica έχει έναν πομποδέκτη 916MHz ή 433MHz στα 40kbps, ενώ ο κόμβος Mica2 έχει έναν πομποδέκτη 433/868/916MHz στα 40kbps. Οι κόμβοι MicaZ και IRIS έχουν εξοπλιστεί με πομποδέκτες συμβατούς με το πρότυπο IEEE 802.15.4 οι οποίοι λειτουργούν στα 2.4GHz με 250kbps ρυθμό μετάδοσης δεδομένων. Κάθε ασύρματο δομοστοιχείο έχει περιορισμένη μνήμη τύπου τυχαίας προσπέλασης (RAM) χωρητικότητας από 4-8kB και μνήμη δεδομένων 512kB. Επίσης κάθε έκδοση είναι εξοπλισμένη με σύνδεσμο 51 ακροδεκτών που χρησιμοποιείται για να συνδέει επιπλέον πλακέτα αισθητήρων και πλακέτα προγραμματισμού στο ασύρματο δομοστοιχείο.
- **Οικογένεια Telos/Tmote**: Η οικογένεια αυτή έχει παρεμφερή αρχιτεκτονική με τον κόμβο MicaZ της οικογένειας Mica. Ο πομποδέκτης είναι ακριβώς ο ίδιος με αυτόν του κόμβου MicaZ αλλά οι κόμβοι Telos/Tmote έχουν μεγαλύτερη μνήμη τυχαίας προσπέλασης (RAM) επειδή χρησιμοποιείται ένας 8MHz TI MSP430 μικροελεγκτής με 10kB RAM. Επιπλέον οι κόμβοι Telos/Tmote έχουν ενσωματωμένους αρκετούς αισθητήρες όπως αισθητήρα φωτός, αισθητήρα υπερύθρων, αισθητήρα υγρασίας και θερμοκρασίας καθώς επίσης διαθέτουν και θύρα USB που καταργεί την ανάγκη για επιπλέον πλακέτα αισθητήρων και πλακέτα προγραμματισμού. Τέλος διαθέτουν και συνδέσμους 6 και 10 ακροδεκτών για επιπλέον αισθητήρες [1].

- **EYE**: Η πλατφόρμα EYE σχεδιάστηκε σαν απόρροια μιας 3ετούς Ευρωπαϊκής έρευνας και είναι παρεμφερής με την αρχιτεκτονική των κόμβων της οικογένειας Telos/Tmote. Στους κόμβους EYE χρησιμοποιείται ένας 16-bit μικροελεγκτής με 60kB προγραμματιζόμενη μνήμη και 2kB μνήμη δεδομένων. Επιπλέον στους κόμβους EYE είναι ενσωματωμένοι οι ακόλουθοι αισθητήρες: πυξίδα, αισθητήρας επιτάχυνσης, αισθητήρας θερμοκρασίας, αισθητήρας φωτός και αισθητήρας πίεσης. Οι κόμβοι EYE διαθέτουν τον πομποδέκτη TR1001 που υποστηρίζει ρυθμούς μετάδοσης μέχρι 115,2kbps με ενεργειακή κατανάλωση της τάξης των 14.4mW, 16.0mW και 15.0μW κατά την διάρκεια λήψης, εκπομπής και αδράνειας αντίστοιχα. Οι κόμβοι τέλος διαθέτουν και μια RS232 σειριακή διεπαφή για προγραμματισμό [2].

Ασύρματα Δομοστοιχεία Υψηλού Επιπέδου

Εκτός των αισθητήρων, της τοπικής επεξεργασίας και της επικοινωνίας πολλαπλών βημάτων (multi-hop) τα Ασύρματα Δίκτυα Αισθητήρων χρειάζονται επιπλέον λειτουργικότητες οι οποίες δεν μπορούν να μεταφερθούν αποδοτικά από τα ασύρματα δομοστοιχεία χαμηλού επιπέδου. Λειτουργίες υψηλού επιπέδου όπως η διαχείριση του δικτύου χρειάζονται μεγαλύτερη επεξεργαστική ισχύ και μνήμη σε σχέση με τις δυνατότητες που μπορούν να προσφέρουν τα ασύρματα δομοστοιχεία χαμηλού επιπέδου. Επιπλέον η ολοκλήρωση των Ασύρματων Δικτύων Αισθητήρων με τις ήδη υπάρχουσες υποδομές δικτύωσης χρειάζεται πολλαπλές τεχνικές επικοινωνίας να είναι ενσωματωμένες διαμέσου ενός δομοστοιχείου-δρομολογητή (gateway module). Επιπροσθέτως, στα δίκτυα όπου πλήμνες (hubs) επεξεργασίας ή αποθήκευσης είναι ενσωματωμένες με κόμβους αισθητήρων χρειάζονται κόμβοι με μεγαλύτερη χωρητικότητα. Για τους παραπάνω λόγους σχεδιάστηκαν τα ασύρματα δομοστοιχεία υψηλού επιπέδου.

- Stargate: Η πλακέτα Stargate είναι μια πλατφόρμα υψηλών επιδόσεων που σχεδιάστηκε για να υλοποιεί αίσθηση (sensing), επεξεργασία σήματος, έλεγχο και διαχείριση δικτύωσης αισθητήρων. Η πλατφόρμα Stargate είναι βασισμένη στον Intel PXA-255 Xscale 400MHz RISC επεξεργαστή. Η Stargate διαθέτει flash μνήμη 32MB, 64MB SDRAM μνήμη τυχαίας προσπέλασης και ενσωματωμένους συνδέσμους για τα ασύρματα δομοστοιχεία της οικογένειας Mica καθώς επίσης και κάρτες διασύνδεσης PCMCIA Bluetooth επικοινωνίας ή ασύρματης επικοινωνίας συμβατής με τα πρότυπα IEEE 802.11. Ως εκ τούτου μπορεί να λειτουργήσει σαν ασύρματος δρομολογητής και υπολογιστική πλήμνη (hub) για αλγορίθμους επεξεργασίας μέσα στο δίκτυο. Μπορεί να συνδεθεί με κάμερα ή άλλες οπτικοακουστικές συσκευές και να λειτουργήσει σαν αισθητήρας πολυμέσων μεσαίας ανάλυσης αν και έχει μεγάλη ενεργειακή κατανάλωση.
- Imote: Η Intel ανέπτυξε δύο πρότυπες γενιές ασύρματων αισθητήρων για υψηλών επιδόσεων εφαρμογές. Η πλατφόρμα Imote κατασκευάστηκε με βάση έναν ολοκληρωμένο ασύρματο μικροελεγκτή που περιελάμβανε έναν 8-bit 12MHz ARM7 επεξεργαστή, για ασύρματη επικοινωνία μια πλακέτα Bluetooth, 64kB RAM μνήμη τυχαίας προσπέλασης και 32kB μνήμη flash

καθώς επίσης και πολλές επιλογές εισόδων/εξόδων. Η αρχιτεκτονική του λειτουργικού ήταν βασισμένη σε μια έκδοση του λειτουργικού συστήματος TinyOS για τον επεξεργαστή ARM.

- Η δεύτερη γενιά του Imote, το Imote2 κατασκευάστηκε με βάση ένα νέο χαμηλής κατανάλωσης επεξεργαστή τον 32-bit PXA271 XScale χρονισμένο στα 320/416/520 MHz ο οποίος χρησιμοποιούσε λειτουργίες ψηφιακής επεξεργασίας σήματος (DSP) για αποθήκευση ή συμπίεση, και για ασύρματη επικοινωνία χρησιμοποιήθηκε μια πλακέτα ChipCon CC2420 συμβατή με το πρότυπο IEEE 802.15.4. Είχε μεγάλη ενσωματωμένη μνήμη τυχαίας προσπέλασης (RAM) και μνήμη flash, υποστήριξη για επιπλέον πομποδέκτες και μια ποικιλία από υψηλής ταχύτητας συνδέσμους εισόδου/εξόδου για να συνδέονται αισθητήρες ή κάμερες. Τέλος μπορούσε να χρησιμοποιεί σαν λειτουργικό σύστημα το Linux και να εκτελεί εφαρμογές υλοποιημένες στην γλώσσα προγραμματισμού Java.

Τυποποίηση

Η μεγάλη ετερογένεια στις πλατφόρμες αισθητήρων είχε σαν αποτέλεσμα την δημιουργία προβλημάτων συμβατότητας στις εφαρμογές, ως εκ τούτου άρχισαν να γίνονται προσπάθειες τυποποίησης συγκεκριμένα για τα χαρακτηριστικά της επικοινωνίας. Δημιουργήθηκε το πρότυπο IEEE 802.15.4 που περιγράφει τις προδιαγραφές για τους ασύρματους πομποδέκτες χαμηλού ρυθμού δεδομένων με σκοπό την μεγάλη διάρκεια ζωής της μπαταρίας και την χαμηλή πολυπλοκότητα. Τρία φάσματα συχνοτήτων επιλέχτηκαν για την επικοινωνία, 2,4GHz (γενικό), 915MHz (για την Αμερική), 868MHz (για την Ευρώπη). Το φυσικό στρώμα χρησιμοποιεί BPSK για τις συχνότητες 868/915MHz και O-QPSK για την συχνότητα 2,4GHz, το στρώμα ζεύξης δεδομένων παρέχει επικοινωνία για διάφορες τοπολογίες όπως αστέρα και δέντρου με χρήση ελεγκτών. Η εμβέλεια εκπομπής των κόμβων είναι 10-100 μέτρα με ρυθμούς μετάδοσης από 20-250kbps.

Πάνω από την τυποποίηση IEEE 802.15.4 βρίσκονται διάφορα άλλα “σώματα” τυποποιήσεων που εμπλουτίζουν τις δυνατότητες των δικτύων χαμηλής κατανάλωσης σε διάφορους τομείς. Είναι γενικά αποδεκτό ότι πρότυπα σαν το Bluetooth και το WiFi δεν ταιριάζουν καλά με εφαρμογές αισθητήρων χαμηλής κατανάλωσης γι αυτό προτιμούνται πρότυπα όπως τα ZigBee, WirelessHART, WINA, SP100.11a, που ικανοποιούν ακριβώς ότι χρειάζονται τα δίκτυα χαμηλής κατανάλωσης [3].

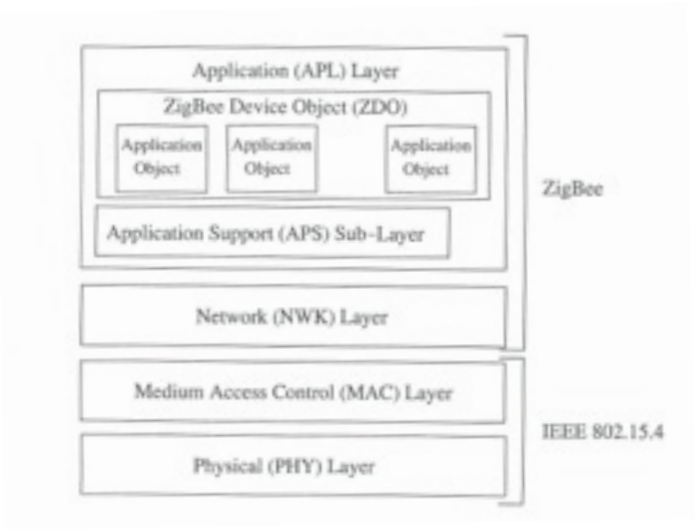
ZigBee

Το πρότυπο ZigBee σχεδιάστηκε για να ικανοποιήσει τις ανάγκες για αποδοτικότητα στις ασύρματες επικοινωνίες που θα υποστηρίζουν χαμηλούς ρυθμούς μετάδοσης δεδομένων, χαμηλή κατανάλωση ενέργειας, ασφάλεια και αξιοπιστία. Δόθηκε έμφαση σε πέντε βασικές περιοχές εφαρμογών: αυτοματισμοί

σπιτιών, έξυπνη ενέργεια, αυτοματισμοί κτηρίων, υπηρεσίες τηλεπικοινωνιών, και προσωπική φροντίδα υγείας.

Το πρότυπο ZigBee σχεδιάστηκε συγκεκριμένα για να συνδυάζεται με το πρότυπο IEEE802.15.4. Στην εικόνα 1,1 που ακολουθεί φαίνεται η κατηγοριοποίηση των επιπέδων αναλόγως του προτύπου που ανήκουν. Το φυσικό επίπεδο και το επίπεδο ζεύξης δεδομένων ορίζονται από το πρότυπο IEEE 802.15.4, καθώς το πρότυπο ZigBee ορίζει το επίπεδο δικτύου και το πλαίσιο της εφαρμογής. Τα αντικείμενα της εφαρμογής ορίζονται από τον χρήστη. Για να υποστηρίζεται μεγάλο εύρος εφαρμογών τρείς τύποι κίνησης έχουν οριστεί. Αρχικά είναι η περιοδική κίνηση δεδομένων για να παρακολουθεί τις εφαρμογές όπου αισθητήρες παρέχουν συνεχόμενες πληροφορίες σχετικά με διάφορα φυσικά φαινόμενα. Η ανταλλαγή δεδομένων ελέγχεται από ελεγκτές δικτύων ή με χρήση δρομολογητών. Έπειτα είναι η διαλειπτόμενη κίνηση δεδομένων που εφαρμόζεται στις εφαρμογές που βασίζονται σε ανίχνευση γεγονότων και ενεργοποιείται είτε από την εφαρμογή είτε από κάποιον εξωτερικό παράγοντα. Αυτού του τύπου η κίνηση διεκπεραιώνεται από κάθε κόμβο δρομολόγησης. Για εξοικονόμηση ενέργειας η συσκευή μπορεί να λειτουργεί σε κατάσταση αποσύνδεσης από το υποσύστημα ραδιομετάδοσης για εξοικονόμηση ενέργειας, ενώ λειτουργεί σε κατάσταση ύπνου τις περισσότερες φορές. Όποτε οι πληροφορίες χρειάζεται να αποσταλούν, ο πομποδέκτης ενεργοποιείται και η συσκευή γνωστοποιεί την παρουσία της στο δίκτυο. Τέλος η επαναλαμβανόμενη χαμηλής καθυστέρησης δεδομένων κίνηση ορίζεται για συγκεκριμένες επικοινωνίες όπως ένα κλικ του ποντικιού που χρειάζεται να ολοκληρωθεί σε συγκεκριμένο χρόνο.

Το επίπεδο δικτύου ZigBee (NWK) προσφέρει λειτουργίες διαχείρισης δικτύου για τον χειρισμό του δικτύου. Καθορίζονται οι διαδικασίες για την δημιουργία ενός καινούργιου δικτύου και αν οι συσκευές θα αποκτήσουν δικαιώματα ή θα παραιτηθούν από το δίκτυο. Επιπλέον ανάλογα με τις λειτουργίες του δικτύου, η στοίβα επικοινωνίας της κάθε συσκευής μπορεί να παραμετροποιηθεί. Επειδή μια συσκευή ZigBee μπορεί να είναι μέρος από διαφορετικά δίκτυα κατά τον κύκλο ζωή της, η προτυποποίηση ορίζει έναν ευέλικτο μηχανισμό διευθυνσιοδότησης. Επομένως, ο συντονιστής δικτύου αναθέτει μια διεύθυνση στην συσκευή καθώς αυτή γίνεται μέλος του δικτύου. Σαν αποτέλεσμα, η μοναδική ταυτότητα της κάθε συσκευής δεν χρησιμοποιείται για επικοινωνία αλλά μια συντομότερη διεύθυνση ανατίθεται για να βελτιώσει την αποδοτικότητα κατά την διάρκεια της επικοινωνίας. Σε μια αρχιτεκτονική τύπου δένδρου, η διεύθυνση μιας συσκευής επιπλέον γνωστοποιεί και τον γονέα της, το οποίο είναι χρήσιμο για λόγους δρομολόγησης. Το επίπεδο δικτύου ZigBee προσφέρει επιπλέον συγχρονισμό μεταξύ συσκευών και ελεγκτών δικτύου. Τέλος οι διαδρομές εκπομπής πολλαπλών βημάτων δημιουργούνται από το επίπεδο δικτύου ZigBee σε συμφωνία με τα ορισμένα πρωτόκολλα.



Εικόνα 4: Στοίβα πρωτοκόλλων ZigBee και 802.15.4

Όπως παρουσιάζεται στην εικόνα 4 το πρότυπο ZigBee ορίζει επίσης συγκεκριμένα συστατικά στο επίπεδο εφαρμογής. Το επίπεδο αυτό αποτελείται από το APS υποεπίπεδο, το αντικείμενο συσκευή ZigBee, και τα αντικείμενα εφαρμογής που ορίζει ο κατασκευαστής. Οι εφαρμογές υλοποιούνται με χρήση των ορισμένων από τον κατασκευαστή αντικειμένων και η υλοποίηση είναι βασισμένη στις απαιτήσεις που ορίζει το πρότυπο. Το αντικείμενο συσκευή ZigBee ορίζει τις λειτουργίες που παρέχονται από την συσκευή για τις λειτουργίες του δικτύου. Πιο συγκεκριμένα, ο ρόλος συσκευών όπως ο συντονιστής δικτύου ή ο δρομολογητής ορίζονται στο αντικείμενο συσκευή ZigBee. Επιπλέον, όποτε μια συσκευή χρειάζεται να συνεργαστεί με το δίκτυο, τα αιτήματα σύνδεσης αντιμετωπίζονται από το αντικείμενο συσκευή ZigBee. Τέλος το υποεπίπεδο APS προσφέρει δυνατότητες ανακάλυψης στις συσκευές έτσι ώστε οι γείτονες μιας συσκευής και οι λειτουργικότητες που αυτοί παρέχουν να μπορούν αποθηκευτούν. Αυτές οι πληροφορίες χρησιμοποιούνται επίσης για να ταιριάζουν τα αιτήματα συνεργασίας των γειτόνων με συγκεκριμένες λειτουργίες [4].

Λειτουργικό Σύστημα

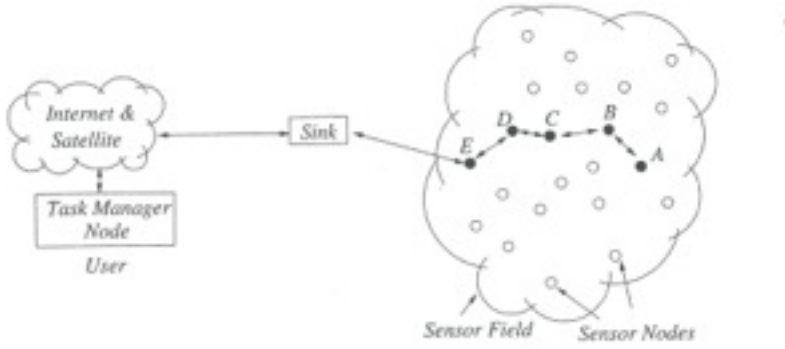
Επιπλέον των πλατφόρμων υλικού και των προτυποποιήσεων, αρκετές πλατφόρμες λειτουργικού έχουν αναπτυχθεί συγκεκριμένα για τα ασύρματα δίκτυα αισθητήρων. Ανάμεσα σε αυτές η πιο διαδεδομένη είναι το TinyOS, το οποίο είναι ένα ανοικτού κώδικα λειτουργικό σύστημα σχεδιασμένο για ασύρματα δίκτυα ενσωματωμένων αισθητήρων.

Το TinyOS ενσωματώνει μια αρχιτεκτονική που στηρίζεται στα “στοιχεία” (components), η οποία ελαχιστοποιεί το μέγεθος του κώδικα και παρέχει μια ευέλικτη πλατφόρμα για την υλοποίηση νέων πρωτοκόλλων επικοινωνιών. Η βιβλιοθήκη του που βασίζεται στα “στοιχεία” περιέχει πρωτόκολλα δικτύων, παρεχόμενες υπηρεσίες, οδηγούς αισθητήρων, και εργαλεία απόκτησης δεδομένων οι οποίες μπορούν να τροποποιηθούν, και να βελτιστοποιηθούν με βάση τις απαιτήσεις των εκάστοτε εφαρμογών. Το TinyOS είναι βασισμένο σε μοντέλο

εκτέλεσης οδηγούμενης από γεγονός (event driven) η οποία είναι μια καλή προσέγγιση από άποψη διαχείρισης ενέργειας [5].

Το Contiki είναι ένα λειτουργικό σύστημα ανοικτού κώδικα εκτέλεσης πολλαπλών εργασιών (multitasking) που αναπτύχθηκε για μια μεγάλη ποικιλία πλατφορμών που περιλαμβάνουν μικροελεγκτές όπως ο TI MSP430 και ο Atmel AVR, οι οποίοι χρησιμοποιούνται στις οικογένειες Telos, Tmote, και Mica. Το Contiki είναι κατασκευασμένο με βάση έναν πυρήνα (kernel) οδηγούμενο από γεγονός. Σαν αποτέλεσμα, σε σύγκριση με το TinyOS το οποίο είναι στατικά συνδεδεμένο με τον χρόνο μεταγλώττισης, το Contiki επιτρέπει στα προγράμματα και τους οδηγούς να μπορούν να αντικατασταθούν σε χρόνο εκτέλεσης χωρίς να κάνουν επανασύνδεση. Επιπλέον η υποστήριξη TCP/IP παρέχεται από την στοίβα μIP [6].

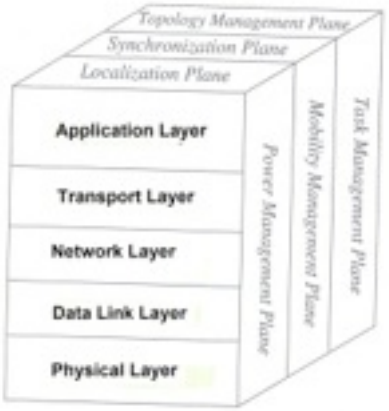
Αρχιτεκτονική Ασύρματων Δικτύων Αισθητήρων και Στοίβα Πρωτοκόλλου.



Εικόνα 5: Λειτουργία Δικτύου Ασύρματων Αισθητήρων.

Οι κόμβοι αισθητήρες είναι συνήθως διασκορπισμένοι σε ένα πεδίο αίσθησης όπως στην εικόνα 5. Καθένας από αυτούς τους διασκορπισμένους κόμβους αισθητήρες έχει την δυνατότητα να συλλέγει δεδομένα και να δρομολογεί τα δεδομένα πίσω στην πύλη και τον τελικό χρήστη. Τα δεδομένα δρομολογούνται πίσω στον τελικό χρήστη με εκπομπή πολλαπλών βημάτων αδόμητης αρχιτεκτονικής διαμέσου της πύλης όπως φαίνεται στην εικόνα 5. Η πύλη μπορεί να επικοινωνεί με τον διαχειριστή διεργασιών/τελικό χρήστη μέσω του Internet ή δορυφόρου ή οποιασδήποτε άλλης μορφής ασύρματο δίκτυο, ή χωρίς κανένα δίκτυο όπου η πύλη μπορεί να είναι συνδεδεμένη απευθείας με τον τελικό χρήστη.
Στα Ασύρματα Δίκτυα Αισθητήρων, οι κόμβοι αισθητήρων έχουν διπλή λειτουργικότητα συγκεκριμένα είναι ικανοί να δημιουργούν δεδομένα και να δρομολογούν δεδομένα. Ως εκ τούτου η επικοινωνία γίνεται για δύο λόγους:

- **Λειτουργία Πηγής**: Κόμβοι Πηγής με πληροφορίες γεγονότος εκτελούν λειτουργίες επικοινωνίας για να μεταδώσουν τα πακέτα τους στην πύλη.
- **Λειτουργία Δρομολόγησης**: Κόμβοι αισθητήρες συμμετέχουν στην προώθηση των πακέτων που λαμβάνουν από άλλους κόμβους στον επόμενο προορισμό.



Εικόνα 6: Στοίβα Πρωτόκολλου Ασύρματων Δικτύων Αισθητήρων.

Η στοίβα πρωτοκόλλου που χρησιμοποιείται από την πύλη και όλους τους κόμβους αισθητήρες φαίνεται στην εικόνα 6. Αυτή η στοίβα πρωτοκόλλου συνδυάζει ενημέρωση για θέματα ισχύος και δρομολόγησης, ενσωματώνει δεδομένα με πρωτόκολλα δικτύωσης, αναμεταδίδει πληροφορίες σχετικά με την αποδοτικότητα ισχύος μέσω του ασύρματου μέσου και προωθεί την συνεργατική προσπάθεια ανάμεσα στους ασύρματους κόμβους αισθητήρων. Η στοίβα πρωτοκόλλου αποτελείται από το φυσικό στρώμα, το στρώμα ζεύξης δεδομένων, το στρώμα δικτύου, το στρώμα μεταφοράς, το στρώμα εφαρμογής, καθώς επίσης και από ένα επίπεδο συγχρονισμού, επίπεδο εντοπισμού, επίπεδο διαχείρισης τοπολογίας, επίπεδο διαχείρισης ενέργειας, επίπεδο διαχείρισης κινητικότητας, και ένα επίπεδο διαχείρισης διεργασιών.

Το φυσικό στρώμα διευθετεί τις ανάγκες απλής αλλά ισχυρής διαμόρφωσης, τις τεχνικές εκπομπής και τεχνικές λήψης. Καθώς το περιβάλλον είναι θορυβώδες και οι κόμβοι αισθητήρων είναι φορητοί, το στρώμα ζεύξης δεδομένων είναι υπεύθυνο στο να διασφαλίσει μια αξιόπιστη επικοινωνία με χρήση τεχνικών ελέγχου σφαλμάτων και να διαχειρίζεται την πρόσβαση στο κανάλι για να ελαχιστοποιεί τις συγκρούσεις λόγω γειτονικών εκπομπών. Το στρώμα δικτύου φροντίζει την δρομολόγηση των δεδομένων που παρέρχονται από το στρώμα μεταφοράς. Το στρώμα μεταφοράς βοηθά στην συντήρηση της ροής δεδομένων αν η εφαρμογή του δικτύου αισθητήρων το χρειάζεται. Επιπρόσθετα τα επίπεδα ισχύος, κινητικότητας, και διαχείρισης διεργασιών παρακολουθούν την ισχύ, την κινητικότητα και την διανομή διεργασιών μεταξύ των κόμβων αισθητήρων. Αυτά τα επίπεδα συντονίζουν τις διεργασίες αίσθησης και χαμηλώνουν την κατανάλωση ισχύος.

Το επίπεδο διαχείρισης ενέργειας διαχειρίζεται με ποιο τρόπο ένας κόμβος αισθητήρα χρησιμοποιεί την ενέργεια του. Για παράδειγμα ο κόμβος αισθητήρα μπορεί να κλείσει τον δέκτη του όταν λάβει ένα μήνυμα από έναν γείτονα του. Αυτό γίνεται για την αποφυγή διπλών μηνυμάτων. Επίσης όταν τα επίπεδα ενέργειας ενός κόμβου είναι χαμηλά αυτός εκπέμπει στους γείτονες του ότι δεν μπορεί να συμμετάσχει στην διαδικασία δρομολόγησης. Η υπολειπόμενη ενέργεια που έχει είναι για να υλοποιεί διεργασίες αίσθησης.

Το επίπεδο διαχείρισης κινητικότητας ανιχνεύει και καταχωρεί την κίνηση των κόμβων αισθητήρα, έτσι ώστε μια δρομολόγηση προς τα πίσω στον χρήστη να είναι πάντα εφικτή και επίσης για να μπορούν οι κόμβοι να είναι ενημερωμένοι πάντα

για τους γείτονες τους. Γνωρίζοντας τους γείτονες τους οι κόμβοι αισθητήρων μπορούν να ισορροπούν την χρήση της ισχύς τους και των διεργασιών τους.

Το επίπεδο διαχείρισης διεργασιών μπορεί να ισορροπεί και να οργανώνει τις διεργασίες αίσθησης για μια συγκεκριμένη περιοχή. Δεν χρειάζεται όλοι οι αισθητήρες σε μια περιοχή να εκτελούν διεργασίες αίσθησης την ίδια στιγμή. Σαν αποτέλεσμα κάποιοι κόμβοι αισθητήρες εκτελούν την διεργασία αίσθησης περισσότερες φορές ανάλογα με τα επίπεδα ενέργειας τους.

Αυτά τα επίπεδα διαχείρισης χρειάζονται για να δουλεύουν συνεργατικά οι κόμβοι αισθητήρων με ενεργειακά αποδοτικό τρόπο, να δρομολογούν δεδομένα σε κινητά δίκτυα αισθητήρων και να μοιράζονται πόρους μεταξύ τους. Χωρίς αυτά τα επίπεδα ο κάθε κόμβος θα δούλευε απομονωμένος. Σαν αποτέλεσμα των ανωτέρω είναι αποδοτικότερο οι κόμβοι αισθητήρων να συνεργάζονται για να επιτυγχάνεται η βέλτιστη διάρκεια ζωής ενός δικτύου αισθητήρων.

Εφαρμογές Ασύρματων Δικτύων Αισθητήρων

Τα ασύρματα δίκτυα αισθητήρων μπορούν να αποτελούνται από πολλών διαφορετικών ειδών αισθητήρες όπως , μαγνητικός, θερμικός, οπτικός, υπέρυθρος, ακουστικός, οι οποίοι μπορούν να παρακολουθήσουν ένα μεγάλο εύρος από περιβαλλοντικές συνθήκες όπως: θερμοκρασία, υγρασία, πίεση, ταχύτητα, κατεύθυνση, κίνηση, φως, επίπεδα θορύβου, την παρουσία ή την απουσία συγκεκριμένων αντικειμένων, την μηχανική αντοχή αντικειμένων. Σαν αποτέλεσμα είναι πολύ μεγάλο το εύρος των εφαρμογών που μπορούν να επιτευχθούν. Αυτό το φάσμα των εφαρμογών περιλαμβάνει εφαρμογές Εθνικής Ασφάλειας, καιρικές και κλιματολογικές αναλύσεις και προβλέψεις, παρακολούθηση σεισμικής επιτάχυνσης και δραστηριότητας, παρακολούθηση του περιβάλλοντος κ.α. [7].

Εφαρμογές για έξυπνο σπίτι

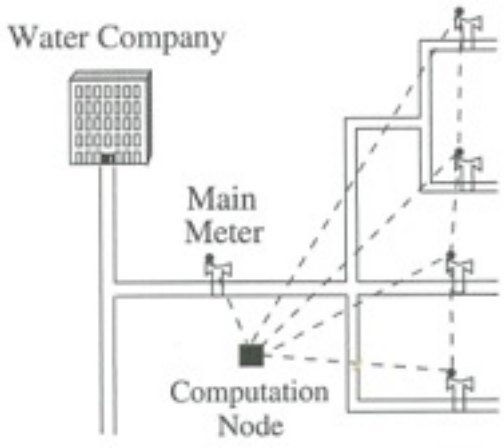
Καθώς η τεχνολογία εξελίσσεται, έξυπνοι κόμβοι αισθητήρων και ενεργοποιητές (actuators) μπορούν να ενσωματωθούν σε συσκευές και εφαρμογές όπως, ηλεκτρικές σκούπες, φούρνους μικροκυμάτων, ψυγεία και καταψύκτες, συσκευές αναπαραγωγής βίντεο καθώς και σε συστήματα παρακολούθησης νερού. Αυτοί οι κόμβοι αισθητήρων μέσα σε συσκευές μπορούν να αλληλεπιδρούν μεταξύ τους και με ένα εξωτερικό δίκτυο μέσω του διαδικτύου ή δορυφορικών επικοινωνιών. Επιτρέπουν στον τελικό χρήστη να διαχειρίζεται εύκολα οικιακές συσκευές είτε τοπικά είτε απομακρυσμένα.

Παρακολούθηση νερού

Ο κύριος στόχος του συστήματος παρακολούθησης νερού για οικιακή χρήση είναι να εντοπίζει την σπατάλη στην χρήση του νερού και να ενημερώνει τους ενδιαφερομένους για μια πιο αποδοτική χρήση. Επειδή οι εταιρίες ύδρευσης παρέχουν μόνο την συνολική χρήση του νερού για ένα σπίτι δεν είναι εύκολο να

καθορίσει κάποιος αναλυτικά τις ποσότητες που συνεισφέρουν οι διαφορετικές πηγές στην συνολική κατανάλωση. Χρησιμοποιώντας όμως ένα ασύρματο δίκτυο αισθητήρων, τοποθετώντας σε κάθε σωλήνα ξεχωριστά έναν ασύρματο κόμβο αίσθησης μπορεί να γίνει παρακολούθηση του συστήματος ύδρευσης του σπιτιού με χαμηλό κόστος.

Η αρχή λειτουργίας του συστήματος αυτού είναι βασισμένη στο γεγονός ότι η ροή νερού σε συγκεκριμένο σωλήνα μπορεί να υπολογιστεί μετρώντας τις δονήσεις του σωλήνα λόγω τις αναλογικής σχέσης μεταξύ των δύο. Επομένως ασύρματοι αισθητήρες έχουν συνδεθεί στους σωλήνες νερού για να μετράνε τις δονήσεις με χρήση επιταχυνσιόμετρου. Λόγω της μη γραμμικής σχέσης μεταξύ των δονήσεων και της ροής του νερού ο κάθε κόμβος αισθητήρα θα πρέπει να είναι ρυθμισμένος να ανακαλύπτει τους βέλτιστους συνδυασμούς παραμέτρων που συνδέουν τις πληροφορίες επιτάχυνσης με την ροή του νερού. Αντί της χειροκίνητης ρύθμισης που μπορεί να εκτελεστεί με επιπλέον εγκατάσταση αισθητήρα μέτρησης ροής νερού σε κάθε σωλήνα στο παραπάνω σύστημα αυτές οι ρυθμίσεις υλοποιούνται αυτόματα με την βοήθεια του γενικού μετρητή νερού.



Εικόνα 7: Δίκτυο Ασύρματων Αισθητήρων για την παρακολούθηση νερού.

Η αρχιτεκτονική του συστήματος φαίνεται στη εικόνα 7, και αποτελείται από τρία είδη συστατικών. Ένας ασύρματος κόμβος αισθητήρα είναι προσαρμοσμένος στον κεντρικό μετρητή νερού για να συλλέγει πραγματική ροή νερού και να την στέλνει στο υπόλοιπο δίκτυο. Αφού ο κόμβος αυτός είναι ρυθμισμένος, λόγω του κεντρικού ρολογιού από την εταιρία ύδρευσης λειτουργεί σαν κατώφλι αλήθειας για το υπόλοιπο σύστημα. Το επόμενο είδος συστατικού του συστήματος είναι οι αισθητήρες κραδασμών που είναι ενσωματωμένοι σε κάθε σωλήνα για να παρακολουθούνται οι κραδασμοί. Οι πληροφορίες σχετικά με τους κραδασμούς στέλνονται στο κεντρικό κόμβο επεξεργασίας ο οποίος ρυθμίζει αυτόματα τις παραμέτρους των αισθητήρων και υπολογίζει την ροή νερού από κάθε σωλήνα του σπιτιού ξεχωριστά. Η παραμετροποίηση στηρίζεται σε έναν αλγόριθμο βελτιστοποίησης που βασίζεται στο γεγονός ότι το άθροισμα των ροών νερού σε κάθε μη παραμετροποιημένο σωλήνα θα πρέπει να είναι ίσο με την ένδειξη του κεντρικού ρολογιού ροής νερού. Στη συνέχεια, οι κραδασμοί χρησιμοποιούνται σε ένα μοντέλο ροής για να υπολογιστεί ο ρυθμός ροής σε χωρικά διανεμημένες περιοχές.

Το σύστημα παρέχει πληροφορίες πραγματικού χρόνου σχετικά με την χρήση του νερού σε διαφορετικές τοποθεσίες του συστήματος ύδρευσης του σπιτιού, το οποίο μπορεί να χρησιμοποιηθεί για να βελτιώσει την αποδοτικότητα των σπιτιών στο μέλλον [8].

Περιβαλλοντολογικές Εφαρμογές

Τα ασύρματα δίκτυα αισθητήρων έχουν χρησιμοποιηθεί επίσης σε ακραία περιβάλλοντα όπου η συνεχόμενη παρουσία του ανθρώπου δεν είναι δυνατή. Η παρακολούθηση ηφαιστείων είναι ένα παράδειγμα εφαρμογής σε ακραίες περιβαλλοντικές συνθήκες όπου η παρουσία ανθρώπου δεν είναι δυνατή. Σε αυτή την εφαρμογή το, δίκτυο αισθητήρων τοποθετείται κοντά σε ενεργά ηφαίστεια για συνεχή παρακολούθηση της δραστηριότητας τους.

Δύο μελέτες έχουν αναπτυχθεί σε δύο ηφαίστεια στο Εκουαδόρ κατά την περίοδο 2004-2005 ως απόδειξη της μελέτης για την παρακολούθηση ηφαιστείων με χρήση ασύρματων δικτύων αισθητήρων. Το 2004 ένα μικρό δίκτυο τριών κόμβων αισθητήρων με ενσωματωμένα μικρόφωνα χρησιμοποιήθηκε για να παρακολουθήσει ένα ηφαίστειο που ξεσπούσε στο κεντρικό Εκουαδόρ. Το 2005, 16 κόμβοι Tmote Sky εξοπλισμένοι με σεισμικούς και ακουστικούς αισθητήρες χρησιμοποιήθηκαν για 19 ημέρες για την παρακολούθηση ενός ενεργού ηφαιστείου στο Βόρειο Εκουαδόρ. Οι κόμβοι αισθητήρων ήταν εξοπλισμένοι με εξωτερικές κεραίες μεγάλου κέρδους για να βελτιώσουν την εμβέλεια της επικοινωνίας και τρείς κόμβοι επικοινωνίας μεγάλης απόστασης χρησιμοποιήθηκαν για να εκπέμπουν τα δεδομένα σε έναν κεντρικό ελεγκτή. Ένας φορητός υπολογιστής εξοπλισμένος με κατευθυντική κεραία χρησιμοποιήθηκε ως πύλη για να συλλέγει τις πληροφορίες και να διαχειρίζεται το δίκτυο απομακρυσμένα.

Ο κύριος στόχος της εφαρμογής ήταν να συλλέξει σεισμικές πληροφορίες που βασίζονταν σε σεισμούς που συνέβαιναν κοντά σε ηφαίστεια. Καθώς οι σεισμοί συνήθως κρατούν λιγότερο από 60 δευτερόλεπτα, χρησιμοποιήθηκε μεγάλος ρυθμός δειγματοληψίας στον σεισμικό αισθητήρα (π.χ. 100Hz), όμως αυτό περιόρισε την δυνατότητα τοπικής αποθήκευσης πληροφοριών σε 20 λεπτά ηφαιστειακής δραστηριότητας. Κάθε κόμβος αισθητήρα χρησιμοποιεί την τοπική μνήμη σαν κυκλικό ρυθμιστή που φιλτράρει την συλλεγόμενη πληροφορία με την χρήση ανιχνευτή κατωφλίου βραχυπρόθεσμου μέσου όρου/μακροπρόθεσμου μέσου όρου για να ξεχωρίζει τα γεγονότα που σχετίζονται με την ηφαιστειακή δραστηριότητα. Όταν ένα γεγονός δημιουργείται αυτό το γεγονός αναφέρεται στην πύλη. Αν ένα γεγονός υψηλής εμπιστοσύνης δημιουργηθεί με πολλαπλές αναφορές η πύλη στέλνει πίσω στο δίκτυο μια εντολή συλλογής δεδομένων. Σαν απάντηση σε αυτή την εντολή κάθε κόμβος εκπέμπει τις πληροφορίες που έχει αποθηκευμένες στην μνήμη του. Συνεπώς ένα μεγάλο κομμάτι από δεδομένα εκπέμπεται στην πύλη μόνο όταν ένα γεγονός ενδιαφέροντος ανιχνευθεί από ένα μικρό αριθμό κόμβων. Αυτό μειώνει την σπατάλη εύρους ζώνης και την κατανάλωση ενέργειας σε κάθε κόμβο.

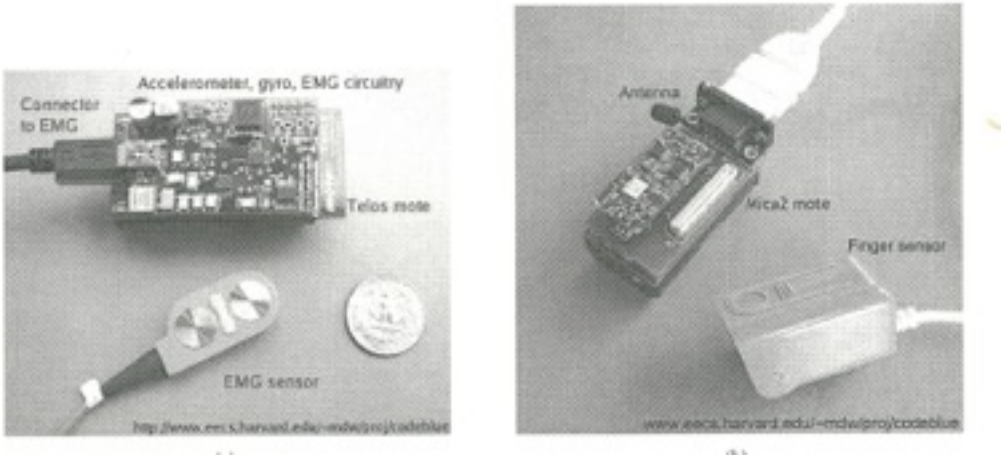
Οι κόμβοι αισθητήρων είναι συγχρονισμένοι χρησιμοποιώντας το πρωτόκολλο συγχρονισμού με πλημμύρα χρόνου (flooding time synchronization) και μια μονάδα GPS με ενσωματωμένο κόμβο MicaZ παρέχει πληροφορίες σχετικά με την τοποθεσία. Τα αποτελέσματα αυτής της εφαρμογής παρέχουν σημαντικές

πληροφορίες που σχετίζονται με φυσικές διεργασίες που συμβαίνουν στο εσωτερικό των ηφαιστείων [9].

Εφαρμογές σε Υπηρεσίες Υγείας

Εφαρμογή Παρακολούθησης Ασθενή

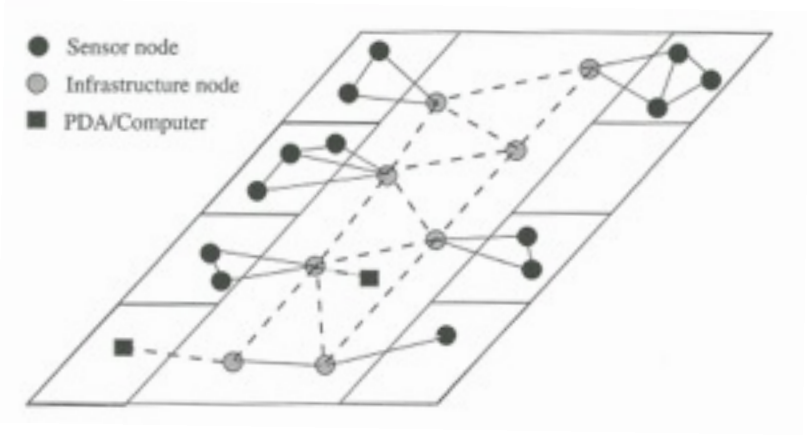
Το σχέδιο CodeBlue του Πανεπιστημίου Χάρβαρντ εστιάζει σε φορετούς αισθητήρες που παρακολουθούν τα ζωτικά σήματα ενός ασθενή στην καθημερινή ζωή του. Για να γίνει εφικτό αυτό πλακέτες αισθητήρων με κυκλώματα παλμικού οξύμετρου, ηλεκτροκαρδιογράφου, και ηλεκτρομυογράφου έχουν σχεδιαστεί για τους κόμβους MicaZ και Telos όπως φαίνονται στην εικόνα 8. Συνεπώς οι σφυγμοί, ο κορεσμός οξυγόνου στο αίμα, οι ηλεκτρικές δραστηριότητες της καρδιάς, η κίνηση του ασθενή και η μυϊκή δραστηριότητα μπορούν να καταγράφονται σε συνεχή βάση. Η πλατφόρμα CodeBlue χρησιμοποιεί τρείς κόμβους που λειτουργούν σε δικτυακή ρύθμιση και έτσι μπορεί το ιατρικό προσωπικό να παρακολουθεί τους ασθενείς με την χρήση PDA.



Εικόνα 8: Υλοποιήσεις της εφαρμογής Code Blue με Telos mote και MicaZ mote.

Η αρχιτεκτονική του δικτύου του Code Blue φαίνεται στην εικόνα 9 όπως θα ήταν οργανωμένο σε ένα νοσοκομείο. Συνεπώς κόμβοι αισθητήρων με αρκετούς αισθητήρες είναι τοποθετημένοι πάνω σε ασθενείς για παρακολούθηση. Επιπρόσθετοι κόμβοι είναι τοποθετημένοι μέσα στο νοσοκομείο για να παρέχουν την υποδομή για την παροχή μηνυμάτων από και προς τους κόμβους. Το ιατρικό προσωπικό μπορεί να αποκτήσει πρόσβαση στο δίκτυο μέσω υπολογιστών και PDA. Το δίκτυο είναι βασισμένο σε έναν μηχανισμό δημοσίευσης/εγγραφής, όπου οι κόμβοι αισθητήρων δημοσιοποιούν τα διαθέσιμα δεδομένα τους και το ιατρικό προσωπικό εγγράφεται σε αυτά τα δεδομένα με βάση συγκεκριμένες συνθήκες. Ο μηχανισμός δημοσίευσης/εγγραφής είναι βασισμένος στο πρωτόκολλο δρομολόγησης προσαρμοστικής ζήτησης πολλαπλής διανομής (Adaptive Demand-Drive Multicast Routing (ADMR)) το οποίο εκτελεί πληροφορίες δρομολόγησης βασισμένο στις δημοσιευμένες πληροφορίες. Κάθε φορά που ένας κόμβος

δημοσιεύει δεδομένα κάθε κόμβος τους δικτύου ενημερώνεται σχετικά με το βέλτιστο μονοπάτι για αυτόν τον αρχικό κόμβο. Η εύρεση του μονοπατιού ανάμεσα σε κόμβους γίνεται περιοδικά με περιορισμένη πλημμύρα για να προσαρμόζεται το ασύρματο κανάλι στις αλλαγές.



Εικόνα 9: Αρχιτεκτονική Δικτύου εφαρμογής Code Blue.

Όταν ένας κόμβος αισθητήρα είναι συνδεδεμένος στο δίκτυο οι δυνατότητες του εκπέμπονται με την χρήση ενός πρωτοκόλλου ανακάλυψης. Η εγγραφή σε δημοσιευμένα δεδομένα γίνεται μέσω του στρώματος ερώτησης CodeBlue (CBQ) που προσδιορίζει την ερώτηση με χρήση μιας πλειάδας {S, τ, chan, ρ, C, p} όπου S είναι η ομάδα εγγεγραμμένων κόμβων, το τ είναι ο τύπος του εγγεγραμμένου αισθητήρα, chan είναι ο τύπος του καναλιού που χρησιμοποιείται για την δρομολόγηση, το ρ είναι ο ρυθμός δειγματοληψίας, και το C είναι ο βέλτιστος μέγιστο αριθμός που πρέπει να αναφερθούν. Για να μοντελοποιηθεί το ενδιαφέρον των χρηστών ένα φίλτρο p χρησιμοποιείται στην ερώτηση όπου φιλτράρει τις πληροφορίες ανάλογα με τους τύπους των αισθητήρων. Σύμφωνα με τα παραπάνω ένας γιατρός μπορεί να καθορίσει ένα συγκεκριμένο αισθητήρα να στέλνει πληροφορίες για έναν συγκεκριμένο ασθενή όταν οι ενδείξεις του υπερβούν τα φυσιολογικά όρια.

Επιπροσθέτως του μηχανισμού δημοσίευσης/εγγραφής το πρόγραμμα CodeBlue παρέχει υπηρεσίες τοποθεσίας με την βοήθεια μιας ομάδας κόμβων που εκπέμπουν περιοδικά πιλοτικά μηνύματα. Ένας κινητός κόμβος μπορεί να υπολογίσει την θέση του με βάση ένα λαμβανόμενο πιλοτικό σήμα. Αυτή η υπηρεσία είναι βασική σε νοσοκομεία για τον εντοπισμό ιατρών, ασθενών, νοσοκόμων και σημαντικών μηχανημάτων. Τέλος παρέχεται και ένα γραφικό περιβάλλον στους χρήστες και στο ιατρικό προσωπικό έτσι ώστε να μπορούν εύκολα να χρησιμοποιούν το σύστημα [10].

Επίλογος- Συμπεράσματα

Στο Κεφάλαιο 1 έγινε μια εκτενής ανάλυσης σε θεωρητικά θέματα που αφορούν τα Δίκτυα Ασύρματων Αισθητήρων. Συγκεκριμένα έγινε αναφορά στην διαδικασία της αίσθησης, στις προδιαγραφές των αισθητήρων ανάλογα με τα μετρούμενα μεγέθη που καλούνται να παρακολουθήσουν, στις προδιαγραφές των

Ασύρματων Δικτύων Αισθητήρων σε επίπεδο υλοποιήσεων, λειτουργικού και επικοινωνιών και τέλος παρατέθηκαν αναφορικά κάποιες εφαρμογές που βασίζονται στα Δίκτυα Ασύρματων Αισθητήρων. Τα συμπεράσματα που προκύπτουν από το Κεφάλαιο 1 υποδεικνύουν την πολυπλοκότητα και την έμφαση στην βελτιστοποίηση παραμέτρων που διέπει τα Δίκτυα Ασύρματων Αισθητήρων, ώστε να λειτουργούν αποδοτικά. Επιπροσθέτως, διαφαίνεται το εύρος των τεχνολογιών που σχετίζονται με την σχεδίαση και την κατασκευή των Δικτύων Ασύρματων Αισθητήρων.

Στο Κεφάλαιο 2 που ακολουθεί πραγματοποιείται ανάλυση χαρακτηριστικών συγκεκριμένων Κόμβων Ασύρματων Αισθητήρων και τεχνολογιών που σχετίζονται με την υλοποίηση της εφαρμογής ευφυούς οικείας που πραγματεύεται η συγκεκριμένη Διπλωματική Εργασία.

ΚΕΦΑΛΑΙΟ 2:Τεχνολογίες Ασύρματων Δικτύων Αισθητήρων

Tmote Sky

Το Tmote Sky είναι μια εξαιρετικά χαμηλής κατανάλωσης ασύρματη πλακέτα για χρήση σε δίκτυα αισθητήρων, σε εφαρμογές παρακολούθησης, και στην ταχεία προτυποποίηση εφαρμογών. Το Tmote Sky αξιοποιεί τα πρότυπα της βιομηχανίας, όπως USB και IEEE 802.15.4 για να λειτουργεί απρόσκοπτα με άλλες συσκευές. Κάνοντας χρήση των προτύπων της βιομηχανίας, των ενσωματωμένων αισθητήρων (υγρασίας, θερμοκρασίας, φωτός), και προσφέροντας ευέλικτα συστήματα διασύνδεσης με περιφερειακά, το Tmote Sky προτιμάται σε ένα μεγάλο εύρος δικτυακών εφαρμογών τύπου πλέγματος (mesh). Το Tmote Sky είναι μια αναπτυγμένη πλακέτα της Moteiv που στοχεύει στην αντικατάσταση της πλακέτας

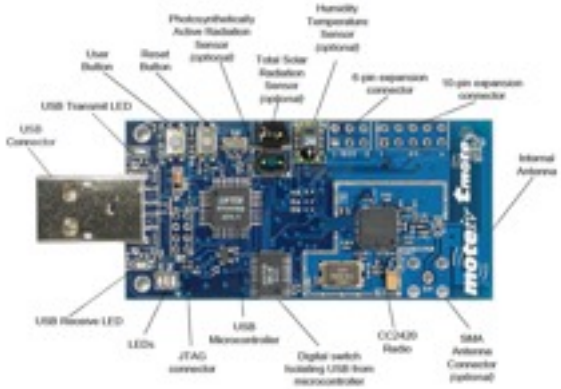
Telos της ίδιας εταιρίας. Το Tmote Sky επιδεικνύει αυξημένη απόδοση, λειτουργικότητα, και επεκτασιμότητα. Το TinyOS υποστηρίζεται εξ αρχής από το Tmote Sky, και έτσι μπορεί να εκμεταλλευτεί εύκολα τα ασύρματα πρωτόκολλα επικοινωνίας και λογισμικού ανοικτού κώδικα. Τέλος το Tmote Sky χρησιμοποιεί ενσωματωμένους αισθητήρες για να αυξήσει την αμεσότητα και να μειώσει το κόστος και το μέγεθος της συσκευασίας του.

Πίνακας 2: Συγκεντρωτικός Πίνακας Χαρακτηριστικών Tmote Sky

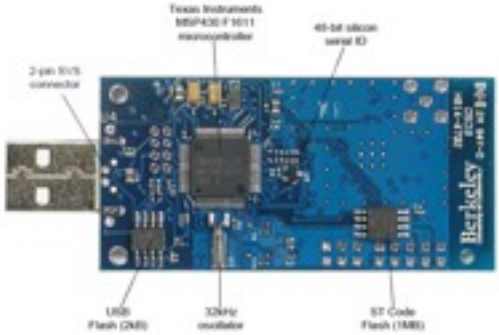
<table>
<tr><th colspan="4">Βασικά Χαρακτηριστικά Tmote Sky</th></tr>
<tr><td>Μικροελγκτής</td><td>Μικροελεγκτής 8MHz Texas Instruments MSP430 (10k RAM, 48k Flash)</td><td>DSP and Modules</td><td>Ενσωματωμένος μετατροπέας από αναλογικό σε ψηφιακό σήμα (ADC), ενσωματωμένος μετατροπέας από ψηφιακό σε αναλογικό σήμα (DAC), ελεγκτή τάσης τροφοδοσίας, και ελεγκτή άμεσης πρόσβασης μνήμης (DMA)</td></tr>
<tr><td rowspan="4">Συνδεσιμότητα</td><td>Ασύρματος πομποδέκτης με ρυθμό μετάδοσης 250kbps στη συχνότητα 2.4GHz IEEE 802.15.4 της Chipcon</td><td>Αισθητήρες</td><td>Ενσωματωμένους αισθητήρες θερμοκρασίας, υγρασίας και φωτός</td></tr>
<tr><td>Διαλειτουργικότητα με άλλες συσκευές IEEE 802.15.4</td><td rowspan="6">Περαιτέρω</td><td>Εξαιρετικά χαμηλή κατανάλωση ρεύματος</td></tr>
<tr><td>Ενσωματωμένη κεραία με ακτίνα κάλυψης 50m για εσωτερικούς χώρους / 125m για εξωτερικούς χώρους</td><td>Γρήγορους χρόνους επαναφοράς από κατάσταση ύπνου (<6μs)</td></tr>
<tr><td>Δυνατότητα επέκταση με σύνδεσμο 16-pin και προαιρετικό σύνδεσμο κεραίας τύπου SMA</td><td>Κωδικοποίηση και πιστοποίηση για το στρώμα ζεύξης δεδομένων</td></tr>
<tr><td colspan="2" rowspan="4"></td><td>Προγραμματισμός και συλλογή δεδομένων μέσω USB</td></tr>
<tr><td>Υποστήριξη TinyOS και δικτύων πλέγματος (mesh)</td></tr>
<tr><td>Δημιουργήθηκε και πιστοποιήθηκε με βάση τους κανονισμούς του οργανισμού FCC και της βιομηχανίας του Καναδά</td></tr>
</table>

Περιγραφή Πλακέτας

Το Tmote Sky είναι μια πλακέτα χαμηλής ισχύος, με ενσωματωμένους αισθητήρες, δυνατότητες διασύνδεσης, κεραία, μικροελεγκτή, και δυνατότητες προγραμματισμού. Οι παρακάτω εικόνες παρουσιάζουν τα δομοστοιχεία από τα οποία αποτελείται το Tmote Sky.



Εικόνα 10: Επάνω όψη της πλακέτας Tmote Sky.



Εικόνα 11: Κάτω όψη της πλακέτας Tmote Sky.

Ισχύς

Το Tmote Sky τροφοδοτείται με δύο μπαταρίες τύπου AA. Η πλακέτα έχει σχεδιαστεί για να χωράει δύο μπαταρίες τύπου AA. Η μπαταρία AA μπορεί να λειτουργεί σε εύρος 2,1 με 3,6 V συνεχούς τάσης, όμως η τάση τροφοδοσίας πρέπει να είναι τουλάχιστον 2,7V κατά την διάρκεια του προγραμματισμού του μικροελεγκτή ή της εξωτερικής μνήμης.

Αν το Tmote Sky είναι συνδεδεμένο σε θύρα USB για προγραμματισμό ή συλλογή δεδομένων, λαμβάνει τροφοδοσία από τον υπολογιστή. Όταν είναι συνδεδεμένο στον υπολογιστή η τάση λειτουργίας είναι 3V. Αν το Tmote Sky είναι μονίμως συνδεδεμένο με υπολογιστή τότε δεν χρειάζονται μπαταρίες για τροφοδοσία.

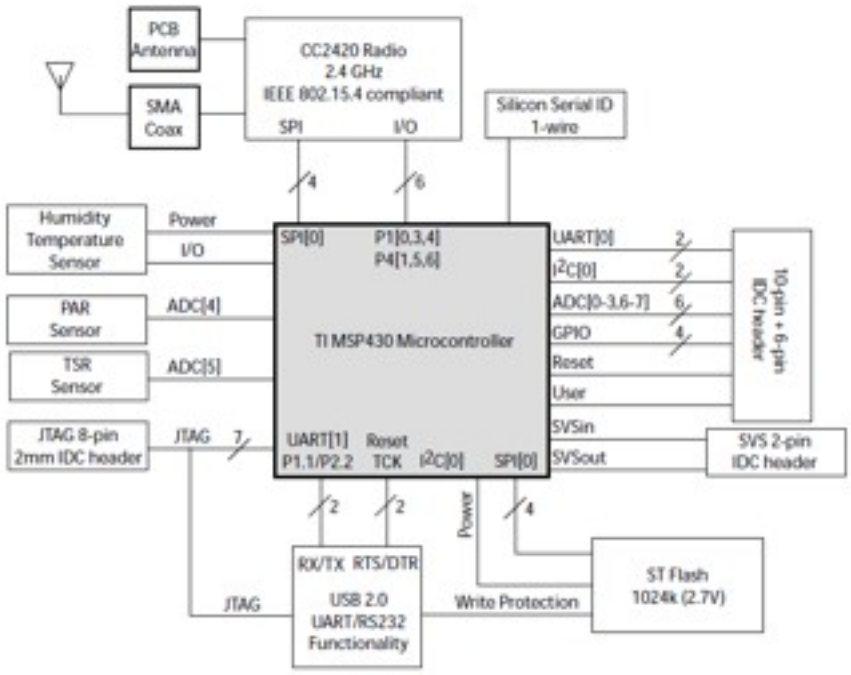
Ο σύνδεσμος 16 ακροδεκτών μπορεί να παρέχει επίσης τροφοδοσία στην πλακέτα. Σε καμία περίπτωση δεν πρέπει να παρέχεται μεγαλύτερη τάση τροφοδοσίας από 3,6 V στην πλακέτα καθώς υπάρχει κίνδυνος καταστροφής του μικροελεγκτή ή των άλλων συστατικών του συστήματος.

Πίνακας 3: Χαρακτηριστικά μεγέθη λειτουργίας του Tmote Sky

	MIN	NOM	MAX	UNIT
Supply voltage	2.1		3.6	V
Supply voltage during flash memory programming	2.7		3.6	V
Operating free air temperature	-40		85	°C
Current Consumption: MCU on, Radio RX		21.8	23	mA
Current Consumption: MCU on, Radio TX		19.5	21	mA
Current Consumption: MCU on, Radio off		1800	2400	µA
Current Consumption: MCU idle, Radio off		54.5	1200	µA
Current Consumption: MCU standby		5.1	21.0	µA

Διάγραμμα Μπλοκ

Στο διάγραμμα μπλοκ που ακολουθεί φαίνεται η διασύνδεση των δομοστοιχείων του Tmote Sky σε λογικό επίπεδο. Συγκεκριμένα παρατηρείται η διασύνδεση των δομοστοιχείων με τον μικροελεγκτή ο οποίος αποτελεί την βασική μονάδα λειτουργίας του Tmote Sky.



Εικόνα 12: Διάγραμμα μπλοκ συνδέσεων δομοστοιχείων της πλακέτας Tmote Sky με τον μικροελεγκτή.

Μικροελεγκτής

Η χαμηλή ισχύς λειτουργίας του Tmote Sky οφείλεται στον εξαιρετικά χαμηλής κατανάλωσης μικροελεγκτή της Texas Instruments MSP430 F1611 που περιλαμβάνει 10kB μνήμη RAM, 48kB μνήμη flash, και 128B μνήμη αποθήκευσης

δεδομένων. Αυτός ο 16-bit επεξεργαστής τύπου RISK έχει εξαιρετικά χαμηλή κατανάλωση ρεύματος σε ενεργή κατάσταση και κατάσταση ύπνου και επιτρέπει στο Tmote Sky να λειτουργεί για μακρά χρονικά διαστήματα με ένα σετ μπαταρίες AA. Ο MSP430 έχει έναν ενσωματωμένο ελεγχόμενο ψηφιακά ταλαντωτή που λειτουργεί στα 8MHz. Ο ταλαντωτής μπορεί να ενεργοποιηθεί από κατάσταση ύπνου σε 6μs, αν και 292 ns είναι η τυπική τιμή σε θερμοκρασίες δωματίου. Όταν ο ταλαντωτής είναι κλειστός ο μικροελεγκτής MSP430 λειτουργεί με βάση ένα εξωτερικό κρυσταλλικό ρολόι συχνότητας 32768 Hz. Αν και η συχνότητα του ψηφιακού ταλαντωτή μεταβάλλεται ανάλογα με την τροφοδοσία και την θερμοκρασία μπορεί επίσης να ρυθμιστεί με τη χρήση ενός άλλου 32kHz ταλαντωτή.

Επιπρόσθετα με τον ψηφιακό ταλαντωτή, ο MSP430 διαθέτει 8 εξωτερικές θύρες αναλογικού σήματος σε ψηφιακό και 8 εσωτερικές θύρες αναλογικού σήματος σε ψηφιακό. Οι εσωτερικές θύρες μετατροπής σήματος από αναλογική σε ψηφιακή μορφή μπορούν να χρησιμοποιηθούν για να διαβάζουν το εσωτερικό θερμίστορ ή να παρακολουθούν την τάση της μπαταρίας. Ο MSP430 διαθέτει επίσης και ένα στοιχείο δύο θυρών 12bit μετατροπέα ψηφιακού σήματος σε αναλογικό, ένα κύκλωμα που επιβλέπει την τάση τροφοδοσίας και ένα κύκλωμα 3 θυρών για έλεγχο της άμεσης πρόσβαση μνήμης.

Σύνδεση με τον υπολογιστή

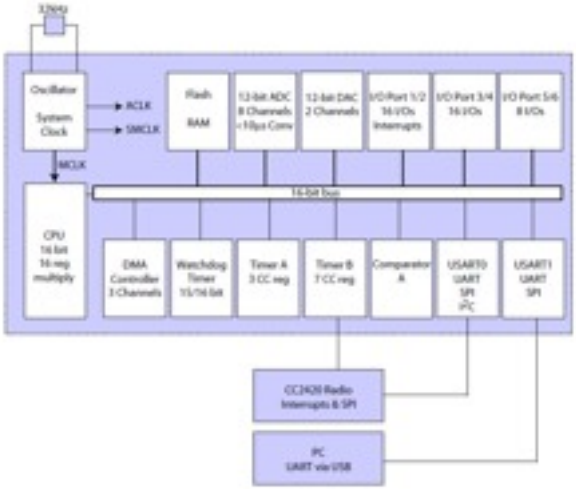
Το Tmote Sky χρησιμοποιεί ένα ελεγκτή USB της FTDI για να επικοινωνεί με τον εκάστοτε υπολογιστή. Για να επικοινωνήσει ο υπολογιστής με την πλακέτα Tmote πρέπει να έχει γίνει εγκατάσταση των κατάλληλων οδηγών (drivers). Η FTDI παρέχει οδηγούς για τα λειτουργικά συστήματα Windows, Linux, Macintosh, Windows CE.

Το Tmote Sky στην διαχείριση συσκευών των Windows εμφανίζεται σαν θύρα επικοινωνίας (COM Port). Πολλαπλά Tmotes μπορούν να συνδεθούν σε έναν υπολογιστή κάνοντας χρήση των θυρών USB που αυτός διαθέτει. Κάθε Tmote θα λάβει ένα ξεχωριστό αναγνωριστικό για την θύρα επικοινωνίας του.

Μια εφαρμογή μπορεί να διαβάσει δεδομένα από το Tmote ανοίγοντας την θύρα επικοινωνίας που έχει ανατεθεί στο συγκεκριμένο Tmote. Τα Tmote επικοινωνεί με τον υπολογιστή κάνοντας χρήση του USART1 του TI MSP430.

Επικοινωνία

Το Tmote Sky για την υλοποίηση της ασύρματης επικοινωνίας του διαθέτει τον πομποδέκτη της Chipcon CC2420. Το CC2420 είναι ένας πομποδέκτης συμβατός με το πρότυπο IEEE 802.15.4 και παρέχει λειτουργίες φυσικού στρώματος και πρόσβασης στο μέσο. Με ευαισθησία που υπερβαίνει τις προδιαγραφές του προτύπου IEEE 802.15.4 και λειτουργία χαμηλής κατανάλωσης το CC2420 παρέχει αξιόπιστη ασύρματη επικοινωνία.



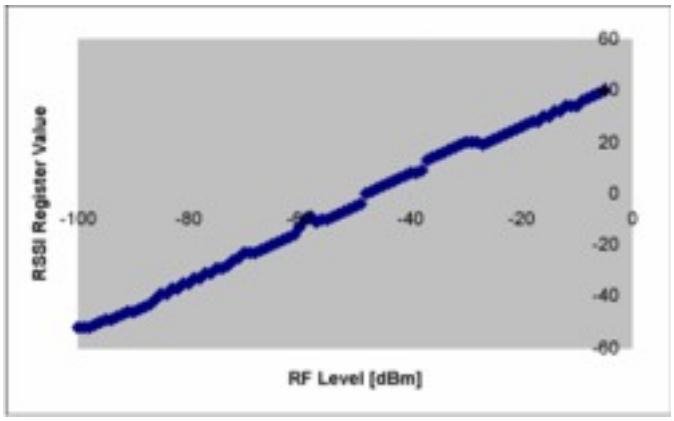
Εικόνα 13: Διασύνδεση σε λογικό επίπεδο της ασύρματης διεπαφής του Tmote Sky.

Το CC2420 ελέγχεται από τον μικροελεγκτή MSP430 με χρήση της θύρας SPI και μιας σειράς από γραμμές εισόδου/εξόδου. Η επικοινωνία μπορεί να κλείσει από τον μικροελεγκτή όταν χρειάζεται λειτουργία πολύ χαμηλής κατανάλωσης ισχύος. Το CC2420 έχει προγραμματιζόμενες ισχείς εξόδου. Τυπικές τιμές μαζί με τα αντίστοιχα ρεύματα που καταναλώνουν παρατίθενται στον πίνακα 4.

Πίνακας 4: Τυπικές τιμές ισχύος και κατανάλωσης ρεύματος.

PA_LEVEL	TXCTRL register	Output Power [dBm]	Current Consumption [mA]
31	0xA0FF	0	17.4
27	0xA0FB	-1	16.5
23	0xA0F7	-3	15.2
19	0xA0F3	-5	13.9
15	0xA0EF	-7	12.5
11	0xA0EB	-10	11.2
7	0xA0E7	-15	9.9
3	0xA0E3	-25	8.5

Το CC2420 παρέχει ένα ενδείκτη ισχύος λαμβανόμενου ψηφιακού σήματος (RSSI) ο οποίος παρέχει δυνατότητα πληροφόρησης προς τον χρήστη ανά πάσα στιγμή. Επιπρόσθετα, σε κάθε λήψη πακέτου το CC2420 υπολογίζει τον ρυθμό σφαλμάτων και παράγει μια ένδειξη για την ποιότητα της σύνδεσης (LQI) που υπολογίζεται για κάθε ληφθέν πακέτο.



Εικόνα 14: Γραφική παράσταση με τυπικές τιμές ισχύς Tmote Sky.

Τυπικές Καταστάσεις Λειτουργίας

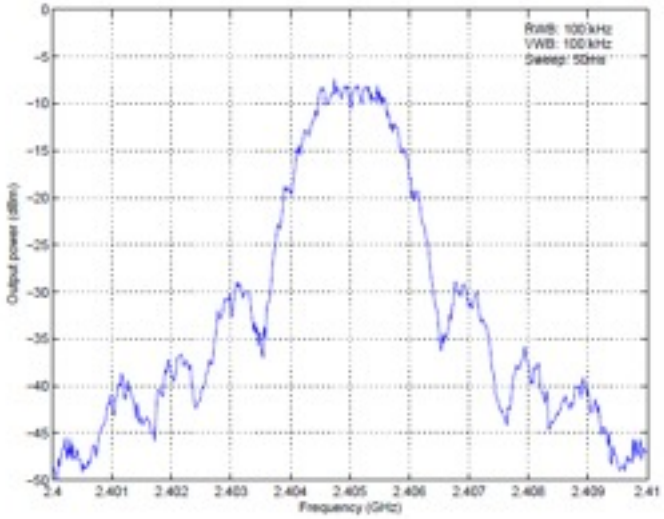
Στον πίνακα 5 απεικονίζονται οι συνήθεις τιμές των βασικών μεγεθών του Tmote Sky ανάλογα με την κατάσταση λειτουργία στην οποία βρίσκεται. Ιδιαίτερο ενδιαφέρον σημειώνουν οι εξεταστικά χαμηλές απαιτήσεις σε ισχύ, καθώς και οι ταχύτατοι χρόνοι απόκρισης του συστήματος.

Πίνακας 5: Χαρακτηριστικά μεγέθη Tmote Sky.

	MIN	NOM	MAX	UNIT
Supply voltage during radio operation (Vreg on)	2.1		3.6	V
Operating free air temperature	-40		85	°C
RF frequency range	2400		2483.5	MHz
Transmit bit rate	250		250	kbps
Nominal output power	-3	0		dBm
Programmable output power range		40		dBm
Receiver sensitivity	-90	-94		dBm
Current consumption: Radio transmitting at 0 dBm		17.4		mA
Current consumption: Radio receiving		19.7		mA
Current consumption: Radio on, Oscillator on		365		µA
Current consumption: Idle mode, Oscillator off		20		µA
Current consumption: Power Down mode, Vreg off			1	µA
Voltage regulator current draw	13	20	29	µA
Radio oscillator startup time		580	860	µs

Μετρούμενη Ισχύς Εξόδου

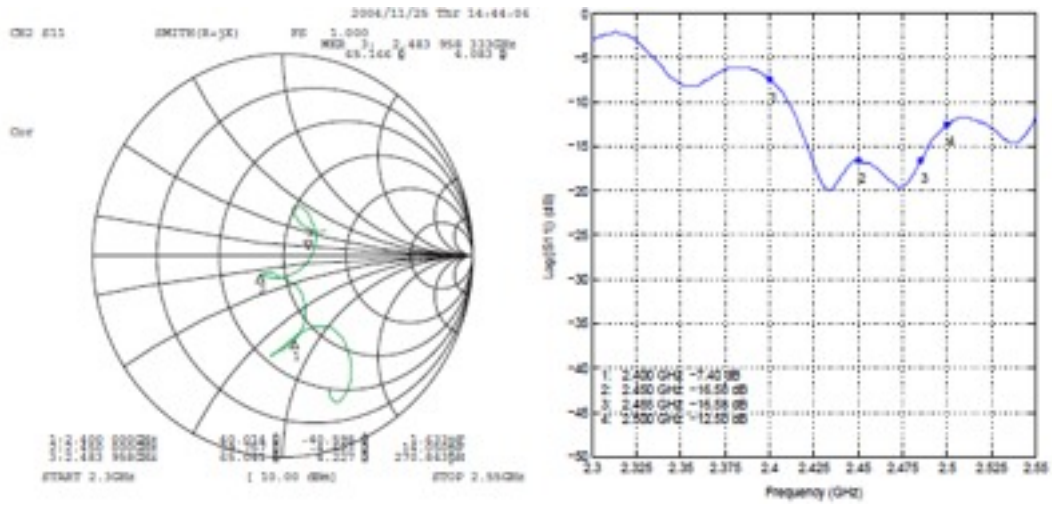
Η ισχύς εξόδου του Tmote Sky από τον πομποδέκτη CC2420 φαίνεται στην παρακάτω εικόνα. Για αυτή την δοκιμή το Tmote Sky εκπέμπει στα 2.405GHz χρησιμοποιώντας διαμόρφωση O-QPSK με DSSS. Η προγραμματιζόμενη ισχύς εξόδου του CC2420 είναι ρυθμισμένη στα 0 dBm.



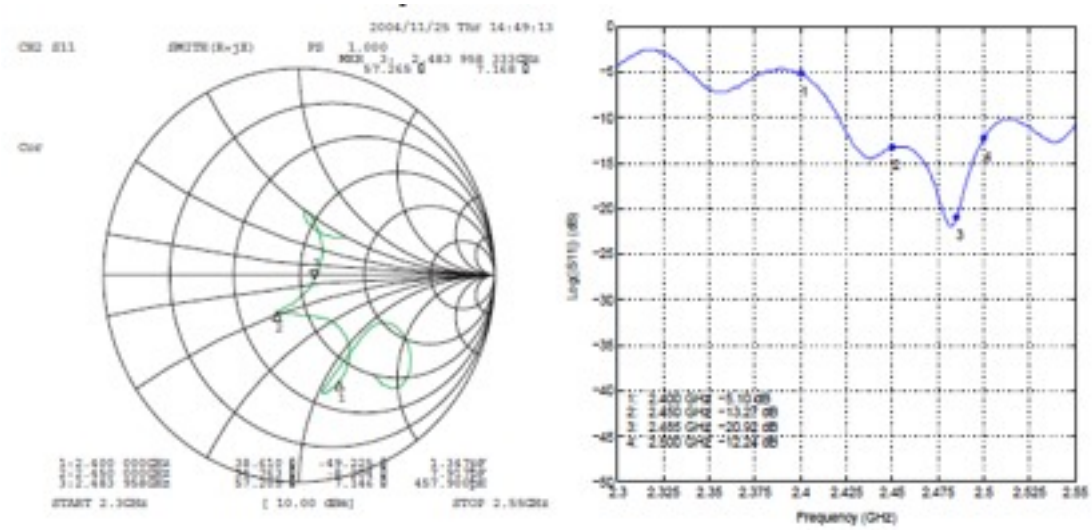
Εικόνα 15: Διάγραμμα ισχύς συναρτήσει της συχνότητας.

Κεραία

Η εσωτερική κεραία του Tmote Sky είναι μια τυπωμένη Inverted-F, σχεδιασμένη να προεξέχει στο πίσω μέρος της πλακέτας μακριά από τις μπαταρίες. Η κεραία Inverted-F είναι ένα μονόπολο όπου το αρχικό της κομμάτι είναι διπλωμένο προς τα κάτω έτσι ώστε να βρίσκεται παράλληλα με το επίπεδο γείωσης. Αν και δεν είναι η καλύτερη ομοιοκατευθυντική λύση, η κεραία μπορεί να πετύχει 50m κάλυψης σε εσωτερικούς χώρους και 125m κάλυψης σε εξωτερικούς χώρους. Δείγματα της επίδοσης της κεραίας με τις μπαταρίες τοποθετημένες στην πλακέτα αλλά και χωρίς φαίνονται στις παρακάτω εικόνες.

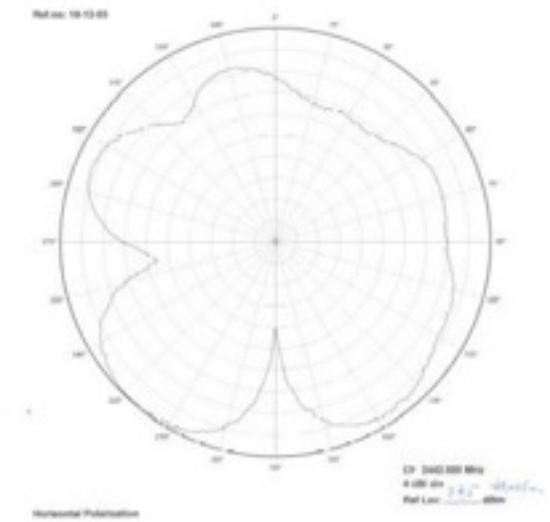


Εικόνα 16: Επιδώσεις κεραίας Tmote Sky συναρτήσει της συχνότητας σε εσωτερικό χώρο.

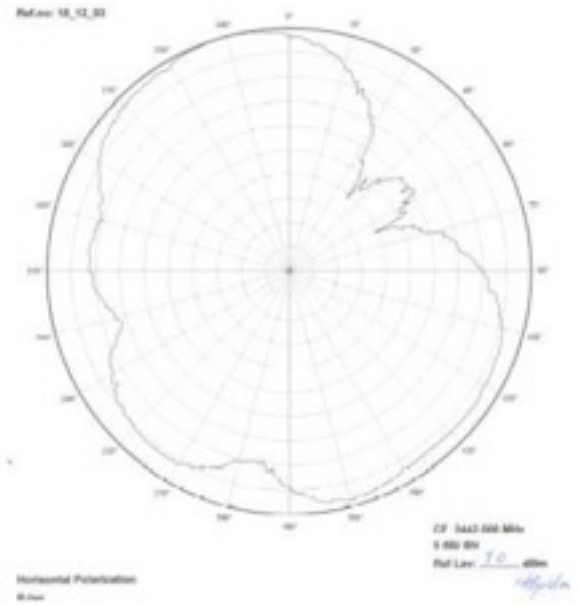


Εικόνα 17: Επιδώσεις κεραίας Tmote Sky συναρτήσει της συχνότητας σε εξωτερικό χώρο.

Διαγράμματα Ακτινοβολίας



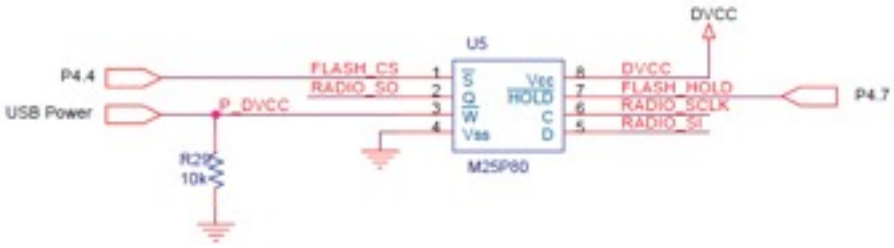
Εικόνα 18: Διάγραμμα ακτινοβολίας της ενσωματωμένης κεραίας στο Tmote Sky σε εσωτερικούς χώρους.



Εικόνα 19: Διάγραμμα ακτινοβολίας της ενσωματωμένης κεραίας στο Tmote Sky σε εξωτερικούς χώρους.

Εξωτερική Μνήμη

Το Tmote Sky χρησιμοποιεί για εξωτερική μνήμη δεδομένων και αποθήκευσης κώδικα, το ST M25P80 40MHz. Η μνήμη χωράει 1024kB δεδομένων και τα αποσυμπιέζει σε 16 κομμάτια των 64kB το καθένα. Η μνήμη μοιράζεται SPI γραμμές επικοινωνίας με τον πομποδέκτη CC2420. Οπότε θα πρέπει να υπάρχει προσοχή κατά την ανάγνωση ή την εγγραφή της μνήμης.



Εικόνα 20: Διάγραμμα μπλοκ της ενσωματωμένης μνήμης στο Tmote Sky.

Αισθητήρες

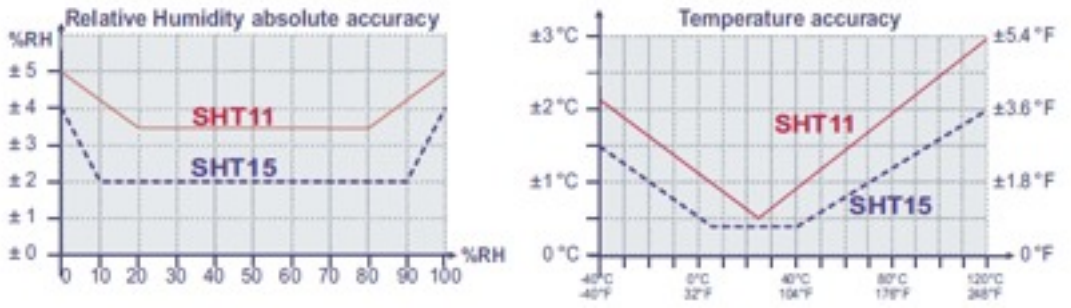
Αισθητήρας Υγρασίας/Θερμοκρασίας

Ο αισθητήρας υγρασίας/θερμοκρασίας κατασκευάστηκε από την Sensirion A.G. Οι SHT11 και SHT15 μπορεί να είναι απευθείας τοποθετημένοι πάνω στο Tmote στην θέση στοιχείων U3. Οι SHT11/SHT15 είναι ρυθμισμένοι να παράγουν

ένα ψηφιακό αποτέλεσμα εξόδου. Οι συντελεστές ρύθμισης είναι αποθηκευμένοι στην ενσωματωμένη EEPROM του αισθητήρα. Η διαφορά ανάμεσα στον SHT11 και στον SHT15 είναι ότι ο τελευταίος παράγει αποτελέσματα μεγαλύτερης ακρίβειας όπως φαίνεται στην παρακάτω εικόνα. Ο αισθητήρας κατασκευάστηκε χρησιμοποιώντας επεξεργασία CMOS και ενώθηκε και με έναν 14-bit μετατροπέα αναλογικού σε ψηφιακό.

Πίνακας 6: Τυπικές τιμές αισθητήρα SHT15.

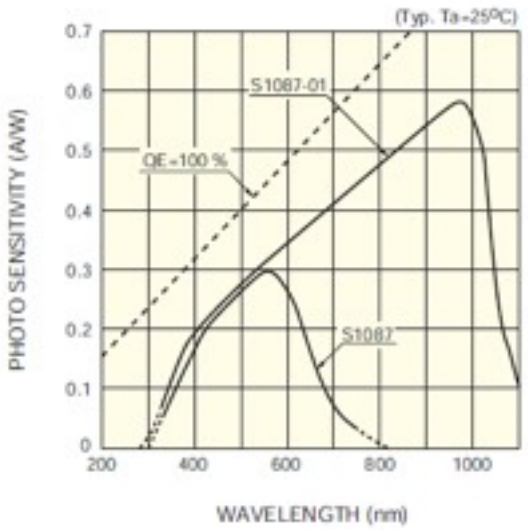
Parameter	MIN	TYP	MAX	Units
Humidity				
Resolution	0.5	0.03	0.03	%RH
	8	12	12	Bit
Repeatability		±0.1		%RH
Range	0		100	%RH
Temperature				
Resolution	0.04	0.01	0.01	°C
	0.07	0.02	0.02	°F
	12	14	14	bit
Repeatability		±0.1		°C
		±0.2		°F
Range	-40		123.8	°C
	-40		254.9	°F



Εικόνα 21: Διαφορές των δύο αισθητήρων θερμοκρασίας και υγρασίας του Tmote Sky.

Αισθητήρας Φωτός

Μεγάλο εύρος από αισθητήρες φωτός μπορούν να χρησιμοποιηθούν με το Tmote Sky. Το Tmote Sky έχει συνδέσεις για δύο φωτοδιόδους. Η εταιρία παραγωγής του Tmote Sky χρησιμοποίει τις φωτοδιόδους της εταιρίας Hamamatsu. Αν το Tmote έχει φωτοδιόδους, αυτές θα είναι η S1087 που θα μετράει την φωτοσυνθετική ενεργή ακτινοβολία και η S1087-01 που θα μετράει όλο το ορατό οπτικό φάσμα και την υπέρυθρη ακτινοβολία.



Εικόνα 22: Διάγραμμα ευαισθησίας συναρτήσει του μήκους κύματος και υγρασίας για τον αισθητήρα φωτός του Tmote Sky.

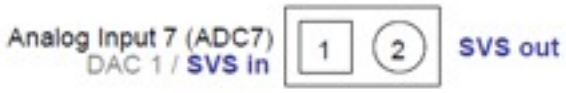
Επιπρόσθετες Συνδέσεις

Το Tmote έχει δύο συνδέσμους για επιπλέον συνδέσεις και ένα ζευγάρι από ενσωματωμένους διακόπτες που μπορούν να ρυθμίσουν επιπρόσθετες συσκευές (αναλογικούς αισθητήρες, ψηφιακές οθόνες, ψηφιακά περιφερειακά) για να ελέγχονται από το Tmote. Στην επάνω όψη του Tmote υπάρχουν στην θέση U2 ένας σύνδεσμος 10 ακροδεκτών και στην θέση U28 ένας σύνδεσμος 6 ακροδεκτών. Ο σύνδεσμος των 10 ακροδεκτών έχει τις ίδιες συνδέσεις όπως έχει το Tmote και θεωρείται ο βασικός του σύνδεσμος. Παρέχει σήματα εισόδου και εξόδου ψηφιακά και αναλογικά. Ο σύνδεσμος των 6 ακροδεκτών παρέχει πρόσβαση σε εξειδικευμένα χαρακτηριστικά του Tmote Sky. Παρέχονται δύο επιπρόσθετες είσοδοι από αναλογικό σήμα σε ψηφιακό, οι οποίες μπορούν να αναπρογραμματιστούν με χρήση προγράμματος και να γίνουν δύο 12-bit έξοδοι σήματος από ψηφιακή μορφή σε αναλογική. Η αναλογική σε ψηφιακή είσοδος 7 μπορεί να χρησιμοποιηθεί και ως είσοδος στο κύκλωμα επίβλεψης τροφοδοσίας.

Εικόνα 23: Διάγραμμα ακροδεκτών του Tmote Sky.

Εικόνα 24: Διάγραμμα ακροδεκτών του Tmote Sky.

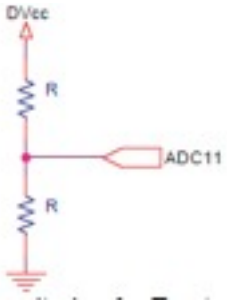
Ένας ξεχωριστός επιβλέπων τροφοδοσίας δύο ακροδεκτών είναι τοποθετημένος κάτω από το USB στην θέση U7. Ακολουθεί εικόνα με το βύσμα του επιβλέποντα τροφοδοσίας που περιλαμβάνει είσοδο και έξοδο από τον μικροελεγκτή.



Εικόνα 25: Διάγραμμα ακροδεκτών του Tmote Sky.

Εσωτερική τροφοδοσία και παρακολούθηση τάσης

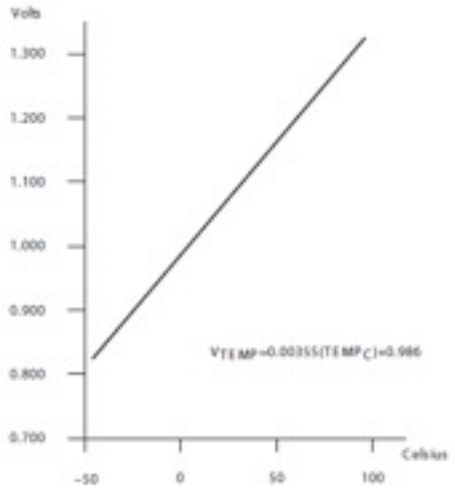
Ο μικροελεγκτής MSP430 διαθέτει εσωτερικούς αισθητήρες θερμοκρασίας και τάσης που μπορούν να χρησιμοποιηθούν από τον μετατροπέα αναλογικού σήματος σε ψηφιακό.



Εικόνα 26: Κύκλωμα μετατροπής σε ενδείξεις τάσης.

Η μετατροπή από τις μονάδες ADC σε ενδείξεις τάσης μπορεί να γίνει με βάση τον τύπο:

Το δεδομένο θερμοκρασίας προέρχεται από μια θερμική δίοδο που είναι συνδεδεμένη εσωτερικά στο μετατροπέα αναλογικού σήματος σε ψηφιακό στον ακροδέκτη 10. Όταν χρησιμοποιείται αυτός ο αισθητήρας θερμοκρασίας η περίοδος δειγματοληψίας πρέπει να είναι μεγαλύτερη από 30μs. Η απόκλιση του αισθητήρα εσωτερικής θερμοκρασίας μπορεί να είναι μεγάλη και να πρέπει να γίνει ρύθμιση του αισθητήρα από το επίπεδο της εφαρμογής. Η τυπική απόκριση του αισθητήρα φαίνεται παρακάτω στην εικόνα που ακολουθεί.



Εικόνα 27: Διάγραμμα αισθητήρα εσωτερικής θερμοκρασίας του Tmote Sky.

Arduino Uno

Το Arduino Uno είναι μια πλακέτα μικροελεγκτή βασισμένο στον επεξεργαστή ATMega 328. Έχει 14 συνδέσμους ψηφιακής εισόδου/εξόδου (έξι από τους οποίους μπορούν να χρησιμοποιηθούν για έξοδο τροφοδοσίας), 6 αναλογικές εισόδους, έναν κρυσταλλικό ταλαντωτή στα 16MHz, μια σύνδεση USB, ένα βύσμα τροφοδοσίας, βύσμα ICSP κεφαλίδας και ένα κουμπί επανεκκίνησης. Περιέχει ότι χρειάζεται για να υπάρχει η δυνατότητα ελέγχου του μικροελεγκτή.

Το Arduino Uno διαφέρει από τις άλλες αντίστοιχες πλακέτες διότι δεν χρησιμοποιεί τον οδηγό FTDI για μετατροπή του USB σε σειριακή σύνδεση, αλλά έχει το ολοκληρωμένο ATmega16U2 που είναι προγραμματισμένο να λειτουργεί σαν μετατροπέας σύνδεσης από USB σε σειριακή μορφή [11].

Εικόνα 28: Όψεις της πλακέτας Arduino Uno.

Πίνακας 7: Χαρακτηριστικά Arduino Uno.

Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Clock Speed	16 MHz

Τροφοδοσία

Το Arduino Uno μπορεί να τροφοδοτηθεί είτε από την θύρα USB είτε με εξωτερική τροφοδοσία. Η εξωτερική τροφοδοσία μπορεί να είναι ή με την χρήση μετασχηματιστή από εναλλασσόμενη τάση σε συνεχή ή με την χρήση μπαταριών. Ο μετασχηματιστής συνδέεται μέσω βύσματος 2,1 χιλιοστών με τον θετικό πόλο στο κέντρο, στο βύσμα που βρίσκεται ενσωματωμένο στο Arduino. Αν δεν υπάρχει μετασχηματιστής, μπορεί να συνδεθεί μπαταρία στο ενσωματωμένο βύσμα 2,1 χιλιοστών.

Η πλακέτα μπορεί να λειτουργήσει με εξωτερική τροφοδοσία από 6 έως 20 V. Αν όμως τροφοδοτηθεί με λιγότερο από 7 V, ο σύνδεσμος τροφοδοσίας 5 V που διαθέτει η πλακέτα ίσως να μην μπορέσει να παράξει 5 V και η πλακέτα να είναι ασταθής. Αν χρησιμοποιηθούν περισσότερα από 12 V για τροφοδοσία ο ρυθμιστής τάσης μπορεί να υπερθερμανθεί και να χαλάσει την πλακέτα. Οπότε προκύπτει ότι η προτεινόμενη τροφοδοσία κυμαίνεται από 7 έως 12 V.

Οι σύνδεσμοι τροφοδοσίας είναι όπως ακολουθούν:

- **Vin**: Είναι ίση με την τάση τροφοδοσίας που έχει εφαρμοστεί στο Arduino. Μπορεί να τροφοδοτήσει εξωτερικές πλακέτες.
- **5V**: Προσφέρει τάση 5V που έχει προκύψει από τον ρυθμιστή τάσης της πλακέτας σε περίπτωση που η τάση τροφοδοσίας είναι μεγαλύτερη των 5V.

- **3,3V**: Προσφέρει 3,3 V που προέρχονται από τον ρυθμιστή τάσης της πλακέτας. Το μεγαλύτερο ρεύμα που καταναλώνεται είναι 50mA.
- **GND**: Σύνδεσμος γείωσης.

Μνήμη

Ο μικροελεγκτής ATmega328 διαθέτει 32KB μνήμη (όπου τα 0,5kB χρησιμοποιούνται από τον φορτωτή εκκίνησης (bootloader)). Διαθέτει επίσης 2kB μνήμη τυχαίας προσπέλασης τύπου SRAM και 1kB μνήμη τύπου EEPROM.

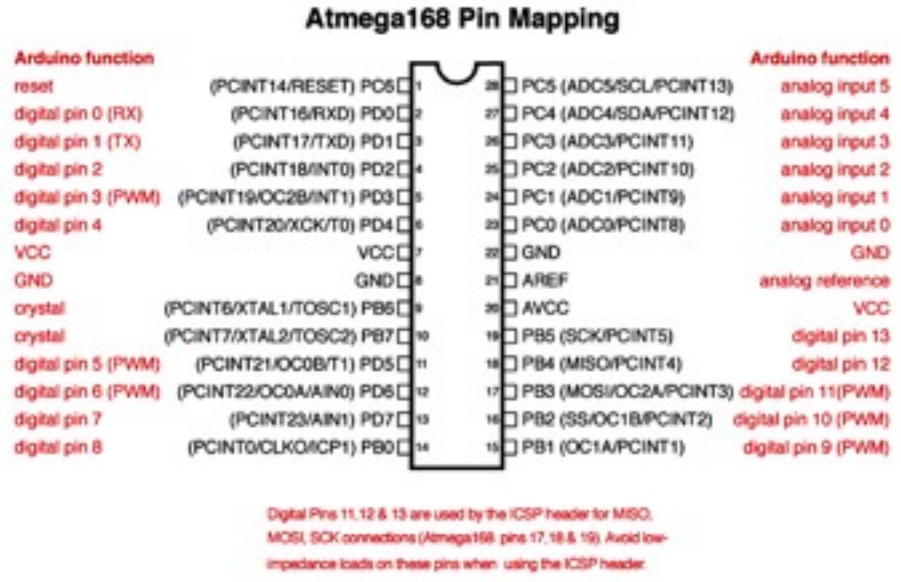
Είσοδοι και Έξοδοι

Κάθε ένας από τους 14 συνδέσμους του Arduino Uno μπορεί να χρησιμοποιηθεί σαν είσοδος ή σαν έξοδος κάνοντας χρήση της συνάρτησης pinMode(), digitalWrite() και digitalRead(). Λειτουργούν στα 5V. Κάθε σύνδεσμος μπορεί να προσφέρει ή να καταναλώσει το μέγιστο 40mA και διαθέτει μια εσωτερική αντίσταση της τάξης των 20-50kΩ. Επιπρόσθετα μερικοί σύνδεσμοι έχουν κάποιες ξεχωριστές δυνατότητες.

- **Serial**: 0 (RX) και 1 (TX). Αυτοί οι σύνδεσμοι μπορούν να χρησιμοποιηθούν για να λάβουν ή να εκπέμψουν σειριακά δεδομένα και συνδέονται στους αντίστοιχους του ATmega8U2.
- **PWM**: Οι σύνδεσμοι 3,5,6,9,10 και 11 παρέχουν 8-bit PWM έξοδο κάνοντας χρήση της συνάρτησης analogWrite().
- **SPI**: Οι σύνδεσμοι 10(SS), 11(MOSI), 12(MISO) και 13(SCK) χρησιμοποιούνται για επικοινωνίες τύπου SPI κάνοντας χρήση της αντίστοιχης βιβλιοθήκης SPI.
- **LED**: Ο σύνδεσμος 13 έχει συνδεδεμένο ένα led και όταν η κατάσταση της θύρας είναι HIGH αυτό ανοίγει και όταν είναι LOW αυτό σβήνει.

Το Arduino Uno έχει και 6 αναλογικές θύρες με ονόματα από A0 έως A5 που η κάθε μια είναι σε θέση να προσφέρει 10 bit ανάλυση. Σαν προεπιλογή μπορούν να μετρήσουν από 0 μέχρι 5V, όμως μπορεί να αλλάξει το επάνω αυτό όριο του εύρους μέτρησης χρησιμοποιώντας την θύρα AREF και την συνάρτηση analogReference().

- **TWI**: Οι θύρες A4 ή SDA και A5 ή SCL υποστηρίζουν επικοινωνία τύπου TWI με χρήση της βιβλιοθήκης Wire.
- **AREF**: Η θύρα αυτή προσφέρει την τάση αναφοράς και μπορεί να χρησιμοποιηθεί μαζί με την συνάρτηση analogReference().
- **Refresh**: Αυτή η θύρα όταν βρεθεί στην κατάσταση του λογικού 0 επανακκινεί τον μικροελεγκτή.



Εικόνα 29: Διασυνδέσεις ολοκληρωμένου μικροελεγκτή του Arduino Uno.

Επικοινωνία

Το Arduino Uno διαθέτει μια σειρά από υποδομές για να μπορεί να επικοινωνεί με τον υπολογιστή, άλλα Arduino καθώς και με άλλους εξωτερικούς μικροελεγκτές. Ο μικροελεγκτής ATmega328 προσφέρει σειριακή επικοινωνία με χρήση του UART TTl(5V), το οποίο είναι διαθέσιμο στις ψηφιακές θύρες 0 και 1 (RX) και (TX) αντίστοιχα. Ο μικροελεγκτής ATmega16U2 μετατρέπει την σειριακή επικοινωνία σε μορφή USB και έτσι προκύπτει η εμφάνιση μιας εικονικής θύρας επικοινωνίας στον υπολογιστή. Τα led Tx και Rx αναβοσβήνουν όποτε η πλακέτα του Arduino δέχεται ή στέλνει δεδομένα διαμέσου της θύρας USB.

Προγραμματισμός

Το Arduino μπορεί να προγραμματιστεί με χρήση του προγράμματος Arduino που παρέχει η εταιρία κατασκευής του. Ο μικροελεγκτής ATmega328 έχει προεγκατεστημένο ένα πρόγραμμα εκκίνησης που επιτρέπει να φορτώνονται κώδικες στην πλακέτα χωρίς να χρειάζεται εξωτερική πλακέτα προγραμματισμού. Η επικοινωνία με τον μικροελεγκτή γίνεται με χρήση του πρωτοκόλλου STK500.

Arduino Ethernet Shield

Η πλακέτα Arduino Ethernet Shield συνδέει το Arduino με το Internet. Η πλακέτα αυτή συνδέεται στην πλακέτα του Arduino μέσω των θυρών SPI. Η τάση

λειτουργίας της είναι 5V που τα προσφέρει η πλακέτα του Arduino. Η πλακέτα Ethernet διαθέτει έναν ελεγκτή Ethernet W5100 με ενσωματωμένη 16kB μνήμη. Η ταχύτητα επικοινωνίας είναι της μορφής 10/100 Mb. Το ολοκληρωμένο W5100 της Wiznet παρέχει την στοίβα δικτύου (IP) που είναι ικανή για υλοποίηση προτύπων TCP και UDP. Υποστηρίζει μέχρι και 5 ταυτόχρονες συνδέσεις τύπου socket. Με την χρήση της βιβλιοθήκης Ethernet μπορούν να γραφτούν προγράμματα που θα έχουν πρόσβαση στο Internet.

Εικόνα 30: Όψεις πλακέτας Ethernet Shield.

Η πλακέτα Ethernet διαθέτει βύσμα τύπου RJ-45 για καλώδιο ethernet και κουμπί για επανεκκίνηση της λειτουργίας της. Επίσης διαθέτει ενσωματωμένη υποδοχή για κάρτα μνήμης τύπου micro-SD η οποία μπορεί να χρησιμοποιηθεί για αποθήκευση αρχείων που θα χρησιμοποιούνται ή θα διακινούνται στο δίκτυο. Η υποδοχή της κάρτας μνήμης micro-SD μπορεί επίσης να διαβάσει την κάρτα μνήμης με χρήση της βιβλιοθήκης SD.

Το Arduino επικοινωνεί με την πλακέτα Ethernet μέσω των θυρών SPI. Αυτές οι θύρες είναι οι 11, 12 και 13. Η πλακέτα επίσης διαθέτει και μια σειρά από led που ο ρόλος τους είναι να παράγουν βοηθητικές ενδείξεις.

- **PWR**: Το led αυτό δείχνει ότι η πλακέτα Ethernet έχει τροφοδοσία.
- **LINK**: Το Led αυτό δείχνει αν υπάρχει συνδεδεμένο καλώδιο δικτύου στην πλακέτα και επίσης αναβοσβήνει όταν η πλακέτα δέχεται ή στέλνει δεδομένα.
- **FULLD**: Αυτό το led δείχνει αν η σύνδεση είναι πλήρως αμφίδρομη.
- **100Mb**: Αυτό το led δείχνει την ταχύτητα της σύνδεσης στο δίκτυο, αν είναι αναμμένο τότε υποστηρίζεται ταχύτητα 100Mb.
- **RX**: Αυτό το led αναβοσβήνει όταν η πλακέτα λαμβάνει δεδομένα.
- **TX**: Αυτό το led αναβοσβήνει όταν η πλακέτα στέλνει δεδομένα.
- **COLL**: Το led αυτό αναβοσβήνει όταν εντοπιστεί κάποια σύγκρουση στο δίκτυο.

Relay Shield

Η πλακέτα ρελέ (Relay Shield) είναι μια πλακέτα με τέσσερα ρελέ που παρέχει ευκολία στις εφαρμογές που απαιτούν έλεγχο υψηλής και χαμηλής τάσης.

Η μέγιστη ισχύς που μπορεί να χρησιμοποιηθεί είναι 90W για συνεχή τάση ή 360V για εναλλασσόμενη τάση. Η πλακέτα μπορεί να ελεγχθεί απευθείας από το Arduino διαμέσου των ψηφιακών συνδέσμων εισόδου/εξόδου που παρέχει το Arduino με εξωτερική τροφοδοσία 9V όπως φαίνεται στην εικόνα 32. Με την υποδοχή RFBee και την πλακέτα 315/433M RF όπως εμφανίζεται στην εικόνα 33 η πλακέτα ρελέ μπορεί να ελεγχθεί απομακρυσμένα, γεγονός που την κάνει ιδανική για εφαρμογές ρομποτικής, έλεγχο βιομηχανικής φύσεως και εφαρμογές έξυπνων σπιτιών.

Εικόνα 31: Πλακέτα Relay Shield.

Εικόνα 32: Πλακέτα Relay Shield ενσωματωμένη σε πλακέτα Arduino.

Εικόνα 33: Πλακέτα Relay Shield με ενσωματωμένο 315M RF για ασύρματη επικοινωνία.

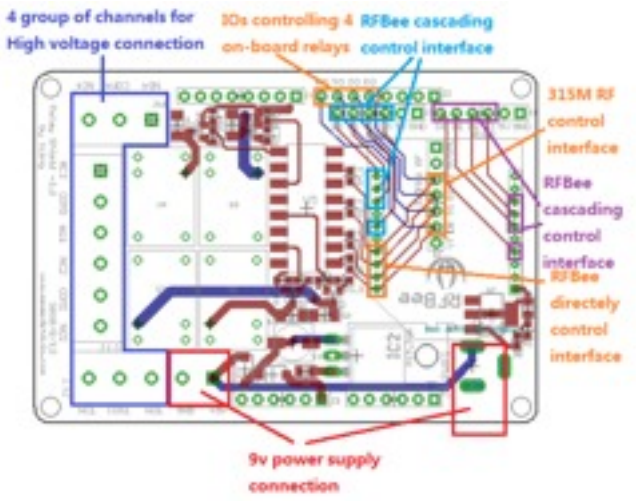
Τα κύρια χαρακτηριστικά της πλακέτας Relay Shield συνοψίζονται ως εξής:

- Τέσσερα ρελέ με κύκλωμα φωτοσύζευξης
- Εύκολη εγκατάσταση με βίδες
- Μικρό βάρος
- Μικρό μέγεθος
- Συμβατή με Arduino

Πίνακας 8: Χαρακτηριστικά λειτουργίας ανά κανάλι

Contact Rating	3A AC 120V / DC 24V
Maximal Switching Voltage	AC 240V / DC 60V
Max Switching Current	5A
Max Switching Power	AC 360VA / DC 90W
Electrical Life Expectancy	100,000 Operations
Mechanical Life Expectancy	10,000,000 Operations
Safety Standard(relay)	UL cUL TUV CQC
Working Voltage	9VDC
Weight	165g
Working temperature	-30℃ to +85℃
Working Humidity	40% - 85%

Διάγραμμα Μπλοκ



Εικόνα 34: Διάγραμμα μπλοκ του Relay Shield.

Στην ανωτέρω εικόνα αναδεικνύεται η διασύνδεση των δομοστοιχείων της πλακέτα Relay Shield σε λογικό επίπεδο. Διακρίνονται τα εξής στοιχεία:

Ομάδα 1

- **COM1**: Κοινός ακροδέκτης
- **NC1**: Κανονικά κλειστός, στην περίπτωση που ο NC1 είναι συνδεδεμένος με τον COM1 τότε στην είσοδο D0 εφαρμόζεται λογικό μηδέν αλλιώς αν εφαρμόζεται λογικό ένα ο NC1 δεν είναι συνδεδεμένος με τον COM1.
- **NO1**: Κανονικά ανοικτός, στην περίπτωση που εφαρμόζεται λογικό μηδέν στην είσοδο D0 τότε ο NO1 δεν είναι συνδεδεμένος με τον COM1. Αν στην είσοδο D0 εφαρμόζεται λογικό ένα τότε το NO1 είναι συνδεδεμένος με τον COM1.

Η πλακέτα Relay Shield διαθέτει άλλες τρείς ομάδες που έχουν ακριβώς την ίδια λογική με την παραπάνω μόνο που αλλάζει η είσοδος ελέγχου και γίνεται αντίστοιχα D1,D2,D3.

Βύσμα τροφοδοσίας για 9V πηγή τάσης

Είσοδοι/Έξοδοι ελέγχου ρελέ: Στην πλακέτα υπάρχουν τέσσερεις ακροδέκτες ελέγχου D0, D1, D2, D3 που με αυτούς ελέγχονται τα ρελέ και πρέπει να είναι συνδεδεμένοι στους ακροδέκτες του Arduino 4 έως 7, για να υπάρχει δυνατότητα να ελέγχονται τα ρελέ από το Arduino.
Διεπαφή Ελέγχου 315M RF: Μαζί με τους ακροδέκτες της γείωσης GND και των 5V υπάρχει και η διεπαφή για την πλακέτα 315M RF για να ελέγχονται τα ρελέ απομακρυσμένα.
Διεπαφή απευθείας ελέγχου RFBee: Τα τέσσερα ρελέ μπορούν να ελεγχθούν απευθείας από την πλακέτα RFBee με την χρήση των 4 DIO συνδέσμων του RFBee.
RFBee cascading διεπαφή ελέγχου: Καθώς το RFBee έχει 12 συνδέσμους DIO ένα RFBee μπορεί να ελέγξει δύο πλακέτες Relay Shield όπως υποδεικνύεται στην εικόνα 35.

Εικόνα 35: Ένα RFBee ελέγχει τρείς πλακέτες Relay Shield.

Πλακέτα XBEE Shield

Εικόνα 36: XBee Shield στα δεξιά και XBee module στα αριστερά.

Η πλακέτα XBEE Shield επιτρέπει στο Arduino να επικοινωνεί ασύρματα με άλλους κόμβους κάνοντας χρήση του πρωτοκόλλου ZigBee. Επίσης επιτρέπει τον προγραμματισμό του δομοστοιχείου XBee κάνοντας χρήση της θύρας USB του Arduino.

Πλακέτα XBee

Το δομοστοιχείο XBee κατασκευάστηκε με βάση τα χαρακτηριστικά του προτύπου IEEE 802.15.4 για να πληρεί τις ανάγκες για χαμηλή κατανάλωση και χαμηλό κόστος. Η συχνότητα λειτουργίας του είναι 2.4GHz όπως προβλέπεται από το πρότυπο IEEE 802.15.4.

Το XBee προσφέρει ασύρματη συνδεσιμότητα σε συσκευές. Χρησιμοποιεί το πρωτόκολλο IEEE802.15.4 για γρήγορη δικτύωση σημείου σε πολλαπλά σημεία και σημείο σε σημείο. Σχεδιάστηκε για εφαρμογές που απαιτούν μεγάλη διαπερατότητα, χαμηλή καθυστέρηση και προβλεπόμενο χρόνο επικοινωνιών.

Τεχνικά Χαρακτηριστικά

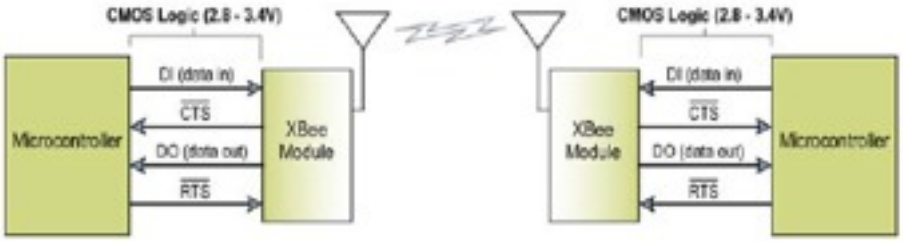
Πίνακας 9: Βασικά τεχνικά χαρακτηριστικά λειτουργίας του δομοστοιχείου XBee.

Specification	XBee	XBee-PRO
Performance		
Indoor/Urban Range	up to 100 ft. (30 m)	Up to 300' (100 m)
Outdoor RF line-of-sight Range	up to 300 ft. (100 m)	Up to 1 mile (1500 m)
Transmit Power Output (software selectable)	1mW (0 dBm)	60 mW (18 dBm) conducted, 100 mW (20 dBm) EIRP*
RF Data Rate	250,000 bps	250,000 bps
Serial Interface Data Rate (software selectable)	1200 - 115200 bps (non-standard baud rates also supported)	1200 - 115200 bps (non-standard baud rates also supported)
Receiver Sensitivity	-92 dBm (1% packet error rate)	-100 dBm (1% packet error rate)
Power Requirements		
Supply Voltage	2.8 – 3.4 V	2.8 – 3.4 V
Transmit Current (typical)	45mA (@ 3.3 V)	If PL=0 (10dBm): 137mA(@3.3V), 139mA(@3.0V) PL=1 (12dBm): 155mA (@3.3V), 153mA(@3.0V) PL=2 (14dBm): 170mA (@3.3V), 171mA(@3.0V) PL=3 (16dBm): 186mA (@3.3V), 195mA(@3.0V) PL=4 (18dBm): 215mA (@3.3V), 227mA(@3.0V)
Idle / Receive Current (typical)	50mA (@ 3.3 V)	55mA (@ 3.3 V)
Power-down Current	< 10 µA	< 10 µA
General		
Operating Frequency	ISM 2.4 GHz	ISM 2.4 GHz
Dimensions	0.960" x 1.087" (2.438cm x 2.761cm)	0.960" x 1.297" (2.438cm x 3.294cm)
Operating Temperature	-40 to 85º C (industrial)	-40 to 85º C (industrial)
Antenna Options	Integrated Whip, Chip or U.FL Connector	Integrated Whip, Chip or U.FL Connector
Networking & Security		
Supported Network Topologies	Point-to-point, Point-to-multipoint & Peer-to-peer	
Number of Channels (software selectable)	16 Direct Sequence Channels	12 Direct Sequence Channels
Addressing Options	PAN ID, Channel and Addresses	PAN ID, Channel and Addresses
Agency Approvals		
United States (FCC Part 15.247)	OUR-XBEE	OUR-XBEEPRO
Industry Canada (IC)	4214A XBEE	4214A XBEEPRO
Europe (CE)	ETSI	ETSI (Max. 10 dBm transmit power output)*
Japan	n/a	005NYCA0378 (Max. 10 dBm transmit power output)**

Το XBee έχει ισχύ εξόδου 1mW και η ακτίνα εκπομπής και λήψης του κυμαίνεται από τα 30 μέτρα για εσωτερικούς χώρους έως τα 90 μέτρα για εξωτερικούς χώρους. Ο ρυθμός μετάδοσης δεδομένων στο ασύρματο κανάλι είναι 250kbps και η ευαισθησία του δέκτη του είναι -91dBm. Υποστηρίζει έλεγχο σφαλμάτων με χρήση επαναλήψεων ή επιβεβαιώσεων. Έχει χωρητικότητα 16 καναλιών με το κάθε κανάλι να μπορεί να έχει μέχρι και 65.000 διευθύνσεις. Η τάση τροφοδοσίας του κυμαίνεται από 2,8V έως 3,4V. Ο πομπός του καταναλώνει 45mA ρεύμα και ο δέκτης του 50mA ρεύμα, αλλά το XBee σε κατάσταση ύπνου καταναλώνει μόλις 10μA.

Το XBee συνδέεται με το Arduino με χρήση της πλακέτας XBee Shield και έτσι τροφοδοτείται αυτόματα από το Arduino και δεν χρειάζεται εξωτερική πηγή ισχύος. Επίσης κάνει χρήση των ακροδεκτών 0 και 1 του Arduino με αποτέλεσμα να μπορεί να εκπέμπει ασύρματα τα δεδομένα που παράγει το Arduino για σειριακή επικοινωνία καθώς επίσης και να λαμβάνει τα αντίστοιχα δεδομένα. Κάνει δηλαδή μετατροπή της σειριακής επικοινωνίας σε ασύρματη, ενσωματώνοντας τα δεδομένα

που λαμβάνονται και αποστέλλονται στην στοίβα του προτύπου IEEE802.15.4 και ZigBee.



Εικόνα 37: Λογική λειτουργίας της μονάδας XBee module.

Πίνακας 10: Σήματα ακροδεκτών του δομοστοιχείου XBee.

Pin #	Name	Direction	Description
1	VCC	-	Power supply
2	DOUT	Output	UART Data Out
3	DIN / CONFIG	Input	UART Data In
4	DO8*	Output	Digital Output 8
5	RESET	Input	Module Reset (reset pulse must be at least 200 ns)
6	PWM0 / RSSI	Output	PWM Output 0 / RX Signal Strength Indicator
7	PWM1	Output	PWM Output 1
8	[reserved]	-	Do not connect
9	DTR / SLEEP_RQ / DI8	Input	Pin Sleep Control Line or Digital Input 8
10	GND	-	Ground
11	AD4 / DIO4	Either	Analog Input 4 or Digital I/O 4
12	CTS / DIO7	Either	Clear-to-Send Flow Control or Digital I/O 7
13	ON / SLEEP	Output	Module Status Indicator
14	VREF	Input	Voltage Reference for A/D Inputs
15	Associate / AD5 / DIO5	Either	Associated Indicator, Analog Input 5 or Digital I/O 5
16	RTS / AD6 / DIO6	Either	Request-to-Send Flow Control, Analog Input 6 or Digital I/O 6
17	AD3 / DIO3	Either	Analog Input 3 or Digital I/O 3
18	AD2 / DIO2	Either	Analog Input 2 or Digital I/O 2
19	AD1 / DIO1	Either	Analog Input 1 or Digital I/O 1
20	AD0 / DIO0	Either	Analog Input 0 or Digital I/O 0

Επίλογος – Συμπεράσματα

Στο Κεφάλαιο 2 εξετάστηκαν λεπτομερώς όλες οι βασικές διατάξεις που θα αποτελέσουν τα δομικά στοιχεία της κατασκευής του δικτύου που υλοποιεί μέρος της εφαρμογής ευφυούς οικείας. Η ανάλυση βασίστηκε κυρίως στα τεχνικά χαρακτηριστικά των διατάξεων, καθώς και στις δυνατότητες που παρέχουν. Από αυτή την ανάλυση προκύπτει το μεγάλο εύρος που υπάρχει διαθέσιμο σε επίπεδο διατάξεων για την υλοποίηση Δικτύων Ασύρματων Αισθητήρων, όπως επίσης και το βάρος που πρέπει να δοθεί στις σχεδιαστικές παραμέτρους του Δικτύου έτσι ώστε το αποτέλεσμα να είναι βέλτιστο.

Στο Κεφάλαιο 3 που ακολουθεί παρατίθεται η ανάλυση προγραμματιστικών τεχνικών που θα χρησιμοποιηθούν για την υλοποίηση των διεπαφών χρήστη. Λόγω της δυικής συμπεριφοράς της εφαρμογής θα αναλυθούν γλώσσες προγραμματισμού διαδικτύου καθώς και η γλώσσα προγραμματισμού Android κινητών τηλεφώνων.

ΚΕΦΑΛΑΙΟ 3: Γλώσσες Προγραμματισμού

NesC

Ο προγραμματισμός του λειτουργικού συστήματος TinyOS που χρησιμοποιείται στους κόμβους αισθητήρων Tmote Sky γίνεται με την χρήση της γλώσσας NesC. Η γλώσσα NesC είναι μια γλώσσα που παρουσιάζει αρκετές ομοιότητες με την γλώσσα προγραμματισμού C. Το σημείο όμως όπου η γλώσσα NesC διαφέρει αρκετά από την γλώσσα C είναι στο μοντέλο σύνδεσης που χρησιμοποιεί. Η πρόκληση και η πολυπλοκότητα που προκύπτει δεν βρίσκεται στην συγγραφή κώδικα για τα στοιχεία αλλά στον συνδυασμό των στοιχείων έτσι ώστε να δημιουργηθεί μια λειτουργική εφαρμογή [12].

Στοιχεία και Διεπαφές

Η γλώσσα NesC είναι μια γλώσσα που βασίζεται στα στοιχεία και χρησιμοποιεί διάλεκτο C. Κατά κάποιο τρόπο τα στοιχεία της NesC είναι σαν τα αντικείμενα σε άλλες γλώσσες προγραμματισμού. Για παράδειγμα τα στοιχεία ενσωματώνουν μια κατάσταση και ζευγαρώνουν την κατάσταση με την λειτουργικότητα. Η διακριτική διαφορά βρίσκεται στην πλευρά του πεδίου ονομάτων. Αντίθετα με τα αντικείμενα στην C++ και την Java που αναφέρονται στις συναρτήσεις και τις μεταβλητές χρησιμοποιώντας πεδίο ονομάτων με μη τοπική ισχύ, στην NesC το πεδίο ονομάτων έχει καθαρά τοπικό χαρακτήρα. Αυτό σημαίνει ότι επιπρόσθετα με την δήλωση των συναρτήσεων που είναι προς εκτέλεση, ένα στοιχείο πρέπει επίσης να δηλώνει τις συναρτήσεις που αυτό καλεί. Τα ονόματα που αυτό το στοιχείο χρησιμοποιεί για να καλέσει μια συνάρτηση είναι καθαρά τοπικά. Όταν ένα στοιχείο Α καλεί μια συνάρτηση Β, βασικά γνωστοποιεί το όνομα Α.Β σε ένα πεδίο ονομάτων. Αντίστοιχα όταν ένα στοιχείο C καλέσει μια συνάρτηση Β τότε ορίζεται το όνομα C.B σε ένα πεδίο ονομάτων. Αν και τα δύο στοιχεία Α και C φαίνεται να αναφέρονται στην ίδια συνάρτηση Β, μπορεί να αναφέρονται σε τελείως διαφορετικές εκτελέσεις.

Μια εφαρμογή NesC αποτελείται λοιπόν από ένα ή πολλά στοιχεία που συναρμολογούνται ή διασυνδέονται μεταξύ τους για να υλοποιήσουν μια εκτελέσιμη μορφή εφαρμογής. Τα στοιχεία έχουν δύο σκοπούς: έναν για τις προδιαγραφές τους, όπου περιέχει τα ονόματα των διεπαφών τους και έναν δεύτερο για την εκτέλεση τους. Ένα στοιχείο προσφέρει και χρησιμοποιεί διεπαφές. Οι προσφερόμενες διεπαφές έχουν ως σκοπό να αναπαραστήσουν την λειτουργικότητα που προσφέρει το στοιχείο στον χρήστη. Οι χρησιμοποιούμενες διεπαφές αναπαριστούν την λειτουργικότητα που χρειάζεται το στοιχείο για να εκτελέσει αυτό που του έχει ανατεθεί.

Οι διεπαφές είναι αμφίδρομες, ορίζουν μια ομάδα από εντολές οι οποίες είναι συναρτήσεις που θα εκτελεστούν από τον πάροχο της διεπαφής και μια ομάδα από γεγονότα που είναι συναρτήσεις που θα εκτελεστούν από τον χρήστη της διεπαφής. Για να καλέσει ένα στοιχείο τις εντολές μιας διεπαφής πρέπει να εκτελέσει τα γεγονότα αυτής της διεπαφής. Ένα στοιχείο μπορεί να προσφέρει ή να χρησιμοποιεί πολλαπλές διεπαφές και πολλαπλές εκφάνσεις της ίδιας διεπαφής. Το σύνολο των διεπαφών που ένα στοιχείο χρησιμοποιεί ή προσφέρει αποτελούν την υπογραφή του στοιχείου αυτού.

Παραμετροποιήσεις και Ενότητες

Υπάρχουν δύο τύποι στοιχείων στην γλώσσα NesC και αυτά είναι οι ενότητες και οι παραμετροποιήσεις. Οι ενότητες προσφέρουν τις προδιαγραφές μιας ή περισσοτέρων διεπαφών. Οι παραμετροποιήσεις χρησιμοποιούνται για να ενώσουν άλλα στοιχεία μαζί. Κάθε εφαρμογή που υλοποιείται με γλώσσα NesC περιγράφεται από μια προδιαγραφή αρχικού επιπέδου που συνδέει τα στοιχεία που βρίσκονται εσωτερικά.

Παράδειγμα Εφαρμογής NesC

Εφαρμογή Blink

Το παράδειγμα που ακολουθεί απεικονίζει έναν μετρητή στα τρία led ενός Tmote Sky. Αυτό που κάνει ο κώδικας του παραδείγματος είναι να αναβοσβήνει το LED0 με συχνότητα 4Hz, το LED1 με συχνότητα 2Hz και το LED2 με συχνότητα 1Hz. Το αποτέλεσμα που προκύπτει είναι σαν τα τρία led να αναπαριστούν έναν δυαδικό μετρητή από το μηδέν μέχρι το επτά κάθε δύο δευτερόλεπτα. Η εφαρμογή Blink αποτελείται από δύο στοιχεία. Μια ενότητα που ονομάζεται BlinkC.nc και μια παραμετροποίηση που ονομάζεται BlinkAppC.nc. Όλες οι εφαρμογές χρειάζονται ένα αρχείο παραμετροποίησης αρχικού επιπέδου που συνήθως ονομάζεται από την ίδια την εφαρμογή. Στην περίπτωση του παραδείγματος το BlinkAppC.nc είναι η παραμετροποίηση για την εφαρμογή Blink και το αρχείο που χρησιμοποιεί ο μεταγλωττιστής της NesC για να παράγει το εκτελέσιμο αρχείο. Το αρχείο BlinkC.nc από την άλλη πλευρά προσφέρει την υλοποίηση της εφαρμογής Blink. Το αρχείο BlinkAppC.nc χρησιμοποιείται για να γίνει η σύνδεση της ενότητας BlinkC.nc με άλλα στοιχεία που χρειάζεται η εφαρμογή Blink για να λειτουργήσει.

```
configuration BlinkAppC {
}
implementation {
  components MainC, BlinkC, LedsC;
  components new TimerMilliC() as Timer0;
  components new TimerMilliC() as Timer1;
  components new TimerMilliC() as Timer2;

  BlinkC -> MainC.Boot;
  BlinkC.Timer0 -> Timer0;
  BlinkC.Timer1 -> Timer1;
  BlinkC.Timer2 -> Timer2;
  BlinkC.Leds -> LedsC;
}
```

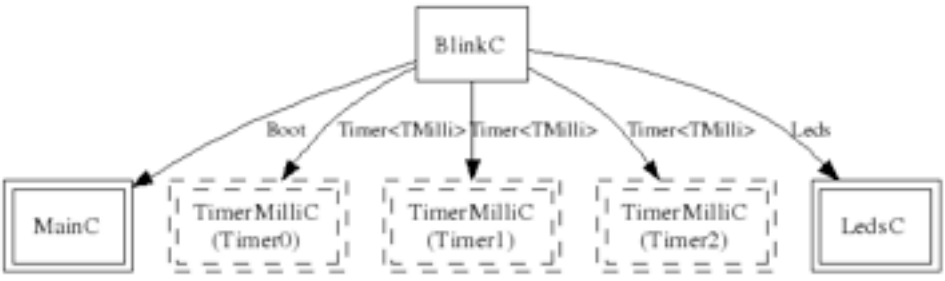
Στην πρώτη γραμμή παρατηρείται η λέξη configuration όπου δηλώνει ότι ο κώδικας είναι κώδικας παραμετροποίησης. Μέσα στα { } παρατηρείται η υλοποίηση της παραμετροποίησης με όνομα BlinkAppC όπου δηλώνονται τα στοιχεία που θα χρησιμοποιηθούν. Το MainC.Boot είναι διεπαφή που ορίζεται στο λειτουργικό σύστημα TinyOS. Οι τελευταίες τέσσερεις γραμμές συνδέουν διεπαφές που χρησιμοποιεί το στοιχείο BlinkC σε διεπαφές όπου τα TimerMilliC και LedsC στοιχεία προσφέρουν.

```
module BlinkC {
  uses interface Timer<TMilli> as Timer0;
  uses interface Timer<TMilli> as Timer1;
  uses interface Timer<TMilli> as Timer2;
  uses interface Leds;
  uses interface Boot;
}
implementation
{
  // implementation code omitted
}
```

Το πρώτο κομμάτι του κώδικα της ενότητας δείχνει ότι η ενότητα ονομάζεται BlinkC και δηλώνονται οι διεπαφές που αυτή χρησιμοποιεί αλλά και προσφέρει. Η ενότητα BlinkC χρησιμοποιεί τρείς διεπαφές. Τρείς εκφάνσεις της διεπαφής Timer<TMilli>, την διεπαφή Leds και την διεπαφή Boot.

Μετά από το παραπάνω απόσπασμα γίνεται πιο καθαρό τι συμβαίνει στον κώδικα παραμετροποίησης BlinkAppC.nc. Η γραμμή BlinkC.Timer0 -> Timer0 συνδέει τις τρείς διεπαφές Timer<TMilli> που χρησιμοποιούνται από το BlinkC στην διεπαφή Timer<TMilli> προσφέροντας τρία στοιχεία TimerMilliC. Αντίστοιχα το ίδιο συμβαίνει και για τα led.

Οπτικοποίηση του διαγράμματος στοιχείων



Εικόνα 38: Διάγραμμα εφαρμογής Blink.

Στο παραπάνω διάγραμμα το απλό κουτί είναι ενότητα και το διπλό κουτί είναι παραμετροποίηση. Τα κουτιά με τις διακεκομμένες γραμμές δηλώνουν ότι τα στοιχεία είναι γενικής χρήσης.

Ενότητες και Καταστάσεις

Κάνοντας μεταγλώττιση μια εφαρμογής TinyOS παράγεται μια δυαδική εικόνα που υποθέτει ότι έχει τον πλήρη έλεγχο του υλικού της πλακέτας. Για αυτό το λόγο η πλακέτα για παράδειγμα Tmote Sky μπορεί να εκτελεί μόνο μια εικόνα εφαρμογής TinyOS την φορά. Η εικόνα αποτελείται από τα στοιχεία που χρειάζονται για μια εφαρμογή. Επειδή οι περισσότερες πλατφόρμες κόμβων αισθητήρων δεν έχουν προστασία μνήμης βασισμένη στο υλικό, δεν υπάρχει διαχωρισμός ανάμεσα στις διευθύνσεις χώρου χρήστη και συστήματος, υπάρχει μόνο μια διεύθυνση χώρου που όλα τα στοιχεία μοιράζονται. Για αυτό το λόγο πολλά στοιχεία του TinyOS προσπαθούν να κρατήσουν την κατάσταση τους ιδιωτική και αποφεύγουν να περνάνε δείκτες μνήμης.

Οι ενότητες μπορούν να δηλώσουν κατάσταση μεταβλητών. Ότι κατάσταση δηλώσει ένα στοιχείο είναι ιδιωτική, κανένα άλλο στοιχείο δεν μπορεί να την ονομάσει ή να αποκτήσει απευθείας πρόσβαση σε αυτή. Η μόνη περίπτωση δύο στοιχεία να αλληλεπιδρούν απευθείας είναι μέσω των διεπαφών. Ακολουθεί εμπλουτισμένος κώδικας για την εφαρμογή Blink.

```
apps/Blink/BlinkC.nc:
module BlinkC @safe(){
  uses interface Timer<TMilli> as Timer0;
  uses interface Timer<TMilli> as Timer1;
  uses interface Timer<TMilli> as Timer2;
  uses interface Leds;
  uses interface Boot;
}
implementation
{
  event void Boot.booted()
  {
    call Timer0.startPeriodic( 250 );
    call Timer1.startPeriodic( 500 );
    call Timer2.startPeriodic( 1000 );
  }

  event void Timer0.fired()
  {
    call Leds.led0Toggle();
  }

  event void Timer1.fired()
  {
    call Leds.led1Toggle();
  }

  event void Timer2.fired()
  {
    call Leds.led2Toggle();
  }
}
```

Το αρχικό αρχείο κώδικα BlinkC.nc δεν διανέμει καμία κατάσταση. Αυτό τώρα το αλλάξαμε με το ανωτέρω απόσπασμα. Στο κομμάτι κώδικα τώρα πλέον έχει προκληθεί εκκίνηση και αρχικοποίηση της πλακέτας. Αφού αυτή έχει εκκινήσει κανονικά ενεργοποιούνται οι μετρητές που ορίζουν την συχνότητα που θα αναβοσβήνουν τα led και τέλος έχει ζητηθεί από τα led να αρχίσουν να αναβοσβήνουν όποτε τα καλέσει ο εκάστοτε μετρητής.

Ο κώδικας του αρχείου BlinkAppC.nc παραμένει ακριβώς ο ίδιος διότι κάνει αυτό που πρέπει, δηλαδή την δήλωση των στοιχείων που θα χρησιμοποιηθούν και την σύνδεση μεταξύ τους.

Γλώσσα Προγραμματισμού Arduino

Η γλώσσα προγραμματισμού για την πλακέτα Arduino βασίζεται στην τεχνολογία Wiring η οποία είναι ένα πλαίσιο προγραμματισμού ανοικτού κώδικα για μικροελεγκτές. Η διάλεκτος και οι εντολές της είναι παρόμοιες με την γλώσσα

προγραμματισμού C. Παρακάτω ακολουθούν μερικές από τις βασικές αρχές για τον προγραμματισμό της πλακέτας Arduino.

Σχεδιάγραμμα (Sketch)

Το σχεδιάγραμμα είναι το όνομα που χρησιμοποιεί το Arduino για το εκάστοτε πρόγραμμα. Είναι η μονάδα κώδικα που φορτώνεται και εκτελείται στην πλακέτα Arduino.

Σχόλια (Comments)

```
/*
 * Blink
 *
 * The basic Arduino example.  Turns on an LED on for one second,
 * then off for one second, and so on...  We use pin 13 because,
 * depending on your Arduino board, it has either a built-in LED
 * or a built-in resistor so that you need only an LED.
 *
 * http://www.arduino.cc/en/Tutorial/Blink
 */
```

Ότι βρίσκεται μεταξύ /* και */ αγνοείται από το Arduino όταν τρέχει τον κώδικα που βρίσκεται στο σχεδιάγραμμα που του έχει φορτωθεί. Αυτό χρησιμοποιείται για τους ανθρώπους που διαβάζουν τον κώδικα μιας εφαρμογής για να μπορούν να καταλάβουν τι κάνει ο κώδικας.

Μεταβλητές (Variables)

Οι μεταβλητές είναι ένας χώρος που αποθηκεύονται τα δεδομένα. Έχουν όνομα, τύπο και τιμή. Για παράδειγμα στην παρακάτω γραμμή κώδικα παρατηρείται η δήλωση μεταβλητής με όνομα ledPin με τύπο int και τιμή 13. Χρησιμοποιείται για να δείξει σε ποιον ακροδέκτη του Arduino συνδέεται ένα led.

```
int ledPin = 13;                // LED connected to digital pin 13
```

Με αυτόν τον τρόπο κάθε φορά που αναφέρεται το όνομα ledPin ανακαλείται και η τιμή που της έχει ανατεθεί.

Συναρτήσεις (Functions)

Μια συνάρτηση είναι ένα κομμάτι κώδικα που του έχει δοθεί ένα όνομα και μπορεί να χρησιμοποιηθεί από οπουδήποτε μέσα στο σχεδιάγραμμα. Για παράδειγμα ακολουθεί ο ορισμός της συνάρτησης setup():

```
void setup()
{
  pinMode(ledPin, OUTPUT);      // sets the digital pin
as output
}
```

Η πρώτη γραμμή προσφέρει πληροφορίες σχετικά με την συνάρτηση όπως το όνομα της. Οι λέξεις πριν και μετά το όνομα της συνάρτησης μας δείχνουν τον τύπο του αποτελέσματος της συνάρτησης και τις παραμέτρους που μπορεί να περνάμε σε αυτήν. Ο κώδικας μεταξύ των { } καλείται σώμα της συνάρτησης και δείχνει τι κάνει αυτή η συνάρτηση. Υπάρχουν συναρτήσεις οι οποίες είναι έτοιμες και παρέχονται από την ομάδα σχεδιασμού του Arduino αλλά υπάρχει και η δυνατότητα ο χρήστης της γλώσσας να δημιουργεί τις δικές του συναρτήσεις ανάλογα με το τι θέλει να υλοποιήσει μέσα σε μια εφαρμογή.

Χρήσιμες συναρτήσεις (Common Functions)

Μερικές χρήσιμες έτοιμες συναρτήσεις είναι οι ακόλουθες: pinMode(), digitalWrite(), και delay(). Η PinMode() ρυθμίζει αν ένας ακροδέκτης θα χρησιμοποιηθεί σαν είσοδος ή σαν έξοδος. Για να γίνει αυτό θα πρέπει η συνάρτηση να έχει ως παραμέτρους τον αριθμό του ακροδέκτη που θα χρησιμοποιηθεί καθώς επίσης και μια εκ των δύο σταθερών INPUT ή OUTPUT. Η συνάρτηση digitalWrite() δίνει σαν έξοδο σε έναν ακροδέκτη μια τιμή για παράδειγμα:

```
digitalWrite(ledPin, HIGH);
```

η οποία θέτει τον ακροδέκτη 13 του Arduino σε κατάσταση λογικού 1 οπότε αυτό θα παρέχει 5V.

Η συνάρτηση delay() προκαλεί στο Arduino μια καθυστέρηση για έναν συγκεκριμένο αριθμό milliseconds που έχει δηλωθεί σε αυτήν σαν παράμετρος.

```
delay(1000);
```

Το παραπάνω παράδειγμα προκαλεί αναμονή ενός δευτερολέπτου.

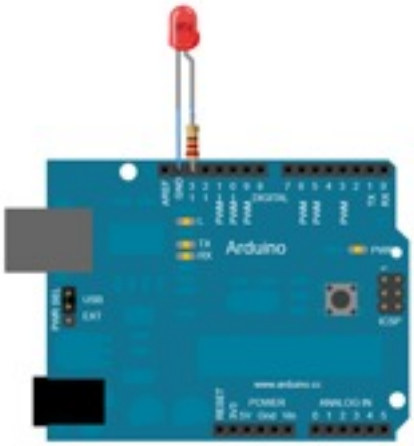
Συναρτήσεις setup() και loop() (Special Functions)

Η συνάρτηση setup() καλείται μία φορά όταν ξεκινάει το Arduino. Είναι ένα καλό σημείο για να κάνει κανείς αρχικοποίηση διαφόρων μεταβλητών και

ρυθμίσεων που θα χρειαστούν στον κώδικα που θα ακολουθήσει παρακάτω. Η συνάρτηση loop() είναι μια συνάρτηση που εκτελείται ξανά και ξανά και θεωρείται η καρδία της εφαρμογής. Για να λειτουργήσει σωστά ένα σχεδιάγραμμα πρέπει να περιλαμβάνει και τις δύο αυτές συναρτήσεις ακόμα και αν δεν υλοποιούν τίποτα.

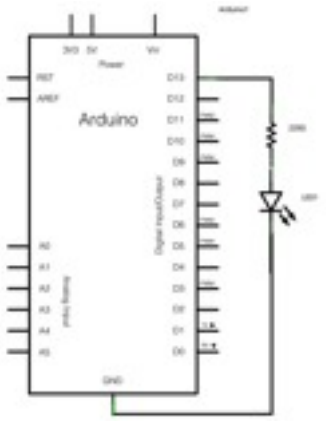
Παραδείγματα

Παρακάτω ακολουθεί ένα παράδειγμα σχεδιαγράμματος για το Arduino που περιλαμβάνει όλες τις παραπάνω βασικές έννοιες. Στο σχεδιάγραμμα αυτό υλοποιείται μια εφαρμογή που ανάβει και σβήνει ένα led κάθε ένα δευτερόλεπτο. Τα αντικείμενα που είναι απαραίτητα για την υλοποίηση του παραδείγματος είναι μια πλακέτα Arduino ένα led και μια αντίσταση.



Εικόνα 39: Ένα Arduino Uno συνδεδεμένο με ένα led.

Το κυκλωματικό διάγραμμα της συνδεσμολογίας είναι το ακόλουθο:



Εικόνα 40: Το κυκλωματικό διάγραμμα της συνδεσμολογίας.

Ο κώδικας που υλοποιεί την παραπάνω εφαρμογή είναι:

```
/*
  Blink
  Turns on an LED on for one second, then off for one second,
repeatedly.

  This example code is in the public domain.
 */

void setup() {
  // initialize the digital pin as an output.
  // Pin 13 has an LED connected on most Arduino boards:
  pinMode(13, OUTPUT);
}

void loop() {
  digitalWrite(13, HIGH);   // set the LED on
  delay(1000);              // wait for a second
  digitalWrite(13, LOW);    // set the LED off
  delay(1000);              // wait for a second
}
```

Εδώ παρατηρείται ότι έχει ορισθεί ο ακροδέκτης 13 για να χρησιμοποιηθεί ως έξοδος του κυκλώματος. Αυτό συμβαίνει μέσα στην συνάρτηση setup() η οποία όπως αναφέρθηκε τρέχει μια φορά όταν ξεκινάει το Arduino. Παρακάτω ορίζεται η συνάρτηση loop() στην οποία παρατηρείται ανάθεση στον ακροδέκτη 13 της τάση των 5V. Με την χρήση της συνάρτηση delay(1000) διατηρείται η τάση στον ακροδέκτη 13 για ένα δευτερόλεπτο. Με την συνάρτηση digitalWrite(13, LOW) αφαιρείται από τον ακροδέκτη η τάση 5V για ένα επίσης δευτερόλεπτο. Η συνάρτηση loop() τρέχει ξανά και ξανά για όση ώρα είναι σε λειτουργία η πλακέτα του Arduino.

Γλώσσα προγραμματισμού για κινητά Android

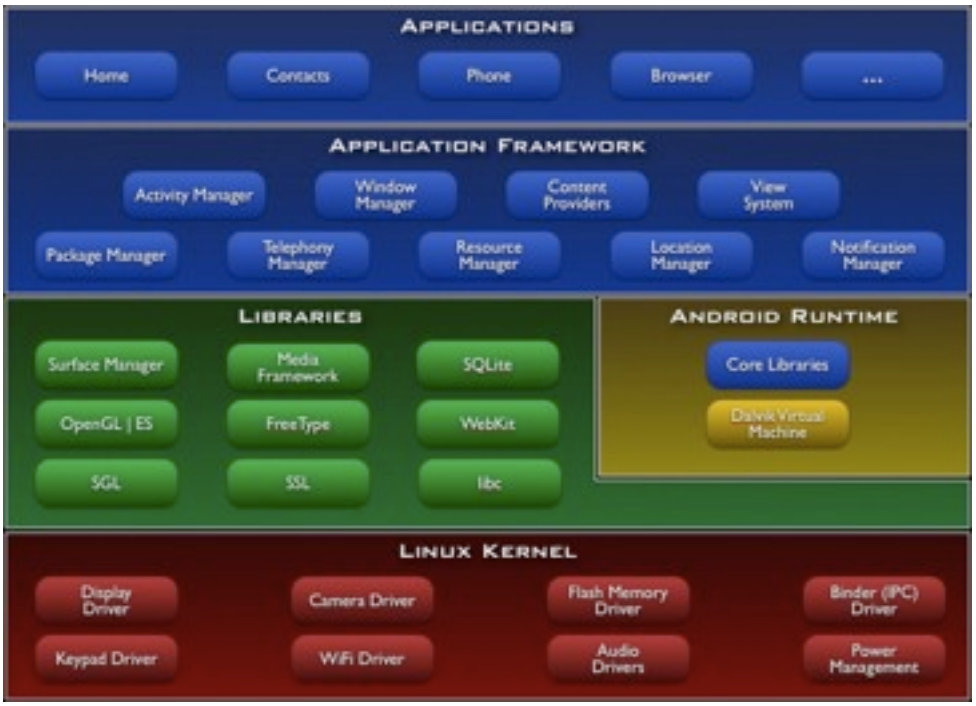
Το Android είναι μια αλυσίδα εργαλείων ανοικτού κώδικα για κινητές συσκευές που κατασκευάστηκε από την Google και την Open Handset Alliance. Τα κύρια σημεία που κάνουν το Android αρκετά σημαντικό είναι τα ακόλουθα:

- Είναι μια πραγματικά ανοικτή πλατφόρμα προγραμματισμού που βασίζεται στο Linux.
- Έχει αρχιτεκτονική βασισμένη σε στοιχεία που εμπνεύστηκε από το Internet, δηλαδή κομμάτια μια εφαρμογής μπορούν να χρησιμοποιηθούν με τρόπους που οι αρχικοί προγραμματιστές της δεν είχαν φανταστεί. Μπορεί ακόμα ο προγραμματιστής μιας εφαρμογής να αλλάξει έτοιμα ενσωματωμένα στοιχεία με δικές του βελτιωμένες εκδόσεις.
- Έχει πάρα πολλές έτοιμες υπηρεσίες όπως εντοπισμός θέσης με χρήση τριγωνοποίησης κυψελών ή με χρήση δέκτη GPS. Επίσης έχει πλήρη συνεργασία με βάσεις δεδομένων SQL.

- Παρέχει αυτοματοποιημένη διαχείριση του κύκλου ζωής μιας εφαρμογής. Τα προγράμματα είναι απομονωμένα το ένα από το άλλο από πολλαπλά επίπεδα ασφαλείας πράγμα που δεν έχει εφαρμοστεί ξανά σε πλατφόρμες έξυπνων συσκευών και προσδίδει μεγάλη αξιοπιστία.
- Υποστηρίζονται πολύ καλής ποιότητας γραφικά και ήχοι κάνοντας χρήση του προτύπου OpenGL και κωδικοποιητών ήχου και βίντεο.
- Τέλος υπάρχει μεγάλη φορητότητα καθώς υποστηρίζονται αρκετοί τύποι υλικού. Όλα τα προγράμματα είναι γραμμένα σε γλώσσα προγραμματισμού Java και εκτελούνται από την εικονική μηχανή του Android που ονομάζεται Dalvik. Επίσης υποστηρίζονται πολύ τύποι εισόδου όπως πληκτρολόγια, αφή, και οπτικός κέρσορας.

Έννοιες Κλειδιά

Κάθε επίπεδο χρησιμοποιεί υπηρεσίες που προσφέρονται από κατώτερα επίπεδα. Αρχίζοντας από το κατώτερο ακολουθεί ανάλυση των επιπέδων που παρέχονται από το Android.



Εικόνα 41: Διάγραμμα επιπέδων Android.

Πυρήνας Linux

Το Linux παρέχει το αφαιρετικό επίπεδο του υλικού για το Android, επιτρέποντας σε αυτό να μπορεί να φορτωθεί σε μια μεγάλη ποικιλία από πλατφόρμες. Εσωτερικά το Android χρησιμοποιεί το Linux για την διαχείριση

μνήμης του, την διαχείριση διεργασιών, την δικτύωση και άλλες υπηρεσίες λειτουργικού συστήματος.

Μητρικές Βιβλιοθήκες

Το επόμενο επίπεδο πάνω από τον πυρήνα είναι το επίπεδο από μητρικές βιβλιοθήκες του Android. Αυτές οι βιβλιοθήκες είναι γραμμένες σε C ή C++ και είναι μεταγλωττισμένες για την συγκεκριμένη αρχιτεκτονική υλικού που χρησιμοποιεί το κινητό τηλέφωνο. Κάποιες από τις πιο σημαντικές βιβλιοθήκες είναι:

- **Διαχειριστής Επιφάνειας (Surface Manager)**: Το Android χρησιμοποιεί έναν διαχειριστή σύνθεσης παραθύρων παρόμοιο με των Windows Vista αλλά είναι πιο απλός. Αντί να σχεδιάζει τις εντολές απευθείας στον προσωρινό αποθηκευτή της οθόνης, οι εντολές σχεδίασης πηγαίνουν σε έναν πίνακα bit εκτός οθόνης και μετά συνδυάζονται με άλλους πίνακες για να σχηματίσουν τελικά την οθόνη που βλέπει ο χρήστης.
- **Γραφικά δύο και τριών διαστάσεων**: Στοιχεία δύο και τριών διαστάσεων μπορούν να συνδυαστούν σε μια διεπαφή χρήστη στο Android. Η βιβλιοθήκη θα χρησιμοποιήσει υλικό 3D αν υπάρχει στην συσκευή ή αλλιώς θα χρησιμοποιηθεί πρόγραμμα γρήγορης σχεδίασης γραφικών.
- **Κωδικοποίηση Πολυμέσων**: Το Android μπορεί να αναπαραγάγει βίντεο και να εγγράψει και να αναπαραγάγει ηχητικά αποσπάσματα διαφόρων τύπων όπως AAC, AVC (H.264), H.263, MP3 και MPEG4.
- **Βάσεις Δεδομένων SQL**: Το Android περιλαμβάνει μιας ελαφριάς μορφής βάση δεδομένων με όνομα SQLite που χρησιμοποιείται επίσης και από τον φυλλομετρητή Firefox και από τα κινητά της εταιρίας Apple.
- **Μηχανή Φυλλομετρητή**: Για γρήγορη απεικόνιση περιεχομένου HTML το Android χρησιμοποιεί την βιβλιοθήκη WebKit. Η μηχανή αυτή χρησιμοποιείται επίσης και από τον φυλλομετρητή Chrome, από τον φυλλομετρητή Safari, από το iPhone και από την πλατφόρμα της Nokia S60.

Αυτές οι βιβλιοθήκες δεν είναι προγράμματα που λειτουργούν από μόνα τους. Υπάρχουν για να κληθούν από προγράμματα ανωτέρου επιπέδου.

Χρόνο εκτέλεσης Android

Πάνω από το επίπεδο του πυρήνα υπάρχει επίσης και το επίπεδο χρόνου εκτέλεσης Android που περιλαμβάνει την εικονική μηχανή Dalvik και τις βασικές βιβλιοθήκες Java. Η εικονική μηχανή Dalvik είναι υλοποίηση της Google για την Java που είναι βελτιστοποιημένη για φορητές συσκευές. Όλος ο κώδικας που γράφεται για το Android είναι γραμμένος σε Java και τρέχει σε αυτήν την εικονική μηχανή.

Πλαίσιο Εφαρμογής

Πάνω από το επίπεδο μητρικών βιβλιοθηκών και του χρόνου εκτέλεσης του Android βρίσκεται το επίπεδο πλαίσιο εφαρμογής. Αυτό το επίπεδο παρέχει τα μπλοκ κατασκευής υψηλού επιπέδου που χρησιμοποιούνται για να κατασκευαστούν οι εφαρμογές. Τα πιο σημαντικά κομμάτια του πλαισίου είναι τα ακόλουθα:

- **Διαχειριστής Δραστηριότητας**: Αυτός ο διαχειριστής ελέγχει τον κύκλο ζωής των εφαρμογών και διατηρεί μια στοίβα προς τα πίσω για την πλοήγηση του χρήστη.
- **Πάροχοι Περιεχομένου**: Αυτά τα αντικείμενα ενσωματώνουν δεδομένα που χρειάζεται να διαμοιράζονται μεταξύ των εφαρμογών, όπως παραδείγματος χάρη οι Επαφές.
- **Διαχειριστής Πόρων**: Οι Πόροι είναι οτιδήποτε βρίσκεται στην εφαρμογή που δεν είναι κώδικας.
- **Διαχειριστής Τοποθεσίας**: Ένα κινητό τηλέφωνο Android πάντα γνωρίζει που βρίσκεται!
- **Διαχειριστής Ειδοποιήσεων**: Γεγονότα όπως η άφιξη μηνύματος, ραντεβού, ειδοποιήσεις και πολλά άλλα μπορούν να παρουσιάζονται στον χρήστη.

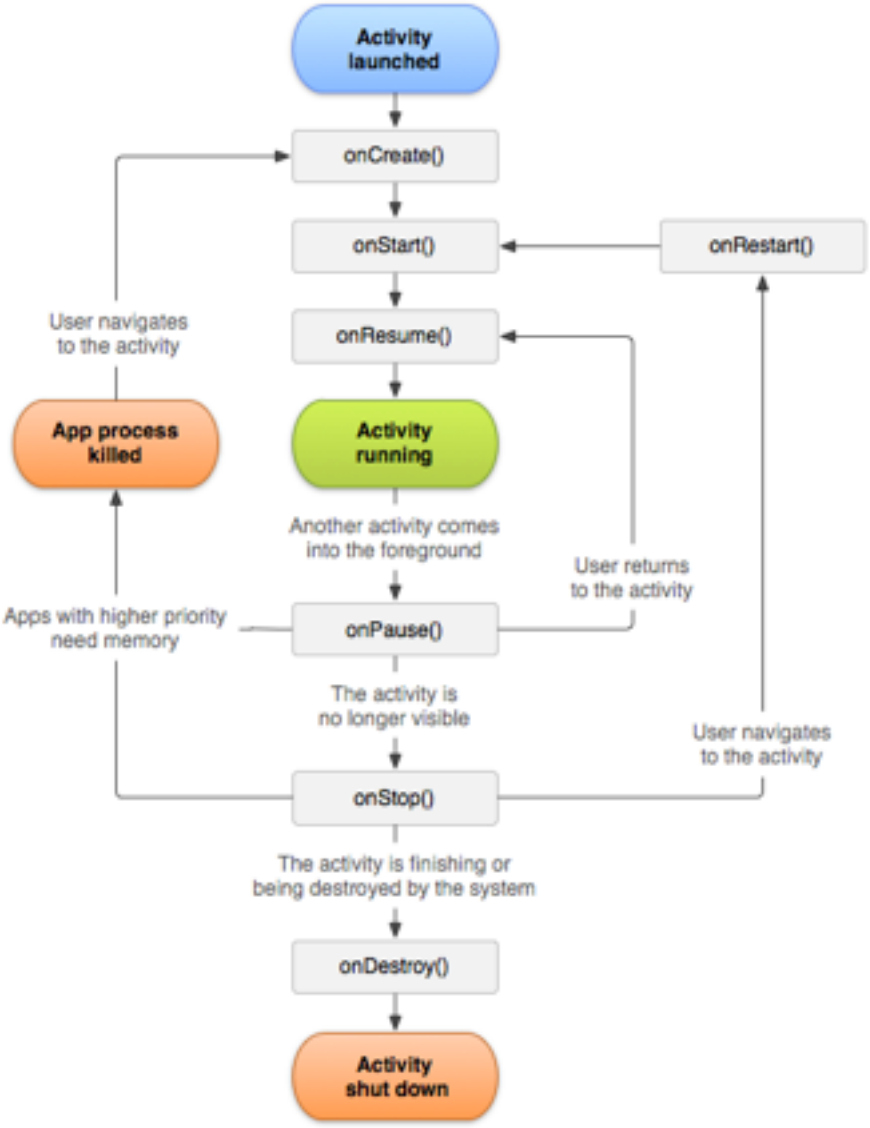
Η διεργασία δεν είναι εφαρμογή

Εσωτερικά κάθε οθόνη διεπαφής χρήστη αναπαριστάται από μια κλάση Δραστηριότητα (Activity). Κάθε δραστηριότητα έχει τον δικό της κύκλο ζωής. Μια εφαρμογή είναι μια ή περισσότερες δραστηριότητες με επιπλέον μια διεργασία Linux που τις περιέχει.

Στο Android μια εφαρμογή μπορεί να είναι ¨ζωντανή¨ ακόμα και αν η διεργασία της έχει ¨σκοτωθεί¨. Κατά μια άλλη έννοια ο κύκλος ζωής μιας δραστηριότητας δεν είναι δεμένος με τον κύκλο ζωής μιας διεργασίας. Οι διεργασίες είναι αναλώσιμα ¨δοχεία¨ για τις δραστηριότητες.

Κύκλος Ζωής

Κατά την διάρκεια ζωής μιας δραστηριότητας μιας εφαρμογής Android μπορεί αυτή να βρίσκεται σε μια από αρκετές καταστάσεις, όπως φαίνεται στην εικόνα που ακολουθεί. Ο προγραμματιστής δεν έχει έλεγχο σε ποια κατάσταση θα βρίσκεται το πρόγραμμα. Αυτό διαχειρίζεται από το σύστημα. Παρόλα αυτά ο προγραμματιστής μπορεί να ειδοποιηθεί όταν μια κατάσταση είναι έτοιμη να αλλάξει με χρήση της μεθόδου onXX().



Εικόνα 42: Διάγραμμα κύκλου ζωής δραστηριότητας Android.

Ο προγραμματιστής μπορεί να παρακάμψει αυτές τις μεθόδους στην κλάση Δραστηριότητα και το Android θα τις καλέσει την κατάλληλη στιγμή:

- **onCreate(Bundle)**: Αυτή η μέθοδος καλείται όταν η δραστηριότητα ξεκινάει. Μπορεί να χρησιμοποιηθεί για να εκτελέσει αρχικοποίηση όπως να δημιουργήσει το γραφικό περιβάλλον του χρήστη. Η μέθοδος onCreate() παίρνει για παράμετρο είτε την τιμή null ή μια πληροφορία κατάστασης που έχει αποθηκευτεί προηγουμένως από την μέθοδο onSaveInstanceState().
- **onStart()**: Αυτή η μέθοδος δείχνει ότι η δραστηριότητα είναι έτοιμη να απεικονιστεί στον χρήστη.
- **onResume()**: Αυτή η μέθοδος καλείται όταν η δραστηριότητα μπορεί να αλληλεπιδράσει με τον χρήστη. Αυτό είναι ένα καλό μέρος για να ξεκινάνε τυχόν κινούμενα γραφικά ή μουσική.

- **onPause()**: Αυτή η μέθοδος τρέχει όταν η δραστηριότητα είναι έτοιμη να πάει στο παρασκήνιο, συνήθως επειδή μια άλλη δραστηριότητα έχει εκκινήσει μπροστά από την τρέχουσα.
- **onStop()**: Η μέθοδος αυτή καλείται όταν η δραστηριότητα δεν είναι πλέον ορατή στον χρήστη και δεν χρειάζεται για λίγο χρονικό διάστημα.
- **onRestart()**: Αν κληθεί αυτή η μέθοδος σηματοδοτεί ότι η δραστηριότητα είναι επανασχεδιασμένη στον χρήστη από την κατάσταση stop.
- **onDestroy()**: Αυτή η μέθοδος καλείται αμέσως πριν την καταστροφή της δραστηριότητας.
- **onSaveInstanceState(Bundle)**: Το Android καλεί αυτή τη μέθοδο για να επιτρέψει στη δραστηριότητα να αποθηκεύσει κατάσταση ανά περίσταση, όπως η θέση του κέρσορα μέσα σε ένα πεδίο κειμένου.
- **onRestoreInstanceState(Bundle)**: Αυτή η μέθοδος καλείται όταν μια δραστηριότητα ξανά αρχικοποιείται από μια κατάσταση που είχε αποθηκευτεί προηγουμένως από την μέθοδο onSaveInstanceState().

Δραστηριότητες που δεν εκτελούνται στο προσκήνιο μπορεί να έχουν σταματήσει, ή οι διεργασίες του Linux που τις κρατάνε να τις έχουν τερματίσει για να ελευθερώσουν χώρο για νέες δραστηριότητες. Επιπρόσθετα με την διαχείριση του κύκλου ζωής μιας εφαρμογής το πλαίσιο του Android παρέχει έναν αριθμό μπλοκ κατασκευής που χρησιμοποιούνται για την κατασκευή της εφαρμογής.

Μπλοκ Κατασκευής

Δραστηριότητες

Μια δραστηριότητα είναι μια οθόνη γραφικού περιβάλλοντος για τον χρήστη. Οι εφαρμογές μπορούν να ορίζουν μια η περισσότερες δραστηριότητες για να διαχειρίζονται διάφορες φάσεις ενός προγράμματος. Κάθε δραστηριότητα είναι υπεύθυνη να αποθηκεύει την κατάσταση της για να μπορεί να ανακληθεί αργότερα σαν κομμάτι του κύκλου ζωής της εφαρμογής. Οι δραστηριότητες επεκτείνουν την κλάση Περιεχόμενο (Context), έτσι ώστε να μπορούν να την χρησιμοποιούν για να λαμβάνουν γενικές πληροφορίες για την εφαρμογή.

Προθέσεις

Μια πρόθεση είναι ένας μηχανισμός για την περιγραφή μιας συγκεκριμένης δράσης όπως η επιλογή μιας φωτογραφίας. Στο Android σχεδόν τα πάντα περνάνε από τις προθέσεις και έτσι υπάρχουν αρκετές ευκαιρίες αντικατάστασης ή χρήσης στοιχείων. Για παράδειγμα υπάρχει μια πρόθεση για την αποστολή ενός μηνύματος ηλεκτρονικού ταχυδρομείου. Αν η εφαρμογή χρειάζεται να στείλει ένα τέτοιο μήνυμα μπορεί να επικαλεσθεί αυτή την πρόθεση.

Υπηρεσίες

Μια υπηρεσία είναι μια ενέργεια που εκτελείται στο παρασκήνιο χωρίς να χρειάζεται κάποια αλληλεπίδραση από τον χρήστη. Το Android έχει πολλές ενσωματωμένες υπηρεσίες που μπορούν εύκολα να χρησιμοποιηθούν.

Πάροχοι Περιεχομένου

Ένας πάροχος περιεχομένου είναι μια ομάδα από δεδομένα που βρίσκονται σε μια παραμετροποιημένη διεπαφή προγράμματος εφαρμογής (API) που υπάρχει δυνατότητα να διαβαστεί και να εγγραφθεί. Για παράδειγμα η Google παρέχει έναν πάροχο περιεχομένου για τις επαφές. Όλες οι πληροφορίες σχετικά με τις επαφές όπως τα ονόματα, οι διευθύνσεις, τα τηλέφωνα και άλλα μπορούν να διαμοιραστούν σε οποιαδήποτε εφαρμογή θέλει να τα χρησιμοποιήσει.

Ασφάλεια

Στο Android κάθε εφαρμογή εκτελεί την δικιά της διεργασία Linux. Το ίδιο το υλικό απαγορεύει μια διεργασία να αποκτήσει πρόσβαση στην μνήμη μιας άλλης διεργασίας. Επιπροσθέτως σε κάθε εφαρμογή έχει ανατεθεί ένα συγκεκριμένο και μοναδικό αναγνωριστικό. Ότι αρχεία δημιουργεί μια εφαρμογή δεν μπορούν να γραφούν ή να διαβαστούν από καμία άλλη εφαρμογή. Επιπλέον πρόσβαση σε συγκεκριμένες κρίσιμες λειτουργίες είναι απαγορευμένη και πρέπει να ζητηθεί άδεια από τον χρήστη ώστε μια εφαρμογή να μπορεί να τις χρησιμοποιήσει. Αυτό γίνεται μέσω του αρχείου AndroidManifest.xml. Μερικές άδειες που ζητούνται συχνά είναι οι παρακάτω:

- **INTERNET**: Πρόσβαση στο Internet.
- **READ_CONTACTS**: Άδεια για να μπορούν να διαβαστούν οι επαφές του χρήστη.
- **WRITE_CONTACTS**: Άδεια για να μπορούν να εγγραφούν οι επαφές του χρήστη.
- **RECEIVE_SMS**: Άδεια για να μπορούν να παρακολουθηθούν τα εισερχόμενα μηνύματα του χρήστη.

Παράδειγμα Εφαρμογής Android

```
package com.example.helloandroid;

import android.app.Activity;
import android.os.Bundle;
import android.widget.TextView;

public class HelloAndroid extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        TextView tv = new TextView(this);
        tv.setText("Hello, Android");
        setContentView(tv);
    }
}
```

Στην ανωτέρω εφαρμογή παρατηρείται μια δραστηριότητα όπου εκτυπώνει στην οθόνη του χρήστη το κείμενο Hello, Android. Αναλυτικά έχουν εισαχθεί τρείς βασικές βιβλιοθήκες και μετά έχει δημιουργηθεί μια κλάση όπου επεκτείνει την βασική λειτουργία της δραστηριότητας έτσι ώστε όταν αυτή ξεκινήσει να εκτυπωθεί στην οθόνη του χρήστη το επιθυμητό κείμενο. Η σχεδίαση (οπτική) της δραστηριότητας μπορεί να γίνει είτε από τον βασικό κώδικα της εφαρμογής, όπως γίνεται παραπάνω με την χρήση του TextView είτε από ένα ξεχωριστό αρχείο σχεδίασης του γραφικού περιβάλλοντος που είναι γραμμένο σε γλώσσα xml το οποίο μετά θα κληθεί από την δραστηριότητα [13].

Γλώσσες Προγραμματισμού Διαδικτυακών Εφαρμογών

Για την ανάπτυξη διαδικτυακών εφαρμογών έχουν δημιουργηθεί αρκετές γλώσσες προγραμματισμού όπου η κάθε μια έχει τα θετικά και τα αρνητικά της στοιχεία. Οι γλώσσες αυτές χωρίζονται σε κατηγορίες ανάλογα με το αν απευθύνονται στον φυλλομετρητή (πελάτη) ή τον διακομιστή (server). Παρακάτω θα ακολουθήσει μια ανάλυση για τις πιο διαδεδομένες γλώσσες προγραμματισμού διαδικτυακών εφαρμογών που υπάρχουν σήμερα. Ιδιαίτερη έμφαση δόθηκε στο γεγονός ότι γλώσσες που επιλέχτηκαν είναι όλες φιλοσοφίας ανοικτού κώδικα.

Γλώσσα HTML

Η γλώσσα HTML είναι μια γλώσσα που περιγράφει τις ιστοσελίδες. Τα αρχικά HTML αντιπροσωπεύουν τις λέξεις Hyper Text Markup Language. Η γλώσσα HTML δεν είναι γλώσσα προγραμματισμού αλλά γλώσσα σήμανσης, δηλαδή περιλαμβάνει διάφορες ετικέτες σήμανσης για την περιγραφή και σχεδίαση του περιεχομένου των ιστοσελίδων [14].

Οι ετικέτες σήμανσης συχνά αποκαλούνται και ετικέτες HTML. Οι ετικέτες αυτές είναι κάποιες λέξεις κλειδιά που βρίσκονται μεταξύ < >, για παράδειγμα <html> και συνήθως συναντώνται σε ζευγάρια όπως . Το σηματοδοτεί την αρχή της ετικέτας και το το τέλος της.
Τα κείμενα HTML περιγράφουν τις ιστοσελίδες. Περιλαμβάνουν μια σειρά από ετικέτες HTML και απλού κειμένου τα οποία διαβάζονται από τον φυλλομετρητή του χρήστη και αυτός τα απεικονίζει ως ιστοσελίδες. Οι ετικέτες HTML δεν εμφανίζονται μέσα στις ιστοσελίδες.

Ένα παράδειγμα κειμένου HTML είναι το παρακάτω:

```
<html>
<body>

<h1>My First Heading</h1>

<p>My first paragraph.</p>

</body>
</html>
```

Η πρώτη ετικέτα δείχνει ότι πρόκειται για μια ιστοσελίδα και η γλώσσα που έχει χρησιμοποιηθεί είναι η HTML. Μετά ακολουθεί το σώμα της ιστοσελίδας που είναι αυτό που θα απεικονιστεί στον χρήστη από τον φυλλομετρητή του. Η ετικέτα <h1> σημαίνει ότι το κείμενο που ακολουθεί είναι επικεφαλίδα τύπου 1 και εμφανίζεται από τον φυλλομετρητή με μεγαλύτερη γραμματοσειρά και με έντονη σχεδίαση. Η ετικέτα σημαίνει ότι αυτό που ακολουθεί είναι μια παράγραφος και εμφανίζεται κάτω από την επικεφαλίδα.

Η σύνταξη της HTML έχει λοιπόν μερικούς κανόνες. Κάθε στοιχείο της HTML ξεκινάει με μια ετικέτα ανοίγματος και τελειώνει με μια ετικέτα κλεισίματος. Το περιεχόμενο του στοιχείου μπορεί να είναι οτιδήποτε βρίσκεται ανάμεσα σε μια ετικέτα ανοίγματος και κλεισίματος. Βέβαια υπάρχουν και στοιχεία HTML που δεν περιέχουν περιεχόμενο ανάμεσα στις ετικέτες τους. Αυτά τα στοιχεία ανοίγουν και κλείνουν στην αρχική ετικέτα όπως το στοιχείο που μας οδηγεί στη επόμενη γραμμή </br>. Τα περισσότερα στοιχεία HTML μπορούν να χαρακτηρίζονται από κάποιες ιδιότητες (attributes). Οι ιδιότητες παρέχουν περισσότερες πληροφορίες σχετικά με τα στοιχεία. Τοποθετούνται πάντα στην ετικέτα ανοίγματος ενός στοιχείου HTML και ποτέ στην ετικέτα κλεισίματος. Η μορφή μιας ιδιότητας ακολουθεί την δομή ζεύγους ονόματος-τιμή, όπως για παράδειγμα name=”value”. Μερικές ιδιότητες φαίνονται στον πίνακα που ακολουθεί.

Πίνακας 11: Ιδιότητες HTML.

Ιδιότητα	Τιμή	Περιγραφή
class	όνομα κλασης	Δηλώνει το όνομα κλάσης ενός στοιχείου
id	όνομα αναγνωριστικού	Δηλώνει το όνομα αναγνωριστικού ενός στοιχείου
style	ορισμός μορφοποίησης	Περιγράφει την εσωτερική μορφοποίηση του στοιχείου
title	όνομα τίτλου ιστοσελίδας	Ορίζει επιπλέον πληροφορίες για ένα στοιχείο και εμφανίζεται στο παράθυρο του φυλλομετρητή

Γλώσσα μορφοποίησης CSS

Η γλώσσα CSS είναι μια γλώσσα μορφοποίησης. Τα αρχικά CSS σημαίνουν Cascading Style Sheet. Η γλώσσα αυτή ορίζει μορφοποιήσεις που δηλώνουν πώς θα απεικονιστεί ένα στοιχείο HTML. Οι μορφοποιήσεις αυτές ενσωματώθηκαν στην έκδοση HTML 4.0. Εξωτερικά φύλλα μορφοποιήσεων μπορούν να αποθηκευτούν σε αρχεία CSS. Η γλώσσα μορφοποιήσεων CSS δημιουργήθηκε για να λύσει ένα μεγάλο πρόβλημα που παρουσίαζε η γλώσσα HTML. Η γλώσσα HTML δεν σχεδιάστηκε ποτέ για να περιέχει και ετικέτες που περιγράφουν μορφοποίηση των περιεχομένων των στοιχείων. Όταν αυτό άρχισε να γίνεται με την χρήση της ετικέτας <font> και διάφορων ιδιοτήτων μορφοποίησης στην έκδοση HTML 3.2 η διαχείριση και επόπτευση μεγάλων ιστοσελίδων έγινε πολύ δύσκολη. Για αυτό το λόγο δημιουργήθηκε η CSS. Στην έκδοση HTML 4.0 όλη η μορφοποίηση των στοιχείων αφαιρέθηκε από την HTML και άρχισε να γίνεται μέσω εξωτερικών αρχείων CSS που συνδέονταν με την HTML. Οπότε η CSS ορίζει πώς τα στοιχεία της HTML θα πρέπει να απεικονιστούν και έτσι η διαμόρφωση της απεικόνισης μπορεί να γίνει με επεξεργασία μόνο ενός εξωτερικού αρχείου CSS [15].

Σύνταξη CSS

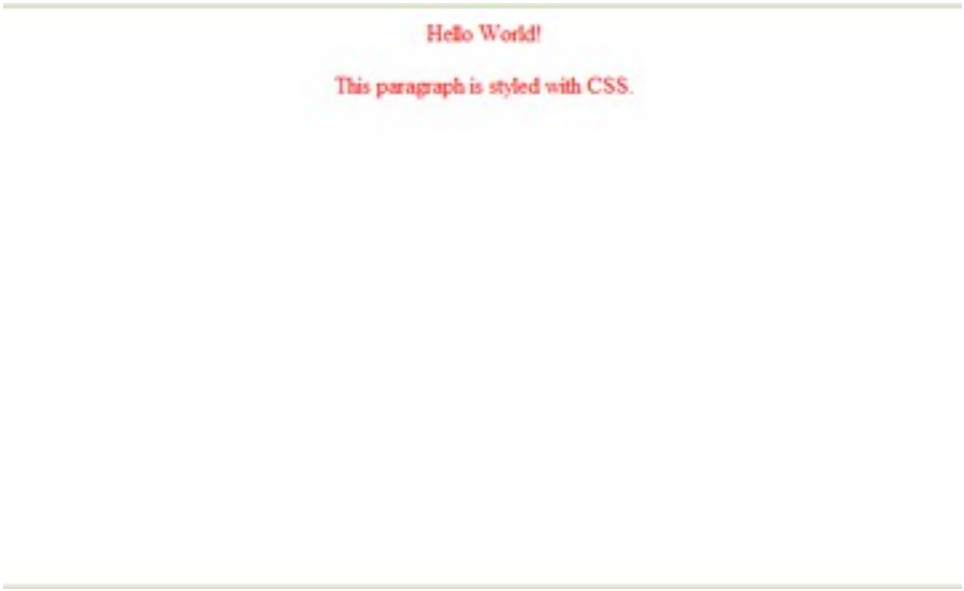
Ένας κανόνας CSS αποτελείται από δύο κυρίως μέρη, έναν επιλογέα και μια ή περισσότερες δηλώσεις. Ο επιλογέας συνήθως είναι το στοιχείο HTML που επιθυμείται να μορφοποιηθεί. Κάθε δήλωση περιλαμβάνει ένα ζεύγος από όνομα και τιμή. Το όνομα είναι η ιδιότητα μορφοποίησης που επιθυμείται να τροποποιηθεί και η τιμή είναι η τιμή που επιθυμείται να έχει αυτή η ιδιότητα.

Εικόνα 43: Ανάλυση χρήσης γλώσσας CSS .

Ένα παράδειγμα για την χρήση της CSS είναι το ακόλουθο:

```
<html>
<head>
<style type="text/css">
p
{
color:red;
text-align:center;
}
</style>
</head>

<body>
<p>Hello World!</p>
<p>This paragraph is styled with CSS.</p>
</body>
</html>
```

Αυτό το κομμάτι κώδικα έχει σαν αποτέλεσμα:



Εικόνα 44: Αποτέλεσμα χρήσης CSS.

Επιπρόσθετα με την παραμετροποίηση της μορφοποίησης ενός στοιχείου HTML, η CSS επιτρέπει και την χρήση συγκεκριμένων επιλογέων του χρήστη που ονομάζονται id και class. Ο επιλογέας id χρησιμοποιείται για την μορφοποίηση ενός συγκεκριμένου και μοναδικού στοιχείου HTML. Αυτός ο επιλογέας χρησιμοποιεί την ιδιότητα id του στοιχείου HTML και ορίζεται με ένα #. Ακολουθεί ένα παράδειγμα:

```
#para1
{
text-align:center;
color:red;
}
```

Σε αυτό το παράδειγμα σαν επιλογέας έχει χρησιμοποιηθεί η ιδιότητα id=”para1” σε ένα στοιχείο HTML.

Ένας άλλος επιλογέας CSS είναι ο class. Ο επιλογέας αυτός χρησιμοποιείται για να μορφοποιήσει μια ομάδα από στοιχεία της HTML. Αντίθετα με τον επιλογέα id ο επιλογέας class μπορεί να χρησιμοποιηθεί σε πολλαπλά στοιχεία HTML. Αυτό επιτρέπει να χρήση της ίδιας μορφοποίησης σε πολλά στοιχεία που έχουν την ίδια ιδιότητα class. Για την επιλογή αυτού του επιλογέα στην CSS χρησιμοποιείται το σύμβολο “.”. Ακολουθεί ένα παράδειγμα για την χρήση του επιλογέα.

```
.center {text-align:center;}
```

Σε αυτό το παράδειγμα σαν επιλογέας έχει χρησιμοποιηθεί η ιδιότητα class=”center” σε ένα στοιχείο HTML.

Υπάρχουν τρείς τρόποι για την εισαγωγή της CSS σε ένα κείμενο HTML. Ο ένας τρόπος είναι με σύνδεση ενός εξωτερικού αρχείου που περιέχει όλους τους κανόνες μορφοποίησης για τα στοιχεία της HTML. Ο άλλος τρόπος είναι να ενσωμάτωση των κανόνων μορφοποίησης της CSS στην ετικέτα επικεφαλίδας του κειμένου HTML και τέλος ο τρίτος τρόπος είναι να εισαγωγή των κανόνων CSS σαν ιδιότητες στην ετικέτα του στοιχείου HTML που επιθυμείται να μορφοποιήθεί. Ακολουθούν παραδείγματα για τον κάθε τρόπο εισαγωγής της CSS σε κείμενο HTML.

- Εξωτερική σύνδεση:

```
<head>
<link rel="stylesheet" type="text/css" href="mystyle.css" />
</head>
```

- Ενσωμάτωση στο κείμενο HTML:

```
<head>
<style type="text/css">
hr {color:sienna;}
p {margin-left:20px;}
body {background-image:url("images/back40.gif");}
</style>
</head>
```

- Ενσωμάτωση σαν ιδιότητα στο στοιχείο HTML που μας ενδιαφέρει:

```
<p style="color:sienna;margin-left:20px">This is a paragraph.</p>
```

Γλώσσα Προγραμματισμού PHP και MySQL

Η γλώσσα PHP είναι ένα ισχυρό εργαλείο για την κατασκευή δυναμικών και διαδραστικών ιστοσελίδων. Η PHP είναι μια γλώσσα που εκτελείται από την πλευρά του διακομιστή σαν την ASP της Microsoft αλλά η PHP είναι φιλοσοφίας ανοικτού λογισμικού. Η PHP υποστηρίζει πολλές βάσεις δεδομένων όπως η MySQL, η Oracle, η PostgreSQL και άλλες. Τα αρχεία PHP μπορούν να περιλαμβάνουν κείμενο, ετικέτες HTML και εκτελέσιμο κώδικα. Τα αρχεία αυτά επιστρέφουν στον φυλλομετρητή σαν απλό HTML άρα η PHP παράγει HTML.

Η PHP μπορεί και εκτελείται στα περισσότερα λειτουργικά συστήματα όπως τα Windows, Linux, Unix και σε πολλά ακόμα. Είναι συμβατή με τους πιο συχνά χρησιμοποιούμενους διακομιστές όπως ο Apache και ο IIS της Microsoft [16].

Σύνταξη PHP

Ο κώδικας της PHP αρχίζει πάντα με την εισαγωγή του <?php και τελειώνει με ?>. Μέσα σε αυτές τις ετικέτες περιλαμβάνεται όλος ο κώδικας της PHP. Κάθε εντολή της PHP πρέπει να τελειώνει με “;” το οποίο καλείται διαχωριστής και διαχωρίζει την μια εντολή από την επόμενη της. Ακολουθεί ένα παράδειγμα κώδικα PHP το οποίο βρίσκεται ανάμεσα σε κάποιες ετικέτες κειμένου HTML.

```
<html>
<body>

<?php
echo "Hello World";
?>

</body>
</html>
```

Η εντολή echo θα προκαλέσει την εκτύπωση του συνόλου των χαρακτήρων που βρίσκονται μέσα στα εισαγωγικά.

Η δήλωση μεταβλητών στην PHP γίνεται με την εισαγωγή του συμβόλου $ πριν το όνομα της μεταβλητής, για παράδειγμα $variable = value;. Στην PHP δεν χρειάζεται να δηλώνονται οι μεταβλητές πριν την ανάθεση μιας τιμής σε αυτές, όπως επίσης και δεν χρειάζεται να δηλώνεται ο τύπος των μεταβλητών. Η PHP αυτόματα αποφασίζει τον τύπο μιας μεταβλητής ανάλογα με την τιμή που της έχει ανατεθεί.

```
<?php
$txt="Hello World";
echo $txt;
?>
```

Στο ανωτέρω παράδειγμα γίνεται δημιουργία μιας μεταβλητής με όνομα txt και ταυτόχρονα πραγματοποιείται και η ανάθεση της τιμής της. Έπειτα ακολουθεί η εντολή εκτύπωσης όπου θα εκτυπώσει το περιεχόμενο της μεταβλητής δηλαδή Hello, World.

Γενικά στην γλώσσα PHP ισχύουν όλοι οι τελεστές και οι εντολές εκτέλεσης υπό συνθήκη που υπάρχουν και σε άλλες γλώσσες όπως η C. Επίσης υπάρχουν και

πάρα πολλές έτοιμες συναρτήσεις αλλά μπορεί και ο προγραμματιστής να γράψει τις δικές του.

Η MySQL είναι ένας διακομιστής βάσης δεδομένων, που είναι κατάλληλος για μεγάλες και μικρές εφαρμογές. Η MySQL υποστηρίζει την γλώσσα ερωτημάτων βάσεων δεδομένων SQL. Μπορεί να λειτουργήσει σε αρκετές πλατφόρμες και είναι της φιλοσοφίας ανοικτού λογισμικού.

Τα δεδομένα στην MySQL αποθηκεύονται σε αντικείμενα βάσης δεδομένων που ονομάζονται πίνακες. Ένας πίνακας είναι μια συλλογή από σχετικά δεδομένα και αποτελείται από γραμμές και στήλες.

Πίνακες Βάσης Δεδομένων

Μια βάση δεδομένων συχνά περιλαμβάνει έναν ή περισσοτέρους πίνακες. Κάθε πίνακας αναγνωρίζεται από ένα όνομα. Οι πίνακες περιλαμβάνουν εγγραφές με δεδομένα. Ακολουθεί ένα παράδειγμα πίνακα βάσης δεδομένων με όνομα Persons.

Πίνακας 12: Παράδειγμα Πίνακα MySQL.

LastName	FirstName	Address	City
Hansen	Ola	Timoteivn 10	Sandnes
Svendson	Tove	Borgvn 23	Sandnes
Pettersen	Kari	Storgt 20	Stavanger

Ο παραπάνω πίνακας περιλαμβάνει τρεις εγγραφές και τέσσερις στήλες.

Ερωτήματα MySQL

Στην MySQL υπάρχει δυνατότητα δημιουργίας ερωτημάτων προς την βάση δεδομένων για μια συγκεκριμένη πληροφορία και να δυνατότητα λήψης ως απάντηση μια ομάδα από εγγραφές. Μια ερώτηση συντάσσεται όπως φαίνεται παρακάτω.

```
SELECT LastName FROM Persons
```

Στην παραπάνω ερώτηση παρατηρείται η εντολή εξόρυξης από τον πίνακα με όνομα Persons των αποτελεσμάτων που σχετίζονται μόνο με το πεδίο που αφορά το LastName δηλαδή το επίθετο. Ο πίνακας που παράγεται ως αποτέλεσμα θα είναι ο παρακάτω.

Πίνακας 13: Παράδειγμα αποτελέσματος σε ερώτημα MySQL.

LastName
Hansen
Svendson
Pettersen

Δημιουργία σύνδεσης PHP-MySQL

Προτού εγκατασταθεί πρόσβαση στην βάση δεδομένων μια εφαρμογής πρέπει πρώτα να δημιουργηθεί μια σύνδεση σε αυτή. Στην PHP αυτό γίνεται με την εντολή mysql_connect(severname, username, password); Η συνάρτηση αυτή παίρνει σαν παραμέτρους το όμονα του διακομιστή που φιλοξενεί την βάση δεδομένων, το όνομα του χρήστη που την διαχειρίζεται και τον κωδικό πρόσβασης σε αυτή. Ένα παράδειγμα σύνδεσης με την βάση δεδομένων ακολουθεί.

```
<?php
$con = mysql_connect("localhost","peter","abc123");
if (!$con)
  {
  die('Could not connect: ' . mysql_error());
  }

// some code

mysql_close($con);
?>
```

Στο ανωτέρω απόσπασμα παρατηρείται η σύνδεση σε μια βάση δεδομένων που υπάρχει στον διακομιστή ‘localhost’ του χρήστη ‘peter’ με κωδικό ‘abc123’. Αυτή η συνάρτηση ανατίθεται σε μια μεταβλητή. Αν συμβεί κάποιο πρόβλημα και δεν υπάρχει επιτυχής σύνδεση στην βάση, αυτό προβλέπεται με την δεύτερη εντολή που ελέγχει αν η μεταβλητή $con είναι ‘false’. Αν είναι ‘false’ εκτυπώνεται αυτό που βρίσκεται μέσα στη παρένθεση της συνάρτησης die που εκτός από την εκτύπωση του κειμένου προκαλεί και τερματισμό των διεργασιών. Επίσης η σύνταξη ‘.mysql_error()’ προκαλεί ένωση αλφαριθμητικών λόγω της ¨.¨ οπότε αυτό που θα εκτυπωθεί τελικά είναι ‘Could not connect και το λάθος που έχει προκύψει’. Τέλος η εντολή mysql_close($con); προκαλεί απόλυση της σύνδεσης με την βάση δεδομένων.

Εισαγωγή Δεδομένων στην Βάση Δεδομένων

Για την εισαγωγή δεδομένων σε μια βάση πρέπει να υπάρχει μια ανοιχτή σύνδεση με τον διακομιστή της βάσης και θα πρέπει να είναι συνδεδεμένος και ένας υπεύθυνος χρήστης. Για την εισαγωγή πρέπει να επιλεγεί ένας συγκεκριμένος πίνακας και τα πεδία αυτού του πίνακα που θα συμπληρωθούν.

```
<?php
$con = mysql_connect("localhost","peter","abc123");
if (!$con)
  {
  die('Could not connect: ' . mysql_error());
  }

mysql_select_db("my_db", $con);

mysql_query("INSERT INTO Persons (FirstName, LastName, Age)
VALUES ('Peter', 'Griffin', '35')");

mysql_query("INSERT INTO Persons (FirstName, LastName, Age)
VALUES ('Glenn', 'Quagmire', '33')");

mysql_close($con);
?>
```

Στο ανωτέρω παράδειγμα υλοποιείται σύνδεση, επιλογή της βάσης και εισαγωγή στον επιθυμητό πίνακα τα δεδομένα στα αντίστοιχα πεδία του. Τέλος υλοποιείται απόλυση της σύνδεσης με τον διακομιστή της βάσης.

Απεικόνιση των αποτελεσμάτων

```
<?php
$con = mysql_connect("localhost","peter","abc123");
if (!$con)
  {
  die('Could not connect: ' . mysql_error());
  }

mysql_select_db("my_db", $con);

$result = mysql_query("SELECT * FROM Persons");

while($row = mysql_fetch_array($result))
  {
  echo $row['FirstName'] . " " . $row['LastName'];
  echo "<br />";
  }

mysql_close($con);
?>
```

Στο παραπάνω απόσπασμα κώδικα υλοποιείται η σύνδεση σε μια βάση δεδομένων, η επιλογή της βάσης με την εντολή mysql_select_db(“my_db”, $con);, η δημιουργία μιας μεταβλητής που επιλέγει από τον πίνακα με όνομα Persons όλα τα πεδία του και μετά καλείται η εκτύπωση των δεδομένων που αντιστοιχούν σε αυτά τα πεδία με τον επαναλαμβανόμενο κώδικα:

```
while($row = mysql_fetch_array($result))
  {
  echo $row['FirstName'] . " " . $row['LastName'];
  echo "<br />";
  }
```

Αυτό το κομμάτι εκτελείται για όσο υπάρχουν εγγραφές στον πίνακα και με χρήση της echo εκτυπώνει τα δεδομένα που υπάρχουν κάτω από την στήλη FirstName και LastName σαν ζεύγη που χωρίζονται με ένα κενό. Επίσης με το echo “</br>”; προκαλείται και αλλαγή γραμμής μετά από κάθε αποτέλεσμα. Έτσι προκύπτει αποτέλεσμα παρόμοιο αυτής της μορφής που ακολουθεί.

```
Peter Griffin
Glenn Quagmire
```

Γλώσσα Προγραμματισμού JavaScript

Η JavaScript είναι η απόλυτη γλώσσα εκτελέσιμου κώδικα στο Διαδίκτυο. Χρησιμοποιείται από εκατομμύρια ιστοσελίδες και προσθέτει χρηστικότητα, επικύρωση φορμών, επικοινωνία με διακομιστές και πάρα πολλά άλλα. Υποστηρίζεται από όλους τους σύγχρονους φυλλομετρητές όπως ο Internet Explorer, ο Chrome, ο Safari, και ο Firefox.

Η JavaScript σχεδιάστηκε για να παρέχει διαδραστικότητα στις σελίδες HTML. Είναι γλώσσα εκτελέσιμου κώδικα που ενσωματώνεται απευθείας στης σελίδες HTML. Είναι δωρεάν και ανήκει στην φιλοσοφία του ανοικτού λογισμικού. Η JavaScript και η Java είναι δύο εντελώς διαφορετικές γλώσσες προγραμματισμού [17].

Τι μπορεί να κάνει η JavaScript

- Η JavaScript δίνει στους σχεδιαστές των σελίδων HTML ένα προγραμματιστικό εργαλείο.
- Μπορεί να αντιδρά σε γεγονότα όπως το τέλος φόρτωσης μια ιστοσελίδας, ή την επιλογή του χρήστη σε κάποιο στοιχείο της HTML.
- Η JavaScript μπορεί να διαβάζει και να γράφει τα στοιχεία HTML, όπως για παράδειγμα τα περιεχόμενα ενός στοιχείου.
- Μπορεί να κάνει έλεγχο και επιβεβαίωση στα δεδομένα φορμών πριν αυτά υποβληθούν στον διακομιστή, εξοικονομώντας με αυτόν τον τρόπο πόρους που σχετίζονται με την επεξεργασία των δεδομένων.
- Μπορεί να εντοπίσει τον τύπο του φυλλομετρητή που χρησιμοποιεί ο χρήστης και ανάλογα με αυτόν να απεικονίσει διαφορετικές εκδόσεις μιας ιστοσελίδας.
- Η JavaScript μπορεί να χρησιμοποιηθεί για να αποθηκεύσει ή να κάνει ανάκληση σε διάφορα δεδομένα προσωπικά για έναν χρήστη κάνοντας χρήση των cookies.

Παραδείγματα JavaScript

Παρακάτω παρατηρείται ένα απόσπασμα κώδικα JavaScript. Η σύνταξη πρέπει να ξεκινάει με <script type=”text/javascript”> και να τελειώνει με </script>. Οτιδήποτε βρίσκεται μεταξύ αυτών των δύο ετικετών HTML θεωρείται από τον φυλλομετρητή εκτελέσιμος κώδικας JavaScript. Στο συγκεκριμένο κομμάτι παρατίθεται η δημιουργία μιας παραγράφου   HTML και μέσα σε αυτήν υπάρχουν σαν δεδομένα η ημερομηνία που παράγεται από την συνάρτηση Date(). Με την σύνταξη +Date()+ το αποτέλεσμα της συνάρτησης μετατρέπεται σε ακολουθία χαρακτήρων (string).

```
<html>
<body>

<h1>My First Web Page</h1>

<script type="text/javascript">
document.write("<p>" + Date() + "</p>");
</script>

</body>
</html>
```

Στον παρακάτω κώδικα υπάρχει η χρήση του επιλογέα DOM της JavaScript όπου εντοπίζει το στοιχείο με id=”demo” με χρήση του document.getelementById (“demo”) όπου είναι μια παράγραφος και εισάγει σε αυτό την ημερομηνία με χρήση της εντολής innerHTML=Date() παρόμοια με το παραπάνω παράδειγμα. Το οπτικό αποτέλεσμα και στις δύο περιπτώσεις θα είναι το ίδιο.

```
<html>
<body>

<h1>My First Web Page</h1>

<p id="demo"></p>

<script type="text/javascript">
document.getElementById("demo").innerHTML=Date();
</script>

</body>
</html>
```

Στο τελευταίο ενδεικτικό παράδειγμα παρατίθεται ένα απόσπασμα JavaScript ενσωματωμένο στην ετικέτα επικεφαλίδας του κειμένου HTML. Αυτός είναι ένας εναλλακτικός τρόπος ενσωμάτωσης της JavaScript σε κείμενο HTML και διαφέρει από τα προηγούμενα παραδείγματα που ενσωματώνονταν μέσα στην ετικέτα σώματος της HTML. Αυτό το κομμάτι JavaScript υλοποιεί μια συνάρτηση με όνομα displayDate() όπου ενσωματώνει στο στοιχείο HTML με id = “demo” την ημερομηνία. Εδώ παρατηρείται ότι αυτό συμβαίνει κάνοντας χρήση τις διαχείρισης γεγονότων της JavaScript. Πιο αναλυτικά φαίνεται η ύπαρξη μια ετικέτας HTML με

όνομα button type = “button” που δημιουργεί ένα κουμπί που γράφει επάνω του Display Date. Μέσα στην ετικέτα του κουμπιού της HTML υπάρχει η ιδιότητα onclick =”displayDate()”, η οποία καλεί την συνάρτηση της JavaScript που υπάρχει παραπάνω. Έτσι με το πάτημα του κουμπιού εμφανίζεται στην παράγραφο με id = “demo” η τρέχουσα ημερομηνία. Σε αυτό το παράδειγμα φαίνεται δηλαδή η χρήση της JavaScript για την αντιμετώπιση γεγονότων όπως το πάτημα ενός κουμπιού.

```
<html>

<head>
<script type="text/javascript">
function displayDate()
{
document.getElementById("demo").innerHTML=Date();
}
</script>
</head>

<body>

<h1>My First Web Page</h1>

<p id="demo"></p>

<button type="button" onclick="displayDate()">Display Date</button>

</body>
</html>
```

Επίλογος – Συμπεράσματα

Στο Κεφάλαιο 3 παρατέθηκαν όλα τα προγραμματιστικά εργαλεία, όπου βασίστηκε η υλοποίηση της εφαρμογής ευφυούς οικείας. Αναλύθηκαν μια πληθώρα από προγραμματιστικές γλώσσες διαφορετικών επιπέδων και αντικειμένων. Συγκεκριμένα πραγματοποιήθηκε εκτενής αναφορά σε προγραμματιστικές τεχνικές που θα εφαρμοστούν στις διατάξεις των κόμβων αισθητήρων και ενεργοποιητών, καθώς επίσης και προγραμματιστικές τεχνικές που θα εφαρμοστούν με σκοπό την υλοποίηση των διεπαφών του χρήστη.

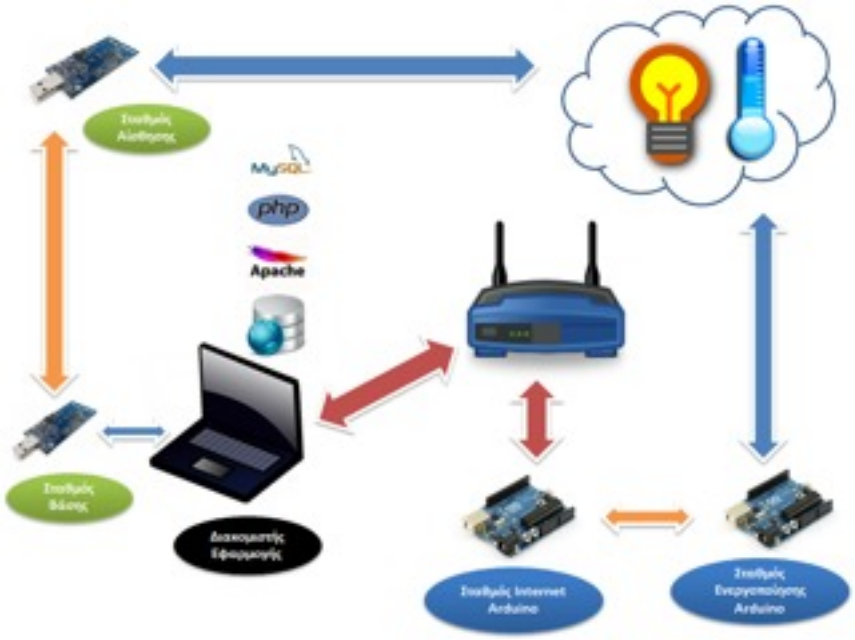
Λόγω του μεγάλου πλήθους των προγραμματιστικών τεχνικών που αφορούν την υλοποίηση της εφαρμογής γίνεται σαφές το μέγεθος της πολυπλοκότητας σε επίπεδο υλοποίησης προκειμένου να υπάρξει ενοποίηση και διαφάνεια ως προς τον τελικό χρήστη. Σαν αποτέλεσμα των ανωτέρω, στην παρούσα Διπλωματική δόθηκε ιδιαίτερη έμφαση έτσι ώστε να επιτευχθεί η βέλτιστη υλοποίηση σε όλους τους τομείς.

Στο Κεφάλαιο 4 που ακολουθεί παρατίθεται η ανάλυση της υλοποίησης που αφορά το Δίκτυο Ασύρματων Αισθητήρων και ενεργοποιητών, καθώς και η λογική που χρησιμοποιήθηκε για την κατασκευή του δικτύου σε επίπεδο διατάξεων.

ΚΕΦΑΛΑΙΟ 4: Υλοποίηση Αρχιτεκτονικής Ασύρματου Δικτύου Αισθητήρων

Σχεδιαστική Λογική

Για την υλοποίηση της εφαρμογής της ευφυούς οικείας χρησιμοποιήθηκαν δύο πλατφόρμες ασύρματων κόμβων. Στόχος ήταν η επίδειξη της διαλειτουργικότητας δύο δικτύων από διαφορετικές τεχνολογίες κόμβων αισθητήρων. Το πρώτο ασύρματο δίκτυο υλοποιεί την παρακολούθηση φυσικών παραμέτρων σε ένα χώρο, ενώ το δεύτερο ασύρματο δίκτυο υλοποιεί τη δυνατότητα παρέμβασης σε ηλεκτρικές συσκευές ώστε να καθορίζονται οι λειτουργίες τους με βάση τις επιλογές χρήστη. Ένα παράδειγμα μια τέτοιας υλοποίησης αποτελεί η δειγματοληψία της θερμοκρασίας και της έντασης της φωτεινότητας σε ένα χώρο και ο απομακρυσμένος έλεγχος αυτών των συνθηκών με την παρέμβαση του χρήστη. Συγκεκριμένα, αν η τιμή της έντασης της φωτεινότητας δεν ήταν η επιθυμητή, ο χρήστης να παρεμβαίνει απομακρυσμένα ενεργοποιώντας ένα φωτιστικό σώμα.

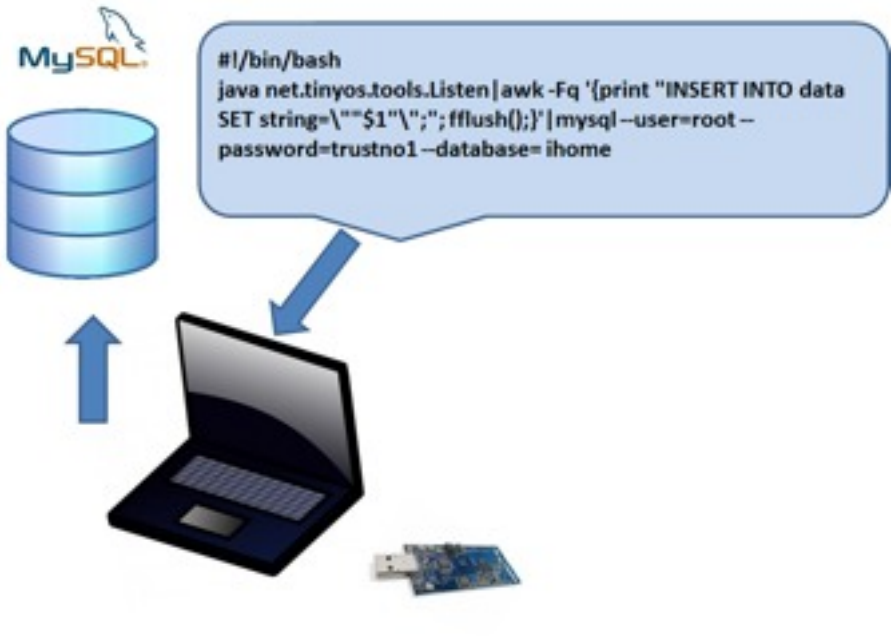


Εικόνα 45: Σχεδιαστική λογική της εφαρμογής ευφυούς οικείας.

Για την υλοποίηση του ανωτέρω σκοπού πραγματοποιήθηκε ενοποίηση δύο τεχνολογιών ασύρματων κόμβων που βασίστηκε στην διεκπεραίωση διαφορετικών διεργασιών. Συγκεκριμένα οι διεργασίες που έπρεπε να επιτελεσθούν ήταν η παρακολούθηση συνθηκών μέσα σε έναν χώρο και στην συνέχεια ο έλεγχος συσκευών με βάση την απόφαση του χρήστη ή κάποιο γεγονός.

Για την διαδικασία της αίσθησης χρησιμοποιήθηκαν ασύρματοι κόμβοι αισθητήρων της οικογένειας Tmote και συγκεκριμένα το μοντέλο Tmote Sky. Για την

πραγματοποίηση αυτής της λειτουργίας ένα Tmote Sky προγραμματίστηκε να εκτελεί λειτουργίες σταθμού βάσης συνδεδεμένο μέσω USB σε έναν διακομιστή. Ο σταθμός αυτός παρακολουθεί το ασύρματο κανάλι και περιμένει για πακέτα από τους κόμβους αίσθησης. Τα πακέτα αυτά αμέσως μετά την λήψη τους, εισάγονται σε μια βάση δεδομένων MySQL όπως φαίνεται και στην εικόνα 46.



Εικόνα 46: Διαδικασία εισόδου ακατέργαστων δεδομένων στην βάση δεδομένων.

Η πρώτη εντολή που βρίσκεται στο μπλε πλαίσιο αρχικά υποδηλώνει ότι ακολουθεί εκτελέσιμος κώδικας. Η δεύτερη εντολή λαμβάνει τα ακατέργαστα δεδομένα που εισέρχεται στην θύρα USB από τον σταθμό βάσης Tmote Sky και το εισάγει σε ένα διαχειρίσιμο δοχείο (string) με όνομα S1. Έπειτα τα δεδομένα του δοχείου εισάγονται στην βάση δεδομένων MySQL με όνομα ihome για λογαριασμό του χρήστη root. Σημαντικό είναι ότι στο τέλος κάθε εκτέλεσης το δοχείο $1 αδειάζει, για να μην προκύψει θέμα επανάληψης ίδιων δεδομένων.

Αντίστοιχα ένα δεύτερο Tmote Sky προγραμματίστηκε να λειτουργεί ως κόμβος αίσθησης και να υλοποιεί δειγματοληψία κάθε ένα δευτερόλεπτο στους αισθητήρες θερμοκρασίας και φωτός που έχει ενσωματωμένους.

Τα δεδομένα που αποθηκεύονται στη βάση δεδομένων MySQL χρησιμοποιούνται από μια διαδικτυακή εφαρμογή που εκτελείται στον ίδιο διακομιστή που είναι συνδεδεμένος ο σταθμός βάσης. Η εφαρμογή αυτή επεξεργάζεται τα δεδομένα και τα μετατρέπει στην επιθυμητή μορφή από όπου αντλούνται για να απεικονιστούν στον χρήστη. Η διαδικτυακή εφαρμογή έχει διπλή χρήση. Ο χρήστης μπορεί να ενημερώνεται σε πραγματικό χρόνο μέσω κάποιου φυλλομετρητή, είτε η διαδικτυακή εφαρμογή να χρησιμοποιείται σαν πυρήνας για την εφαρμογή που έχει υλοποιηθεί για το κινητό του τηλέφωνο.

Για την υλοποίηση των λειτουργιών της ενέργειας από τον χρήστη μπορεί και η διαδικτυακή εφαρμογή αλλά και η εφαρμογή που βρίσκεται στο κινητό να στέλνουν εντολές στους κόμβους ενεργοποίησης που υλοποιούνται με την χρήση της πλατφόρμας Arduino. Συγκεκριμένα χρησιμοποιήθηκαν δύο Arduino Uno, δύο XBee modules, μια Ethernet Shield και ένα Relay Shield.

Σταθμός Βάσης Tmote Sky

Ο ρόλος του σταθμού βάσης είναι να συγκεντρώνει τις απαραίτητες μετρήσεις που εκπέμπονται από τον κόμβο αισθητήρα. Είναι συνδεδεμένος στον διακομιστή της εφαρμογής. Το πρωτόκολλο που χρησιμοποιείται για την ασύρματη επικοινωνία με τον κόμβο αισθητήρα είναι το ZigBee πάνω στο πρότυπο IEEE 802.15.4. Με αυτό τον τρόπο επιτυγχάνεται πολύ χαμηλή κατανάλωση σε ισχύ για την επικοινωνία μεταξύ σταθμού βάσης και κόμβου παρακολούθησης. Ο σταθμός βάσης τροφοδοτείται από την θύρα USB που διαθέτει, καθώς επίσης και μέσω αυτής εισάγει τα δεδομένα σειριακά στον διακομιστή.

Το λειτουργικό σύστημα που χρησιμοποιήθηκε είναι το TinyOS και η εφαρμογή που υλοποιείται από το Tmote Sky είναι η Base Station. Λόγω της αρχιτεκτονικής του λειτουργικού συστήματος TinyOS το μόνο πρόγραμμα που εκτελείται στο Tmote Sky είναι η εφαρμογή σταθμού βάσης (Base Station) .

Εφαρμογή Base Station

Η εφαρμογή Base Station παρέχεται έτοιμη από την διανομή του TinyOS. Λειτουργεί σαν μια γέφυρα ενεργών μηνυμάτων ανάμεσα στο σειριακό και το ασύρματο κανάλι. Στο σειριακό κανάλι η εφαρμογή μπορεί να στέλνει και να λαμβάνει ενεργά μηνύματα. Στο ασύρματο κανάλι στέλνει ασύρματα ενεργά μηνύματα των οποίων η μορφή εξαρτάται από την στοίβα δικτύου που έχει επιλεγεί. Η εφαρμογή Base Station αντιγράφει το μεταγλωττισμένο αναγνωριστικό ομάδας (Group ID) που της έχει ανατεθεί κατά την εγκατάσταση της στο Tmote Sky στα μηνύματα που κινούνται από το σειριακό κανάλι στο ασύρματο και φιλτράρει και αποκόπτει μηνύματα που έχουν διαφορετικό αναγνωριστικό ομάδας. Επίσης διαθέτει ουρές αναμονής και στις δύο κατευθύνσεις με εγγύηση ότι αν φτάσει ένα πακέτο και μπει στη ουρά τότε αυτό θα φύγει κάποια στιγμή και στην επόμενη διεπαφή. Οι ουρές επιτρέπουν στην εφαρμογή να διαχειρίζεται ακμές φόρτου πιο αποτελεσματικά.

Τα led που διαθέτει το Tmote Sky είναι προγραμματισμένα να αναβοσβήνουν ανάλογα με τα εξής γεγονότα:

- **Εναλλαγή κόκκινου**: Γεφύρωση μηνύματος από το σειριακό κανάλι στο ασύρματο.
- **Εναλλαγή πράσινου**: Γεφύρωση μηνύματος από το ασύρματο κανάλι στο σειριακό.
- **Εναλλαγή κίτρινου/μπλε**: Απόρριψη μηνύματος από ουρά αναμονής λόγω υπερχείλισης είτε στην μια κατεύθυνση είτε στην άλλη.

Η εφαρμογή αυτή χρησιμοποιήθηκε έτσι ώστε να πραγματοποιείται η συλλογή ασύρματων πακέτων που εκπέμπει ο απομακρυσμένος κόμβος αίσθησης Tmote Sky και με αυτόν τον τρόπο να λαμβάνεται η εικόνα των περιβαλλοντικών συνθηκών

που επικρατούν σε έναν χώρο. Οι περιβαλλοντολογικές συνθήκες που παρακολουθούνται είναι η ένταση της φωτεινότητας καθώς και η θερμοκρασία του χώρου κάνοντας δειγματοληψία των ενσωματωμένων αισθητήρων φωτός και φωτεινότητας του Tmote Sky κάθε ένα δευτερόλεπτο.

Τέλος τα πακέτα που λαμβάνει ο κόμβος βάσης αποθηκεύονται σε μια βάση δεδομένων MySQL όπως αναφέρθηκε, με όνομα πίνακα data και τοποθετούνται κάτω από την στήλη string. Η βάση δεδομένων βρίσκεται στον διακομιστή και τα πακέτα αποθηκεύονται σε αυτήν απευθείας με άντληση τους από το σειριακό κανάλι.

Ο κώδικας του προγράμματος Base Station αποτελείται από δύο αρχεία. Ένα που παρέχει τις παραμετροποιήσεις και ένα που παρέχει την ενότητα μεταξύ των στοιχείων. Ολόκληρος ο κώδικας της εφαρμογής Base Station βρίσκεται στο ***Παράρτημα Α***.

Σταθμός Συλλογής Δεδομένων Φυσικών Παραμέτρων

Ο σταθμός συλλογής δεδομένων φυσικών παραμέτρων υλοποιήθηκε κάνοντας χρήση ενός δεύτερου ασύρματου κόμβου Tmote Sky. Ο ρόλος αυτού του κόμβου είναι να παρέχει την πληροφορία για τις συνθήκες που επικρατούν σε έναν χώρο. Χρησιμοποιήθηκαν δύο από τους ενσωματωμένους αισθητήρες του κόμβου για την παροχή των απαιτούμενων πληροφοριών. Ο αισθητήρας θερμοκρασίας Sensirion SHT11, και ο αισθητήρας φωτεινότητας Hamamatsu TSR (S-1087). Οι αισθητήρες αυτοί δειγματοληπτούνται κάθε ένα δευτερόλεπτο από τρείς φορές. Αυτές οι μετρήσεις ενθυλακώνονται σε ένα πακέτο στο ωφέλιμο κομμάτι του (payload) και αποστέλλονται στο ασύρματο κανάλι. Επίσης ο λόγος που δειγματοληπτούνται τρείς φορές ο καθένας είναι για αύξηση της ακρίβειας στο τελικό αποτέλεσμα που παρέχεται στον χρήστη καθώς προκύπτει ως ο μέσος όρος τριών μετρήσεων.

Το πρόγραμμα που είναι εγκατεστημένο στον κόμβο αίσθησης είναι μια τροποποιημένη μορφή του κώδικα της εφαρμογή Oscilloscope που διατίθεται από το λειτουργικό TinyOS.

Εφαρμογή Oscilloscope

Η εφαρμογή Oscilloscope υλοποιεί την συλλογή δεδομένων. Δειγματοληπτεί περιοδικά έναν προκαθορισμένο αισθητήρα και εκπέμπει δημοσίως στο ασύρματο κανάλι ένα μήνυμα κάθε δέκα μετρήσεις που θα πραγματοποιηθούν στον αισθητήρα. Αυτές οι μετρήσεις μπορούν να ληφθούν από έναν κόμβο που εκτελεί την εφαρμογή Base Station και να απεικονιστούν με την χρήση μιας εφαρμογής που είναι υλοποιημένη σε γλώσσα Java στον χρήστη. Ο ρυθμός δειγματοληψίας αρχίζει στα 4Hz, αλλά μπορεί να αλλάξει μέσω της Java εφαρμογής.

Στην παρούσα εφαρμογή το θεμιτό αποτέλεσμα είναι το πρόγραμμα Oscilloscope να δειγματοληπτεί δύο αισθητήρες ταυτόχρονα και να συλλέγει τρείς μετρήσεις από τον καθένα. Για να συμβεί αυτό τροποποιήθηκε το πρόγραμμα

Oscilloscope που παρέχεται από το TinyOS διότι αυτό δειγματοληπτεί μόνο έναν αισθητήρα και παρέχει δέκα μετρήσεις του. Αρχικά έπρεπε να τροποποιηθεί το αρχείο Oscilloscope.h που παρέχει διάφορες δηλώσεις και αρχικοποιήσεις στην εφαρμογή. Αυτό μετά τις τροποποιήσεις έγινε όπως ακολουθεί παρακάτω:

```
#ifndef OSCILLOSCOPE_H
#define OSCILLOSCOPE_H

enum {

  NREADINGS = 3,

  DEFAULT_INTERVAL = 256,

  AM_OSCILLOSCOPE = 0x93
};

typedef nx_struct oscilloscope {
  nx_uint16_t version;
  nx_uint16_t interval;
  nx_uint16_t id;
  nx_uint16_t count;
  nx_uint16_t tsr[NREADINGS];
  nx_uint16_t temp[NREADINGS];
} oscilloscope_t;

#endif
```

Στο παραπάνω απόσπασμα τροποποιήθηκε η τιμή NREADINGS από δέκα σε τρία και με αυτό τον τρόπο παρέχονται οι τρείς μετρήσεις του αισθητήρα. Επίσης προστέθηκαν οι εντολές nx_uint16_t tsr[NREADINGS] και nx_uint16_t temp[NREADINGS], που δημιουργούν την επιθυμητή μορφή της δομής του πακέτου που θα εκπέμψει ο κόμβος στο ασύρματο κανάλι. Ουσιαστικά η τροποποίηση αυτή προσθέτει χώρο για τις τρείς μετρήσεις του αισθητήρα φωτός και αμέσως μετά για τις άλλες τρείς μετρήσεις του αισθητήρα θερμοκρασίας δηλώνοντας ότι είναι δεκαεξαδικής μορφής.

Επίσης τροποποιήθηκε το αρχείο OscilloscopeAppC.nc που παρέχει τις παραμετροποιήσεις στην εφαρμογή και έγινε:

```
configuration OscilloscopeAppC { }
implementation
{
  components OscilloscopeC, MainC, ActiveMessageC, LedsC,
    new TimerMilliC(), new SensirionSht11C(), new HamamatsuS10871TsrC(),
    new AMSenderC(AM_OSCILLOSCOPE), new AMReceiverC(AM_OSCILLOSCOPE);

  OscilloscopeC.Boot -> MainC;
  OscilloscopeC.RadioControl -> ActiveMessageC;
  OscilloscopeC.AMSend -> AMSenderC;
```

```
  OscilloscopeC.Receive -> AMReceiverC;
  OscilloscopeC.Timer -> TimerMilliC;
  OscilloscopeC.Leds -> LedsC;
  OscilloscopeC.ReadTSR -> HamamatsuS10871TsrC;
  OscilloscopeC.ReadExtTemp -> SensirionSht11C.Temperature;
}
```

Στο ανωτέρω απόσπασμα με **bold** απεικονίζονται τα κομμάτια που προστέθηκαν. Αρχικά προστέθηκαν οι εντολές new SensirionSht11C() και new HamamatsuS10871TsrC(), που δηλώνουν ότι θα χρησιμοποιηθούν στην εφαρμογή τα στοιχεία αυτά. Δηλαδή οι δύο διαφορετικοί αισθητήρες που θα χρησιμοποιηθούν για την δειγματοληψία των περιβαλλοντολογικών συνθηκών. Έπειτα πραγματοποιήθηκε η σύνδεση τους με την εφαρμογή και δηλώθηκε ότι τα στοιχεία OscilloscopeC.ReadTSR και OscilloscopeC.ReadExtTemp θα διαβάζουν τους αισθητήρες HamamatsuS10871TsrC και SensirionSht11C.Temperature αντίστοιχα.

Τέλος στο κύριο αρχείο της εφαρμογής δηλαδή το OscilloscopeC.nc που παράγει το εκτελέσιμο πρόγραμμα έγιναν οι παρακάτω τροποποιήσεις:

```
module OscilloscopeC @safe()
{
  uses {
    interface Boot;
    interface SplitControl as RadioControl;
    interface AMSend;
    interface Receive;
    interface Timer<TMilli>;
    interface Read<uint16_t> as ReadTSR;
    interface Read<uint16_t> as ReadExtTemp;
    interface Leds;
  }
}
```

Αρχικά δηλώθηκε ότι θα χρησιμοποιηθούν δύο διεπαφές ανάγνωσης, μια για τις τιμές της φωτεινότητας interface Read<uint16_t> as ReadTSR σε δεκαεξαδική μορφή και μια για τις τιμές της θερμοκρασίας interface Read<uint16_t> as ReadExtTemp και αυτή σε δεκαεξαδική μορφή. Έπειτα δηλώθηκαν δύο μεταβλητές για χρήση ως μετρητές που το εύρος τους είναι από το μηδέν μέχρι τον αριθμό των δειγμάτων που έχουν δηλώθει στο κομμάτι των αρχικοποιήσεων ως NREADINGS. Αυτό έγινε προσθέτοντας στον κώδικα τα ακόλουθα:

```
uint8_t readingTemperatures; /* 0 to NREADINGS */
uint8_t readingLights; /* 0 to NREADINGS */
```

Ακολουθεί το κομμάτι του κώδικα της εφαρμογής που αρχικοποιεί τις τιμές των μετρητών και καλεί την διεπαφή που ξεκινάει το χρονόμετρο για την περιοδική λειτουργία του προγράμματος:

```
void startTimer() {
  call Timer.startPeriodic(local.interval);
```

```
  readingTemperatures = 0;
  readingLights = 0;
 }
```

Στο παρακάτω κομμάτι που ακολουθεί παρατηρείται τι συμβαίνει με την έναρξη του μετρητή:

```
event void Timer.fired() {

if (readingTemperatures == NREADINGS && readingLights == NREADINGS)
    {
        if (!sendBusy && sizeof local <= call AMSend.maxPayloadLength())
         {
           // Don't need to check for null because we've already checked length
           // above
              memcpy(call AMSend.getPayload(&sendBuf, sizeof(local)), &local, sizeof
local);
             if (call AMSend.send(AM_BROADCAST_ADDR, &sendBuf, sizeof local) ==
SUCCESS)
            sendBusy = TRUE;
         }
        if (!sendBusy)
          report_problem();

        readingTemperatures = 0;
        readingLights = 0;
```

Αρχικά ελέγχεται αν υπάρχει ο επιθυμητός αριθμός των μετρήσεων και στους δύο αισθητήρες μαζί. Αν αυτό ισχύει ελέγχεται αν το κανάλι αποστολής δεν είναι κατειλημμένο και ότι δεν υφίσταται υπέρβαση στο μέγιστο μέγεθος του ωφέλιμου χώρου δεδομένων. Αν και αυτό συμβαίνει τότε καλείται η διεπαφή που ενσωματώνει την πληροφορία των αισθητήρων από την μνήμη στο πακέτο που θα αποσταλεί στο ασύρματο κανάλι. Αν αυτή η διαδικασία στεφθεί με επιτυχία και ξεκινήσει η αποστολή τότε μεταβάλλεται σε αληθής η μεταβλητή που δείχνει αν το κανάλι είναι απασχολημένο. Αν όμως η μεταβλητή που δείχνει αν το κανάλι είναι κατειλημμένο η όχι δεν αλλάξει, τότε καλείται η συνάρτηση δήλωσης προβλήματος. Επίσης αρχικοποιούνται στο μηδέν οι τιμές των μετρητών για το πλήθος των μετρήσεων.

Παρακάτω ακολουθεί η εντολή που καλεί τις συναρτήσεις που κάνουν την ανάγνωση στους αισθητήρες φωτός και θερμοκρασίας. Επίσης ελέγχεται αν οι συναρτήσεις που θα κληθούν επιστρέψουν κατάσταση επιτυχίας. Αν έστω και μια από τις δύο συναρτήσεις που θα κληθούν δεν επιστρέψουν κατάσταση επιτυχίας τότε καλείται η συνάρτηση δήλωσης προβλήματος.

```
  if ((call ReadTSR.read() != SUCCESS) || (call ReadExtTemp.read() != SUCCESS) )
    report_problem();
```

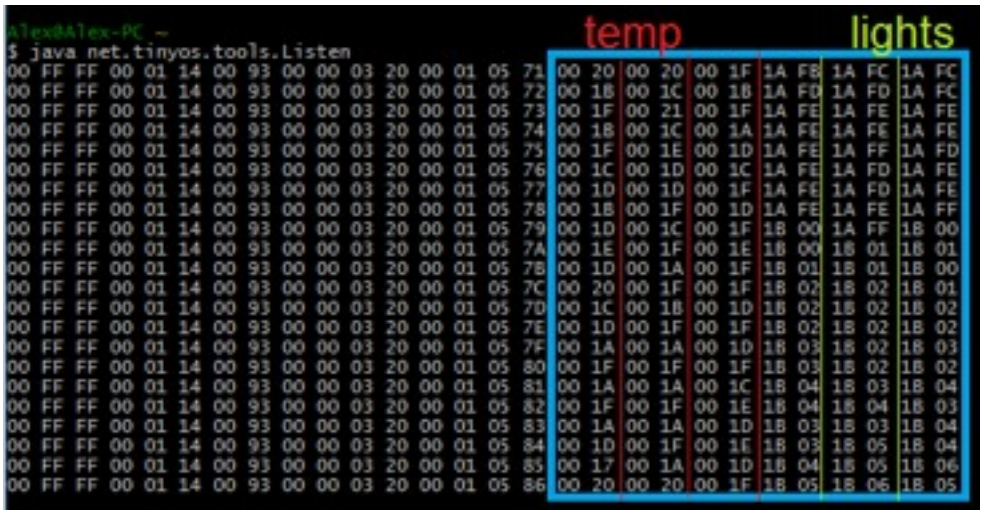
Τέλος παρατίθενται οι συναρτήσεις που υλοποιούν την ανάγνωση των αισθητήρων και είναι οι ακόλουθες:

```
event void ReadTSR.readDone(error_t result, uint16_t data) {
   if (result != SUCCESS)
    {
         data = 0xffff;
         report_problem();
    }
   local.tsr[readingLights++] = data;
 }

 event void ReadExtTemp.readDone(error_t result, uint16_t data) {
   if (result != SUCCESS)
    {
         data = 0xffff;
         report_problem();
    }
   local.temp[readingTemperatures++] = data;
 }
}
```

Στις παραπάνω συναρτήσεις υλοποιείται η υλοποίηση της ανάγνωσης των δεδομένων από την μνήμη. Αν τα αποτελέσματα που θα διαβαστούν δεν είναι τα σωστά καλείται η συνάρτηση δήλωσης προβλήματος και σαν δεδομένα εμφανίζονται μηδενικές τιμές. Αν όμως τα αποτελέσματα είναι σωστά τότε σαν δεδομένα περνιούνται οι τιμές που βρίσκονται στην θέση μνήμης που αντιστοιχεί στο νούμερο της μέτρησης και αυξάνεται κατά ένα μέχρι να διαβαστούν όλες οι μετρήσεις.

Στα παραπάνω αποσπάσματα με **bold** επισημαίνονται τα κομμάτια του κώδικα που αποτελούν μέρος των τροποποιήσεων που έγιναν στην εφαρμογή Oscilloscope. Ολόκληρος ο κώδικας της εφαρμογής παρατίθεται στο ***Παράρτημα Α***.

Τα δεδομένα που εκπέμπονται από τον σταθμό αίσθησης και λαμβάνονται από τον σταθμό βάσης έχουν την παρακάτω μορφή:



Εικόνα 47: Στιγμιότυπο ροής ακατέργαστων δεδομένων από τον σταθμό συλλογής στην σειριακή θύρα του σταθμού βάσης.

Για να απεικονιστούν στον τελικό χρήστη πρέπει να μετατραπούν από μορφή ροής δεκαεξαδικών δεδομένων σε φυσικές μονάδες του συστήματος SI. Αυτό πραγματοποιείται με βάση τύπους που εξάγονται από το φυλλάδιο τεχνικών προδιαγραφών των αισθητήρων. Συγκεκριμένα για να μετατραπεί σε επιθυμητή μορφή η τιμή της θερμοκρασίας που μετράει ο αισθητήρας Sensirion SHT11 χρησιμοποιείται ο ακόλουθος τύπος:

Όπου R.D είναι η δεκαδική μορφή του αριθμού ροής δεδομένων που λαμβάνεται από το Tmote Sky.

Για την μετατροπή της τιμής της έντασης φωτεινότητας που μετράει ο αισθητήρας Hamamatsu TSR (S-1087) χρησιμοποιείται η εξής μεθοδολογία:

Η φωτοδίοδος που υλοποιεί αυτόν τον αισθητήρα τροφοδοτείται με τάση Vref = 1.5V. Αυτό που έχει ενδιαφέρον με βάση το φυλλάδιο προδιαγραφών του αισθητήρα είναι η τιμή του ρεύματος που διαρρέει κατά μήκος την αντίσταση R = 100kΩ που διαθέτει η φωτοδίοδος. Η τιμή της τάσης του αισθητήρα υπολογίζεται από τον τύπο:

Μετατρέπεται αυτή η τιμή σε ρεύμα με βάση τον τύπο:

Άρα το ρεύμα της φωτοδιόδου βρίσκεται από τον τύπο:

Τέλος αυτό μετατρέπεται σε μονάδα lux κάνοντας χρήση του τύπου:

Σταθμός Βάσης Arduino

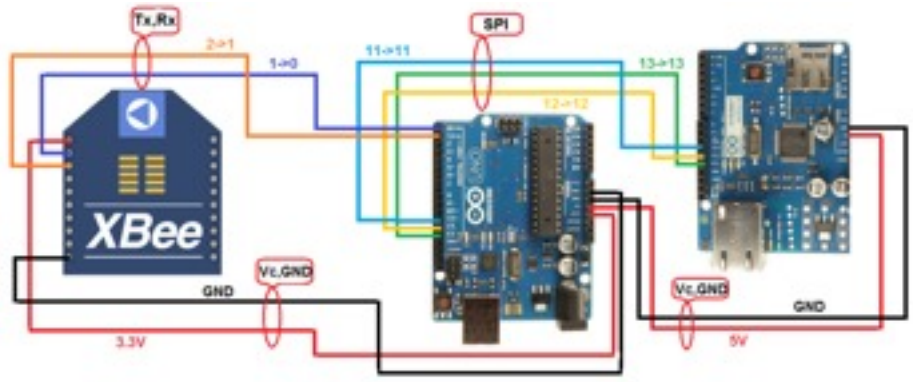
Για την υλοποίηση της επικοινωνίας με το internet και την διαχείριση των αιτήσεων για ενεργοποίηση των φωτιστικών σωμάτων με την χρήση απομακρυσμένου υπολογιστή ή κινητού τηλεφώνου επιλέχθηκε η πλατφόρμα Arduino.

Συγκεκριμένα χρησιμοποιήθηκε σαν πυρήνας της κατασκευής ένα Arduino Uno, πάνω στο οποίο συνδέθηκε η πλακέτα Ethernet Shield που παρέχει στο Arduino βιβλιοθήκες υλοποίησης και διαχείρισης της επικοινωνίας με το Internet. Επιπλέον συνδέθηκε μια πλακέτα XBee Module που παρέχει στον όλο κόμβο την δυνατότητα ασύρματης επικοινωνίας με χρήση του προτύπου χαμηλής κατανάλωσης ZigBee.

Ο σκοπός αυτού του κόμβου είναι να δημιουργηθεί ένας διακομιστής διαδικτύου, όπου ανάλογα με τις αιτήσεις http που θα λαμβάνει, να αντιδρά στα

ζητούμενα τους. Για παράδειγμα να ενημερώνει για την κατάσταση στην οποία βρίσκονται τα φωτιστικά σώματα, και να τα ενεργοποιεί ή να τα απενεργοποιεί ανάλογα με την επιθυμία του χρήστη.

Η συνδεσμολογία που υλοποιήθηκε φαίνεται στη εικόνα 48. Αρχικά συνδέθηκε στο Arduino η πλακέτα Ethernet Shield και πάνω σε αυτή συνδέθηκε η πλακέτα XBee Shield. Οι ακροδέκτες που χρειάζεται να συνδεθούν φαίνονται στην εικόνα 48. Για να λειτουργήσει η πλακέτα Ethernet Shield χρειάζεται να συνδεθούν οι ακροδέκτες 11,12 και 13 του Arduino με αυτούς τις πλακέτας. Αυτοί οι ακροδέκτες παρέχουν την σύγχρονη σειριακή επικοινωνία δεδομένων (SPI) μεταξύ του μικροελεγκτή που βρίσκεται στο Arduino και του μικροελεγκτή που βρίσκεται στην πλακέτα Ethernet Shield. Επίσης χρειάζεται να συνδεθούν και οι ακροδέκτες τροφοδοσίας που παρέχει το Arduino με τους αντίστοιχους της πλακέτας Ethernet Shield. Συγκεκριμένα η πλακέτα Ethernet Shield τροφοδοτείται από τα 5V που παρέχει το Arduino. Η επόμενη πλακέτα που πρέπει να συνδεθεί είναι η XBee Shield. Για αυτή την πλακέτα χρειαζόμαστε τους ακροδέκτες 0 και 1 του Arduino που αφορούν την σειριακή επικοινωνία του Arduino η οποία μετατρέπεται σε ασύρματη μέσω της πλακέτας XBee Shield. Τέλος για την τροφοδοσία της πλακέτας XBee Shield χρησιμοποιείται ο ακροδέκτης του Arduino που παρέχει 3,3V και ο ακροδέκτης της γείωσης.



Εικόνα 48: Συνδεσμολογία σταθμού διαδικτύου.

Το σχεδιάγραμμα Arduino που υλοποιεί τα παραπάνω είναι το ακόλουθο και αναλύεται τμηματικά:

```
#include <Ethernet.h>
#include <SPI.h>
boolean reading = false;
boolean state = false;

byte ip[] = { 192, 168, 0, 199 };
byte gateway[] = { 192, 168, 0, 1 };
byte subnet[] = { 255, 255, 255, 0 };
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };

 Server server = Server(8889);
```

Σε αυτό το πρώτο κομμάτι γίνεται η εισαγωγή δύο βιβλιοθηκών της Ethernet και της SPI και η δήλωση δύο μεταβλητών που θα μας χρειαστούν παρακάτω στο πρόγραμμά. Η βιβλιοθήκη Ethernet παρέχει τις δυνατότητες επικοινωνίας μέσω του προτύπου Ethernet. Η βιβλιοθήκη SPI παρέχει την δυνατότητα επικοινωνίας του μικροελεγκτή του Arduino με τον μικροελεγκτή της πλακέτας Ethernet Shield. Ακολουθεί η ανάθεση διεύθυνσης IP στον κόμβο καθώς επίσης και η δήλωση της διεύθυνσης IP που επικοινωνεί το router με το Internet. Τέλος πραγματοποιείται η δήλωση της μάσκας υποδικτύου που έχει συνδεθεί το Arduino, και της φυσικής διεύθυνσης του Arduino.

Έπειτα από αυτή την παραμετροποίηση δηλώνεται ο διακομιστής και ρυθμίζεται να δέχεται αιτήσεις στην θύρα 8889 που χρησιμοποιείται για αιτήσεις http.

```
void setup(){
  pinMode(2, OUTPUT);
  Ethernet.begin(mac, ip, gateway, subnet);
  server.begin();
  Serial.begin(9600);
}
```

Στο παραπάνω κομμάτι που εκτελείται κάθε φορά που ξεκινάει το Arduino πραγματοποιείται η αρχικοποίηση διαφόρων παραμέτρων όπως ποιός ακροδέκτης του Arduino θα χρησιμοποιηθεί και ως τί θα χρησιμοποιηθεί. Στον συγκεκριμένο κώδικα χρησιμοποιείται ο ψηφιακός ακροδέκτη 2 ως έξοδος. Παρακάτω εισάγονται οι παραμετροποιήσεις που έγιναν στο πρώτο κομμάτι του κώδικα για να εκκινήσει η επικοινωνία μέσω Ethernet, ενεργοποιείται ο διακομιστής που θα λαμβάνει τις αιτήσεις και τέλος ενεργοποιείται μια σειριακή επικοινωνία με ρυθμό μετάδοσης δεδομένων 9600bps.

Στο κομμάτι που ακολουθεί παρατίθεται ο κώδικας που εκτελείται συνέχεια από το Arduino και αποτελεί τον πυρήνα της εφαρμογής αυτού του κόμβου. Αποτελείται από μια συνάρτηση την checkForClient() που με την σειρά της αποτελείται από διάφορες εντολές και από άλλες δύο συναρτήσεις.

```
void loop(){
  checkForClient();

}
```

Η συνάρτηση checkForClient() ελέγχει αν ο διακομιστής είναι διαθέσιμος για να διαχειριστεί αιτήματα και αν κάποιος πελάτης έχει κάνει αίτηση να συνδεθεί με τον διακομιστή. Οι αιτήσεις που έρχονται από τους πελάτες είναι της μορφής http://192.168.0.199:8889/?0, δηλαδή η διεύθυνση IP που έχει ανατεθεί στον διακομιστή μας καθώς και μια παράμετρος που ακολουθεί μετά από το ερωτηματικό. Ανάλογα με αυτή την παράμετρο προκαλούνται οι διάφορετικές αποκρίσεις του διακομιστή. Αρχικά ελέγχεται αν είναι διαθέσιμος ο διακομιστής και αν είναι συνδεδεμένος κάποιος πελάτης. Εάν είναι συνδεδεμένος τότε σαν απάντηση λαμβάνει την βασική επικεφαλίδα των μηνυμάτων http για επιτυχή σύνδεση και μετά ακολουθεί το περιεχόμενο απάντησης του διακομιστή.

```
void checkForClient(){
```

```
Client client = server.available();

if (client) {

 // an http request ends with a blank line
 boolean currentLineIsBlank = true;
 boolean sentHeader = false;

 while (client.connected()) {
  if (client.available()) {

   if(!sentHeader){
    client.println("HTTP/1.1 200 OK");
    client.println("Content-Type: text/html");
    client.println();
    sentHeader = true;
   }
```

Στο παρακάτω απόσπασμα ξεκινάει να διαβάζει ο διακομιστής την μορφή του αιτήματος που του έχει αποσταλεί από τον πελάτη αφού πρώτα ελέγξει αν η μορφή του αιτήματος είναι η αναμενόμενη. Με το που εντοπίζεται μια παράμετρος μετά από το ? αυτή αποθηκεύεται στην μεταβλητή c και ξεκινάει ο έλεγχος της τιμής αυτής της μεταβλητής. Στο πρόγραμμά έχει ορίσει ότι αν λάβει ο διακομιστής το αίτημα http://192.168.0.199:8889/?2 τότε αυτός θα καλέσει την συνάρτηση onPin που ακολουθεί παρακάτω και ενεργοποιεί τα φωτιστικά σώματα, ενώ αν λάβει αίτημα της μορφής http://192.168.0.199:8889/?3 θα καλέσει την συνάρτηση offPin που απενεργοποιεί τα φωτιστικά σώματα. Τέλος αν λάβει την αίτηση http://192.168.0.199:8889/?0 τότε θα ελέγξει σε ποία κατάσταση βρίσκονται τα φωτιστικά σώματα (ενεργοποιημένα ή απενεργοποιημένα). Αυτό γίνεται μέσω της συνάρτησης switch που ελέγχει την τιμή της μεταβλητής c και ανάλογα εκτελεί την επόμενη ενέργεια.

```
   char c = client.read();

   if(reading && c == ' ') reading = false;
   if(c == '?') reading = true;

if(reading){

     switch (c) {
      case '2':
       onPin(2, client);
       break;
      case '3':
       offPin(2, client);
       break;
      case '0':
       if (state == true){
```

```
            client.print("ON");}
          else {
            client.print("OFF");}
        break;
      }

    }
```

Στο παρακάτω κομμάτι παρατίθενται οι συνθήκες τερματισμού αν δεν εντοπιστεί κάποιος χαρακτήρας μετά το ? ή αντίστοιχα αν ακολουθεί κάποιος άλλος χαρακτήρας μετά τον πρώτο χαρακτήρα που βρίσκεται έπειτα απο το ?, έτσι ώστε να διαβαστεί και αυτός.

```
    if (c == '\n' && currentLineIsBlank)  break;

    if (c == '\n') {
      currentLineIsBlank = true;
    }else if (c != '\r') {
      currentLineIsBlank = false;
    }

   }
  }
```

Αντίστοιχα στο κομμάτι που ακολουθεί ενσωματώνεται η προσθήκη μιας καθυστέρησης της τάξης του 1ms έτσι ώστε να παρέχεται χρόνος στον πελάτη να λάβει τα δεδομένα που θα αποσταλούν από τον διακομιστή του Arduino. Τέλος αν τα ανωτέρω στεφθούν με επιτυχία γίνεται απόλυση της σύνδεσης με τον πελάτη μέχρι την δημιουργία μιας νέας.

```
  delay(1);
  client.stop();
 }

}
```

Στην συνέχεια ακολουθούν οι συναρτήσεις onPin και offPin. Αυτές οι συναρτήσεις παίρνουν σαν παράμετρο τον αριθμό του ακροδέκτη του Arduino που επιθυμείται να οδηγηθεί και τον πελάτη που έκανε το αίτημα. Και οι δύο συναρτήσεις είναι ίδιες απλά εκτελούν τα αντίθετα γεγονότα. Στην συνάρτηση onPin η οποία καλείται αν λάβει ο διακομιστής αίτημα με παράμετρο c=2 στέλνεται στο ασύρματο κανάλι μέσω της πλακέτας XBee module ο χαρακτήρας “H” και αν ληφθεί θετική απόκριση εκτυπώνεται το μήνυμα Turning ON Lights OK. Αντίστοιχα αν το αίτημα του πελάτη έχει παράμετρο c=3 τότε καλείται η συνάρτηση offPin όπου στέλνει στο ασύρματο κανάλι τον χαρακτήρα “L” και αν ληφθεί θετική απόκριση εκτυπώνεται στον πελάτη Turning OFF Lights OK. Το νόημα της αποστολής στο ασύρματο κανάλι αυτών των χαρακτήρων είναι για να τους λάβει ο κόμβος ενεργοποίησης που θα αναλυθεί παρακάτω. Ο κόμβος ενεργοποίησης είναι αυτός που θα ενεργοποιήσει ή θα απενεργοποιήσει τα φωτιστικά σώματα και επιθυμείται

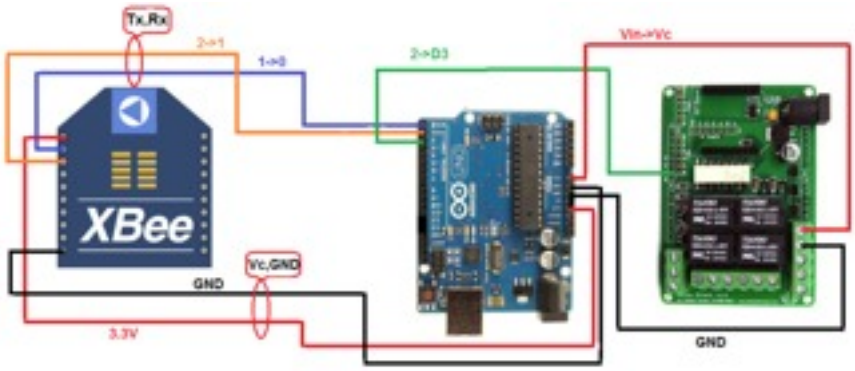
να είναι φορητός. Για αυτό τον λόγο χρησιμοποιήθηκε η ασύρματη επικοινωνία. Τέλος υλοποιείται κατά αντιστοιχία η αλλαγή μιας μεταβλητής status για να γνωρίζει ο κόμβος βάσης σε τι κατάσταση βρίσκονται τα φωτιστικά σώματα που είναι συνδεδεμένα στον σταθμό ενεργοποίησης.

```
void onPin(int pin, Client client){
  client.print("Turning ON Lights");
  client.print("<br>");
  Serial.print('H');
  delay(10);
  if (Serial.available()>0){
    if (Serial.read() == 'K'){
      delay(10);
      client.print("OK");
    }
  }
  state = true;

}

void offPin(int pin, Client client){
  client.print("Turning OFF Lights");
  client.print("<br>");
  Serial.print('L');
  delay(10);
  if (Serial.available()>0){
    if (Serial.read() == 'K'){
      delay(10);
      client.print("OK");
    }
  }
  state = false;

}
```

Εικόνα 49: Υλοποίηση Κόμβου Σταθμού Βάσης.

Κόμβος Ενεργοποίησης Arduino

Ο κόμβος ενεργοποίησης υλοποιήθηκε με την χρήση ενός δεύτερου Arduino Uno, μιας πλακέτας XBee module για την παροχή της ασύρματης επικοινωνίας και μιας πλακέτας Relay Shield για την οδήγηση των φωτιστικών σωμάτων. Η χρήση της πλακέτας Relay Shield ήταν αναγκαία διότι για την οδήγηση των φωτιστικών σωμάτων γίνεται χρήση μεγάλων τάσεων της τάξης των 230V.



Εικόνα 50: Συνδεσμολογίες Κόμβου Ενεργοποίησης.

Στην εικόνα 50 παρατηρούνται επακριβώς οι συνδεσμολογίες του κόμβου ενεργοποίησης. Έχει συνδεθεί όπως και πριν η πλακέτα XBee Shield στους ακροδέκτες του Arduino 0 και 1, όπως επίσης και την απαραίτητη τροφοδοσία των 3,3V καθώς και την γείωση που παρέχονται από το Arduino. Τέλος έχει συνδεθεί η πλακέτα Relay Shield. Ο ακροδέκτης 2 του Arduino ο οποίος χρησιμοποιείται σαν ψηφιακή είσοδος ή έξοδος συνδέεται στον ακροδέκτη D3 της πλακέτας Relay Shield για να μπορεί να οδηγεί το τρίτο ρελέ που ενεργοποιεί και απενεργοποιεί το φωτιστικό σώμα. Επίσης στους ακροδέκτες Vc και GND της πλακέτας Relay Shield

έχουν συνδεθεί οι ακροδέκτες Vin και GND αντίστοιχα του Arduino που παρέχουν την τροφοδοσία των 9V που χρειάζεται η πλακέτα Relay Shield για να λειτουργήσει. Ο κόμβος αυτός είναι φορητός και μπορεί να λειτουργήσει με την χρήση μιας μπαταρίας τάσης 9V. Με αυτόν τον τρόπο επιτυγχάνεται μεγάλη ευελιξία και παρέχεται η δυνατότητα ελέγχου συσκευών οπουδήποτε μέσα σε ένα οικιακό περιβάλλον χωρίς να χρειάζονται παροχές τροφοδοσίας και επιπλέον καλώδια.

Το σχεδιάγραμμα Arduino που είναι φορτωμένο σε αυτόν τον κόμβο είναι το ακόλουθο:

Ξεκινάει με την αρχικοποίηση μέσα στην συνάρτηση setup() που τρέχει μόλις εκκινήσει ο κόμβος. Σε αυτή την συνάρτηση δηλώνεται ποιος ακροδέκτης θα χρησιμοποιηθεί και επίσης γίνεται η εκκίνηση της επικοινωνίας με ρυθμό μετάδοσης δεδομένων 9600bps.

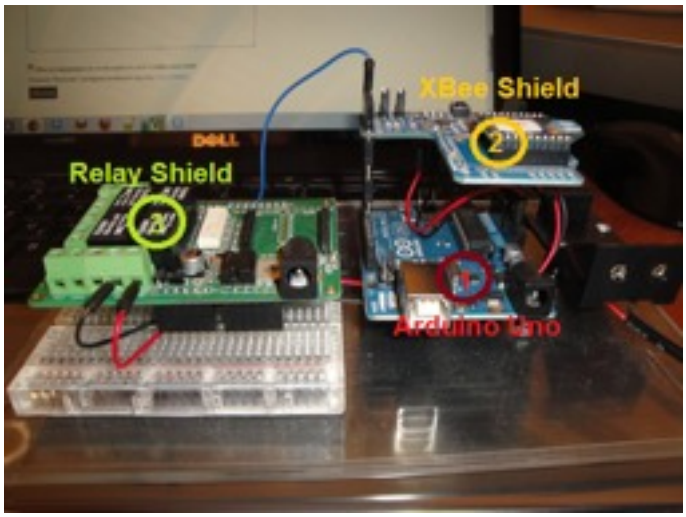
```
void setup(){
  pinMode(2, OUTPUT);
  Serial.begin(9600);
}
```

Στο παρακάτω κομμάτι που εκτελείται συνέχεια στο Arduino πραγματοποιείται ο έλεγχος αν υπάρχει κάποιο πακέτο στο ασύρματο κανάλι. Αν όντως υπάρχει, τότε το Arduino το διαβάζει. Αν είναι ο χαρακτήρας “H” που έχει προέλθει από την συνάρτηση onPin του κόμβου βάσης Arduino, τότε στον ακροδέκτη 2 του τρέχοντος Arduino εφαρμόζεται τάση 5V. Μετά αποστέλλεται προς τον κόμβο βάσης ο χαρακτήρας “K” αν όλα έχουν υλοποιηθεί σωστά και ο σταθμός βάσης εκτυπώνει το μήνυμα επιτυχίας που αναφέρθηκε στην παράγραφο ανάλυσης του. Αντίστοιχα αν ο χαρακτήρας που λαμβάνεται από τον κόμβο ενεργοποίησης είναι ο “L” τότε αφαιρείται η τάση των 5V από τον ακροδέκτη 2 του τρέχοντος Arduino και εκπέμπεται πίσω στον σταθμό βάσης ο χαρακτήρας “K” ότι όλα πήγαν καλά που με την σειρά του εκτυπώνει το αντίστοιχο μήνυμα στον πελάτη.

```
void loop(){
  if (Serial.available() > 0){

        switch (Serial.read()) {
         case 'H':
           digitalWrite(2, HIGH);
            Serial.print('K');
           break;
         case 'L':
           digitalWrite(2, LOW);
            Serial.print('K');
           break;
       }
  }
}
```

Αυτό που συμβαίνει με τον παραπάνω κώδικα είναι να εφαρμόζεται τάση 5V στον ακροδέκτη 2 του τρέχοντος Arduino ή να αφαιρείται η τάση από αυτόν. Σε αυτόν

τον ακροδέκτη είναι συνδεδεμένος ο ακροδέκτης ελέγχου των ρελέ που βρίσκονται στην πλακέτα Relay Shield. Όταν εφαρμόζεται τάση τότε το στοιχείο ρελέ κλείνει κύκλωμα και εφαρμόζεται η τάση των 230V στα φωτιστικά σώματα. Όταν αφαιρείται η τάση από αυτόν τον ακροδέκτη τότε το στοιχείο ρελέ ανοίγει και σταματάει να εφαρμόζεται η τάση των 230V στα φωτιστικά σώματα. Με αυτόν τον τρόπο προκαλείται ενεργοποίηση και απενεργοποίηση των φωτιστικών σωμάτων με χρήση αιτημάτων πρωτοκόλλου Internet http.



Εικόνα 51: Υλοποίηση Κόμβου Ενεργοποίησης.

Προγραμματισμός Ασύρματης Μονάδας XBee

Στην υλοποίηση της εφαρμογής χρησιμοποιήθηκε η μονάδα XBee Series 1, η οποία παρέχει βασική ασύρματη επικοινωνία σημείο σε σημείο κάνοντας χρήση του πρωτοκόλλου χαμηλής κατανάλωσης ZigBee. Σε ένα δίκτυο ZigBee υπάρχει ένας συντονιστής (cordinator) και διάφοροι άλλοι κόμβοι όπως δρομολογητές ή τελικοί σταθμοί. Οι υποχρεώσεις του συντονιστή είναι να δίνει μορφή στο δίκτυο, να κρατάει τις διευθύνσεις και να διαχειρίζεται και άλλες λειτουργίες του δικτύου όπως η ασφάλεια και η εξασφάλιση καλής λειτουργίας του. Επίσης ο συντονιστής πρέπει να είναι μοναδικός σε ένα δίκτυο ZigBee. Στην εφαρμογή ευφυούς οικείας, συντονιστής του δικτύου είναι η μονάδα XBee που βρίσκεται ενσωματωμένη στον σταθμό του Internet, ενώ τελικός σταθμός είναι η μονάδα που βρίσκεται ενσωματωμένη στον σταθμό ενεργοποίησης.

Για τον προγραμματισμό της ασύρματης μονάδας XBee που παρέχει την ασύρματη επικοινωνία ανάμεσα στον σταθμό Internet και τον κόμβο ενεργοποίησης πραγματοποιήθηκε η χρήση της πλακέτας XBee Shield. Αυτή η πλακέτα ενσωματώνει την μονάδα XBee στο Arduino Uno και έτσι παρέχεται η δυνατότητα προγραμματισμού της μονάδας XBee μέσω της θύρας USB. Είναι πολύ σημαντικό ότι για να μπορέσει αυτό να γίνει εφικτό θα πρέπει πρώτα να αφαιρεθεί ο ολοκληρωμένος μικροελεγκτής του Arduino Uno, ATMEGA. Αυτό συμβαίνει διότι με το που τροφοδοτηθεί το Arduino με τάση απαντάει αμέσως ο φορτωτής εκκίνησης (bootloader) που διαθέτει ο μικροελεγκτής με αποτέλεσμα αυτό να δημιουργεί πρόβλημα κατά τον προγραμματισμό της μονάδας XBee. Εναλλακτικά αν δεν επιθυμείται η αφαίρεση του μικροελεγκτή υπάρχει η δυνατότητα φόρτωσης

του παρακάτω σχεδιαγράμματος στο Arduino το οποίο παρακάμπτει την απάντηση του bootloader.

```
void setup ( ) {
}
void loop ( ) {
}
```

Για τον προγραμματισμό της μονάδας XBee απαιτείται ένα πρόγραμμα τερματικού για να παρέχεται η δυνατότητα παραμετροποίησης παραμέτρων στην μονάδα XBee. Στην παρούσα διπλωματική χρησιμοποιήθηκε το X-CTU. Κάθε πομποδέκτης XBee έχει τυπωμένο στην κάτω πλευρά του έναν αριθμό ο οποίος υποδηλώνει την 64-bit διεύθυνση του. Αυτός ο αριθμός αποτελείται από δύο μέρη. Το πρώτο μέρος είναι η αρχική διεύθυνση όπου σε όλες τις μονάδες XBee είναι η ίδια. Αυτή η διεύθυνση είναι η 0013A200. Το δεύτερο κομμάτι της διεύθυνσης είναι μοναδικό για κάθε πομποδέκτη και αποτελεί το αναγνωριστικό του.

Αρχικά παραμετροποιείται ο συντονιστής, συνδέοντας τον στην πλακέτα XBee Shield που βρίσκεται πάνω στο Arduino. Έπειτα συνδέεται η διάταξη στον υπολογιστή μέσω της θύρας USB και με την βοήθεια του X-CTU τροποποιείται η μονάδα XBee έτσι ώστε να βρεθεί σε κατάσταση προγραμματισμού. Αυτό συμβαίνει γράφοντας στο πρόγραμμα τερματικού την εντολή +++ και αναμένεται η απόκριση OK που σημαίνει ότι η μονάδα βρίσκεται στην κατάσταση προγραμματισμού. Το επόμενο βήμα είναι η ανάθεση ταυτότητας στο δίκτυο που κατασκευάζεται μεταξύ των δύο μονάδων XBee. Αυτό υλοποιείται με την εντολή ATID 3332 και αυτό το νούμερο είναι κοινό και για τις δύο μονάδες XBee. Το επόμενο βήμα είναι η δήλωση της αρχικής διεύθυνσης προορισμού και αυτό υλοποιείται με την εντολή ATDH 0013A200. Έπειτα ακολουθεί η ανάθεση της διεύθυνσης αναγνωριστικού του προορισμού. Αυτό για τον συντονιστή γίνεται με χρήση της εντολής ATDL 4071E794 που είναι η διεύθυνση αναγνωριστικού του τελικού σταθμού. Τέλος για να αποθηκευτούν οι αλλαγές που έγιναν παραπάνω πρέπει να εκτελεστεί η εντολή ATWR και να ληφθεί η απάντηση OK αν όλα εκτελέστηκαν σωστά.

Όσον αφορά τον τελικό σταθμό όλες οι εντολές και οι παράμετροι θα πρέπει να είναι ίδιοι με του συντονιστή για να ανήκει στο ίδιο δίκτυο με αυτόν. Το μόνο που θα αλλάξει στην πλευρά του τελικού σταθμού θα είναι η διεύθυνση αναγνωριστικού του προορισμού που αυτή τη φορά θα είναι η διεύθυνση του συντονιστή. Αυτό θα υλοποιηθεί με την εντολή ATDL 4071E791.

Αναλυτικά οι εντολές που εκτελέστηκαν στον συντονιστή και στον τελικό σταθμό μαζί με τις επιθυμητές αποκρίσεις τους είναι:

Συντονιστής

```
+++
OK
ATID 3332
OK
ATDH 0013A200
OK
ATDL 4071E794
OK
ATID
```

3332
ATDH
0013A200
ATDL
4071E794
ATWR
OK

Τελικός Σταθμός

+++
OK
ATID 3332
OK
ATDH 0013A200
OK
ATDL 4071E791
OK
ATID
3332
ATDH
0013A200
ATDL
4071E791
ATWR
OK

Δύο πολύ σημαντικά σημεία που πρέπει να δοθεί ιδιαίτερη προσοχή :

- Το πρώτο σημαντικό σημείο είναι ότι μετά από οποιοδήποτε σύνολο αλλαγών που έχουν υλοποιηθεί στις ασύρματες μονάδες XBee πρέπει να ακολουθεί αποθήκευση με χρήση της εντολής ATWR, αλλιώς με την απομάκρυνση της τροφοδοσίας θα χαθούν.
- Το δεύτερο σημείο είναι ότι όταν η μονάδα XBee βρίσκεται στην κατάσταση προγραμματισμού τότε αν δεν παρέχεται κάποια είσοδο από το πληκτρολόγιο στην γραμμή εντολών του τερματικού ή δεν πραγματοποιηθεί η ολοκλήρωση της εντολής που απαιτείται να εκτελεστεί για χρονικό διάστημα μεγαλύτερο του ενός δευτερολέπτου τότε θα πρέπει να ξαναεκτελείται η εντολή από την αρχή. Επίσης αν περάσουν δέκα δευτερόλεπτα σιγής εισόδου τότε η μονάδα XBee βγαίνει από την κατάσταση προγραμματισμού.

Επίλογος – Συμπεράσματα

Στο Κεφάλαιο 4 αναλύθηκε διεξοδικά η λογική σχεδιασμού καθώς και η υλοποίηση σε επίπεδο διατάξεων της κατασκευής του Ασύρματου Δικτύου που χρησιμοποιείται από την εφαρμογή ευφυούς οικείας. Συγκεκριμένα σε πρώτο

επίπεδο η σχεδιαστική λογική του δικτύου βασίστηκε στην εξόρυξη ροής ακατέργαστων δεδομένων από το Ασύρματο Δίκτυο Αισθητήρων που παρέχει την υποδομή παρακολούθηση περιβαλλοντολογικών συνθηκών ενός χώρου και έπειτα πραγματοποιήθηκε η εισαγωγή αυτών των δεδομένων σε μια βάση δεδομένων MySQL. Η παροχή δεδομένων από το Ασύρματο Δίκτυο πραγματοποιήθηκε με την υλοποίηση δύο προγραμμάτων για τους ασύρματους κόμβους Tmote Sky. Το πρώτο πρόγραμμα (Base Station) αφορούσε την υλοποίηση ενός σταθμού συλλογής δεδομένων από το ασύρματο κανάλι και χρησιμοποιήθηκε αυτούσιο από την διανομή του λειτουργικού συστήματος TinyOS. Το δεύτερο πρόγραμμα που υλοποιήθηκε αφορούσε την δειγματοληψία δύο αισθητήρων που βρίσκονται ενσωματωμένοι στον ασύρματο κόμβο Tmote Sky και την εκπομπή αυτών των μετρήσεων στο ασύρματο κανάλι που παρακολουθεί ο σταθμός συλλογής δεδομένων. Το πρόγραμμα αυτό προέκυψε από έναν αριθμό παραμετροποιήσεων που πραγματοποιήθηκαν στην εφαρμογή Oscilloscope που παρέχει η διανομή λειτουργικού συστήματος TinyOS.

Σε δεύτερο επίπεδο υλοποιήθηκε το Ασύρματο Δίκτυο που παρέχει την δυνατότητα στον χρήστη για απομακρυσμένη παρέμβαση σε ηλεκτρικές συσκευές και συγκεκριμένα σε φωτιστικά σώματα. Αυτό κατέστει δυνατό με χρήση της πλατφόρμας ανοικτού υλικού Arduino. Δημιουργήθηκαν δύο κόμβου με διαφορετικούς ρόλους. Ο πρώτος κόμβος υλοποιήθηκε για να παρέχει την δυνατότητα επικοινωνίας της διάταξης με τον διαδίκτυο έτσι ώστε να μπορεί ο χρήστης να επενεργεί σε αυτή απομακρυσμένα. Ο δεύτερος κόμβος παρέχει την δυνατότητα ελέγχου ενός φωτιστικού σώματος κάνοντας χρήση διατάξεων οδήγησης διακοπτών υψηλών τάσεων (ρελέ). Οι δύο κόμβοι που υλοποιήθηκαν με χρήση της πλατφόρμας Arduino έχουν την δυνατότητα να επικοινωνούν μεταξύ τους ασύρματα προσδίδοντας με αυτόν τον τρόπο ευελιξία στην δομή του δικτύου.

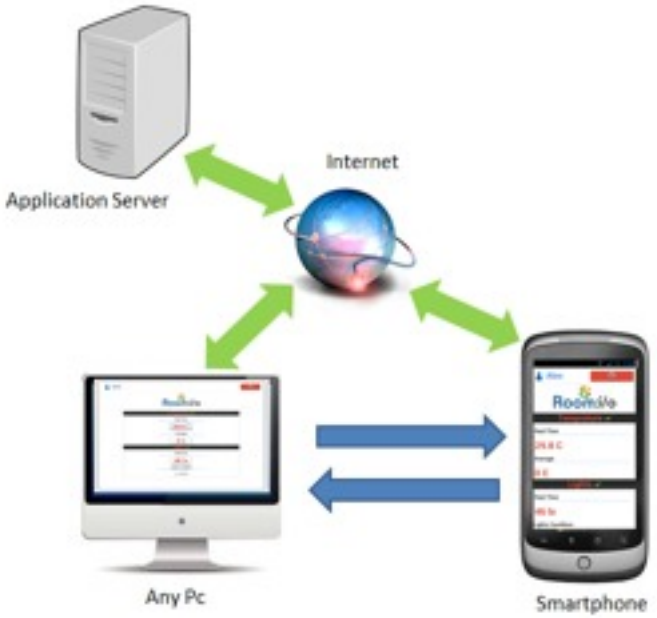
Τα συμπεράσματα που προκύπτουν από αυτό το κεφάλαιο είναι το μεγάλο εύρος σε δυνατότητες που παρέχει το λειτουργικό σύστημα TinyOS καθώς και οι ασύρματοι κόμβοι Tmote Sky, και η ευελιξία σε επίπεδο διατάξεων της πλατφόρμας Arduino που κάνει χρήση της λογικής επεκτάσιμων δομοστοιχείων. Με βάση τους ανωτέρω λόγους προκύπτει η σπουδαιότητα του κλάδου των Δικτύων Ασύρματων Αισθητήρων, καθώς και η ραγδαία ανάπτυξη που γνωρίζουν, διότι μπορούν να παρέχουν πληθώρα εφαρμογών είτε αυτοματισμών, είτε ευκολιών, είτε παρακολούθησης συνθηκών.

Στο Κεφάλαιο 5 που ακολουθεί πραγματοποιείται η ανάπτυξη σε επίπεδο προγραμματιστικών τεχνικών που υλοποιεί τις διεπαφές του χρήστη της εφαρμογής ευφυούς οικείας. Επίσης παρατίθεται ο τρόπος που χρησιμοποιήθηκε για την επίτευξη της ενοποίησης των διαφορετικών Ασύρματων Δικτύων που κατασκευάστηκαν επιτυχώς στο παρόν κεφάλαιο, έτσι ώστε να παρέχεται στον χρήστη διαφανείς λειτουργία της εφαρμογής.

ΚΕΦΑΛΑΙΟ 5: Κατασκευή Διεπαφών Χρήστη

Σχεδιαστική Λογική

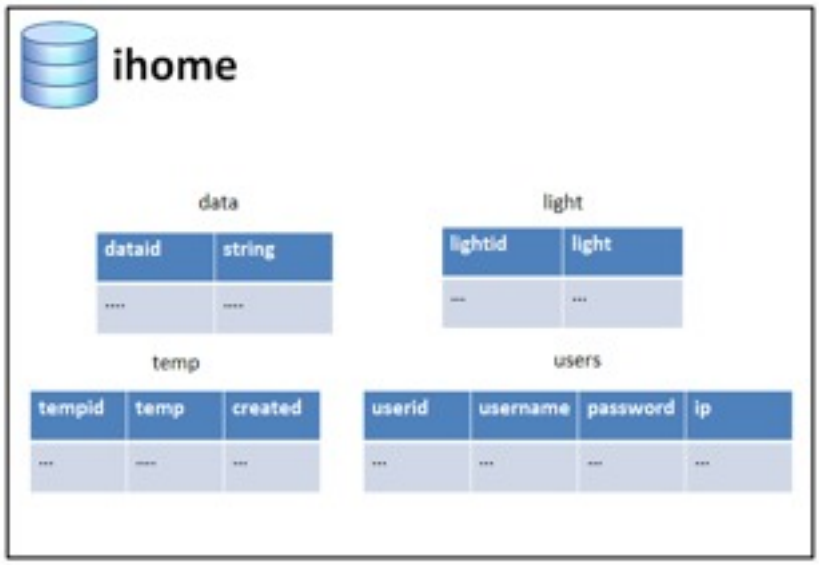
Η σχεδιαστική λογική που εφαρμόστηκε για να την υλοποίηση των διεπαφών του χρήστη βασίστηκε στις απαιτήσεις για φορητότητα, και ευκολία χρήσης. Η κεντρική ιδέα που στηρίζονται οι διεπαφές έχει να κάνει με την πραγματοποίηση άντλησης των ακατέργαστων δεδομένων από τον κεντρικό διακομιστή της εφαρμογής ευφυούς οικείας που αφορούν τις φυσικές παραμέτρους ενός χώρου, την επεξεργασία των δεδομένων αυτών, την απεικόνιση τους στον χρήστη και την παροχή δυνατότητας ελέγχου ηλεκτρικών συσκευών. Επίσης στόχος ήταν η σύγχρονη επικοινωνία και αλληλεπίδραση μεταξύ των επιμέρους ασύρματων δικτύων που συνθέτουν την εφαρμογή, καθώς και ο συγχρονισμός μεταξύ των δύο διεπαφών έτσι ώστε να προκύπτει διαφανής λειτουργία και ενοποίηση όλων των τεχνολογιών. Στην εικόνα 52 παρατίθεται σχηματικά η λειτουργίας των διεπαφών του τελικού χρήστη.



Εικόνα 52: Σχεδιαστική λογική διεπαφών τελικού χρήστη.

Για την επίτευξη των ανωτέρω σχεδιαστικών στόχων έγινε χρήση προγραμματιστικών τεχνικών Διαδικτύου με σκοπό την υλοποίηση μιας διαδικτυακής εφαρμογής που θα παρέχει πρόσβαση στον χρήστη από οποιοδήποτε υπολογιστή που διαθέτει πρόσβαση στο Διαδίκτυο και προγραμματιστικών τεχνικών λειτουργικού συστήματος Android με σκοπό την υλοποίηση μιας εφαρμογής που θα εκτελείται από το έξυπνο κινητό τηλέφωνο που θα διαθέτει ο χρήστης.

Το πρωταρχικό στάδιο ήταν η κατασκευή της βάσης δεδομένων της εφαρμογής. Η υλοποίηση πραγματοποιήθηκε με χρήση του εργαλείου phpMyadmin, η γλώσσα που χρησιμοποιήθηκε για τα ερωτήματα στην βάση ήταν η MySQL και το σχήμα της βάσης δεδομένων προέκυψε αυτό που απεικονίζεται στην εικόνα 53. Όπως απεικονίζεται και στην εικόνα 53 η βάση αποτελείται από τέσσερεις πίνακες. Ο πίνακας data αποθηκεύει τα ακατέργαστα δεδομένα από τον Σταθμό Βάσης Tmote Sky του δικτύου αισθητήρων, ο πίνακας temp αποθηκεύει τα επεξεργασμένα δεδομένα που προκύπτουν από την εξόρυξη και μορφοποίηση των μετρήσεων θερμοκρασίας καθώς και μια χρονική σφραγίδα που άφορα την στιγμή που πραγματοποιήθηκε η μέτρηση, ο πίνακας light αποθηκεύει τα δεδομένα που προκύπτουν από την εξόρυξη και μορφοποίηση των μετρήσεων έντασης φωτεινότητας και τέλος ο πίνακας users αποθηκεύει δεδομένα που αφορούν τους εγγεγραμμένους χρήστες της εφαρμογής.



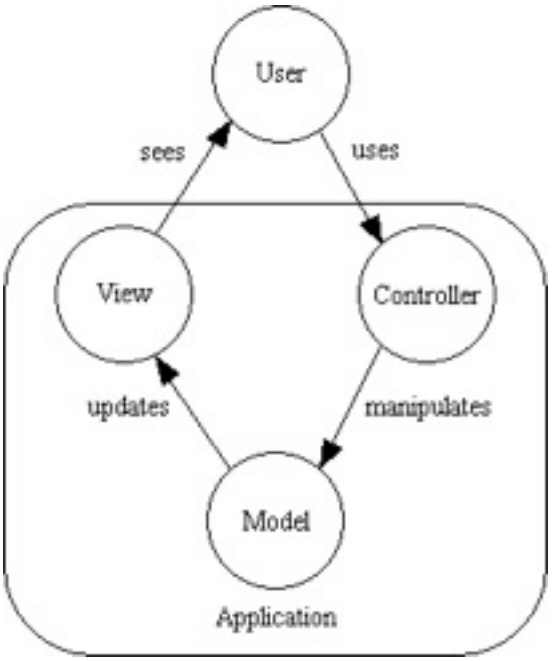
Εικόνα 53:Βάση Δεδομένων.

Διαδικτυακή Εφαρμογή Room: i/o

Για την ανάπτυξη της Διαδικτυακής Εφαρμογής Room: i/o που αποτελεί την πρώτη διεπαφή του χρήστη έγινε χρήση προγραμματιστικών τεχνικών σε γλώσσα PHP. Συγκεκριμένα χρησιμοποιήθηκε το σχεδιαστικό πρότυπο M.V.C. το οποίο παρέχει δομημένη μορφή στον τρόπο λειτουργίας της Διαδικτυακής Εφαρμογής. Τα αρχικά M.V.C. παραπέμπουν στους όρους Model, View, Controller που αποτελούν τις λογικές οντότητες του τρόπου λειτουργίας της Διαδικτυακής Εφαρμογής και αναλύονται παρακάτω:

- **Model**: Το μοντέλο είναι ένα αντικείμενο το οποίο αντιπροσωπεύει δεδομένα όπως για παράδειγμα του πίνακες μιας βάσης δεδομένων. Ο ρόλος του είναι να διαχειρίζεται την συμπεριφορά της Διαδικτυακής Εφαρμογής καθώς και τα δεδομένα που αυτή χρησιμοποιεί. Επίσης το μοντέλο δεν διαθέτει γνώση για του ελεγκτές (controllers) και τις προβολές (views) που το χρησιμοποιούν.
- **View**: Μια προβολή είναι σαν μια κατάσταση στην οποία βρίσκεται το μοντέλο (model) την παρούσα χρονική στιγμή. Διαχειρίζεται την απεικόνιση ολόκληρου του γραφικού περιβάλλοντος της Διαδικτυακής Εφαρμογής κάνοντας χρήση δεδομένων από το μοντέλο (model) της εφαρμογής.
- **Controller**: Ο ελεγκτής προσφέρει τις δυνατότητες της αλλαγής στην κατάσταση του μοντέλου (model) και της προβολής. Ο ελεγκτής ερμηνεύει τις ενέργειες από το ποντίκι ή το πληκτρολόγιο του χρήστη έτσι ώστε να αλλάζει τις καταστάσεις του μοντέλου (model) κατά την περίσταση. Δηλαδή παρέχει τα μέσα για την αλληλεπίδραση του χρήστη με την εφαρμογή. Αυτό συμβαίνει με την μετάφραση των εισόδων από τις προβολές (view) της εφαρμογής σε αλλαγές κατάστασης.

Διαγραμματικά οι αλληλεπιδράσεις των αντικειμένων αναπαρίστανται στην εικόνα 54.



Εικόνα 54: Λογική σχεδιαστικού προτύπου MVC.

Σύμφωνα με τα ανωτέρω η δομή της Διαδικτυακής Εφαρμογής αποτελείται από έξι φακέλους, τον φάκελο models, ο οποίος περιέχει τα μοντέλα της εφαρμογής, τον φάκελο views, ο οποίος περιέχει τις προβολές της εφαρμογής, τον φάκελο controllers, οποίος περιέχει τους ελεγκτές της εφαρμογής, τον φάκελο css που περιέχει τα αρχεία με τους κανόνες μορφοποίησης που θα χρησιμοποιηθούν από τις προβολές τις εφαρμογής, τον φάκελος js που περιέχει τα αρχεία με τις προγραμματιστικές τεχνικές σε γλώσσα εκτελέσιμου κώδικα JavaScript και τέλος τον φάκελο images που περιέχει διάφορα γραφικά στοιχεία που χρησιμοποιούνται από τις προβολές για την κατασκευή του γραφικού περιβάλλοντος της εφαρμογής. Η δομή του φακέλου της εφαρμογής απεικονίζεται στην εικόνα 55.

Εικόνα 55: Δομή φακέλου Διαδικτυακής Εφαρμογής.

Στόχοι και καταστάσεις Διαδικτυακής Εφαρμογής

Στόχοι

Ο πρώτος στόχος της Διαδικτυακής εφαρμογής είναι να δημιουργηθεί μια σύνδεση με την βάση δεδομένων MySQL στην οποία εισέρχονται τα ακατέργαστα δεδομένα από τον Σταθμό Βάσης που έχει υλοποιηθεί με χρήση της πλατφόρμας Tmote Sky. Για επιτευχθεί αυτός ο στόχος δημιουργήθηκε ένα μοντέλο με όνομα αρχείου connection.php που παρέχει αυτή την σύνδεση:

Connection.php

```
<?php
include 'constants.php';

    $connection = mysql_connect(DB_SERVER,DB_USER,DB_PASS);
    if (!$connection) {
        die("Database connection failed: " . mysql_error());
    }

    $db_select = mysql_select_db(DB_NAME,$connection);
    if (!$db_select) {
        die("Database selection failed: " . mysql_error());
    }
?>
```

Στο ανωτέρω απόσπασμα δημιουργήθηκαν δύο μεταβλητές εκ των οποίον η πρώτη παρέχει την σύνδεση με την βάση δεδομένων και η δεύτερη την επιλογή της συγκεκριμένης βάσης δεδομένων που αποθηκεύονται τα ακατέργαστα δεδομένα. Επίσης στο αρχείο αυτό εισάγεται και ένα επιπλέον αρχείο με όνομα constants.php που παρέχει τις σταθερές που χρειάζονται οι συναρτήσεις mysql_connect και mysql_select_db οι οποίες είναι οι ακόλουθες:

Constants.php

```
<?php
define("DB_SERVER", "localhost");
define("DB_USER", "root");
define("DB_PASS", "trustno1");
define("DB_NAME", "ihome");
?>
```

Ο επόμενος στόχος είναι να δημιουργηθεί ένα επιπλέον μοντέλο το οποίο θα διαβάζει τα ακατέργαστα δεδομένα από τον αρχικό πίνακα της βάσης δεδομένων που έχουν εισαχθεί, θα τα διαχωρίζει και θα τα μορφοποιεί από δεκαεξαδική μορφή σε δεκαδική και τέλος θα τα εισάγει σε δύο καινούργιους πίνακες της βάσης δεδομένων όπου ο πρώτος πίνακας θα αφορά τις μετρήσεις τις θερμοκρασίας του χώρου και ο δεύτερος τις μετρήσεις της έντασης της φωτεινότητας. Τα ανωτέρω υλοποιούνται από το μοντέλο parse.php που ακολουθεί.

Parse.php

```
<?php
    include 'connection.php';
```

```
        $res = mysql_query(
            "SELECT
                string,dataid
            FROM
                data
            ORDER BY dataid DESC LIMIT 1;"
        );
            if ($row = mysql_fetch_array( $res )){
                $data = $row['string'];
                $nospaces = str_replace(' ', '', $data);
                $lightone = substr( "$nospaces",32,4);
                $lighttow = substr( "$nospaces",36,4);
                $lightthree = substr( "$nospaces",40,4);
                $temperatureon = substr( "$nospaces",44,4);
                $temperaturetw = substr( "$nospaces",48,4);
                $temperatureth = substr( "$nospaces",52,4);
                $decTemp1 = hexdec($temperatureon);
                $decTemp2 = hexdec($temperaturetw);
                $decTemp3 = hexdec($temperatureth);
                $decLight1 = hexdec($lightone);
                $decLight2 = hexdec($lighttwo);
                $decLight3 = hexdec($lightthree);
                $tempr = ((($decTemp1+$decTemp2+$decTemp3)/
3)*0.01)-39.6);
                $lightt = ((($decLight1+$decLight2+$decLight3)/
3)*2.288);

                mysql_query(
                    "INSERT INTO
                        temp
                    SET
                        temp = '$tempr',
                        created = NOW();"
                );

                    mysql_query(
                        "INSERT INTO
                            light
                        SET
                            light = '$lightt';"
                    );

                        mysql_query(
                        "TRUNCATE data;"
                    );
            }
?>
```

Στο ανωτέρω αρχείο, αρχικά γίνεται χρήση του μοντέλου που δημιουργήθηκε για την σύνδεση στη βάση δεδομένων της εφαρμογής, έπειτα ακολουθούν οι συναρτήσεις που εκτελούν τα ερωτήματα ως προς τον αρχικό πίνακα της βάσης, υλοποιείται η μορφοποίηση των ακατέργαστων δεδομένων, υλοποιείται η εισαγωγή των μορφοποιημένων δεδομένων στους καινούργιους πίνακες και τέλος προκαλείται εξαναγκαστικό άδειασμα των εγγραφών του αρχικού πίνακα με τα ακατέργαστα δεδομένα έτσι ώστε να εξαλειφτεί η πιθανότητα επεξεργασίας των ίδιων δεδομένων παραπάνω από μια φορές.

Σε αυτό το σημείο πρέπει να σημειωθεί το γεγονός ότι το μοντέλο parse.php εκτελείται κάθε ένα δευτερόλεπτο από την εφαρμογή διότι αυτός είναι ο ρυθμός αποστολής των ακατέργαστων δεδομένων από τον Σταθμό Συλλογής Φυσικών

Παραμέτρων προς τον Σταθμό Βάσης. Για να γίνει εφικτή η ανωτέρω διαδικασία δημιουργήθηκε το αρχείο refresh.html το οποίο πρέπει να είναι συνεχώς ανοικτό στον διακομιστή της εφαρμογής έτσι ώστε να παρέχεται η ενημέρωση στους πίνακες της βάσης δεδομένων.

Refresh.html

```
<html>
    <head>
        <script type="text/javascript" src="/ihome/
jquery-1.7.min.js"></script>

            <script type="text/javascript">
                setInterval( function () {
                $.post( './models/parse.php' );
                }, 1000 );
            </script>
    </head>
        <body>
            <p>Keep this page open to refresh database.</p>
        </body>
</html>
```

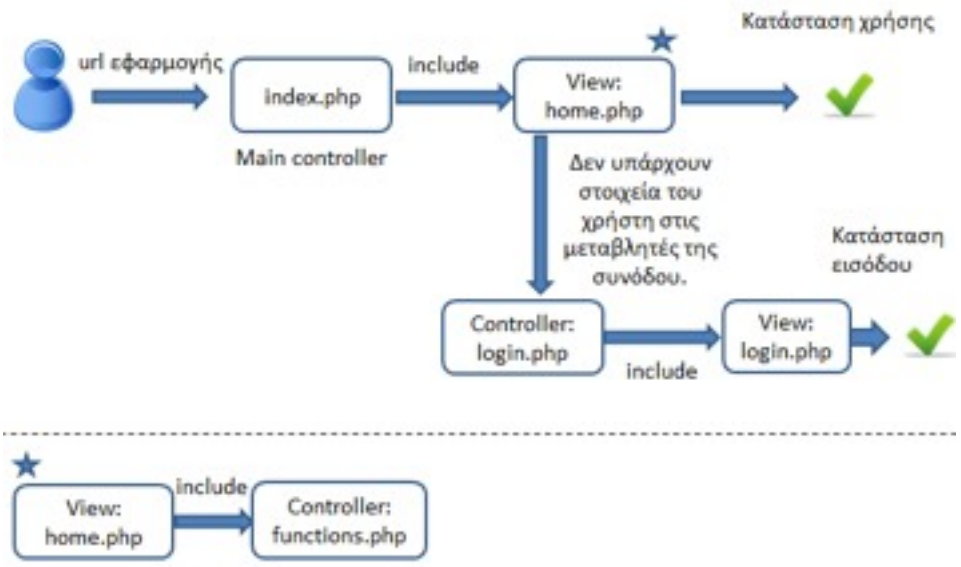
Το αρχείο refresh.html έχει υλοποιηθεί με χρήση JavaScript και αποτελείται από μια συνάρτηση η οποία καλεί κάθε 1000ms το μοντέλο parse.php.

Καταστάσεις

Οι καταστάσεις της Διαδικτυακής Εφαρμογής προκύπτουν ως απόρροια των συνθηκών χρήσης της εφαρμογής από τον χρήστη. Οι καταστάσεις αυτές είναι οι ακόλουθες:

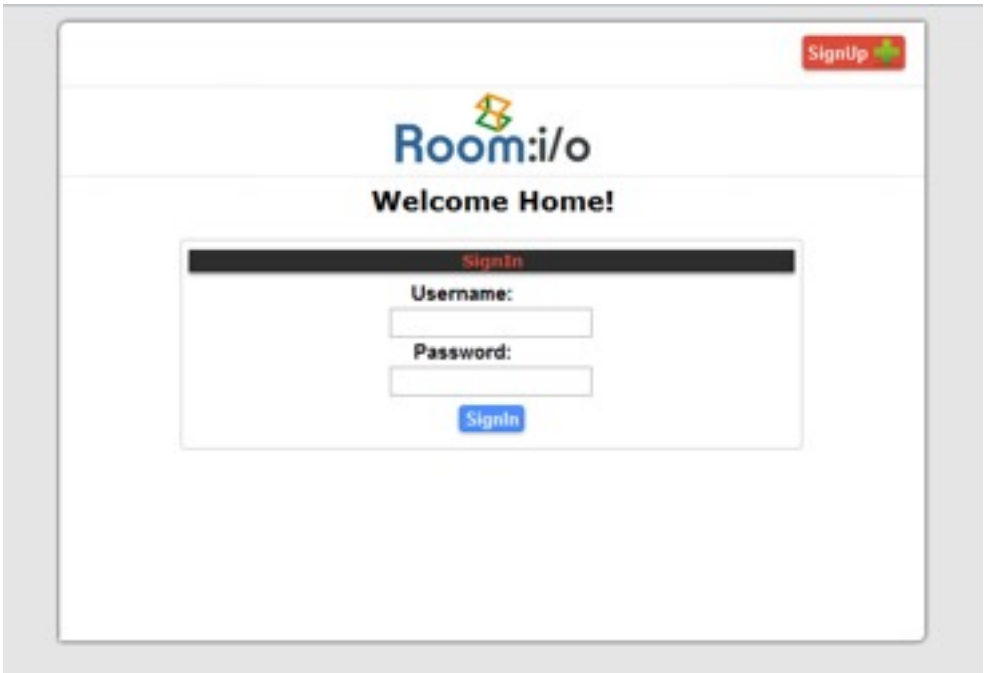
- Κατάσταση εισόδου χρήστη στην εφαρμογή και δημιουργία μνήμης δεδομένων χρήστη με χρήση έναρξης συνόδου.
- Κατάσταση εγγραφής χρήστη στην εφαρμογή.
- Κατάσταση χρήσης της εφαρμογής με παράθεση των απαιτούμενων πληροφοριών και δυνατοτήτων.
- Κατάσταση ελέγχου των περιφερικών δικτύων της εφαρμογής.
- Κατάσταση αποσύνδεσης.

Η λογική που υλοποιεί η Διαδικτυακή Εφαρμογή όταν ένας χρήστης πληκτρολογήσει την διεύθυνση που αντιστοιχεί στον διακομιστή της εφαρμογής απεικονίζεται στην εικόνα 56.



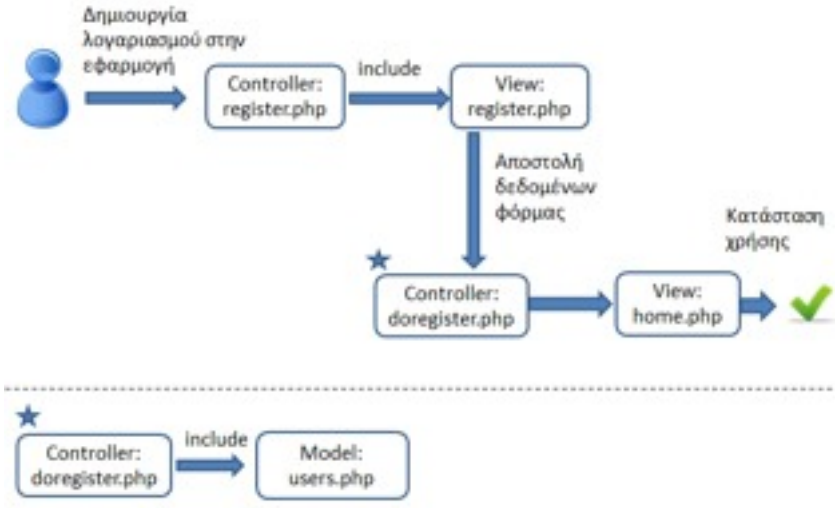
Εικόνα 56: Λογική απόφασης κατάστασης εφαρμογής.

Το αρχικό αίτημα σύνδεσης παραλαμβάνει ο κεντρικός ελεγκτής index.php, ο οποίος ενσωματώνει το αρχείο προβολής home.php. Το αρχείο προβολής home.php με την σειρά του ενσωματώνει το αρχείο ελεγκτή functions.php που του παρέχει δύο χρήσιμες συναρτήσεις. Την συνάρτηση check user και την συνάρτηση create panel. Η πρώτη συνάρτηση ελέγχει αν έχει δημιουργηθεί σύνοδος με τον διακομιστή της εφαρμογής γεγονός που υποδηλώνει αν ο χρήστης είναι συνδεδεμένος ή όχι. Η δεύτερη συνάρτηση υλοποιεί την δημιουργία ενός πλαισίου επιλογών ως προς τον χρήστη, ανάλογα με τις συνθήκες που πληρούνται. Στην αρχική κατάσταση όπου ο χρήστης δεν είναι συνδεδεμένος στην εφαρμογή η συνάρτηση του ελεγκτή functions.php, check user επιστρέφει false στο αρχείο προβολής home.php και με αυτόν τον τρόπο προκαλείται ανακατεύθυνση του χρήστη στον ελεγκτή login.php όπως απεικονίζεται στην εικόνα 56. Όταν ο χρήστης δρομολογηθεί στο αρχείο του ελεγκτή login.php αυτός ενσωματώνει την προβολή login.php με αποτέλεσμα να σχεδιαστεί στον χρήστη η προβολή της σελίδας της κατάστασης εισόδου στην εφαρμογή η οποία απεικονίζεται στην εικόνα 56. Επίσης πρέπει να σημειωθεί ότι και η προβολή login.php κάνει χρήση του ελεγκτή functions.php και συγκεκριμένα της δεύτερης συνάρτησης που αυτός υλοποιεί, δηλαδή της συνάρτησης create panel. Στην προκείμενη κατάσταση η συνάρτηση create panel είναι υπεύθυνη για την δημιουργία του κουμπιού SignUp και την παράθεση του λογοτύπου της εφαρμογής, όπως αυτά απεικονίζονται στην εικόνα 57.



Εικόνα 57: Κατάσταση εισόδου στην εφαρμογή με χρήση του ελεγκτή login.php και της προβολής login.php.

Όταν ο χρήστης βρεθεί στην προβολή login.php όπως απεικονίζεται στην εικόνα 57 έχει την δυνατότητα είτε να συνδεθεί στην εφαρμογή αν έχει εγγραφτεί προηγουμένως, είτε αν είναι καινούργιος χρήστης να επιλέξει το κουμπί που αναγράφει SingUp και να πλοηγηθεί στην προβολή που παρέχει την δυνατότητα δημιουργίας λογαριασμού. Πραγματοποιώντας την επιλογή δημιουργίας λογαριασμού η λογική που εφαρμόζεται στην εφαρμογή είναι να μεταβεί σε κατάσταση εγγραφής όπως απεικονίζεται στην εικόνα 58.



Εικόνα 58: Λογική Κατάστασης Εγγραφής χρήστη.

Ο χρήστης όταν πατήσει το κουμπί SignUp που βρίσκεται στο πλαίσιο επιλογών της προβολής login.php που απεικονίζεται στη εικόνα 57 μεταφέρεται στον ελεγκτή register.php. Στον ελεγκτή register.php ενσωματώνεται η προβολή register.php που απεικονίζεται στην εικόνα 59.

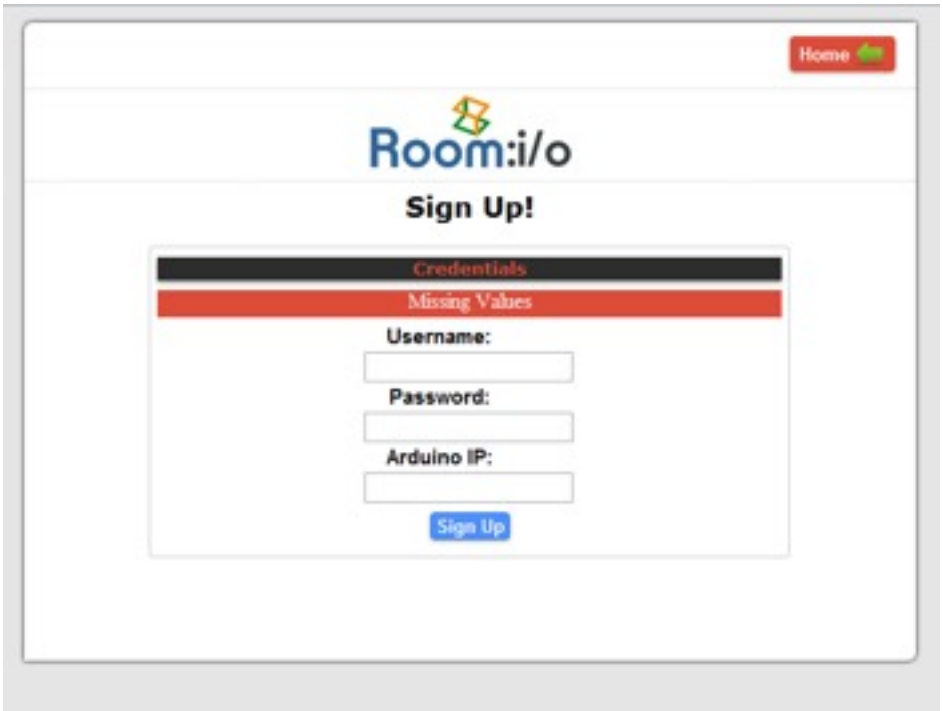
Εικόνα 59: Προβολή register.php για δυνατότητα δημιουργίας λογαριασμού χρήστη.

Στην προβολή register.php ενσωματώνεται ο ελεγκτής fumctions.php για χρήση της συνάρτησης create panel με σκοπό την δημιουργία του πλαισίου επιλογών του χρήστη που στην προκείμενη περίπτωση δημιουργεί εκτός από το λογότυπο της εφαρμογής και ένα κουμπί που δίνει την δυνατότητα στον χρήστη να επιστρέψει στην σελίδα εισόδου της εφαρμογής αν το επιθυμεί.

Συμπληρώνοντας την φόρμα στοιχείων λογαριασμού και επιλέγοντας το κουμπί Sing Up που απεικονίζεται στην εικόνα 59 ο χρήστης δίνει εντολή τα στοιχεία του να αποσταλούν στον διακομιστή της εφαρμογής και συγκεκριμένα στον ελεγκτή doregirster.php όπως αυτό απεικονίζεται στην εικόνα 58. Ο ελεγκτής doregister.php είναι υπεύθυνος για την εγγραφή του χρήστη στην εφαρμογή και την δημιουργία δεδομένων συνόδου έτσι ώστε η εφαρμογή να αποκτήσει μνήμη για την ταυτότητα του χρήστη. Η δημιουργία δεδομένων συνόδου υλοποιείται από την εξόρυξη των στοιχείων που έχει εισάγει ο χρήστης στην φόρμα που απεικονίζεται στην εικόνα 59 και η αποθήκευση τους σε γενικές μεταβλητές τύπου cookie. Επίσης ενσωματώνοντας στον ελεγκτή doregister.php το μοντέλο urers.php από το οποίο καλείται η συνάρτηση create user δημιουργείται η εγγραφή του χρήστη στην βάση δεδομένων της εφαρμογής που αφορά τους χρήστες.

Όλα τα ανωτέρω συμβαίνουν στο παρασκήνιο του διακομιστή της εφαρμογής και το μόνο που αντιλαμβάνεται ο χρήστης όταν συμπληρώσει την φόρμα της προβολής register.php και επιλέξει το κουμπί Sign Up είναι η ανακατεύθυνση του στην προβολή home.php, όπου αυτή την φορά η συνάρτηση check user που έχει ενσωματωμένη η προβολή επιστρέφει true και μεταβάλει την

εφαρμογή σε κατάσταση χρήσης. Αυτό συμβαίνει διότι ο ελεγκτής doregister.php έχει δημιουργήσει πλέον τα απαραίτητα δεδομένα συνόδου για τον συγκεκριμένο χρήστη. Αν όμως ο χρήστης δεν έχει συμπληρώσει κάποιο πεδίο της φόρμας της προβολής register.php αυτό γίνεται αντιληπτό από τον ελεγκτή doregister.php που καλείται με την επιλογή του κουμπιού Sign Up και δρομολογεί τον χρήστη ξανά στον ελεγκτή register.php που με την σειρά του όταν ενσωματώσει την προβολή register.php εμφανίζει ένα μήνυμα σφάλματος όπως απεικονίζεται στην εικόνα 60.



Εικόνα 60: Εντοπισμός σφάλματος από τον ελεγκτή doregister.php.

Όταν ο χρήστης έχει εγγραφεί στην εφαρμογή μέσω της προβολής register.php ή συνδεθεί σε αυτήν μέσω της προβολής login.php η εφαρμογή μεταβάλλεται σε κατάσταση χρήσης. Η κατάσταση χρήσης προκύπτει από την προβολή home.php που ενσωματώνει τον ελεγκτή functions.php. Στην κατάσταση χρήσης η συνάρτηση του ελεγκτή functions.php check user επιστρέφει true στην προβολή home.php με αποτέλεσμα να παρέχονται στον χρήστη οι απαιτούμενες πληροφορίες και δυνατότητες που επιθυμεί. Επίσης η συνάρτηση create panel στην προκείμενη περίπτωση προκαλεί την δημιουργία του πλαισίου επιλογών που απεικονίζεται στην εικόνα 61.

Εικόνα 61: Κατάσταση Χρήσης εφαρμογής.

Στην κατάσταση χρήσης η συνάρτηση του ελεγκτή functions.php, create panel έχει δημιουργήσει στο πλαίσιο επιλογών εκτός από το λογότυπο της εφαρμογής και ένα πεδίο με το όνομα χρήστη και ένα κουπί με επιμέρους επιλογές όπως απεικονίζονται στην εικόνα 61. Επίσης κάτω από το πλαίσιο επιλογών προκαλείται η παράθεση των πληροφοριών της θερμοκρασίας όπου ενημερώνονται σε πραγματικό χρόνο από την βάση δεδομένων της εφαρμογής που αφορά τις μορφοποιημένες μετρήσεις θερμοκρασίας που παράγονται από το μοντέλο parse.php που αναλύθηκε ανωτέρω. Κατά τον ίδιο τρόπο παράγονται και οι μετρήσεις έντασης της φωτεινότητας που παρέχονται κάτω από τις μετρήσεις θερμοκρασίας, όπως απεικονίζεται στην εικόνα 61.

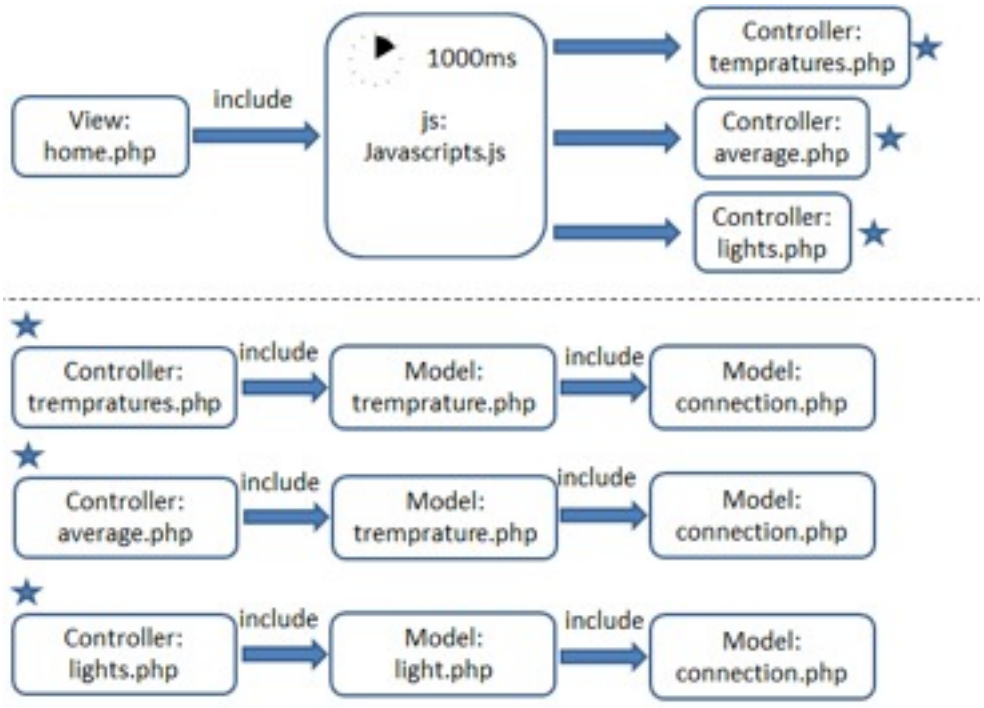
Τέλος παρέχεται στον χρήστη ένα κουμπί με δυική χρήση που απεικονίζεται στην εικόνα 61 κάτω από την τιμή της έντασης φωτεινότητας. Το κουμπί αυτό ενημερώνει τον χρήστη για την κατάσταση που βρίσκονται τα φωτιστικά σώματα που είναι συνδεδεμένα στον Σταθμό Ενεργοποίησης Arduino. Επίσης δίνει την δυνατότητα στον χρήστη να ενεργοποιήσει ή να απενεργοποιήσει τα φωτιστικά σώματα.

Για την υλοποίηση των δυνατοτήτων ενημέρωσης σε πραγματικό χρόνο των μετρήσεων και της κατάστασης των φωτιστικών σωμάτων, καθώς επίσης και την δυνατότητα ενεργοποίησης και απενεργοποίησης η εφαρμογή στην κατάσταση χρήσης ενσωματώνει στην προβολή home.php προγραμματιστικές τεχνικές που έχουν υλοποιηθεί σε γλώσσα JavaScript. Συγκεκριμένα για την ενημέρωση των μετρήσεων έχει ενσωματωθεί το αρχείο javascripts.js που βρίσκεται στον φάκελο js της εφαρμογής και είναι το ακόλουθο.

```
var refreshId = setInterval(function(){
    $('.room1').load('controllers/tempratures.php');
}, 1000);

var refreshId = setInterval(function(){
    $('.average').load('controllers/average.php');
}, 1000);

var refreshId = setInterval( function() {
```

```
    $( '.room2' ).load( 'controllers/lights.php' );
}, 1000 );
```

Με βάση το ανωτέρω απόσπασμα JavaScript προκύπτει η λογική λειτουργίας ενημέρωσης μετρήσεων που απεικονίζεται στην εικόνα 62.



Εικόνα 62: Λογική λειτουργίας ενημέρωσης μετρήσεων.

Αυτό που υλοποιεί το απόσπασμα JavaScript που βρίσκεται ενσωματωμένο στην προβολή home.php κατά την λειτουργία χρήσης είναι να καλεί κάθε 1000ms τους ελεγκτές tempratures.php, average.php και lights.php αντίστοιχα. Οι ελεγκτές αυτοί ενσωματώνουν την χρήση των μοντέλων temperature.php και light.php όπως απεικονίζεται στην εικόνα 62. Τα μοντέλα αυτά επίσης ενσωματώνουν το μοντέλο connection.php που τους παρέχει την δυνατότητα σύνδεσης με την βάση δεδομένων της εφαρμογής. Το μοντέλο temperature.php παρέχει τρείς συναρτήσεις, την confirm query, την get temp και την get tempaver. Η πρώτη συνάρτηση επιβεβαιώνει την εγκυρότητα του ερωτήματος προς την βάση δεδομένων της εφαρμογής, η δεύτερη συνάρτηση προκαλεί εξόρυξη των μορφοποιημένων μετρήσεων θερμοκρασίας του χώρου και η τρίτη συνάρτηση παρέχει τον μέσο όρο των μετρούμενων θερμοκρασιών για τις προηγούμενες τρείς ώρες. Οι πρώτες δύο συναρτήσεις του μοντέλου temprature.php παρέχουν στον ελεγκτή tempratures.php κάθε ένα δευτερόλεπτο τις απαιτούμενες μετρήσεις θερμοκρασίας όπου με την σειρά του παρέχει τα αποτελέσματα τους μέσω του αρχείου javascripts.js στην προβολή home.php. Αντίστοιχα ο ελεγκτής average.php κάνει χρήση των συναρτήσεων confirm query και get tempaver του μοντέλου temperature.php έτσι ώστε να επιστρέφει κάθε ένα δευτερόλεπτο μέσω του αρχείου javascripts.js στην προβολή home.php τον μέσο όρο των μετρούμενων θερμοκρασιών. Τέλος ο ελεγκτής lights.php ενσωματώνει το μοντέλο light.php που του παρέχει δύο συναρτήσεις την confirm query και την get light. Η πρώτη συνάρτηση υλοποιεί ότι ακριβώς και η αντίστοιχη συνάρτηση του μοντέλου temperature.php, η δεύτερη συνάρτηση μέσω του μοντέλου connection.php που

ενσωματώνεται στο μοντέλο light.php παρέχει εξόρυξη μορφοποιημένων μετρήσεων έντασης φωτεινότητας από την βάση δεδομένων της εφαρμογής οι οποίες επιστρέφονται στην προβολή home.php μέσω του αρχείου javascripts.js κάθε ένα δευτερόλεπτο.

Επίσης ο ελεγκτής lights.php που καλείται από το αρχείο javascripts.js κάθε ένα δευτερόλεπτο παρέχει και την υλοποίηση μιας μορφής αυτοματισμού για την εφαρμογή. Συγκεκριμένα στον ελεγκτή lights.php πραγματοποιείται έλεγχος της μέτρησης της φωτεινότητας σε πραγματικό χρόνο και σε περίπτωση που ανιχνευθεί μέτρηση χαμηλότερη μιας κρίσιμης τιμής αποστέλλει αυτόματα εντολή προς το δίκτυο που παρέχει την δυνατότητα ενεργοποίησης των φωτιστικών σωμάτων να τα ενεργοποιήσει.

Για την υλοποίηση της δυνατότητας ενημέρωσης της κατάστασης των φωτιστικών σωμάτων, καθώς και την ενεργοποίηση ή απενεργοποίηση τους η προβολή home.php ενσωματώνει ένα ακόμα απόσπασμα εκτελέσιμου κώδικα JavaScript το οποίο παρατίθεται ακολούθως.

```
<script type="text/javascript">
        $( "#imgon" ).hide();
        $( "#imgoff" ).hide();

        var refreshId = setInterval( function() {
           $.get('./controllers/status1.php', function(data) {
            if (data == "OFF") {
                $( "#imgon" ).hide();
                $( "#imgoff" ).show();
                stateToggle('1');
            } else if (data == "ON") {
                $( "#imgon" ).show();
                $( "#imgoff" ).hide();
                stateToggle('2');
            }
            });
        }, 1000 );

        function stateToggle(data){
            if ( data == "1" ){
                $('#imgoff').click(function() {
                    $.get( 'http://<?php echo $_SESSION['userip']?>/?
2' );
                    $( "#imgoff" ).hide();
                    $( "#imgon" ).show();
                });
            } else if (data == "2"){
                $('#imgon').click(function() {
                    $.get( "http://<?php echo $_SESSION['userip']?>/?
3" );
                    $( "#imgon" ).hide();
                    $( "#imgoff" ).show();
                });
            }
        }

        $('#imgon').click(function() {
                    callbacka();
                });
        $('#imgoff').click(function() {
                    callbackb();
                });
```

```
        function callbacka() {
                var ua = navigator.userAgent.toLowerCase();
                var isAndroid = ua.indexOf( "android" ) > -1;
                if ( isAndroid ) {
                    Android.showToast( "Turning OFF Lights" );
                }
            }
        function callbackb() {
                var ua = navigator.userAgent.toLowerCase();
                var isAndroid = ua.indexOf( "android" ) > -1;
                if ( isAndroid ) {
                    Android.showToast( "Turning ON Lights" );
                }
            }
        </script>
```

Η πρώτη συνάρτηση είναι υπεύθυνη για την ενημέρωση της κατάστασης των φωτιστικών σωμάτων. Για την υλοποίηση αυτής της λειτουργίας πραγματοποιείται η κλίση κάθε ένα δευτερόλεπτο του ελεγκτή status1.php. Ο ελεγκτής αυτός αποστέλλει προς το Σταθμό Βάσης Internet Arduino την παράμετρο 0 που προκαλεί τον σταθμό να αποστείλει ως απάντηση την κατάσταση που βρίσκονται τα φωτιστικά σώματα την προκείμενη στιγμή όπως έχει αναλυθεί στο Κεφάλαιο 4. Η απόκριση του Σταθμού Βάσης Internet Arduino ενημερώνει με αυτό τον τρόπο την κατάσταση του διακόπτη που παρέχει η προβολή home.php. Επίσης προκαλείται η κλίση της συνάρτησης stateToggle με δεδομένα την κατάσταση που βρίσκονται τα φωτιστικά σώματα.

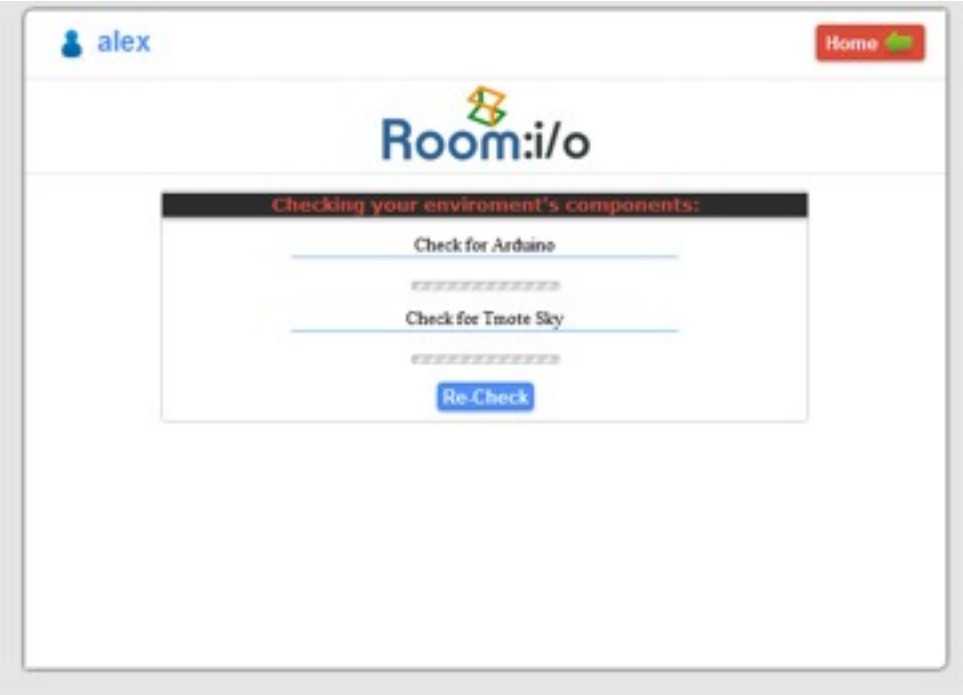
Η συνάρτηση stateToggle αναγνωρίζει την κατάσταση που βρίσκονται τα φωτιστικά σώματα και ανάλογα επιτρέπει την ενεργοποίηση τους ή την απενεργοποίηση τους. Αυτό προκύπτει με την αποστολή στον Σταθμό Βάσης Internet Arduino της παραμέτρου 2 ή 3 αντίστοιχα όταν ο χρήστης πραγματοποιήσει πάτημα στον διακόπτη της προβολής home.php. Τέλος στο ανωτέρω απόσπασμα JavaScript έχουν υλοποιηθεί και οι συναρτήσεις callbacka και callbackb που χρησιμοποιούνται από την δεύτερη διεπαφή του χρήστη που έχει υλοποιηθεί για έξυπνα κινητά τηλέφωνα που διαθέτουν λειτουργικό σύστημα Android και πρόκειται να αναλυθούν στην αντίστοιχη ενότητα.

Για να βρεθεί η εφαρμογή στην κατάσταση ελέγχου των περιφερειακών δικτύων χρειάζεται ο χρήστης να είναι συνδεδεμένος στην εφαρμογή και αυτή να βρίσκεται σε κατάσταση λειτουργίας. Αν πληρούνται αυτές οι συνθήκες τότε μέσω του κουμπιού επιλογών που βρίσκεται στο πλαίσιο επιλογών της προβολής home.php ο χρήστης λαμβάνει το μενού που απεικονίζεται στην εικόνα 63.

Εικόνα 63: Μενού επιλογών στην Κατάσταση Λειτουργίας.

Με την επιλογή Check System του μενού που απεικονίζεται στην εικόνα 63 καλείται ο ελεγκτής systemcheck.php που ενσωματώνει την προβολή check.php. Στην προβολή check.php προκαλείται κλίση μέσω προγραμματιστικών τεχνικών σε JavaScript δύο μοντέλων, του arduino.php και του tmote.php που υλοποιούν τις διαδικασίες ελέγχου λειτουργίας των δικτύων περιφερειακών. Στην εικόνα 64 απεικονίζεται η λογική λειτουργίας της κατάστασης ελέγχου της εφαρμογής και στην εικόνα 65 απεικονίζεται η προβολή check.php.

Εικόνα 64: Λογική λειτουργίας κατάστασης ελέγχου περιφερειακών δικτύων.



Εικόνα 65: Προβολή κατάστασης ελέγχου των περιφερειακών δικτύων.

Τέλος για να μεταβεί η κατάσταση λειτουργίας της εφαρμογής σε κατάσταση αποσύνδεσης πρέπει ο χρήστης να είναι συνδεδεμένος στην εφαρμογή και η εφαρμογή να βρίσκεται σε κατάσταση χρήσης. Σε περίπτωση που θέλει να αποσυνδεθεί από την εφαρμογή μπορεί μέσω του μενού που απεικονίζεται στην εικόνα 63 να επιλέξει το Log Out. Μέσω αυτής της επιλογής καλείται ο ελεγκτής logout.php που προκαλεί απόλυσης και καταστροφή των δεδομένων της συνόδου που είχε δημιουργηθεί για τον χρήστη μέσω της προβολής login.php ή της προβολής register.php. Επίσης προκαλείται ανακατεύθυνση του χρήστη στον κεντρικό ελεγκτή index.php. Ο ελεγκτής index.php λόγω τις απόλυσης των δεδομένων συνόδου προκαλεί την ανακατεύθυνση του χρήστη στον ελεγκτή login.php και με αυτό τον τρόπο η εφαρμογή επιστρέφει στην κατάσταση εισόδου.

Εφαρμογή Λειτουργικού Συστήματος Android Room: i/o

Στο σημείο αυτό θα αναλυθεί η ανάπτυξη της διεπαφής του χρήστη σε περιβάλλον λειτουργικού συστήματος Android. Η επιλογή αυτής της πλατφόρμας έγινε λόγω της ραγδαίας ανάπτυξης της τα τελευταία χρόνια. Το λειτουργικό σύστημα Android έχει κατακτήσει ένα τεράστιο κομμάτι της αγοράς των smartphones και υποστηρίζεται από τους μεγαλύτερους κατασκευαστικούς κολοσσούς του χώρου. Αυτό συμβαίνει λόγω των εξαιρετικών χαρακτηριστικών που διαθέτει τόσο σε επίπεδο τεχνολογιών όσο και ευελιξίας και ευκολίας στην παραμετροποίηση. Επίσης είναι βασισμένο στην φιλοσοφία ανοικτού λογισμικού με αποτέλεσμα να υποστηρίζεται από μια τεράστια κοινότητα προγραμματιστών.

Ο κεντρικός στόχος της υλοποίησης της εφαρμογής Android ήταν να παρέχεται απόλυτος συγχρονισμός με την Διαδικτυακή Εφαρμογή που αναπτύχθηκε, να διατίθεται εύκολο και ευχάριστο γραφικό περιβάλλον, η εφαρμογή να λειτουργεί γρήγορα, απροβλημάτιστα, και όλες οι διεργασίες να υλοποιούνται διάφανα ως προς τον τελικό χρήστη. Για αυτούς τους λόγους ολόκληρη η υλοποίηση βασίστηκε στην έτοιμη διεπαφή WebView που παρέχεται από το Android.

Η διεπαφή WebView ουσιαστικά είναι ένας φυλλομετρητής ο οποίος μπορεί να ενσωματωθεί μέσα στην Εφαρμογή Android. Αυτή η διεπαφή παρέχει όλες τις δυνατότητες ενός κανονικού φυλλομετρητή, δηλαδή μπορεί να φορτώνει ιστοσελίδες, να πλοηγείται μέσα σε αυτές, να εκτελεί JavaScripts καθώς και να αντλεί δεδομένα από αυτά. Το γεγονός αυτό έχει σαν αποτέλεσμα να παρέχεται η δυνατότητα χρήσης της Διαδικτυακής Εφαρμογής που έχει αναπτυχθεί, σχεδόν αυτούσια. Οι αλλαγές που έχουν γίνει στην Διαδικτυακή Εφαρμογή αφορούν κυρίως θέματα γραφικού περιβάλλοντος και έχουν προστεθεί επίσης κάποια κομμάτια JavaScript για να μπορεί η Διαδικτυακή Εφαρμογή να αντιλαμβάνεται ότι εκτελείται σε έξυπνο κινητό τηλέφωνο έτσι ώστε να καλούνται συγκεκριμένα “γεγονότα”. Η δομή της Εφαρμογής Android που αναπτύχθηκε απεικονίζεται στην εικόνα 66.

Εικόνα 66: Λογική υλοποίησης Εφαρμογής Android.

Το πρώτο στάδιο που πραγματοποιήθηκε ήταν η δημιουργία ενός ξεχωριστού αρχείου μορφοποίησης CSS για την Διαδικτυακή Εφαρμογή το οποίο αφορά την σχεδίαση του γραφικού περιβάλλοντος της σε μικρότερη οθόνη έξυπνου κινητού τηλεφώνου. Αυτό το αρχείο έχει την ονομασία android.css και βρίσκεται μέσα στον φάκελο css της δομής της Διαδικτυακής Εφαρμογής και περιλαμβάνει όλους τους κανόνες σχεδίασης του γραφικού περιβάλλοντος της εφαρμογής που αφορούν το λειτουργικό σύστημα Android.

Έπειτα υλοποιήθηκε η σύνδεση αυτού του αρχείου με την Διαδικτυακή Εφαρμογή μέσα στο αρχείο header.php το οποίο ενσωματώνεται σε όλες τις προβολές της Διαδικτυακής Εφαρμογής και αφορά την επικεφαλίδα HTML.

```
<meta name="viewport" content="user-scalable=no, width=device-width,
height=device-height, target-densitydpi=device-dpi" />

<link rel="stylesheet" type="text/css" href="android.css" media="only
screen and (max-width: 480px)" />

<link rel="stylesheet" type="text/css" href="style.css" media="screen
and (min-width: 481px)" />
```

Στο παραπάνω απόσπασμα του αρχείου header.php της Διαδικτυακής Εφαρμογής παρατηρείται η ετικέτα που αφορά τις μεταπληροφορίες της εφαρμογής. Οι παράμετροι που έχουν δηλωθεί στις μεταπληροφορίες επισημαίνουν στην διεπαφή WebView (φυλλομετρητής) του Android να χρησιμοποιήσει όλη την επιφάνεια της οθόνης του κινητού τηλεφώνου καθώς επίσης και την πλήρη ανάλυση της για την σχεδίαση του γραφικού περιβάλλοντος. Έπειτα ακολουθούν οι εντολές σύνδεσης με τα αρχεία μορφοποίησης CSS. Το πρώτο αρχείο αφορά την μορφοποίηση στην οθόνη του κινητού και το δεύτερο αφορά την μορφοποίηση στην οθόνη οποιουδήποτε υπολογιστή.

Πρέπει να δοθεί ιδιαίτερη σημασία στο γεγονός ότι τα παραπάνω βήματα έχουν υλοποιηθεί στην πλευρά του διακομιστή της Διαδικτυακής Εφαρμογής και όχι στην Εφαρμογή Android , όμως συνδέονται άμεσα με αυτή.

Στην πλευρά της Εφαρμογής Android αρχικά όπως απεικονίζεται και στην εικόνα 65 υλοποιείται η παραμετροποίηση του αρχείου που αφορά τα δικαιώματα και τις βασικές ρυθμίσεις της εφαρμογής. Αυτό το αρχείο ονομάζεται AndroidManifest.xml και υλοποιείται με χρήση προγραμματιστικών τεχνικών γλώσσας XML.

AndroidManifest

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.roomio"
    android:versionCode="1"
    android:versionName="1.0" >
```

Αρχικά παρέχεται η δήλωση του ονόματος της εφαρμογής, καθώς και σε ποια έκδοση βρίσκεται. Έπειτα παρέχεται η δήλωση της χαμηλότερης έκδοσης του λειτουργικού Android που μπορεί η εφαρμογή να εκτελεστεί με επιτυχία. Στην προκείμενη περίπτωση αυτή η έκδοση είναι η Android 2.2 (froyo) που αναπαριστάται με τον αριθμό 8. Τέλος ακολουθούν τα δικαιώματα που θα έχει η εφαρμογή στο λειτουργικό σύστημα, τα οποία εμφανίζονται στον χρήστη κατά την εγκατάσταση της εφαρμογής και πρέπει να επιλέξει αν τα αποδέχεται.

Η εφαρμογή που αναπτύχθηκε κάνει χρήση όσο το δυνατόν λιγότερων δικαιωμάτων και αυτά είναι:

- Να μπορεί να έχει πρόσβαση στο διαδίκτυο.
- Να μπορεί να αποκτήσει πρόσβαση στις ρυθμίσεις που αφορούν το κομμάτι διασύνδεσης του κινητού με το διαδίκτυο.
- Να μπορεί να αλλάξει τις ρυθμίσεις διασύνδεσης του κινητού.

```
<uses-sdk android:minSdkVersion="8" />
    <uses-permission android:name="android.permission.INTERNET"/>
```

```
    <uses-permission
android:name="android.permission.ACCESS_NETWORK_STATE"/>
    <uses-permission
android:name="android.permission.CHANGE_NETWORK_STATE"/>
    <uses-permission
android:name="android.permission.WRITE_SETTINGS"/>
```

Επίσης δηλώνονται στο αρχείο AndroidMaifest.xml οι βασικές πληροφορίες της εφαρμογής όπως ποιο θα είναι το εικονίδιο της, ποιο θα είναι το όνομα της, ποια είναι η βασική της δραστηριότητα, και αν θα επηρεάζεται από την περιστροφή της οθόνης του κινητού τηλεφώνου (στην περίπτωση της εφαρμογής που αναπτύχθηκε διατίθεται μόνο κάθετη προβολή).

```
<application android:icon="@drawable/home" android:label="@string/
app_name" >
        <activity
            android:name=".RoomioActivity"
            android:label="@string/app_name"
            android:screenOrientation="portrait"
            android:theme="@android:style/Theme.NoTitleBar">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category
android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

Το επόμενο αρχείο της εφαρμογής Android παρέχει παραμετροποιήσεις που σχετίζονται με την προβολή της εφαρμογής στην οθόνη. Αυτό το αρχείο έχει όνομα main.xlm και βρίσκεται στον φάκελο res/layout της Εφαρμογής Android.

Main.xml

```
<WebView xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/webview"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
```

Στο κώδικα του ανωτέρω αρχείου οι παραμετροποιήσεις που υλοποιούνται είναι η δήλωση της διεπαφής WebView που θα χρησιμοποιηθεί από την εφαρμογή και το γεγονός ότι αυτή η διεπαφή θα καταλαμβάνει όλη την οθόνη του κινητού τηλεφώνου.

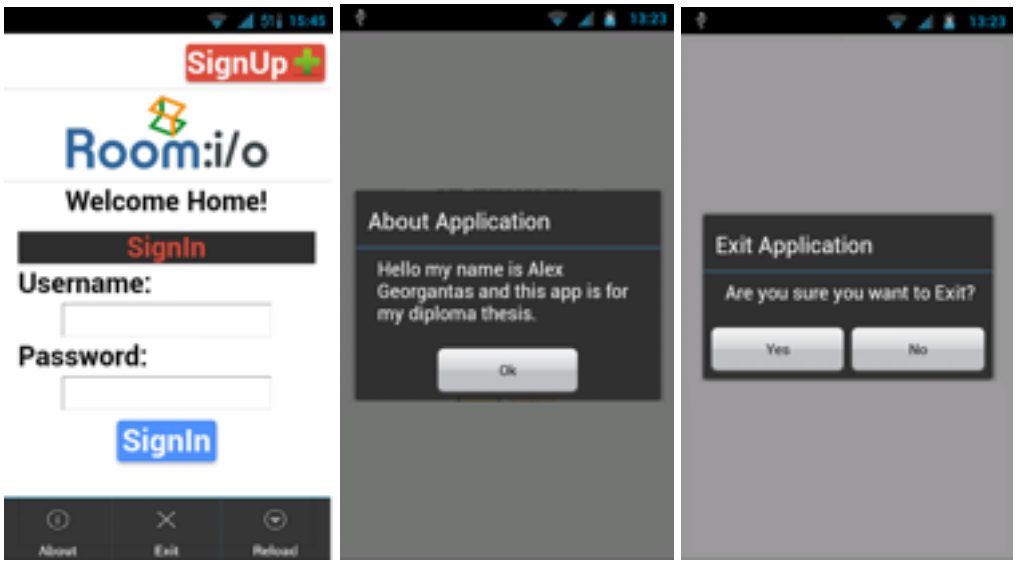
Συνεχίζοντας δημιουργείται το αρχείο mymenu.xml το οποίο αναφέρεται στην προβολή ενός καταλόγου με επιλογές που διαθέτει η εφαρμογή. Αυτό το αρχείο βρίσκεται στον φάκελο res/menu. Στο αρχείο mymenu.xml δηλώνεται ότι θα υπάρχουν τρία αντικείμενα (επιλογές) με ονόματα about, exit και reload. Αντίστοιχα για κάθε αντικείμενο δηλώνεται το εικονίδιο που θα το αντιπροσωπεύει.

Mymenu.xml

```
<menu xmlns:android="http://schemas.android.com/apk/res/android" >
    <item android:id="@+id/about"
```

```
        android:title="About"
        android:icon="@android:drawable/ic_menu_info_details"
    />
    <item android:id="@+id/exit"
        android:title="Exit"
        android:icon="@android:drawable/ic_menu_close_clear_cancel"
        />
    <item android:id="@+id/reload"
        android:title="Reload"
        android:icon="@android:drawable/ic_menu_more"
    />
</menu>
```

Η μορφή του καταλόγου επιλογών καθώς και οι επιλογές about και exit απεικονίζονται στην εικόνα 67.



Εικόνα 67: Στα δεξιά απεικονίζεται η μορφή του καταλόγου επιλογών, στο κέντρο απεικονίζεται η επιλογή About και αριστερά απεικονίζεται η επιλογή Exit.

Τέλος υλοποιείται το αρχείο με όνομα toast_layout.xml και αφορά το οπτικό κομμάτι ενός αναδυόμενου παραθύρου που δημιουργείται όταν ο χρήστης πατήσει την επιλογή ενεργοποίησης ή απενεργοποίησης του φωτιστικού σώματος.

toast_layout.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/
android"
android:id="@+id/toast_root"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:orientation="horizontal"
android:padding="10dp"
android:background="#DAAA"
>
    <ImageView android:id="@+id/image"
        android:layout_width="wrap_content"
        android:layout_height="fill_parent"
        android:src="@drawable/android"
        android:layout_marginRight="10dp"
        android:contentDescription="@string/image"
```

```
    />
    <TextView android:id="@+id/text"
        android:layout_width="wrap_content"
        android:layout_height="fill_parent"
        android:textSize="7pt"
        android:textStyle="bold"
        android:textColor="#000"
        android:layout_gravity="center_vertical"
        android:gravity="center"
        android:layout_marginRight="10dp"
    />
</LinearLayout>
```

Στο ανωτέρω αρχείο παρατηρούνται εντολές μορφοποίησης για το αναδυόμενο παράθυρο, εντολές μορφοποίησης για το κείμενο που θα περιλαμβάνει το παράθυρο και ένα εικονίδιο που θα εμφανίζεται επίσης μέσα στο παράθυρο. Ένα παράδειγμα αυτού του αναδυόμενου παραθύρου ακολουθεί στην εικόνα 68.

Εικόνα 68: Αναδυόμενο Παράθυρο ενεργοποίησης φωτιστικών σωμάτων.

Τα αρχεία main.xml, mymenu.xml και toast_layout.xml που αναπτύχθηκαν ανωτέρω συστήνουν το επίπεδο Android View που απεικονίζεται στην εικόνα 66 και αποτελούν τους κανόνες σχεδίασης του περιεχομένου της εφαρμογής. Το αρχείο που ακολουθεί παρέχει την βασική δραστηριότητα (Android Activity της εικόνας 66) της Εφαρμογής Android η οποία υλοποιεί τις δυνατότητες της εφαρμογής.

Το κύριο αρχείο της εφαρμογής το οποίο παρέχει την δραστηριότητα της εφαρμογής βρίσκεται στον φάκελο src/com/roomio και έχει όνομα Roomio Activity. Για την ανάπτυξη του αρχείου δραστηριότητας της εφαρμογής χρησιμοποιήθηκαν προγραμματιστικές τεχνικές σε γλώσσα προγραμματισμού Java.

Αρχικά υλοποιείται η δήλωση του ονόματος πακέτου της εφαρμογής και η εισαγωγή αρκετών βιβλιοθηκών που απαιτούνται για τις δυνατότητες της εφαρμογής.

```
package com.roomio;

import android.app.Activity;
import android.app.AlertDialog;
```

```
import android.app.ProgressDialog;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.net.ConnectivityManager;
import android.net.NetworkInfo;
import android.os.Bundle;
import android.view.Gravity;
import android.view.LayoutInflater;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.view.ViewGroup;
import android.view.animation.AccelerateInterpolator;
import android.view.animation.Animation;
import android.view.animation.TranslateAnimation;
import android.webkit.WebView;
import android.webkit.WebViewClient;
import android.widget.TextView;
import android.widget.Toast;
```

Οι εντολές που ακολουθούν παρέχουν βασικές λειτουργίες στην δραστηριότητα της εφαρμογής.

```
@Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        mProgressDialog = new ProgressDialog(this);
        mWebView = (WebView)findViewById(R.id.webview);
        mWebView.setVerticalScrollBarEnabled(false);
        mWebView.setHorizontalScrollBarEnabled(false);
        mWebView.getSettings().setJavaScriptEnabled(true);
        mWebView.loadUrl("http://alexator.zapto.org:8888/roomio/");
        mWebView.addJavascriptInterface(new
JavaScriptInterface(this), "Android");
        mWebView.setWebViewClient(new HelloWebClient());
```

Συγκεκριμένα επιτελείται η δημιουργία της δραστηριότητας κατά την εκκίνηση της εφαρμογής και η παραχώρηση της οθόνης σε αυτή. Έπειτα η δραστηριότητα αντλεί πληροφορίες για το πώς θα είναι η προβολή της διαβάζοντας το αρχείο main.xml που έχει δημιουργηθεί. Επίσης υλοποιείται η δήλωση μιας διεπαφής “παραθύρου προόδου” που θα χρησιμοποιηθεί από την δραστηριότητα όταν θα αναμένει τα δεδομένα από τον διακομιστή της Διαδικτυακής Εφαρμογής. Αμέσως μετά ακολουθεί η δήλωση του WebView και των ιδιοτήτων του όπως η απενεργοποίηση δυνατότητας κύλισης οριζόντια και κάθετα, η ενεργοποίηση υποστήριξης JavaScript, η δημιουργία μιας διεπαφής που θα παρακολουθεί JavaScript που βρίσκονται στην Διαδικτυακή Εφαρμογή και αφορούν το WebView και τέλος υλοποιείται η φόρτωση της διεύθυνσης της Διαδικτυακής Εφαρμογής.

Έπειτα παρατίθενται οι εντολές που δημιουργούν τον κατάλογο επιλογών που εμφανίζεται αν ο χρήστης πατήσει το φυσικό πλήκτρο menu που διαθέτουν όλες οι συσκευές που εκτελούν λειτουργικό σύστημα Android.

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

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch(item.getItemId()){
        case R.id.about:
            openOptionsDialog();
            break;
        case R.id.reload:
            mWebView.reload();
            break;
        case R.id.exit:
            exitOptionsDialog();
            break;
        }
        return true;
    }
```

Στις ανωτέρω εντολές δηλώνεται στην διεπαφή που ακούει αν έχει πατηθεί το φυσικό κουμπί menu πως θα είναι ο κατάλογος που θα αναδυθεί στην οθόνη με βάση το αρχείο mymenu.xml που έχει δημιουργηθεί. Στην συνέχεια δημιουργείται η διεπαφή που περιμένει να ακούσει τί επιλογή θα επιλέξει ο χρήστης αφού αναδυθεί ο κατάλογος και αντίστοιχα με την επιλογή που θα κάνει καλείται και από μια συνάρτηση που εμφανίζει τις δυνατότητες της κάθε επιλογής.

Οι συναρτήσεις αυτές είναι οι ακόλουθες:

```
private void openOptionsDialog(){
        new AlertDialog.Builder(this)
        .setTitle("About Application")
        .setMessage("Hello my name is Alex Georgantas and this app is
for my diploma thesis.")
        .setCancelable(false)
        .setPositiveButton ("Ok", new
DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialoginterface, int
i){
            }
        })
        .show();
    }

    private void exitOptionsDialog(){
        new AlertDialog.Builder(this)
        .setTitle("Exit Application")
        .setMessage("Are you sure you want to Exit?")
        .setCancelable(false)
        .setNegativeButton ("No", new
DialogInterface.OnClickListener() {
```

```
            public void onClick(DialogInterface dialoginterface, int
i){
                }
            })
            .setPositiveButton("Yes", new
DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialoginterface , int
i){

android.webkit.CookieManager.getInstance().removeAllCookie();
                mWebView.clearHistory();
                mWebView.clearFormData();
                mWebView.clearCache(true);
                finish();
            }
        })
        .show();
    }
```

Οι τρείς συναρτήσεις αντιστοιχούν στις τρείς επιλογές που διαθέτει ο κατάλογος. Η πρώτη προκαλεί ένα παράθυρο που περιέχει πληροφορίες σχετικά με την εφαρμογή, η δεύτερη προκαλεί την επαναφόρτωση της εκάστοτε σελίδας της Διαδικτυακής Εφαρμογής που βρίσκεται ο χρήστης και η τρίτη συνάρτηση προκαλεί το κλείσιμο της εφαρμογής αφού καθαρίσει πρώτα όλες τις αποθηκευμένες πληροφορίες περιήγησης (cache, cookies).

Μια επιπλέον λειτουργία της εφαρμογής είναι να αντιλαμβάνεται αν το κινητό τηλέφωνο έχει ενεργοποιημένη την σύνδεση του με το διαδίκτυο.

```
ConnectivityManager cm = (ConnectivityManager)
getSystemService(Context.CONNECTIVITY_SERVICE);
        NetworkInfo info = cm.getActiveNetworkInfo();

        if (info !=null){
            if (!info.isConnected()){
                openInternetDialog();
            }
        } else {
            openInternetDialog();
        }
```

Αυτή η δυνατότητα παρέχεται με την χρήση του ConnectivityManager του λειτουργικό συστήματος Android. Όταν η εφαρμογή αντιληφθεί ότι δεν υπάρχει σύνδεση με το διαδίκτυο μεταφέρει αυτόματα τον χρήστη στις ρυθμίσεις του κινητού που έχουν να κάνουν με την ενεργοποίηση της ασύρματης κάρτας δικτύου που διαθέτουν όλες οι συσκευές Android ή στις ρυθμίσεις που έχουν να κάνουν με την ενεργοποίηση των δυνατοτήτων για GPRS/3G. Αυτό υλοποιείται με την συνάρτηση openInternetDialog.

```
private void openInternetDialog(){
        new AlertDialog.Builder(this)
        .setTitle("Oops no Internet!")
        .setMessage("Sorry this is a webapp and you need to open your
Wi-Fi or mobile data")
        .setCancelable(false)
        .setPositiveButton("Enable", new
DialogInterface.OnClickListener(){
            public void onClick(DialogInterface dialoginterface, int
i){
```

```
                finish();
                Intent intent = new Intent(Intent.ACTION_MAIN);

intent.setClassName("com.android.settings","com.android.settings.Wire
lessSettings");
                startActivity(intent);
            }
        })
        .show();
    }
```

Στο σημείο αυτό θα αναλυθεί η δυνατότητα που έχει η εφαρμογή Android για να αλληλεπιδρά με την Διαδικτυακή Εφαρμογή. Η επικοινωνία αυτή μπορεί να υλοποιηθεί χρησιμοποιώντας την δυνατότητα που παρέχει το λειτουργικό Android κάνοντας χρήση έτοιμων συναρτήσεων. Συγκεκριμένα δημιουργείται μια διεπαφή μέσα στην κύρια δραστηριότητα της εφαρμογής που παρακολουθεί αν υπάρχει εκτελέσιμος κώδικας JavaScript που αφορά την εφαρμογή Android. Αυτό υλοποιείται ως εξής:

```
public class JavaScriptInterface{
        Context mContext;
        JavaScriptInterface(Context c){
            mContext = c;
        }
```

Με αυτή την διεπαφή υλοποιείται η επικοινωνία με τα JavaScripts της Διαδικτυακής Εφαρμογής και αποθήκευση των δεδομένων τους. Στην Εφαρμογή Android η δυνατότητα αυτή χρησιμοποιείται για να παράγεται ένα αναδυόμενο παράθυρο στην τελικό χρήστη όταν αυτός πατήσει το κουμπί για να ενεργοποιήσει ή να απενεργοποιήσει το φωτιστικό σώμα, όπως αναφέρθηκε και ανωτέρω. Στην πλευρά της Εφαρμογής Android υλοποιείται ο κώδικας δημιουργίας και αναπαράστασης αυτού του παραθύρου ο οποίος αντλεί τα δεδομένα περιεχομένου του (το μήνυμα που θα προβάλει) από την διεπαφή επικοινωνίας με την Διαδικτυακή Εφαρμογή:

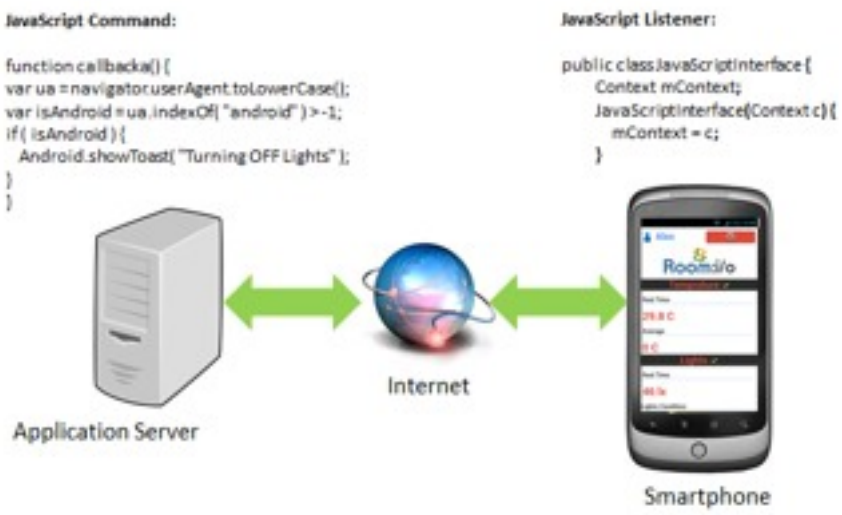
```
LayoutInflater inflater = getLayoutInflater();
        View layout = inflater.inflate(R.layout.toast_layout,
(ViewGroup)findViewById(R.id.toast_root));
        TextView text = (TextView) layout.findViewById(R.id.text);

        public void showToast(String toast){
            Toast t = new Toast(mContext);
            t.setGravity(Gravity.CENTER_VERTICAL, 0, 0);
            t.setView(layout);
            text.setText(toast);
            t.show();
        }
```

Στην πλευρά της Διαδικτυακής Εφαρμογής και συγκεκριμένα στην προβολή home.php παρέχεται το αντίστοιχο απόσπασμα JavaScript που δίνει την δυνατότητα να αντιλαμβάνεται η Διαδικτυακή Εφαρμογή αν εκτελείται σε περιβάλλον λειτουργικού Android με αποτέλεσμα να αποστέλλει συγκεκριμένη εντολή σε αυτό. Το απόσπασμα αυτό βρίσκεται στον ενσωματωμένο εκτελέσιμο κώδικα JavaScript της προβολής home.php της Διαδικτυακής Εφαρμογής:

```
$('#imgon').click(function() {
                    callbacka();
                });
```

```
        $('#imgoff').click(function() {
                    callbackb();
                });


        function callbacka() {
                var ua = navigator.userAgent.toLowerCase();
                var isAndroid = ua.indexOf( "android" ) > -1;
                if ( isAndroid ) {
                    Android.showToast( "Turning OFF Lights" );
                }
            }
        function callbackb() {
                var ua = navigator.userAgent.toLowerCase();
                var isAndroid = ua.indexOf( "android" ) > -1;
                if ( isAndroid ) {
                    Android.showToast( "Turning ON Lights" );
                }
            }
```

Συγκεκριμένα η εντολή Android.showToast("Turning ON Lights") είναι που προκαλεί την αντίδραση της Εφαρμογής Android και την κλήση της διεπαφής που δημιουργεί το αναδυόμενο παράθυρο όπως απεικονίζεται και στην εικόνα 69.



Εικόνα 69: Επικοινωνία Διεπαφών χρήστη σε επίπεδο Javascript.

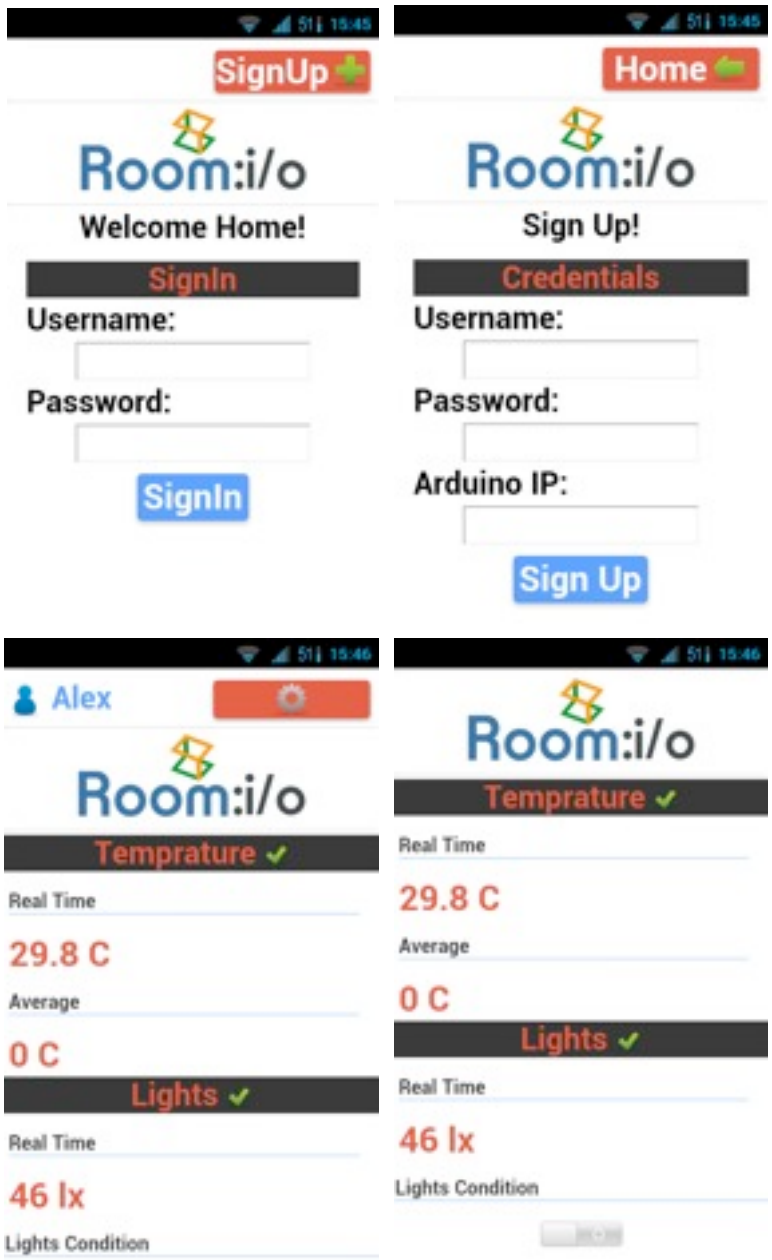
Τέλος στην δραστηριότητα της Εφαρμογής Android υλοποιήθηκε και ένα εφέ κίνησης. Το γραφικό στοιχείο αυτό δημιουργεί κίνηση εισαγωγής των περιεχομένων της εφαρμογής από το αριστερό μέρος της οθόνης με φορά προς το δεξιό και χρησιμοποιείται όταν ο χρήστης πλοηγείται σε καινούργιες σελίδες στον WebView client.

```
private Animation inFromLeftAnimation() {
        Animation inFromLeft = new TranslateAnimation(
        Animation.RELATIVE_TO_PARENT,  -1.0f,
Animation.RELATIVE_TO_PARENT,  0.0f,
        Animation.RELATIVE_TO_PARENT,  0.0f,
Animation.RELATIVE_TO_PARENT,   0.0f
        );
        inFromLeft.setDuration(300);
        inFromLeft.setInterpolator(new AccelerateInterpolator());
```

```
        return inFromLeft;
    }
```

Για να προκύπτει καλύτερο αποτέλεσμα στις μεταβάσεις του εφέ δημιουργήθηκε μια συνάρτηση που εξαφανίζει τα περιεχόμενα της Εφαρμογής Android όσο φορτώνονται τα δεδομένα από την Διαδικτυακή Εφαρμογή και παράγεται ένα παράθυρο που ενημερώνει τον χρήστη για την πρόοδο της φόρτωση των περιεχομένων. Όταν η φόρτωση ολοκληρωθεί πλήρως εμφανίζονται τα περιεχόμενα της εφαρμογής να εισάγονται από τα αριστερά προς τα δεξιά.

```
private class HelloWebClient extends WebViewClient {
        @Override
        public boolean shouldOverrideUrlLoading(WebView view, String
url) {
        view.setVisibility(View.GONE);
        mProgressDialog.setTitle("Processing");
        mProgressDialog.show();
        mProgressDialog.setMessage("Loading");
        return false;
        }
        @Override
        public void onPageFinished(WebView view, String url) {
            view.startAnimation(inFromLeftAnimation());
            view.setVisibility(View.VISIBLE);
            super.onPageFinished(view, url);
        }
    }
```



Εικόνα 70: Στιγμιότυπα Εφαρμογής Room:i/o για λειτουργικό σύστημα Android.

Επίλογος – Συμπεράσματα

Στο Κεφάλαιο 5 αναλύθηκε διεξοδικά η λογική σχεδιασμού καθώς και η υλοποίηση σε επίπεδο λογισμικού των διεπαφών του χρήστη. Αρχικά κατασκευάστηκε η βάση δεδομένων της εφαρμογής η οποία λειτουργεί σαν κόμβος άντλησης δεδομένων και για τις δυο διεπαφές του χρήστη. Έπειτα κατασκευάστηκε η πρώτη διεπαφή του χρήστη που επιλέχθηκε να υλοποιηθεί για λειτουργία σε διαδικτυακό περιβάλλον. Για την κατασκευή της Διαδικτυακής Εφαρμογής έγινε χρήση προγραμματιστικών τεχνικών σε γλώσσες PHP, JavaScript, HTML, MySQL και CSS για να επιτευχθεί ο δυναμικός τρόπος λειτουργίας της διεπαφής. Η υλοποίηση

της Διαδικτυακής Εφαρμογής βασίστηκε στην χρήση του προτύπου δομημένης μορφής MVC και οι λειτουργίες της ομαδοποιήθηκαν με βάση τις λογικές καταστάσεις χρήσης. Τέλος αναπτύχθηκε η δεύτερη διεπαφή για τον χρήστη η οποία υλοποιήθηκε για να εκτελείται σε περιβάλλον λειτουργικού συστήματος Android. Η σχεδιαστική λογική της δεύτερης διεπαφής βασίστηκε στις δυνατότητες άντλησης δεδομένων και συνεργασίας-ανταλλαγής δεδομένων με την Διαδικτυακή Εφαρμογή που παρέχονται από το λειτουργικό σύστημα Android. Με αυτό τον τρόπο αναπτύχθηκαν δύο διεπαφές για τον χρήστη οι οποίες λειτουργούν σε απόλυτο συγχρονισμό, διαθέτουν απλό και χρηστικό γραφικό περιβάλλον, είναι υλοποιημένες με χρήση ανοικτών προτύπων και παρέχουν φορητότητα.

Τα συμπεράσματα που προκύπτουν από το παρόν Κεφάλαιο αφορούν τις δυνατότητες των προγραμματιστικών τεχνικών Διαδικτύου που παρέχουν άμεση διαχείριση δεδομένων ανεξαρτήτως τύπου και μορφής, αποτελεσματική δημιουργία επικοινωνίας με επιμέρους συστήματα, ταχύτατους χρόνους εκτέλεσης καθώς και ευελιξία στην σχεδίαση γραφικού περιβάλλοντος. Επίσης ιδιαιτέρως σημαντικό ήταν και το πλήθος των παρεχόμενων δυνατοτήτων και εργαλείων του λειτουργικού συστήματος Android. Το γεγονός αυτό υποδηλώνει το μεγάλο ενδιαφέρον που παρουσιάζουν οι εφαρμογές έξυπνων κινητών τηλεφώνων που αυξάνονται ραγδαία. Για τον συγκεκριμένο λόγο η εφαρμογή ευφυούς οικείας αποτελεί μια πολύ σύγχρονη υλοποίηση αφού συνδυάζει τεχνολογίες που γνωρίζουν μεγάλη ανάπτυξη και δημοσιότητα.

Συμπεράσματα – Μελλοντικές Προεκτάσεις

Σύνοψη και Συμπεράσματα

Στα πλαίσια αυτής της διπλωματικής εργασίας αναπτύχθηκε με επιτυχία μια πλήρως λειτουργική εφαρμογή που συνδύασε με διαφανή τρόπο πολλούς κλάδους τεχνολογιών όπως, ασύρματα δίκτυα αισθητήρων με κόμβους Tmote Sky και λειτουργικό σύστημα TinyOS, πλακέτες και επεκτάσεις ανοικτού υλικού Arduino, γλώσσες προγραμματισμού ανάπτυξης διαδικτυακών εφαρμογών και τέλος γλώσσα προγραμματισμού εφαρμογών Android.

Συγκριμένα αναπτύχθηκε μια εφαρμογή υλοποίησης ευφυούς οικείας που παρέχει στον χρήστη την δυνατότητα να ενημερώνεται απομακρυσμένα μέσω του έξυπνου κινητού που διαθέτει ή μέσω ενός υπολογιστή με σύνδεση στο διαδίκτυο για τις περιβαλλοντολογικές συνθήκες (θερμοκρασία, ένταση φωτεινότητας) ενός χώρου που επιθυμεί. Επίσης παρέχεται προς τον χρήστη η δυνατότητα να επενεργεί απομακρυσμένα σε ένα φωτιστικό σώματα καθώς και να ενημερώνεται για την κατάσταση στην οποία βρίσκεται αυτό.

Αναλυτικά για την παρακολούθηση περιβαλλοντολογικών συνθηκών υλοποιήθηκε ένα Δίκτυο Ασύρματων Αισθητήρων με χρήση των κόμβων Tmote Sky που αποτελείται από έναν Σταθμό Συλλογής Φυσικών Παραμέτρων και έναν ασύρματο κόμβο δειγματοληψίας αισθητήρων. Για την δυνατότητα παρέμβασης από τον χρήστη δημιουργήθηκε ένα Ασύρματο Δίκτυο ενεργοποιητή (actuator) με χρήση της πλατφόρμας ανοικτού υλικού Arduino, που αποτελείται από έναν κόμβο που υλοποιεί την επικοινωνία με το διαδίκτυο και ένα ασύρματο κόμβο ενεργοποίησης συσκευών. Ο κόμβος επικοινωνίας με το διαδίκτυο παρέχει τον απομακρυσμένο έλεγχο του κόμβου ενεργοποίησης συσκευών. Τέλος κατασκευάστηκε μια διαδικτυακή εφαρμογή που υλοποιεί την διεπαφή του χρήστη με το συνολικό δίκτυο και μια εφαρμογή που εκτελείται από έξυπνα κινητά που διαθέτουν λειτουργικό σύστημα Android.

Τα συμπεράσματα που προκύπτουν από το σύνολο της Διπλωματικής Εργασίας αφορούν την σπουδαιότητα των τεχνικών μετατροπής αναλογικού σήματος σε ψηφιακό, καθώς παρέχεται η δυνατότητα παρακολούθησης φυσικών μεγεθών με χρήση διαφόρων τύπων αισθητήριων και η άμεση επεξεργασία τους από ηλεκτρονικούς υπολογιστές. Πολύ σημαντικό σημείο είναι επίσης η ανάπτυξη ευέλικτων προτύπων ασύρματων επικοινωνιών όπως το 802.15.4 και το ZigBee με έμφαση στην χαμηλή κατανάλωση ισχύος που χρησιμοποιείται κατά κόρον στα Δίκτυα Ασύρματων Αισθητήρων προσδίδοντας την απαιτούμενη αυτονομία στους ασύρματους κόμβους. Με τα πρότυπα αυτά διευκολύνεται ο σχεδιασμός του δικτύου και αυξάνεται η ποιότητα και η αξιοπιστία του με αποτέλεσμα την μεγαλύτερη διείσδυση των Δικτύων Ασύρματων Αισθητήρων σε εφαρμογές. Εξαιρετικά χρήσιμη χαρακτηρίζεται και η ανάπτυξη λειτουργικών συστημάτων όπως το TinyOS και το Contiki που προορίζονται για αποκλειστική χρήση σε κόμβους Ασύρματων Αισθητήρων παρέχοντας βελτιστοποιήσεις σε επίπεδο δυνατοτήτων διασύνδεσης και λειτουργιών με γνώμονα πάντα την χαμηλή κατανάλωση ισχύος.

Επίσης το μεγάλο πλήθος από διατάξεις όπως οι κόμβοι Tmote Sky, οι κόμβοι MicaZ και η πλατφόρμα ανοικτού υλικού Arduino, παρέχει την δυνατότητα επιλογής έτσι ώστε να σχεδιαστεί το βέλτιστο δυνατό δίκτυο που απαιτείται από την εκάστοτε εφαρμογή. Επιπλέον αξίζει να σημειωθεί η σπουδαιότητα των προγραμματιστικών τεχνικών διαδικτύου που προσδίδουν μεγάλη ευελιξία και αμέτρητες δυνατότητες για την κατασκευή εφαρμογών και την ενοποίηση τεχνολογιών και αποτέλεσαν πολύ σημαντικό εργαλείο για την ανάπτυξη της εφαρμογής ευφυούς οικείας. Τέλος η διάδοση των έξυπνων κινητών τηλεφώνων που παρέχουν πλήθος ευκολιών και δυνατοτήτων αποτέλεσε σημαντική παράμετρο για την ανάπτυξη της εφαρμογής ευφυούς οικείας διότι προσέθεσε αμεσότητα στον απομακρυσμένο έλεγχο της εφαρμογής και ευελιξία.

Σημαντικά Σημεία

Στην παρούσα Διπλωματική Εργασία ιδιαίτερη έμφαση δόθηκε στην ενοποίηση και την συνεργασία δύο ξεχωριστών δικτύων που επιτελούν διαφορετικές λειτουργίες. Το πρώτο δίκτυο σχεδιάστηκε έτσι ώστε να παρέχει την παρακολούθηση φυσικών παραμέτρων ενός χώρου και υλοποιήθηκε με χρήση τον ασύρματων κόμβων Tmote Sky. Το δεύτερο δίκτυο σχεδιάστηκε για να παρέχει την δυνατότητα ενέργειας από τον χρήστη σε κάποια ηλεκτρική συσκευή και υλοποιήθηκε με χρήση της πλατφόρμας Arduino. Η ενοποίηση επιτεύχθηκε με την χρήση προγραμματιστικών τεχνικών διαδικτύου καθώς κατασκευάστηκε μια ενιαία βάση δεδομένων με αποτέλεσμα τα διαφορετικά δίκτυα να μπορούν να έχουν κοινό σημείο επικοινωνίας και άντλησης πληροφοριών.

Επίσης έγινε μεγάλη προσπάθεια στον τομέα της διαφάνειας ως προς τον τελικό χρήστη έτσι ώστε να μην γίνεται σε κανένα σημείο εμφανής η πολυπλοκότητα της εφαρμογής. Αυτό επιτεύχθηκε αναθέτοντας όλα τα σημεία επεξεργασίας, επικοινωνίας και λειτουργιών στον κεντρικό διακομιστή της εφαρμογής έτσι ώστε στον χρήστη να παρέχονται μόνο τελικές πληροφορίες και επιλογές μέσω ενός απλού και εύχρηστου γραφικού περιβάλλοντος.

Επιπλέον οι διεπαφές του χρήστη δηλαδή η διαδικτυακή εφαρμογή και η εφαρμογή του λειτουργικού Android κατασκευάστηκαν με τέτοιο τρόπο ώστε να υπάρχει πλήρης συγχρονισμός μεταξύ τους. Δηλαδή να μπορούν να ενημερώνονται παράλληλα για τις ενέργειες του χρήστη καθώς και τις συνθήκες που επικρατούν στον χώρο. Για να υλοποιηθεί αυτός ο στόχος έγινε χρήση προγραμματιστικών τεχνικών του λειτουργικού Android έτσι ώστε να μπορεί η εφαρμογή που εκτελείται στον έξυπνο κινητό τηλέφωνο να ανταλλάσει διαρκώς πληροφορίες από την διαδικτυακή εφαρμογή που εκτελείται στον κεντρικό διακομιστή.

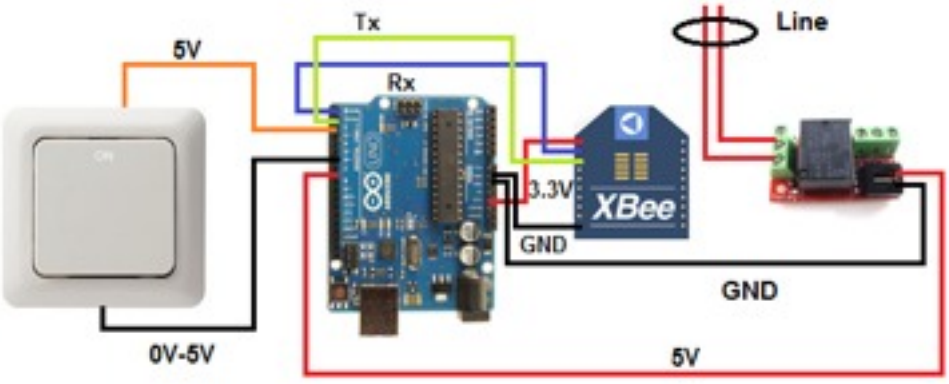
Τέλος αξίζει να σημειωθεί η προσπάθεια που έγινε να χρησιμοποιηθούν αποκλειστικά τεχνολογίες που στηρίζονται σε ανοιχτές κοινότητες ανάπτυξης έτσι ώστε να υπάρχει ελεύθερη πρόσβαση από οποιονδήποτε ενδιαφέρεται να ασχοληθεί ή να επεκτείνει την εφαρμογή ευφυούς οικείας που κατασκευάστηκε. Αυτό επιτεύχθηκε κάνοντας χρήση προγραμματιστικών τεχνικών σε PHP, HTML, CSS, JavaScript, Android, Arduino, και TinyOS που υπάγονται στην φιλοσοφία ανοικτού λογισμικού.

Μελλοντικές Επεκτάσεις

Με βάση αυτή την Διπλωματική Εργασία μπορούν να γίνουν αρκετές προσθήκες σε επίπεδο λειτουργιών και δυνατοτήτων, όπως να προστεθεί δυνατότητα ελέγχου της θερμοκρασίας ενός χώρου ή να προστεθεί δυνατότητα παρακολούθησης της υγρασίας ενός χώρου. Τα ανωτέρω μπορούν να υλοποιηθούν με μικρές προσθήκες στους ήδη υπάρχοντες κώδικες καθώς ο βασικός πυρήνας έχει δημιουργηθεί και επιδέχεται εύκολα παραμετροποιήσεις και αλλαγές.

Μια σημαντική μελλοντική επέκταση θα ήταν να κατασκευαστεί ένα ολοκληρωμένο σύστημα παρακολούθησης συνθηκών σε ολόκληρο το περιβάλλον μιας οικείας και να έχει την δυνατότητα ο χρήστης να ελέγχει και να πληροφορείται για όλα τα φωτιστικά σώματα της οικείας του, την κατανάλωση των ηλεκτρικών του συσκευών, και την θερμοκρασία των διαφόρων δωματίων. Επίσης θα ήταν ενδιαφέρον όλα αυτά να παραμετροποιούνται αυτόματα από ένα κεντρικό σύστημα που θα μαθαίνει τις προτιμήσεις του χρήστη. Τέλος θα μπορούσε να προστεθεί και η λειτουργία φωνητικών εντολών από τον χρήστη που θα μεταφράζονται από ένα σύστημα σε εντολές ενεργειών ή εντολές παράθεσης πληροφοριών.

Στο άμεσο επίπεδο της Διπλωματικής Εργασίας θα μπορούσε να προστεθεί η δυνατότητα συγχρονισμού με οικιακούς διακόπτες φωτιστικών σωμάτων έτσι ώστε να υπάρξει πλήρης ολοκλήρωση στον τομέα ελέγχου των φωτιστικών σωμάτων. Αυτό θα μπορούσε να υλοποιηθεί με την δημιουργία συγκεκριμένων διακοπτών, όπου θα ενσωματώνουν πάνω τους ένα μικρότερο μέλος της πλατφόρμας Arduino, ένα XBee module και ένα ρελέ όπως φαίνεται στην εικόνα 71.



Εικόνα 71: Πρότυπος Διακόπτης.

Με αυτήν την υλοποίηση θα υπάρχει δυνατότητα μέσω του δομοστοιχείου XBee module να παρέχεται επικοινωνία μεταξύ του διακόπτη και ενός σταθμού βάσης που θα μπορούσε να προκύψει με μικρή τροποποίηση του Σταθμού Βάσης Internet που δημιουργήθηκε στα πλαίσια αυτή της Διπλωματικής Εργασίας. Το Arduino που εμφανίζεται στην εικόνα 71 θα μπορούσε να εκτελεί μια τροποποιημένη έκδοση του Σταθμού Ενεργοποιητή που δημιουργήθηκε στα πλαίσια της παρούσας

Διπλωματικής Εργασίας έτσι ώστε να λαμβάνει υπόψη του εκτός από τις απομακρυσμένες εντολές του χρήστη για την ενεργοποίηση του φωτιστικού σώματος και τις εντολές που θα δέχεται απευθείας από τον διακόπτη που θα είναι ενσωματωμένος. Με αυτή την κατασκευή θα μπορεί να επιτευχθεί πλήρης έλεγχος των φωτιστικών σωμάτων είτε απομακρυσμένα είτε τοπικά, δεν θα χρειάζεται ο χρήστης να παρέμβει στις υπάρχουσες ηλεκτρολογικές εγκαταστάσεις της οικείας του αλλά μόνο να αντικαταστήσει τους διακόπτες των φωτιστικών του, και το σημαντικότερο πλεονέκτημα θα είναι ότι στον διακόπτη δεν θα καταλήγουν 230V από την φάση της γραμμής αλλά μόλις 5V που θα χρησιμοποιούνται από το Arduino για να αντιλαμβάνεται την κατάσταση που θα βρίσκεται ο διακόπτης.

Τέλος αξίζει να αναφερθεί ότι οι εξελίξεις που αφορούν το αντικείμενο του ευφυούς οικείας αλλά και του Διαδικτύου των Πραγμάτων (internet of things) που υπάγονται αυτές οι εφαρμογές, έχει αρχίσει να υποστηρίζεται ένθερμα από μεγάλες εταιρίες όπως η Samsung, η LG, η Google, κ.α. παρουσιάζοντας σε διεθνείς εκθέσεις “έξυπνες” οικιακές συσκευές όπως ψυγεία, πλυντήρια, ηχητικά συστήματα, και φωτιστικά σώματα που παραμετροποιούνται ανάλογα με τις προσωπικές επιθυμίες του χρήστη. Αυτές οι εξελίξεις υπονοούν ότι αυτός ο ευρύτερος κλάδος θα γνωρίσει μεγάλη ανάπτυξη και θα απασχολήσει σημαντικά τα επόμενα χρόνια, γεγονός που καταδεικνύει και την σπουδαιότητα της παρούσας Διπλωματικής Εργασίας που πραγματεύεται το συγκεκριμένο αντικείμενο.

Βιβλιογραφία και Αναφορές

[1] Telos B, Tmote Sky. http://www.memsic.com/.
[2] EYES Project. http://www.eyes.eu.org/. March 2002-February 2005.
[3] IECON 2007. 33rd Annual Conference of the IEEE, A Comparative Study of Wireless Protocols: Bluetooth, UWB, ZigBee, and Wi-Fi, Jin-Shyan Lee Ind. Technol. Res. Inst., Hsinchu Yu-Wei Su, Chung-Chou Shen.
[4] ZigBee Alliance. http://www.zigbee.org/.

[5] TinyOS. http://www.tinyos.net/.
[6] A. Dunkels, B. Gronvall, and T. Voigt. Contiki – a lithgweight and flexible operating system for tiny networked sensors. In Proceedings of IEEE EmNets 2004. Tampa. Florida, USA, November 2004.
[7] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: a survey. Computer Networks, 38(4): 393-422, March 2002.
[8] Y. Kim, T. Schmid, Z. M. Charbiwala, J. Friedman, and M. B. Srivastava. NAWMS: Nonintrusive Autonomous Water Monitoring System. In Proceedings of ACM SenSys 2008, pp. 309-322, Raleigh, NC, USA, November 2008.
[9] G. Werner-Allen, K. Lorinez, M. Ruiz, O.Marcillo, J. Johnson, J. Lees, and M. Welsh. Deploying a wireless sensor network on an active volcano. IEEE Internet Computing, 10(2): 18-25, March/April 2006.
[10] D. Malan, T. Fulford-Jones, M. Welsh, and S. Moulton. CodeBlue: an Ad Hoc sensor network infrastructure for emerging medical care. In Proceedings of Workshop on Applications of Mobile Embedded Systems, Boston MA, USA, June 2004.
[11] Arduino Project. http://www.arduino.cc/.
[12] nesC: A Programming Language for Deeply Networked Systems. http:// nescc.sourceforge.net/.
[13] Android OS. http://developer.android.com/index.html/.
[14] HTML. http://www.w3schools.com/html/default.asp/.
[15] CSS. http://www.w3schools.com/css/default.asp/.
[16] PHP. http://www.w3schools.com/php/default.asp/.
[17] JavaScript. http://www.w3schools.com/js/default.asp/.
[18] PHP and MySQL Web Development (4th Edition).
[19] HTML Dog: The Best-Practice Guide to XHTML and CSS.
[20] Wireless Sensor Networks (Advanced Texts in Communications and Networking), Ian F. Akyildiz, Mehmet Can Vuran.
[21] Building Wireless Sensor Networks: with ZigBee, XBee, Arduino, and Processing, Robert Faludi.
[22] Fundamentals of Wireless Sensor Networks: Theory and Practice (Wireless Communications and Mobile Computing) by Waltenegus Dargie and Christian Poellabaue.
[24] Building Android Apps with HTML, CSS, and JavaScript: Making Native Apps with Standards-Based Web Tools by Jonathan Starkand Brian Jepson (Jan 30, 2012).

Παράρτημα Α

Σταθμός Βάσης Tmote Sky

BaseStationp.nc

```
// $Id: BaseStationP.nc,v 1.10 2008/06/23 20:25:14 regehr Exp $

/*                                  tab:4
```

```
 * "Copyright (c) 2000-2005 The Regents of the University  of
California.
 * All rights reserved.
 *
 * Permission to use, copy, modify, and distribute this software and
its
 * documentation for any purpose, without fee, and without written
agreement is
 * hereby granted, provided that the above copyright notice, the
following
 * two paragraphs and the author appear in all copies of this
software.
 *
 * IN NO EVENT SHALL THE UNIVERSITY OF CALIFORNIA BE LIABLE TO ANY
PARTY FOR
 * DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES
ARISING OUT
 * OF THE USE OF THIS SOFTWARE AND ITS DOCUMENTATION, EVEN IF THE
UNIVERSITY OF
 * CALIFORNIA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 *
 * THE UNIVERSITY OF CALIFORNIA SPECIFICALLY DISCLAIMS ANY
WARRANTIES,
 * INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF
MERCHANTABILITY
 * AND FITNESS FOR A PARTICULAR PURPOSE.  THE SOFTWARE PROVIDED
HEREUNDER IS
 * ON AN "AS IS" BASIS, AND THE UNIVERSITY OF CALIFORNIA HAS NO
OBLIGATION TO
 * PROVIDE MAINTENANCE, SUPPORT, UPDATES, ENHANCEMENTS, OR
MODIFICATIONS."
 *
 * Copyright (c) 2002-2005 Intel Corporation
 * All rights reserved.
 *
 * This file is distributed under the terms in the attached INTEL-
LICENSE
 * file. If you do not find these files, copies can be found by
writing to
 * Intel Research Berkeley, 2150 Shattuck Avenue, Suite 1300,
Berkeley, CA,
 * 94704.  Attention:  Intel License Inquiry.
 */

/*
 * @author Phil Buonadonna
 * @author Gilman Tolle
 * @author David Gay
 * Revision:    $Id: BaseStationP.nc,v 1.10 2008/06/23 20:25:14
regehr Exp $
 */

/*
 * BaseStationP bridges packets between a serial channel and the
radio.
 * Messages moving from serial to radio will be tagged with the group
 * ID compiled into the TOSBase, and messages moving from radio to
 * serial will be filtered by that same group id.
 */

#include "AM.h"
#include "Serial.h"
```

```
module BaseStationP @safe() {
  uses {
    interface Boot;
    interface SplitControl as SerialControl;
    interface SplitControl as RadioControl;

    interface AMSend as UartSend[am_id_t id];
    interface Receive as UartReceive[am_id_t id];
    interface Packet as UartPacket;
    interface AMPacket as UartAMPacket;

    interface AMSend as RadioSend[am_id_t id];
    interface Receive as RadioReceive[am_id_t id];
    interface Receive as RadioSnoop[am_id_t id];
    interface Packet as RadioPacket;
    interface AMPacket as RadioAMPacket;

    interface Leds;
  }
}

implementation
{
  enum {
    UART_QUEUE_LEN = 12,
    RADIO_QUEUE_LEN = 12,
  };

  message_t  uartQueueBufs[UART_QUEUE_LEN];
  message_t  * ONE_NOK uartQueue[UART_QUEUE_LEN];
  uint8_t    uartIn, uartOut;
  bool       uartBusy, uartFull;

  message_t  radioQueueBufs[RADIO_QUEUE_LEN];
  message_t  * ONE_NOK radioQueue[RADIO_QUEUE_LEN];
  uint8_t    radioIn, radioOut;
  bool       radioBusy, radioFull;

  task void uartSendTask();
  task void radioSendTask();

  void dropBlink() {
    call Leds.led2Toggle();
  }

  void failBlink() {
    call Leds.led2Toggle();
  }

  event void Boot.booted() {
    uint8_t i;

    for (i = 0; i < UART_QUEUE_LEN; i++)
      uartQueue[i] = &uartQueueBufs[i];
    uartIn = uartOut = 0;
    uartBusy = FALSE;
    uartFull = TRUE;

    for (i = 0; i < RADIO_QUEUE_LEN; i++)
      radioQueue[i] = &radioQueueBufs[i];
    radioIn = radioOut = 0;
    radioBusy = FALSE;
    radioFull = TRUE;
```

```
    call RadioControl.start();
    call SerialControl.start();
  }

  event void RadioControl.startDone(error_t error) {
    if (error == SUCCESS) {
      radioFull = FALSE;
    }
  }

  event void SerialControl.startDone(error_t error) {
    if (error == SUCCESS) {
      uartFull = FALSE;
    }
  }

  event void SerialControl.stopDone(error_t error) {}
  event void RadioControl.stopDone(error_t error) {}

  uint8_t count = 0;

  message_t* ONE receive(message_t* ONE msg, void* payload, uint8_t
len);

  event message_t *RadioSnoop.receive[am_id_t id](message_t *msg,
                            void *payload,
                            uint8_t len) {
    return receive(msg, payload, len);
  }

  event message_t *RadioReceive.receive[am_id_t id](message_t *msg,
                            void *payload,
                            uint8_t len) {
    return receive(msg, payload, len);
  }

  message_t* receive(message_t *msg, void *payload, uint8_t len) {
    message_t *ret = msg;

    atomic {
      if (!uartFull)
    {
      ret = uartQueue[uartIn];
      uartQueue[uartIn] = msg;

      uartIn = (uartIn + 1) % UART_QUEUE_LEN;

      if (uartIn == uartOut)
        uartFull = TRUE;

      if (!uartBusy)
        {
          post uartSendTask();
          uartBusy = TRUE;
        }
    }
      else
    dropBlink();
    }

    return ret;
  }
```

```
  uint8_t tmpLen;

  task void uartSendTask() {
    uint8_t len;
    am_id_t id;
    am_addr_t addr, src;
    message_t* msg;
    atomic
      if (uartIn == uartOut && !uartFull)
    {
      uartBusy = FALSE;
      return;
    }

    msg = uartQueue[uartOut];
    tmpLen = len = call RadioPacket.payloadLength(msg);
    id = call RadioAMPacket.type(msg);
    addr = call RadioAMPacket.destination(msg);
    src = call RadioAMPacket.source(msg);
    call UartPacket.clear(msg);
    call UartAMPacket.setSource(msg, src);

    if (call UartSend.send[id](addr, uartQueue[uartOut], len) ==
SUCCESS)
      call Leds.led1Toggle();
    else
      {
    failBlink();
    post uartSendTask();
      }
  }

  event void UartSend.sendDone[am_id_t id](message_t* msg, error_t
error) {
    if (error != SUCCESS)
      failBlink();
    else
      atomic
    if (msg == uartQueue[uartOut])
      {
        if (++uartOut >= UART_QUEUE_LEN)
          uartOut = 0;
        if (uartFull)
          uartFull = FALSE;
      }
    post uartSendTask();
  }

  event message_t *UartReceive.receive[am_id_t id](message_t *msg,
                           void *payload,
                           uint8_t len) {
    message_t *ret = msg;
    bool reflectToken = FALSE;

    atomic
      if (!radioFull)
    {
      reflectToken = TRUE;
      ret = radioQueue[radioIn];
      radioQueue[radioIn] = msg;
      if (++radioIn >= RADIO_QUEUE_LEN)
        radioIn = 0;
```

```
      if (radioIn == radioOut)
        radioFull = TRUE;

      if (!radioBusy)
        {
          post radioSendTask();
          radioBusy = TRUE;
        }
    }
      else
    dropBlink();

    if (reflectToken) {
      //call UartTokenReceive.ReflectToken(Token);
    }

    return ret;
  }

  task void radioSendTask() {
    uint8_t len;
    am_id_t id;
    am_addr_t addr,source;
    message_t* msg;

    atomic
      if (radioIn == radioOut && !radioFull)
    {
      radioBusy = FALSE;
      return;
    }

    msg = radioQueue[radioOut];
    len = call UartPacket.payloadLength(msg);
    addr = call UartAMPacket.destination(msg);
    source = call UartAMPacket.source(msg);
    id = call UartAMPacket.type(msg);

    call RadioPacket.clear(msg);
    call RadioAMPacket.setSource(msg, source);

    if (call RadioSend.send[id](addr, msg, len) == SUCCESS)
      call Leds.led0Toggle();
    else
      {
    failBlink();
    post radioSendTask();
      }
  }

  event void RadioSend.sendDone[am_id_t id](message_t* msg, error_t
error) {
    if (error != SUCCESS)
      failBlink();
    else
      atomic
    if (msg == radioQueue[radioOut])
      {
        if (++radioOut >= RADIO_QUEUE_LEN)
          radioOut = 0;
        if (radioFull)
          radioFull = FALSE;
      }
```

```
    post radioSendTask();
  }
}
```

BaseStationC.nc

```
// $Id: BaseStationC.nc,v 1.6 2008/09/25 04:08:09 regehr Exp $

/*                                      tab:4
 * "Copyright (c) 2000-2003 The Regents of the University  of
California.
 * All rights reserved.
 *
 * Permission to use, copy, modify, and distribute this software and
its
 * documentation for any purpose, without fee, and without written
agreement is
 * hereby granted, provided that the above copyright notice, the
following
 * two paragraphs and the author appear in all copies of this
software.
 *
 * IN NO EVENT SHALL THE UNIVERSITY OF CALIFORNIA BE LIABLE TO ANY
PARTY FOR
 * DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES
ARISING OUT
 * OF THE USE OF THIS SOFTWARE AND ITS DOCUMENTATION, EVEN IF THE
UNIVERSITY OF
 * CALIFORNIA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 *
 * THE UNIVERSITY OF CALIFORNIA SPECIFICALLY DISCLAIMS ANY
WARRANTIES,
 * INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF
MERCHANTABILITY
 * AND FITNESS FOR A PARTICULAR PURPOSE.  THE SOFTWARE PROVIDED
HEREUNDER IS
 * ON AN "AS IS" BASIS, AND THE UNIVERSITY OF CALIFORNIA HAS NO
OBLIGATION TO
 * PROVIDE MAINTENANCE, SUPPORT, UPDATES, ENHANCEMENTS, OR
MODIFICATIONS."
 *
 * Copyright (c) 2002-2003 Intel Corporation
 * All rights reserved.
 *
 * This file is distributed under the terms in the attached INTEL-
LICENSE
 * file. If you do not find these files, copies can be found by
writing to
 * Intel Research Berkeley, 2150 Shattuck Avenue, Suite 1300,
Berkeley, CA,
 * 94704.  Attention:  Intel License Inquiry.
 */

/**
 * The TinyOS 2.x base station that forwards packets between the UART
 * and radio.It replaces the GenericBase of TinyOS 1.0 and the
 * TOSBase of TinyOS 1.1.
 *
```

```
 * <p>On the serial link, BaseStation sends and receives simple
active
 * messages (not particular radio packets): on the radio link, it
 * sends radio active messages, whose format depends on the network
 * stack being used. BaseStation will copy its compiled-in group ID
to
 * messages moving from the serial link to the radio, and will filter
 * out incoming radio messages that do not contain that group ID.</p>
 *
 * <p>BaseStation includes queues in both directions, with a
guarantee
 * that once a message enters a queue, it will eventually leave on
the
 * other interface. The queues allow the BaseStation to handle load
 * spikes.</p>
 *
 * <p>BaseStation acknowledges a message arriving over the serial
link
 * only if that message was successfully enqueued for delivery to the
 * radio link.</p>
 *
 * <p>The LEDS are programmed to toggle as follows:</p>
 * <ul>
 * <li><b>RED Toggle:</b>: Message bridged from serial to radio</li>
 * <li><b>GREEN Toggle:</b> Message bridged from radio to serial</li>
 * <li><b>YELLOW/BLUE Toggle:</b> Dropped message due to queue
overflow in either direction</li>
 * </ul>
 *
 * @author Phil Buonadonna
 * @author Gilman Tolle
 * @author David Gay
 * @author Philip Levis
 * @date August 10 2005
 */

configuration BaseStationC {
}
implementation {
  components MainC, BaseStationP, LedsC;
  components ActiveMessageC as Radio, SerialActiveMessageC as Serial;

  MainC.Boot <- BaseStationP;

  BaseStationP.RadioControl -> Radio;
  BaseStationP.SerialControl -> Serial;

  BaseStationP.UartSend -> Serial;
  BaseStationP.UartReceive -> Serial.Receive;
  BaseStationP.UartPacket -> Serial;
  BaseStationP.UartAMPacket -> Serial;

  BaseStationP.RadioSend -> Radio;
  BaseStationP.RadioReceive -> Radio.Receive;
  BaseStationP.RadioSnoop -> Radio.Snoop;
  BaseStationP.RadioPacket -> Radio;
  BaseStationP.RadioAMPacket -> Radio;

  BaseStationP.Leds -> LedsC;
}
```

Σταθμός Συλλογής Φυσικών Παραμέτρων

Oscilloscope.h

```
/*
 * Copyright (c) 2006 Intel Corporation
 * All rights reserved.
 *
 * This file is distributed under the terms in the attached INTEL-
LICENSE
 * file. If you do not find these files, copies can be found by
writing to
 * Intel Research Berkeley, 2150 Shattuck Avenue, Suite 1300,
Berkeley, CA,
 * 94704.  Attention:  Intel License Inquiry.
 */

// @author David Gay

#ifndef OSCILLOSCOPE_H
#define OSCILLOSCOPE_H

enum {
  /* Number of readings per message. If you increase this, you may
have to
     increase the message_t size. */
  NREADINGS = 3,

  /* Default sampling period. */
  DEFAULT_INTERVAL = 800,

  AM_OSCILLOSCOPE = 0x93
};

typedef nx_struct oscilloscope {
  nx_uint16_t version; /* Version of the interval. */
  nx_uint16_t interval; /* Samping period. */
  nx_uint16_t id; /* Mote id of sending mote. */
  nx_uint16_t count; /* The readings are samples count * NREADINGS
onwards */
  nx_uint16_t tsr[NREADINGS];
  nx_uint16_t temp[NREADINGS];
} oscilloscope_t;

#endif
```

OscilloscopeAppC.nc

```
/*
 * Copyright (c) 2006 Intel Corporation
```

```
 * All rights reserved.
 *
 * This file is distributed under the terms in the attached INTEL-
LICENSE
 * file. If you do not find these files, copies can be found by
writing to
 * Intel Research Berkeley, 2150 Shattuck Avenue, Suite 1300,
Berkeley, CA,
 * 94704.  Attention:  Intel License Inquiry.
 */

/**
 * Oscilloscope demo application. Uses the demo sensor - change the
 * new DemoSensorC() instantiation if you want something else.
 *
 * See README.txt file in this directory for usage instructions.
 *
 * @author David Gay
 */
configuration OscilloscopeAppC { }
implementation
{
  components OscilloscopeC, MainC, ActiveMessageC, LedsC,
    new TimerMilliC(), new SensirionSht11C(), new
HamamatsuS10871TsrC(),
    new AMSenderC(AM_OSCILLOSCOPE), new AMReceiverC(AM_OSCILLOSCOPE);

  OscilloscopeC.Boot -> MainC;
  OscilloscopeC.RadioControl -> ActiveMessageC;
  OscilloscopeC.AMSend -> AMSenderC;
  OscilloscopeC.Receive -> AMReceiverC;
  OscilloscopeC.Timer -> TimerMilliC;
  OscilloscopeC.Leds -> LedsC;
  OscilloscopeC.ReadTSR -> HamamatsuS10871TsrC;
  OscilloscopeC.ReadExtTemp -> SensirionSht11C.Temperature;

}
```

OscilloscopeC.nc

```
/*
 * Copyright (c) 2006 Intel Corporation
 * All rights reserved.
 *
 * This file is distributed under the terms in the attached INTEL-
LICENSE
 * file. If you do not find these files, copies can be found by
writing to
 * Intel Research Berkeley, 2150 Shattuck Avenue, Suite 1300,
Berkeley, CA,
 * 94704.  Attention:  Intel License Inquiry.
 */

/**
 * Oscilloscope demo application. See README.txt file in this
directory.
 *
 * @author David Gay
 */
#include "Timer.h"
#include "Oscilloscope.h"
```

```
module OscilloscopeC @safe()
{
  uses {
    interface Boot;
    interface SplitControl as RadioControl;
    interface AMSend;
    interface Receive;
    interface Timer<TMilli>;
    interface Read<uint16_t> as ReadTSR;
    interface Read<uint16_t> as ReadExtTemp;
    interface Leds;
  }
}
implementation
{
  message_t sendBuf;
  bool sendBusy;

  /* Current local state - interval, version and accumulated readings
*/
  oscilloscope_t local;

  uint8_t readingTemperatures; /* 0 to NREADINGS */
  uint8_t readingLights; /* 0 to NREADINGS */

  /* When we head an Oscilloscope message, we check it's sample
count. If
     it's ahead of ours, we "jump" forwards (set our count to the
received
     count). However, we must then suppress our next count increment.
This
     is a very simple form of "time" synchronization (for an abstract
     notion of time). */
  bool suppressCountChange;

  // Use LEDs to report various status issues.
  void report_problem() { call Leds.led0Toggle(); }
  void report_sent() { call Leds.led1Toggle(); }
  void report_received() { call Leds.led2Toggle(); }

  event void Boot.booted() {
    local.interval = DEFAULT_INTERVAL;
    local.id = TOS_NODE_ID;
    if (call RadioControl.start() != SUCCESS)
      report_problem();
  }

  void startTimer() {
    call Timer.startPeriodic(local.interval);
    readingTemperatures = 0;
    readingLights = 0;
  }

  event void RadioControl.startDone(error_t error) {
    startTimer();
  }

  event void RadioControl.stopDone(error_t error) {
  }

  event message_t* Receive.receive(message_t* msg, void* payload,
uint8_t len) {
```

```
    oscilloscope_t *omsg = payload;

    report_received();

    /* If we receive a newer version, update our interval.
       If we hear from a future count, jump ahead but suppress our
own change
    */
    if (omsg->version > local.version)
      {
    local.version = omsg->version;
    local.interval = omsg->interval;
    startTimer();
      }
    if (omsg->count > local.count)
      {
    local.count = omsg->count;
    suppressCountChange = TRUE;
      }

    return msg;
  }

  /* At each sample period:
     - if local sample buffer is full, send accumulated samples
     - read next sample
  */
  event void Timer.fired() {
    if (readingTemperatures == NREADINGS && readingLights ==
NREADINGS)
      {
    if (!sendBusy && sizeof local <= call AMSend.maxPayloadLength())
      {
        // Don't need to check for null because we've already checked
length
        // above
        memcpy(call AMSend.getPayload(&sendBuf, sizeof(local)),
&local, sizeof local);
        if (call AMSend.send(AM_BROADCAST_ADDR, &sendBuf, sizeof
local) == SUCCESS)
          sendBusy = TRUE;
      }
    if (!sendBusy)
      report_problem();

    readingTemperatures = 0;
    readingLights = 0;

    /* Part 2 of cheap "time sync": increment our count if we didn't
       jump ahead. */
    if (!suppressCountChange)
      local.count++;
    suppressCountChange = FALSE;
      }
    if ((call ReadTSR.read() != SUCCESS) || (call
ReadExtTemp.read() != SUCCESS) )
      report_problem();

  }

  event void AMSend.sendDone(message_t* msg, error_t error) {
    if (error == SUCCESS)
      report_sent();
```

```
    else
      report_problem();

    sendBusy = FALSE;
  }

  event void ReadTSR.readDone(error_t result, uint16_t data) {
    if (result != SUCCESS)
      {
    data = 0xffff;
    report_problem();
      }
    local.tsr[readingLights++] = data;
  }

  event void ReadExtTemp.readDone(error_t result, uint16_t data) {
    if (result != SUCCESS)
      {
    data = 0xffff;
    report_problem();
      }
    local.temp[readingTemperatures++] = data;
  }
}
```

Σταθμός Βάσης Internet Arduino

```
#include <Ethernet.h>
#include <SPI.h>
boolean reading = false;
boolean state = false;

//////////////////////////////////////////////////////////////////////
///
//CONFIGURE
//////////////////////////////////////////////////////////////////////
///
  byte mac[] = {
  0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
  IPAddress ip(192,168,0, 199);

  //byte ip[] = { 192, 168, 1, 10 };   //ip address to assign the
arduino
  byte gateway[] = { 192, 168, 0, 1 }; //ip address of the gatewa or
router

  //Rarly need to change this
  byte subnet[] = { 255, 255, 255, 0 };

  // if need to change the MAC address (Very Rare)
  //byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
  EthernetServer server(8889);
  //Server server = Server(8889); //port 80
//////////////////////////////////////////////////////////////////////
///

void setup(){
  //Pins 10,11,12 & 13 are used by the ethernet shield
```

```
  //Ethernet.begin(mac, ip);
  Ethernet.begin(mac, ip, gateway, subnet);
  server.begin();
  Serial.begin(9600);
}

void loop(){

  // listen for incoming clients, and process qequest.
  checkForClient();

}

void checkForClient(){

  EthernetClient client = server.available();
  //Client client = server.available();

  if (client) {

    // an http request ends with a blank line
    boolean currentLineIsBlank = true;
    boolean sentHeader = false;

    while (client.connected()) {
      if (client.available()) {

        if(!sentHeader){
          // send a standard http response header
          client.println("HTTP/1.1 200 OK");
          client.println("Content-Type: text/html");
          client.println();
          sentHeader = true;
        }

        char c = client.read();

        if(reading && c == ' ') reading = false;
        if(c == '?') reading = true; //found the ?, begin reading the
info


        if(reading){

           switch (c) {
             case '2':
               onPin(2, client);
               break;
             case '3':
               offPin(2, client);
               break;
             case '0':
               if ( state == true ) {
                 client.print("ON");
               } else {
                 client.print("OFF");
               }
               break;
          }
         }

        if (c == '\n' && currentLineIsBlank)  break;
```

```
          if (c == '\n') {
            currentLineIsBlank = true;
          }else if (c != '\r') {
            currentLineIsBlank = false;
          }

        }
      }

      delay(1); // give the web browser time to receive the data
      client.stop(); // close the connection:

    }
}
void onPin(int pin, EthernetClient client){
  client.print("Turning ON Lights");
  client.print("<br>");
  Serial.print('H');
  delay(10);
  if (Serial.available()>0){
    if (Serial.read() == 'K'){
      delay(10);
      client.print("OK");
    }
  }
  state = true;

}

void offPin(int pin, EthernetClient client){
  client.print("Turning OFF Lights");
  client.print("<br>");
  Serial.print('L');
  delay(10);
  if (Serial.available()>0){
    if (Serial.read() == 'K'){
      delay(10);
      client.print("OK");
    }
  }
  state = false;

}
```

Σταθμός Ενεργοποιητή Arduino

```
void setup(){
  pinMode(2, OUTPUT);
  Serial.begin(9600);
}

void loop(){
  if (Serial.available() > 0){

          switch (Serial.read()) {
           case 'H':
             digitalWrite(2, HIGH);
              Serial.print('K');
```

```
                break;
            case 'L':
                digitalWrite(2, LOW);
                  Serial.print('K');
                break;
        }
    }
}
```

Διαδικτυακή Εφαρμογή Room:i/o

Μοντέλα

Constants.php

```
<?php
define("DB_SERVER", "localhost");
define("DB_USER", "root");
define("DB_PASS", "trustno1");
define("DB_NAME", "ihome");
?>
```

Connection.php

```
<?php
include 'constants.php';

    $connection = mysql_connect(DB_SERVER,DB_USER,DB_PASS);
    if (!$connection) {
        die("Database connection failed: " . mysql_error());
    }

    $db_select = mysql_select_db(DB_NAME,$connection);
    if (!$db_select) {
        die("Database selection failed: " . mysql_error());
    }
?>
```

Users.php

```
<?php
    include 'connection.php';

    function CreateUser( $username, $password, $arduinoip){
        $success = mysql_query(
            "INSERT INTO
                users
            SET
                username = '$username',
                password = '$password',
                ip = '$arduinoip'"
        );
        return $success;
    }
```

```
    function AuthenticateUser( $username, $password ){
        $res = mysql_query(
            "SELECT
                userid
            FROM
                users
            WHERE
                username = '$username'
                AND password = '$password'
            LIMIT 1;"
        );

        if ( mysql_num_rows( $res ) == 1 ) {
            $user = mysql_fetch_array( $res );
            return $user['userid'];
        }

        return false;
    }

    function user_ip( $username, $password ){
        $res = mysql_query(
            "SELECT
                ip
            FROM
                users
            WHERE
                username = '$username'
                AND password = '$password'
            LIMIT 1;"
        );
            if ( mysql_num_rows( $res ) == 1 ) {
            $userip = mysql_fetch_array( $res );
            return $userip['ip'];
        }
        return false;
    }
?>
```

Parse.php

```
<?php
    include 'connection.php';
        $res = mysql_query(
            "SELECT
                string,dataid
            FROM
                data
            ORDER BY dataid DESC LIMIT 1;"
        );
            if ($row = mysql_fetch_array( $res )){
                $data = $row['string'];
                $nospaces = str_replace(' ', '', $data);
                $lightone = substr( "$nospaces",32,4);
                $lighttow = substr( "$nospaces",36,4);
                $lightthree = substr( "$nospaces",40,4);
                $temperatureon = substr( "$nospaces",44,4);
                $temperaturetw = substr( "$nospaces",48,4);
                $temperatureth = substr( "$nospaces",52,4);
                $decTemp1 = hexdec($temperatureon);
```

```
                $decTemp2 = hexdec($temperaturetw);
                $decTemp3 = hexdec($temperatureth);
                $decLight1 = hexdec($lightone);
                $decLight2 = hexdec($lighttwo);
                $decLight3 = hexdec($lightthree);
                $tempr = ((($decTemp1+$decTemp2+$decTemp3)/
3)*0.01)-39.6);
                $lightt = (($decLight1+$decLight2+$decLight3)/
3)*2.288);

                mysql_query(
                    "INSERT INTO
                        temp
                    SET
                        temp = '$tempr',
                        created = NOW();"
                );

                    mysql_query(
                        "INSERT INTO
                            light
                        SET
                            light = '$lightt';"
                    );

                        mysql_query(
                        "TRUNCATE data;"
                    );
            }
?>
```

Temprature.php

```
<?php
include 'connection.php';

    function confirm_query($result_set) {
        if (!$result_set) {
            die("Database query failed: " . mysql_error());
        }
    }

    function get_temp() {
    global $connection;
    $query = "SELECT
                temp
              FROM
                temp
              ORDER BY tempid DESC LIMIT 1";
    $temprature = mysql_query($query, $connection);
    confirm_query($temprature);
    return $temprature;
    }

    function get_tempaver() {
    global $connection;
    $query = "SELECT
                AVG( temp ) AS average
              FROM
                temp
              WHERE
```

```
                created > NOW() - INTERVAL 3 HOUR";
    $average = mysql_query($query, $connection);
    confirm_query($average);
    return $average;
    }
    ?>
```

Light.php

```
<?php
include 'connection.php';

    function confirm_query($result_set) {
        if (!$result_set) {
            die("Database query failed: " . mysql_error());
        }
    }

    function get_light() {
    global $connection;
    $query = "SELECT
                light
              FROM
                light
              ORDER BY lightid DESC LIMIT 1;";
    $light = mysql_query($query, $connection);
    confirm_query($light);
    return $light;
    }
?>
```

Tmote.php

```
<?php
    error_reporting(0);
    $is_open = fsockopen("localhost", 9002,$errno, $errstr, 30);

    if (!$is_open) {
        echo "<p class=\"no\">PROBLEM!</p>";
    } else {
        echo "<p class=\"yes\">OK!</p>";
    }
?>
```

Arduino.php

```
<?php
    session_start();
    $_URL = "http://".$_SESSION[ 'userip' ]."/?0";
    $handle = curl_init($_URL);
    curl_setopt($handle,  CURLOPT_RETURNTRANSFER, TRUE);
    curl_setopt($handle,  CURLOPT_CONNECTTIMEOUT_MS, 20);

    /* Get the HTML or whatever is linked in $url. */
    $response = curl_exec($handle);

    /* Check for 404 (file not found). */
```

```
    $httpCode = curl_getinfo($handle, CURLINFO_HTTP_CODE);
    if($httpCode == 0) {
        echo "<p class=\"no\">PROBLEM!</p>";
    } else {
        echo "<p class=\"yes\">OK!</p>";
    }
?>
```

Ελεγκτές

Functions.php

```
<?php
    function check_user() {
        if( isset( $_SESSION[ 'username' ] ) ) {
        return true;
        } else {
            return false;
        }
    }

    function create_panel() {
        if ( check_user() ) {
            if ( $_SERVER['PHP_SELF'] == "/roomio/index.php" ) {
?>
            <div class="panel">
                <div id="hellouser">
                    <img src="./images/user.png" id="process"
alt="user"/>
                    <?php
                        echo " {$_SESSION[ 'username' ]}";
                    ?>
                </div>
                <div class="menu">
                    <img src="./images/process.png" id="process"
alt="menu"/>
                </div>
                <div id="menu_opt">
                    <ul class="settings">
                        <li><a class="menu_log" href="./controllers/
logout.php">Logout</a></li>
                        <li><a class="menu_check" href="./
controllers/systemcheck.php">Check System</a></li>
                    </ul>
                </div>
            </div>
            <script type="text/javascript">
                $( "#menu_opt" ).hide();
                $('.menu').toggle(function() {
                    $( "#menu_opt" ).show();
                    $('.menu').css('background-color','#D14836');
                },function() {
```

```
                    $( "#menu_opt" ).hide();
                    $('.menu').css('background-color','');
                } );
            </script>
            <h1><img class="imghome" src="./images/io3.jpg"
alt="Room:i/o" /></h1>
      <?php } else { ?>
            <div class="panel">
                <div id="hellouser">
                    <img src="../images/user.png" id="process"
alt="user"/>
                    <?php
                        echo " {$_SESSION[ 'username' ]}";
                    ?>
                </div>
                <div class="menu">
                    <a href="../"><span id="signup"
class="home">Home</span></a>
                    <img src="../images/back.png" id="process"
alt="Home"/>
                </div>
            </div>
            <h1><img class="imghome" src="../images/io3.jpg"
alt="Room:i/o" /></h1>
      <?php }
        } else {
            ?>
            <div class="panel">
                <div class="menu">
                    <?php
                        if ($_SERVER['PHP_SELF'] == "/roomio/
controllers/register.php" ) {
                    ?>
                            <a href="../"><span id="signup"
class="home">Home</span></a>
                            <img src="../images/back.png"
id="process" alt="Home"/>
                </div>
            </div>
                    <?php
                        } else {
                    ?>
                            <a href="../controllers/
register.php"><span id="signup">SignUp</span></a>
                            <img src="../images/add.png" id="process"
alt="add"/>
                </div>
            </div>
            <?php
                        }
            ?>
            <h1><img class="imghome" src="../images/io3.jpg"
alt="Room:i/o" />
      <?php
        }
    }
?>
```

Login.php

```
<?php
    if ( isset( $_GET[ 'error' ] ) && $_GET[ 'error' ] == 'yes' ) {
        $error = true;
    }
    include '../views/login.php';
?>
```

Dologin.php

```
<?php
    session_start();

    include '../models/users.php';

    if ( !isset( $_POST[ 'username' ] ) || !
isset( $_POST[ 'password' ] ) ) {
        die();
    }
    if ( !empty( $_POST[ 'username' ] ) && !
empty( $_POST[ 'password' ] ) ) {
    $userid = AuthenticateUser( $_POST[ 'username' ],
$_POST[ 'password' ] );
    $user_ip = user_ip( $_POST[ 'username' ], $_POST[ 'password' ] );
    if ( $userid !== false ) {
        $_SESSION[ 'userid' ] = $userid;
        $_SESSION[ 'username' ] = $_POST[ 'username' ];
        $_SESSION[ 'userip' ] = $user_ip;
        $username = $_SESSION[ 'username' ];
        header( 'Location: ../index.php' );
    }
    else {
        header( 'Location: login.php?error=yes' );
    }
    }
    else {
        header( 'Location: login.php?error=yes' );
    }
?>
```

Register.php

```
<?php
    if ( isset( $_GET[ 'missing' ] ) && $_GET[ 'missing' ] == 'yes' )
{
        $missing = true;
    }
    include '../views/register.php';
?>
```

Doregister.php

```
<?php
    session_start();
```

```
    include '../models/users.php';
    if( isset( $_POST[ 'username' ] ) &&
        isset( $_POST[ 'password' ] ) && !
empty( $_POST[ 'username' ] ) && !empty( $_POST[ 'password' ] ) ){

        $userid = CreateUser( $_POST[ 'username' ],
$_POST[ 'password' ], $_POST[ 'arduinoip' ] );
        if( $userid ) {
            $_SESSION[ 'username' ] = $_POST[ 'username' ];
            $_SESSION[ 'userid' ] = $userid;
        }
        header( 'Location: ../index.php' );
    }
    else {
        header( 'Location: register.php?missing=yes' );
    }
?>
```

Logout.php

```
<?php
    session_start();
    unset( $_SESSION[ 'username' ] );
    header( 'Location: ../index.php' );
?>
```

Tempratures.php

```
<?php
    include '../models/temprature.php';

    $tempratures = get_temp();
    while ($row = mysql_fetch_array( $tempratures )){
        $tempsfinal =  $row['temp'];
        echo round( $tempsfinal, 1 );
        echo " C";
    }
?>
```

Average.php

```
<?php
    include '../models/temprature.php';

    $tempaver = get_tempaver();
        if ($row2 = mysql_fetch_array( $tempaver )){
            $aver =  $row2['average'];
            echo round($aver, 1);
            echo " C";
        }
?>
```

Lights.php

```
<?php
    session_start();
```

```
    include '../models/light.php';

    $lights = get_light();
    while ($row = mysql_fetch_array( $lights )){
        echo $lightsfinal =  $row['light'];
        echo " lx";
    }
    if ( $lightsfinal <= 5 ) {
        $contents1 = @file_get_contents('http://'.
$_SESSION['userip'].'/?2');
    }
?>
```

Status1.php

```
<?php
    session_start();
    $contents1 = @file_get_contents('http://'.$_SESSION['userip'].'/?
0');
        echo $contents1;
?>
```

Systemcheck.php

```
<?php
    session_start();
    include '../views/check.php';
?>
```

Προβολές

Header.php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML Basic 1.1//EN" "http://
www.w3.org/TR/xhtml-basic/xhtml-basic11.dtd">
    <head>
        <title>Room:i/o</title>
        <meta http-equiv="Content-Type" content="text/html;
charset=utf-8"/>
        <meta name="viewport" content="user-scalable=no,
width=device-width, height=device-height, target-densitydpi=device-
dpi" />
        <link rel="stylesheet" type="text/css" href="/roomio/css/
android.css" media="only screen and (max-width: 480px)" />
        <link rel="stylesheet" type="text/css" href="/roomio/css/
style.css" media="screen and (min-width: 481px)" />
        <script type="text/javascript" src="./js/
jquery-1.7.min.js"></script>
    </head>
    <body>
        <div id="universe">
```

Home.php

```
<?php
```

```
    include 'header.php';
    include './controllers/functions.php';
    if ( check_user() ){
        create_panel();
?>
    <script type="text/javascript" src="./js/javascripts.js"></
script>
    <div class="room">
        <h2>Temprature <img src="./images/accept.png" id="process"
alt="user"/></h2>
        <div id="realtemp">
            <h3>Real Time</h3>
            <p class="room1"></p>
        </div>
        <div id="averagetemp">
            <h3>Average</h3>
            <p class="average"></p>
        </div>
        <h2>Lights <img src="./images/accept.png" id="process"
alt="user"/></h2>
        <div id="reallight">
            <h3>Real Time</h3>
            <p class = "room2"></p>
        </div>
        <div id="condition">
            <h3>Lights Condition</h3>
            <div id="button">
                <img src="./images/switchbuttonon.png" id="imgon"/
><img src="./images/switchbuttonoff.png" id="imgoff"/>
            </div>
        </div>
    </div>
       <script type="text/javascript">
        $( "#imgon" ).hide();
        $( "#imgoff" ).hide();

        var refreshId = setInterval( function() {
           $.get('./controllers/status1.php', function(data) {
            if (data == "OFF") {
                $( "#imgon" ).hide();
                $( "#imgoff" ).show();
                stateToggle('1');
            } else if (data == "ON") {
                $( "#imgon" ).show();
                $( "#imgoff" ).hide();
                stateToggle('2');
            }
            });
        }, 1000 );

        function stateToggle(data){
            if ( data == "1" ){
                $('#imgoff').click(function() {
                    $.get( 'http://<?php echo $_SESSION['userip']?>/?
2' );
                    $( "#imgoff" ).hide();
                    $( "#imgon" ).show();
                });
            } else if (data == "2"){
                $('#imgon').click(function() {
                    $.get( "http://<?php echo $_SESSION['userip']?>/?
3" );
                    $( "#imgon" ).hide();
```

```
                    $( "#imgoff" ).show();
                });
            }
        }

        $('#imgon').click(function() {
                    callbacka();
                });
        $('#imgoff').click(function() {
                    callbackb();
                });


        function callbacka() {
                var ua = navigator.userAgent.toLowerCase();
                var isAndroid = ua.indexOf( "android" ) > -1;
                if ( isAndroid ) {
                    Android.showToast( "Turning OFF Lights" );
                }
            }
        function callbackb() {
                var ua = navigator.userAgent.toLowerCase();
                var isAndroid = ua.indexOf( "android" ) > -1;
                if ( isAndroid ) {
                    Android.showToast( "Turning ON Lights" );
                }
            }
        </script>
<?php
    } else {
        header('Location: ./controllers/login.php');
    }
 ?>
<?php
    include 'footer.php';
?>
```

Footer.php

```
        </div>
    </body>
</html>
```

Register.php

```
<?php
    include 'header.php';
    include '../controllers/functions.php';
    create_panel();
?>
    <h4>Sign Up!</h4>
    <form action="../controllers/doregister.php" method="post"
class="form_login">
        <h2>Credentials</h2>
        <?php
            if ( isset ( $missing )) {
                ?><div class="error">Missing Values</div><?php
            }
        ?>
```

```
        <label>Username:</label>
        <input type="text" name="username" id="credentials"/>
        <label>Password:</label>
        <input type="password" name="password" id="credentials"/>
        <label>Arduino IP:</label>
        <input type="text" name="arduinoip" id="credentials" />
        <input type="submit" class="submit" value="Sign Up" />
    </form>
<?php
    include 'footer.php';
?>
```

Login.php

```
<?php
    include 'header.php';
    include '../controllers/functions.php';
    create_panel();
?>
    <h4>Welcome Home!</h4>
            <form action="../controllers/dologin.php" method="post"
class="form_login">
            <h2>SignIn</h2>
            <?php
                if ( isset( $error ) ) {
            ?><div class="error">Wrong username, or password</div>
            <?php
                }
            ?>
            <label>Username:</label>
            <input type="text" name="username" id="credentials"/>
            <label>Password:</label>
            <input type="password" name="password" id="credentials"/>
            <input type="submit" value="SignIn" class="submit" />
            </form>
<?php
    include 'footer.php';
?>
```

Check.php

```
<?php
    include 'header.php';
    include '../controllers/functions.php';
    create_panel();
?>
<script type="text/javascript" src="../js/jquery-1.7.min.js"></
script>
    <div class="room">
    <h2 >Checking your enviroment's components:</h2>
    <script type="text/javascript">
        $( document ).ready( function() {
            $( "#imgProg1" ).show();
            setTimeout( function() {
                $( '#LoadPage1' ).load( '../models/arduino.php',
function() {
                    $( "#imgProg1" ).hide();
                } );
            }, 4000 );
```

```
        } );

        $(document).ready(function() {
            $("#imgProg2").show();
            setTimeout( function() {
                $('#LoadPage2').load( '../models/tmote.php',
function( data2 ) {
                    $("#imgProg2").hide();
                } );
            }, 4000 );
        } );
    </script>
    <div id="arduino">
        <p class = "checkar">Check for Arduino</p>
        <div id="LoadPage1" class="divPage"></div>
        <img alt="Progress" src="../images/ajax-loader.gif"
id="imgProg1" visible="false" />
    </div>
    <div id="tmote">
        <p class = "checktm">Check for Tmote Sky</p>
        <div id="LoadPage2" class="divPage"></div>
        <img alt="Progress" src="../images/ajax-loader.gif"
id="imgProg2" visible="false" />
    </div>
    <input  class="submit" type="button" value="Re-Check"
onclick="window.location='../controllers/systemcheck.php'"/>
    </div>
<?php
    include '../views/footer.php';
?>
```

Index.php

```
<?php
    session_start();
    include 'views/home.php';
?>
```

JavaScript

Javascripts.js

```
var refreshId = setInterval(function(){
    $('.room1').load('controllers/tempratures.php');
}, 1000);

var refreshId = setInterval(function(){
    $('.average').load('controllers/average.php');
}, 1000);

var refreshId = setInterval( function() {
    $( '.room2' ).load( 'controllers/lights.php' );
}, 1000 );
```

CSS

Style.css

```
html, body {
    margin: 0;
    padding: 0;
    background-color: #E5E5E5;
}
#signup {
    font-family: "Arial";
    font-weight: bold;
    font-size: 25px;
    vertical-align: middle;
    color: #fff;
}
label {
    font-family: "Arial";
    font-weight: bold;
    font-size: 30px;
    margin: 5px 10px 5px 310px;
    padding: 0px;
    width: 200px;
    display: block;
}
#credentials {
    float: none;
    padding: 0px;
    margin: 5px auto 5px;
    text-align: center;
    font-size: 25px;
    clear: both;
    display: block;
    width: 300px;
    height: 40px;
}
.form_login {
    border: solid thin #ccc;
    border-radius: 7px;
    width: 70%;
    padding: 10px;
    margin: 5px auto 5px;
}
.submit {
    background-color: #4D90FE;
    color: #fff;
    font-family: "Arial";
    font-weight: bold;
    text-shadow: 0 1px rgba(0, 0, 0, 0.1);
    text-align: center;
    font-size: 25px;
    float: none;
    margin:10px auto 15px auto;
    border: 1px solid #3079ED;
    border-radius: 6px;
    box-shadow:0px 3px 10px #B9B9B9;
    -webkit-box-shadow:0px 3px 10px #B9B9B9;
    -moz-box-shadow: 0px   3px 10px #B9B9B9;
}
```

```
.submit:hover {background-color: #4787ED;}

.error {
    background-color: #DD4B39;
    color: #fff;
    height: 40px;
    font-size: 30px;
    vertical-align: middle;
    margin-bottom: 10px;
}
.panel {
    margin: 10px auto 15px;
    overflow: hidden;
}
.menu {
    margin: 0px 30px 10px 10px;
    background-color: #DD4B39;
    width: 150px;
    float: right;
    border: 1px solid transparent;
    vertical-align: middle;
    border-radius: 6px;
    box-shadow: 0px 3px 10px #B9B9B9;
    -webkit-box-shadow: 0px 3px 10px #B9B9B9;
    -moz-box-shadow: 0px   3px 10px #B9B9B9;
}
.menu:hover {
    background-color: #D14836;
}
#process {
    vertical-align: middle;
}
#hellouser {
    vertical-align: middle;
    padding: 0px;
    float: left;
    font-family: "arial";
    font-weight: bold;
    font-size: 40px;
    color: #4D90FE;
    width: 220px;
}
#menu_opt {
    margin-top: 55px;
    text-align: left;
    width: 200px;
    height: 100px;
    vertical-align: middle;
    min-height: 40px; /*min height of DIV should be set to at least
2x the width of the arrow*/
    border-left: solid thin #E5E5E5;
    border-right: solid thin #E5E5E5;
    border-bottom: solid thin #E5E5E5;
    margin-left: 1080px;
    background-color: #F8F8F8 ;
    padding: 5px;
    z-index: 0;
    position: absolute;
    -moz-border-radius: 5px; /*add some nice CSS3 round corners*/
    -webkit-border-radius: 5px;
    border-radius: 5px;
    margin-bottom: 2em;
```

```
    -webkit-box-shadow: 1px 2px 5px #E6E6E6;
    -moz-box-shadow: 1px 2px 5px #E6E6E6;
    box-shadow: 1px 2px 5px #E6E6E6;
}
#menu_opt:after { /*arrow added to uparrowdiv DIV*/
    content: '';
    display: block;
    position: absolute;
    top: -20px; /*should be set to -border-width x 2 */
    left: 60px;
    width: 0;
    height: 0;
    border-color: transparent transparent #F8F8F8 transparent; /
*border color should be same as div div background color*/
    border-style: solid;
    border-width: 10px;
}
.settings {
    margin-top: 15px;
    padding: 0px;
}
.menu_log {
    font-family: "arial";
    font-size: 25px;
    font-weight: bold;
    vertical-align: middle;
    color: #2D2D2D;
}
.menu_log:hover {color: #DD4B39;}
.menu_check {
    font-family: "arial";
    font-size: 25px;
    font-weight: bold;
    vertical-align: middle;
    color: #2D2D2D;
}
.menu_check:hover {color: #DD4B39;}
h4 {
    font-family: "Verdana";
    font-size: 40px;
    font-weight: bold;
    color: #000;
    text-align: center;
    margin: 10px auto 35px;
}
h1,h1 a {
    font-family: "Verdana";
    font-size: 50px;
    font-weight: bold;
    text-shadow: 2px 2px 2px #000;
    color: #910000;
    text-align: center;
    margin: 0px auto 5px;
    border-bottom: dotted 2px #ccc;
    border-top: dotted 2px #ccc;
}
p.no{
color : red;
font-family: "Arial";
font-weight: bold;
font-size: 30px;
margin: 5px auto 5px auto;
```

```
}
p.yes{
color : green;
font-family: "Arial";
font-weight: bold;
font-size: 30px;
margin: 5px auto 5px auto;
}
p.checkar{
    text-align: center;
    border-bottom: 2px solid #66AAFF;
    font-size: 25px;
    width:60%;
    margin: 25px auto 20px auto;
}
p.checktm{
    text-align: center;
    border-bottom: 2px solid #66AAFF;
    font-size: 25px;
    width:60%;
    margin: 25px auto 20px auto;
}
}
#tmote{height:80px;
margin:15px auto 10px auto;}
#arduino{height:80px;
margin:15px auto 20px auto;}
#LoadPage2{
margin: 10px auto 10px auto;
}
#LoadPage1{
margin: 10px auto 10px auto;
}
#imgProg1{
margin: 10px auto 10px auto;
}
#imgProg2{
margin: 10px auto 10px auto;
}
ul {
    list-style: none;
    margin-top: 35px;
    margin-bottom:0px;
    margin-left: auto;
    margin-right: auto;
    padding: 0px;
}
h2 {
    font-family: "Verdana";
    text-align: center;
    margin-top: 5px;
    margin-left: auto;
    margin-right: auto;
    margin-bottom: 15px;
    color: #DD4B39;
    background-color: #2D2D2D;
    font-size: 25px;
    height:35px;
    vertical-align:middle;
    width:100%;
    box-shadow:0 5px 8px rgba(0, 0, 0, 0.35);
}
```

```
#realtemp{
width:40%;
margin:5px auto 5px;
}
#averagetemp{
width:40%;
margin:5px auto 5px;
}
#reallight{
width:40%;
margin:5px auto 5px;
}
#condition{
width:40%;
margin:5px auto 5px;
}
h3 {
    font-family: "Verdana";
    text-align: center;
    /*margin-top: 25px;
    margin-left: auto;
    margin-right: auto;
    margin-bottom: 15px;*/
    color: #2D2D2D;
    font-size: 18px;
    padding: 5px;
    vertical-align:middle;
    text-shadow: 0 1px #ccc;
    border-bottom: 2px solid #66AAFF;
}
p.room1 {
    font-family: "Arial";
    font-size: 45px;
    font-weight: bold;
    color: #DD4B39;
    width: 170px;
    vertical-align:middle;
    height:50px;
    border-bottom:1px solid #ccc;
    margin: 5px auto 5px;
    text-shadow: 0 1px #fff;
    box-shadow:0px 1px 9px #000;
    -webkit-box-shadow:0px 1px 9px #000;
    -moz-box-shadow: 0px   1px 9px #000;
    }
p.average {
    font-family: "Arial";
    text-align: center;
    font-size: 45px;
    font-weight: bold;
    vertical-align:middle;
    height:50px;
    color: #DD4B39;
    width: 170px;
    border-bottom:1px solid #ccc;
    margin: 5px auto 15px;
    text-shadow: 0 1px #fff;
    box-shadow:0px 1px 9px #000;
    -webkit-box-shadow:0px 1px 9px #000;
    -moz-box-shadow: 0px   1px 9px #000;
    }
#universe {
```

```
    background-color: #FFF;
    width: 1300px;
    height: 900px;
    margin-left: auto;
    margin-right: auto;
    margin-top: 25px;
    margin-bottom: 30px;
    z-index:1;
    text-align: center;
    -moz-border-radius-topleft: 12px;
    -moz-border-radius-bottomright: 12px;
    border-top-left-radius: 12px;
    border-bottom-right-radius: 12px;
    padding-bottom: 20px;
    box-shadow: 0px 0px 10px #000;
    -moz-box-shadow: 0px 0px 10px #000;
    padding-left:0px;
    padding-right:0px;
    padding-top:10px;
    vertical-align: middle;
}
div.room {
    background-color: #fff;
    width: 70%;
    vertical-align: middle;
    margin-left: auto;
    margin-right: auto;
    border: 1px solid #CDCDCD;
    margin-top:20px;
    margin-bottom:20px;
    border-radius: 7px;
}
.buttonroom{
    width:150px;
    background-color: #4D90FE;
    color: #fff;
    font-weight: bold;
    text-shadow: 0 1px rgba(0, 0, 0, 0.1);
    height:30px;
    font-family: "Veranda";
    font-size: 18px;
    margin-top: 200px;
    border: 1px solid #3079ED;
    border-radius: 2px;
    box-shadow:0px 3px 10px #B9B9B9;
    -webkit-box-shadow:0px 3px 10px #B9B9B9;
    -moz-box-shadow: 0px   3px 10px #B9B9B9;
    }
.buttonroom:hover {
    background-color: #4787ED;
}
.back{
    width:150px;
    height:30px;
    font-family: "Veranda";
    font-size: 18px;
    margin: 60px auto 10px 5px;
    background-color: #4D90FE;
    color: #fff;
    font-weight: bold;
    text-shadow: 0 1px rgba(0, 0, 0, 0.1);
    border: 1px solid #3079ED;
    border-radius: 2px;
```

```
    box-shadow:0px 3px 10px #B9B9B9;
    -webkit-box-shadow:0px 3px 10px #B9B9B9;
    -moz-box-shadow: 0px   3px 10px #B9B9B9;
    }
.back:hover {
    background-color: #4787ED;
}
#button{
height:50px;
}
.back11{
    width:150px;
    height:30px;
    font-family: "Veranda";
    font-size: 18px;
    margin: 60px auto 10px 5px;
    background-color: #4D90FE;
    color: #fff;
    font-weight: bold;
    text-shadow: 0 1px rgba(0, 0, 0, 0.1);
    border: 1px solid #3079ED;
    border-radius: 2px;
    box-shadow:0px 3px 10px #B9B9B9;
    -webkit-box-shadow:0px 3px 10px #B9B9B9;
    -moz-box-shadow: 0px   3px 10px #B9B9B9;
    }
.back11:hover {
    background-color: #4787ED;
}
.back1{
    width:150px;
    height:30px;
    font-family: "Veranda";
    font-size: 18px;
    margin: 25px auto 5px auto;
    background-color: #4D90FE;
    color: #fff;
    font-weight: bold;
    text-shadow: 0 1px rgba(0, 0, 0, 0.1);
    border: 1px solid #3079ED;
    border-radius: 2px;
    box-shadow:0px 3px 10px #B9B9B9;
    -webkit-box-shadow:0px 3px 10px #B9B9B9;
    -moz-box-shadow: 0px   3px 10px #B9B9B9;
    }
.back1:hover {
    background-color: #4787ED;
}
.bcheck{
    width:150px;
    height:30px;
    font-family: "Veranda";
    font-size: 18px;
    margin: 60px 5px 10px auto;
    background-color: #4D90FE;
    color: #fff;
    font-weight: bold;
    text-shadow: 0 1px rgba(0, 0, 0, 0.1);
    border: 1px solid #3079ED;
    border-radius: 2px;
    box-shadow:0px 3px 10px #B9B9B9;
    -webkit-box-shadow:0px 3px 10px #B9B9B9;
    -moz-box-shadow: 0px   3px 10px #B9B9B9;
```

```
    }
.bcheck:hover {
    background-color: #4787ED;
}
.bcheck11{
    width:150px;
    height:30px;
    font-family: "Veranda";
    font-size: 18px;
    margin: 60px 5px 10px auto;
    background-color: #4D90FE;
    color: #fff;
    font-weight: bold;
    text-shadow: 0 1px rgba(0, 0, 0, 0.1);
    border: 1px solid #3079ED;
    border-radius: 2px;
    box-shadow:0px 3px 10px #B9B9B9;
    -webkit-box-shadow:0px 3px 10px #B9B9B9;
    -moz-box-shadow: 0px   3px 10px #B9B9B9;
    }
.bcheck11:hover {
    background-color: #4787ED;
}
div.control{
    width: 550px;
    height: 450px;
    margin-left: auto;
    margin-right: auto;
    margin-top:50px;
    border: 1px solid #CDCDCD;
}
.buttoncontrol1{
    height:40px;
    font-family: "Veranda";
    font-size: 18px;
    margin-top:5px;
    margin-left: 20px;
    margin-right: 20px;
    margin-bottom: 10px;
    background-color: #DD4B39;
    color: #fff;
    font-weight: bold;
    text-shadow: 0 1px rgba(0, 0, 0, 0.1);
    border: 1px solid transparent;
    border-radius: 2px;
    box-shadow:0px 3px 10px #B9B9B9;
    -webkit-box-shadow:0px 3px 10px #B9B9B9;
    -moz-box-shadow: 0px   3px 10px #B9B9B9;
    }
.buttoncontrol1:hover {
    background-color: #D14836;
}
.buttoncontrol2{
    height:40px;
    font-family: "Veranda";
    font-size: 18px;
    margin-top:5px;
    margin-left: 20px;
    margin-right: 20px;
    margin-bottom: 10px;
    background-color: #DD4B39;
    color: #fff;
```

```
    font-weight: bold;
    text-shadow: 0 1px rgba(0, 0, 0, 0.1);
    border: 1px solid transparent;
    border-radius: 2px;
   box-shadow:0px 3px 10px #B9B9B9;
    -webkit-box-shadow:0px 3px 10px #B9B9B9;
    -moz-box-shadow: 0px   3px 10px #B9B9B9;
    }
.buttoncontrol2:hover {
    background-color: #D14836;
}
.imgoption{
padding:0px;
margin-top:50px;
width:130px;
height:70px;
}
.imgoption2{
display: none;
}
    a {
    text-decoration: none;
    color: #37f;
    font-weight: bold;
}
.anh1{
display:none;}

p.room2 {
    font-family: "Arial";
    text-align: center;
    font-size: 45px;
    font-weight: bold;
    color: #DD4B39;
    width: 200px;
    height:50px;
    vertical-align: middle;
    border-bottom:1px solid #ccc;
    margin:15px auto 10px auto;
    text-shadow: 0 1px #fff;
    box-shadow:0px 1px 9px #000;
    -webkit-box-shadow:0px 1px 9px #000;
    -moz-box-shadow: 0px   1px 9px #000;
    }
.button1{
    width:100px;
    height:30px;
    font-family: "Veranda";
    font-size: 18px;
    margin-top: 5px;
    }
p.status1 {
    font-family: "Arial";
    text-align: center;
    font-size: 28px;
    font-weight: bold;
    color: #DD4B39;
    width: 60%;
    border-bottom:1px solid #ccc;
    margin:15px auto 10px auto;
    text-shadow: 0 1px #fff;
    box-shadow:0px 1px 9px #000;
```

```
    -webkit-box-shadow:0px 1px 9px #000;
    -moz-box-shadow: 0px   1px 9px #000;
    }
p.status2 {
    font-family: "Veranda";
    text-align: center;
    font-size: 16px;
    font-weight: bold;
    color: #DD4B39;
    width: 80%;
    height:40px;
    margin:5px auto 5px auto;
    }
.arrow{
    padding:0px;
    margin:0px auto 10px auto;
}
p.iptext {
    font-family: "Veranda";
    text-align: center;
    font-size: 25px;
    width: 90%;
    margin:80px auto 20px auto;
    color: #fff;
    border: 1px solid #D14836;
    background-color: #DD4B39;
    border-radius: 7px;
    text-shadow: 0 1px rgba(0, 0, 0, 0.1);

}
.ipfield {
    width: 90%;
    height: 40px;
    text-align: center;
    font-size: 25px;
    float: none;
    margin:0px auto 10px auto;
    border-radius: 7px;
}
.usrfield {
    width: 90%;
    height: 40px;
    text-align: center;
    font-size: 25px;
    float: none;
    margin:0px auto 10px auto;
    border-radius: 7px;
}
.ipsubmit {
    background-color: #4D90FE;
    color: #fff;
    font-weight: bold;
    text-shadow: 0 1px rgba(0, 0, 0, 0.1);
    width: 70%;
    text-align: center;
    font-size: 25px;
    float: none;
    margin:10px auto 0px auto;
    border: 1px solid #3079ED;
    border-radius: 2px;
    box-shadow:0px 3px 10px #B9B9B9;
    -webkit-box-shadow:0px 3px 10px #B9B9B9;
```

```
    -moz-box-shadow: 0px   3px 10px #B9B9B9;
}
.ipsubmit:hover {
    background-color: #4787ED;
}
```

Android.css

```
html, body {
    margin: 0;
    padding: 0;
}
#signup {
    font-family: "Arial";
    font-weight: bold;
    font-size: 45px;
    vertical-align: middle;
    color: #fff;
}
label {
    font-family: "Arial";
    font-weight: bold;
    font-size: 40px;
    padding: 0px;
    display: block;
    text-align: left;
}
#credentials {
    float: none;
    padding: 0px;
    margin: 5px auto 5px;
    text-align: center;
    font-size: 25px;
    clear: both;
    display: block;
    width: 300px;
    height: 45px;
}
.form_login {
    width: 90%;
    padding: 10px;
    margin: 0px auto 0px;
}
.submit {
    background-color: #4D90FE;
    color: #fff;
    font-family: "Arial";
    font-weight: bold;
    text-shadow: 0 1px rgba(0, 0, 0, 0.1);
    text-align: center;
    font-size: 45px;
    float: none;
    margin:10px auto 15px auto;
    border: 1px solid #3079ED;
    border-radius: 6px;
    box-shadow:0px 3px 10px #B9B9B9;
    -webkit-box-shadow:0px 3px 10px #B9B9B9;
    -moz-box-shadow: 0px   3px 10px #B9B9B9;
}
.submit:hover {background-color: #4787ED;}
```

```
.error {
    background-color: #DD4B39;
    color: #fff;
    height: 40px;
    font-size: 30px;
    vertical-align: middle;
    margin-bottom: 10px;
}
.panel {
    margin: 5px auto 5px;
    overflow: hidden;
}
.menu {
    margin: 0px 10px 5px 20px;
    background-color: #DD4B39;
    width: 200px;
    float: right;
    border: 1px solid transparent;
    vertical-align: middle;
    border-radius: 6px;
    box-shadow: 0px 3px 10px #B9B9B9;
    -webkit-box-shadow: 0px 3px 10px #B9B9B9;
    -moz-box-shadow: 0px   3px 10px #B9B9B9;
}
.menu:hover {
    background-color: #D14836;
}
#process {
    vertical-align: middle;
}
#hellouser {
    vertical-align: middle;
    padding: 0px;
    float: left;
    font-family: "arial";
    font-weight: bold;
    font-size: 40px;
    color: #4D90FE;
    width: 220px;
    margin-left:-40px;
}
#menu_opt {
    margin-top: 55px;
    text-align: left;
    width: 200px;
    height: 100px;
    vertical-align: middle;
    min-height: 40px; /*min height of DIV should be set to at least
2x the width of the arrow*/
    border-left: solid thin #E5E5E5;
    border-right: solid thin #E5E5E5;
    border-bottom: solid thin #E5E5E5;
    margin-left: 250px;
    background-color: #F8F8F8 ;
    padding: 5px;
    z-index: 0;
    position: absolute;
    -moz-border-radius: 5px; /*add some nice CSS3 round corners*/
    -webkit-border-radius: 5px;
    border-radius: 5px;
    margin-bottom: 2em;
```

```
    -webkit-box-shadow: 1px 2px 5px #E6E6E6;
    -moz-box-shadow: 1px 2px 5px #E6E6E6;
    box-shadow: 1px 2px 5px #E6E6E6;
}
#menu_opt:after { /*arrow added to uparrowdiv DIV*/
    content: '';
    display: block;
    position: absolute;
    top: -20px; /*should be set to -border-width x 2 */
    left: 60px;
    width: 0;
    height: 0;
    border-color: transparent transparent #F8F8F8 transparent; /
*border color should be same as div div background color*/
    border-style: solid;
    border-width: 10px;
}
.settings {
    margin-top: 15px;
    padding: 0px;
}
.menu_log {
    font-family: "arial";
    font-size: 25px;
    font-weight: bold;
    vertical-align: middle;
    color: #2D2D2D;
}
.menu_log:hover {color: #DD4B39;}
.menu_check {
    font-family: "arial";
    font-size: 25px;
    font-weight: bold;
    vertical-align: middle;
    color: #2D2D2D;
}
.menu_check:hover {color: #DD4B39;}
h4 {
    font-family: "Verdana";
    font-size: 40px;
    font-weight: bold;
    color: #000;
    text-align: center;
    margin: 5px auto 5px;
}
h1,h1 a {
    text-align: center;
    margin: 0px auto 5px;
    border-bottom: dotted 2px #ccc;
    border-top: dotted 2px #ccc;
}
p.no{
color : red;
font-family: "Arial";
font-weight: bold;
font-size: 30px;
margin: 5px auto 5px auto;
}
p.yes{
color : green;
font-family: "Arial";
font-weight: bold;
```

```
font-size: 30px;
margin: 5px auto 5px auto;
}
p.checkar{
    text-align: center;
    border-bottom: 2px solid #66AAFF;
    font-size: 25px;
    width:60%;
    margin: 25px auto 20px auto;
}
p.checktm{
    text-align: center;
    border-bottom: 2px solid #66AAFF;
    font-size: 25px;
    width:60%;
    margin: 25px auto 20px auto;
}
#tmote{height:80px;
margin:15px auto 10px auto;}
#arduino{height:80px;
margin:15px auto 20px auto;}
#LoadPage2{
margin: 10px auto 10px auto;
}
#LoadPage1{
margin: 10px auto 10px auto;
}
#imgProg1{
margin: 10px auto 10px auto;
}
#imgProg2{
margin: 10px auto 10px auto;
}
ul {
    list-style: none;
    margin-top: 35px;
    margin-bottom:0px;
    margin-left: auto;
    margin-right: auto;
    padding: 0px;
}
h2 {
    font-family: "Verdana";
    text-align: center;
    margin-top: 5px;
    margin-left: auto;
    margin-right: auto;
    margin-bottom: 5px;
    color: #DD4B39;
    background-color: #2D2D2D;
    font-size: 40px;
    vertical-align:middle;
    width:100%;
}
#realtemp{
width:94%;
text-align: left;
margin-left: 5px;
}
#averagetemp{
width:94%;
text-align: left;
```

```
margin-left: 5px;
}
#reallight{
width:94%;
text-align: left;
margin-left: 5px;
}
#condition{
width:100%;
margin: 0px;
}
a{
text-decoration:none;
}
h3 {
    font-family: "Verdana";
    text-align: left;
    color: #2D2D2D;
    width: 100%;
    font-size: 25px;
    vertical-align:middle;
    text-shadow: 0 1px #ccc;
    border-bottom: 1px solid #66AAFF;
}
p.room1 {
    font-family: "Arial";
    font-size: 45px;
    text-align: left;
    font-weight: bold;
    color: #DD4B39;
    width: 170px;
    vertical-align:middle;
    height:50px;
    text-shadow: 0 1px #fff;
    padding: 0px;
    margin:0px;
    }
p.average {
    font-family: "Arial";
    font-size: 45px;
    text-align: left;
    font-weight: bold;
    color: #DD4B39;
    width: 170px;
    vertical-align:middle;
    height:50px;
    text-shadow: 0 1px #fff;
    padding: 0px;
    margin:0px;
    }
#universe {
    background-color: #FFF;
    width: 480px;
    height: 730px;
    margin-left: auto;
    margin-right: auto;
    z-index:1;
    text-align: center;
    padding-top:10px;
}
div.room {
    background-color: #fff;
```

```
    width: 100%;
    vertical-align: middle;
    margin-left: auto;
    margin-right: auto;
    margin-top:0px;
    margin-bottom:0px;
}
#button{
height:50px;
}
p.room2 {
    font-family: "Arial";
    font-size: 45px;
    text-align: left;
    font-weight: bold;
    color: #DD4B39;
    width: 170px;
    vertical-align:middle;
    height:50px;
    text-shadow: 0 1px #fff;
    padding: 0px;
    margin:0px;
    }
p.iptext {
    font-family: "Veranda";
    text-align: center;
    font-size: 25px;
    width: 90%;
    margin:80px auto 20px auto;
    color: #fff;
    border: 1px solid #D14836;
    background-color: #DD4B39;
    border-radius: 7px;
    text-shadow: 0 1px rgba(0, 0, 0, 0.1);

}
.ipfield {
    width: 90%;
    height: 40px;
    text-align: center;
    font-size: 25px;
    float: none;
    margin:0px auto 10px auto;
    border-radius: 7px;
}
.usrfield {
    width: 90%;
    height: 40px;
    text-align: center;
    font-size: 25px;
    float: none;
    margin:0px auto 10px auto;
    border-radius: 7px;
}
.ipsubmit {
    background-color: #4D90FE;
    color: #fff;
    font-weight: bold;
    text-shadow: 0 1px rgba(0, 0, 0, 0.1);
    width: 70%;
    text-align: center;
    font-size: 25px;
```

```
    float: none;
    margin:10px auto 0px auto;
    border: 1px solid #3079ED;
    border-radius: 2px;
    box-shadow:0px 3px 10px #B9B9B9;
    -webkit-box-shadow:0px 3px 10px #B9B9B9;
    -moz-box-shadow: 0px   3px 10px #B9B9B9;
}
.ipsubmit:hover {
    background-color: #4787ED;
}
```

Android Room:i/o

AndroidMaxifest.xml

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

    <uses-sdk android:minSdkVersion="8" />
    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission
android:name="android.permission.ACCESS_NETWORK_STATE"/>
    <uses-permission
android:name="android.permission.CHANGE_NETWORK_STATE"/>
    <uses-permission
android:name="android.permission.WRITE_SETTINGS"/>

    <application android:icon="@drawable/home"
android:label="@string/app_name" >
        <activity
            android:name=".RoomioActivity"
            android:label="@string/app_name"
            android:screenOrientation="portrait"
            android:theme="@android:style/Theme.NoTitleBar">

            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category
android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

Main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<WebView xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/webview"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
/>
```

Mymenu.xml

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android" >
    <item android:id="@+id/about"
        android:title="About"
        android:icon="@android:drawable/ic_menu_info_details"
    />
    <item android:id="@+id/exit"
        android:title="Exit"
        android:icon="@android:drawable/ic_menu_close_clear_cancel"
        />
    <item android:id="@+id/reload"
        android:title="Reload"
        android:icon="@android:drawable/ic_menu_more"
    />
</menu>
```

Toast_layout.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/
android"
android:id="@+id/toast_root"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:orientation="horizontal"
android:padding="10dp"
android:background="#DAAA"
>
    <ImageView android:id="@+id/image"
        android:layout_width="wrap_content"
        android:layout_height="fill_parent"
        android:src="@drawable/android"
        android:layout_marginRight="10dp"
        android:contentDescription="@string/image"
    />
    <TextView android:id="@+id/text"
        android:layout_width="wrap_content"
        android:layout_height="fill_parent"
        android:textSize="7pt"
        android:textStyle="bold"
        android:textColor="#000"
        android:layout_gravity="center_vertical"
        android:gravity="center"
        android:layout_marginRight="10dp"
    />
</LinearLayout>
```

Strings.xml

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

    <string name="hello">Hello World, RoomioActivity!</string>
    <string name="app_name">Roomio</string>
```

```
    <string name="image">Hello</string>

</resources>
```

Roomioactivity.java

```
package com.roomio;

import android.app.Activity;
import android.app.AlertDialog;
import android.app.ProgressDialog;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.net.ConnectivityManager;
import android.net.NetworkInfo;
import android.os.Bundle;
import android.view.Gravity;
import android.view.LayoutInflater;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.view.ViewGroup;
import android.view.animation.AccelerateInterpolator;
import android.view.animation.Animation;
import android.view.animation.TranslateAnimation;
import android.webkit.WebView;
import android.webkit.WebViewClient;
import android.widget.TextView;
import android.widget.Toast;

public class RoomioActivity extends Activity {

    WebView mWebView;
    ProgressDialog mProgressDialog;

        private Animation inFromLeftAnimation() {
        Animation inFromLeft = new TranslateAnimation(
        Animation.RELATIVE_TO_PARENT,  -1.0f,
Animation.RELATIVE_TO_PARENT,  0.0f,
        Animation.RELATIVE_TO_PARENT,  0.0f,
Animation.RELATIVE_TO_PARENT,   0.0f
        );
        inFromLeft.setDuration(300);
        inFromLeft.setInterpolator(new AccelerateInterpolator());
        return inFromLeft;
        }

        private class HelloWebClient extends WebViewClient {

            @Override
            public boolean shouldOverrideUrlLoading(WebView view,
String url) {
                view.setVisibility(View.GONE);
                mProgressDialog.setTitle("Processing");
                mProgressDialog.show();
                mProgressDialog.setMessage("Loading");
                return false;
            }
```

```
            @Override
            public void onPageFinished(WebView view, String url) {
                mProgressDialog.dismiss();
                view.startAnimation(inFromLeftAnimation());
                view.setVisibility(View.VISIBLE);
                super.onPageFinished(view, url);
            }
        }


    @Override
    public void onBackPressed(){
        return;
    }


    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        mProgressDialog = new ProgressDialog(this);
        mWebView = (WebView)findViewById(R.id.webview);
        mWebView.setVerticalScrollBarEnabled(false);
        mWebView.setHorizontalScrollBarEnabled(false);
        mWebView.getSettings().setJavaScriptEnabled(true);
        mWebView.loadUrl("http://alexator.zapto.org:8888/roomio/");
        mWebView.addJavascriptInterface(new
JavaScriptInterface(this), "Android");
        mWebView.setWebViewClient(new HelloWebClient());
        ConnectivityManager cm = (ConnectivityManager)
getSystemService(Context.CONNECTIVITY_SERVICE);
        NetworkInfo info = cm.getActiveNetworkInfo();

        if (info !=null){
            if (!info.isConnected()){
                openInternetDialog();
            }
        }
        else {
            openInternetDialog();
        }
    }

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

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch(item.getItemId()){
        case R.id.about:
            openOptionsDialog();
            break;
        case R.id.reload:
            mWebView.reload();
            break;
        case R.id.exit:
            exitOptionsDialog();
            break;
```

```
        }
        return true;
    }

    private void openOptionsDialog(){
        new AlertDialog.Builder(this)
        .setTitle("About Application")
        .setMessage("Hello my name is Alex Georgantas and this app is
for my diploma thesis.")
        .setCancelable(false)
        .setPositiveButton ("Ok", new
DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialoginterface, int
i){
            }
        })
        .show();
    }

    private void exitOptionsDialog(){
        new AlertDialog.Builder(this)
        .setTitle("Exit Application")
        .setMessage("Are you sure you want to Exit?")
        .setCancelable(false)
        .setNegativeButton ("No", new
DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialoginterface, int
i){
            }
        })
        .setPositiveButton("Yes", new
DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialoginterface , int
i){

android.webkit.CookieManager.getInstance().removeAllCookie();
                mWebView.clearHistory();
                mWebView.clearFormData();
                mWebView.clearCache(true);
                finish();
            }
        })
        .show();
    }

    private void openInternetDialog(){
        new AlertDialog.Builder(this)
        .setTitle("Oops no Internet!")
        .setMessage("Sorry this is a webapp and you need to open your
Wi-Fi or mobile data")
        .setCancelable(false)
        .setPositiveButton("Enable", new
DialogInterface.OnClickListener(){
            public void onClick(DialogInterface dialoginterface, int
i){
                finish();
                Intent intent = new Intent(Intent.ACTION_MAIN);

intent.setClassName("com.android.settings","com.android.settings.Wire
lessSettings");
                startActivity(intent);
            }
        })
```

```
            .show();
    }
    public class JavaScriptInterface{
        Context mContext;
        JavaScriptInterface(Context c){
            mContext = c;
        }

        LayoutInflater inflater = getLayoutInflater();
        View layout = inflater.inflate(R.layout.toast_layout,
(ViewGroup)findViewById(R.id.toast_root));
        TextView text = (TextView) layout.findViewById(R.id.text);

        public void showToast(String toast){
            Toast t = new Toast(mContext);
            t.setGravity(Gravity.CENTER_VERTICAL, 0, 0);
            t.setView(layout);
            text.setText(toast);
            t.show();
        }
    }
}
```