ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Έξυπνο Σπίτι στο Σύννεφο

Αγκοπιάν Εμμανουήλ
Μελέτης Χρήστος

Εισηγητής: Δρ Ιωάννης Έλληνας, Καθηγητής

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Έξυπνο Σπίτι στο Σύννεφο

Αγκοπιάν Εμμανουήλ

ΑΜ:40344

Μελέτης Χρήστος

ΑΜ: 42023

Εισηγητής:

Δρ Ιωάννης Έλληνας, Καθηγητής

Εξεταστική Επιτροπή:

Μαστοροκώστας Πάρις, Καθηγητής

Αμοργίνος Ιωάννης, Λέκτορας Εφαρμογών

Ημερομηνία Εξέτασης: N/A

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΠΤΥΧΙΑΚΗΣ ΕΡΓΑΣΙΑΣ

Οι κάτωθεν υπογεγραμμένοι Εμμανουήλ Αγκοπιάν του Σαρκίς, με αριθμό μητρώου 40344 και Χρήστος Μελέτης του Ανδρέα, με αριθμό μητρώου 42023, φοιτητές του Τμήματος Μηχανικών Η/Υ Συστημάτων Τ.Ε. του Α.Ε.Ι. Πειραιά Τ.Τ. πριν αναλάβουμε την εκπόνηση της Πτυχιακής Εργασίας μας, δηλώνουμε ότι ενημερωθήκαμε για τα παρακάτω:

«Η Πτυχιακή Εργασία (Π.Ε.) αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο του συγγραφέα, όσο και του Ιδρύματος και θα πρέπει να έχει μοναδικό χαρακτήρα και πρωτότυπο περιεχόμενο.

Απαγορεύεται αυστηρά οποιοδήποτε κομμάτι κειμένου της να εμφανίζεται αυτούσιο ή μεταφρασμένο από κάποια άλλη δημοσιευμένη πηγή. Κάθε τέτοια πράξη αποτελεί προϊόν λογοκλοπής και εγείρει θέμα Ηθικής Τάξης για τα πνευματικά δικαιώματα του άλλου συγγραφέα. Αποκλειστικός υπεύθυνος είναι ο συγγραφέας της Π.Ε., ο οποίος φέρει και την ευθύνη των συνεπειών, ποινικών και άλλων, αυτής της πράξης.

Πέραν των όποιων ποινικών ευθυνών του συγγραφέα σε περίπτωση που το Ίδρυμα του έχει απονείμει Πτυχίο, αυτό ανακαλείται με απόφαση της Συνέλευσης του Τμήματος. Η Συνέλευση του Τμήματος με νέα απόφασης της, μετά από αίτηση του ενδιαφερόμενου, του αναθέτει εκ νέου την εκπόνηση της Π.Ε. με άλλο θέμα και διαφορετικό επιβλέποντα καθηγητή. Η εκπόνηση της εν λόγω Π.Ε. πρέπει να ολοκληρωθεί εντός τουλάχιστον ενός ημερολογιακού 6μήνου από την ημερομηνία ανάθεσης της. Κατά τα λοιπά εφαρμόζονται τα προβλεπόμενα στο άρθρο 18, παρ. 5 του ισχύοντος Εσωτερικού Κανονισμού.»

Αγκοπιάν Εμμανουήλ

Μελέτης Χρήστος

ΕΥΧΑΡΙΣΤΙΕΣ

Η παρούσα πτυχιακή υλοποιήθηκε ύστερα από επανειλημμένες προσπάθειες, συζητήσεις και σχεδιασμούς του συστήματος. Θα θέλαμε να ευχαριστήσουμε τον καθηγητή-εισηγητή της πτυχιακής κύριο Έλληνα Ιωάννη για τις συμβουλές του και την άμεση ανταπόκρισή του σε κάθε πρόβλημα που παρουσιάστηκε. Καθώς και τις οικογένειες μας για τη συνεχή υποστήριξη τους σε όλους τους τομείς, που πίστεψαν στις ικανότητες μας και μας πίεσαν προς την ολοκλήρωση των ιδεών μας.

ΠΕΡΙΛΗΨΗ

Η παρούσα πτυχιακή εργασία ασχολείται με την ανάπτυξη συστήματος ελέγχου οικιακών συσκευών. Σε αυτό το πεδίο έχουν αναπτυχθεί ιστορικά αρκετά συστήματα, κάθε ένα από τα οποία έχει ορισμένα πλεονεκτήματα και μειονεκτήματα. Το προτεινόμενο σύστημα λαμβάνει υπ' όψη την οικειότητα του χρήστη με τον web browser και αναλαμβάνει να κάνει εφικτό τον έλεγχο των συσκευών μέσω ενός εύκολου στη χρήση web interface.

SUMMARY

The theme of this thesis is about developing a home automation control system. There have existed many iterations of such systems in this field, each one different from the others. This particular system is based on the user's ever-growing familiarity with web browsers and aims to easily control a household's devices via a user friendly graphical user interface.

Επιστημονική Περιοχή: Οικιακός Αυτοματισμός & Κατανεμημένος Έλεγχος
Έξυπνο Σπίτι, Μικροελεγκτής, Οικιακή Πρίζα, Τυπωμένο Κύκλωμα Πλακέτας.
Smart Home, Microcontroller, Home Plug, Printed Circuit Board.

Αγκοπιάν Εμμανουήλ
Μελέτης Χρήστος

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΑΤΑΛΟΓΟΣ ΣΧΗΜΑΤΩΝ

ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ

1. Η ΙΣΤΟΡΙΑ ΤΩΝ ΕΞΥΠΝΩΝ ΣΠΙΤΙΩΝ (SMART HOMES)

Τα πρώτα έξυπνα σπίτια ήταν ιδέες, όχι πραγματικά κτίσματα. Για πολλές δεκαετίες η επιστημονική φαντασία είχε εξερευνήσει την ιδέα του οικιακού αυτοματισμού. Πρωτοποριακοί συγγραφείς είχαν φανταστεί σπίτια να είναι διαδραστικά και αυτόνομα και μάλιστα σπίτια τα οποία θα ξεπέρναγαν σε ζωή και την ίδια την ανθρωπότητα. Άλλοι οραματίζονταν ρομποτικά συστήματα που θα βοηθούσαν στις δουλειές και τις ασχολίες του κάθε ανθρώπου. Είναι αλήθεια πως ενώ η ιδέα του οικιακού αυτοματισμού υπάρχει καιρό τώρα, τα πραγματικά “έξυπνα σπίτια” έχουν δημιουργηθεί πολύ πρόσφατα.

1901 – 1920 - Οικιακές συσκευές.

Αν και οι οικιακές συσκευές είναι μακριά απ' το να θεωρηθούν “έξυπνες”, ήταν ένα σημαντικό επίτευγμα στις αρχές του εικοστού αιώνα. Αυτά τα επιτεύγματα ξεκινήσανε με τη πρώτη μηχανική σκούπα η οποία εξελίχθηκε σε ηλεκτρική. Ύστερα ακολουθήσανε τα ψυγεία τις επόμενες δύο δεκαετίες καθώς και πλυντήρια, σίδερα, φρυγανιέρες και ακόμη περισσότερα.

1932 – 1939 - Άλφα και Ελέκτρο.

Η εκδήλωση Chicago World's Fair παρουσίασε την ιδέα των έξυπνων σπιτιών όχι τόσο σαν σπίτια αλλά σαν ρομπότ που θα εκτελούσαν τις οδηγίες των ιδιοκτητών τους. Τέτοια ρομπότ ήταν για παράδειγμα ο Alpha ο οποίος μπορούσε να αντιδράσει σε λεκτικές εντολές και ο Ελέκτρο (The smoking robot) ικανός να κάνει διάφορες λειτουργίες που τον καθιστούσαν κατάλληλο για κάθε σπίτι, δημιουργήθηκε από την “Westinghouse Electric” και παρουσιάστηκε το 1939 στο New York World's Fair.

1950 - Έπαυλη με κουμπιά του Emil Mathias.

Ο Emil Mathias μηχανεύτηκε κουρτίνες που έκλειναν μόνες τους, ένα καβουρδιστή καφέ αιολικής ενέργειας και ένα φωτεινό καθρέπτη για τη γυναίκα του. Ήταν ο κλασικός εφευρέτης της εποχής. Το πρώτο πραγματικά έξυπνο σπίτι το οποίο κατασκεύασε και χρησιμοποίησε η μηχανική ιδιοφυΐα Emil Mathias από το Jackson, Michigan των Ηνωμένων Πολιτειών.

1966 – 1967 - ECHO IV και Υπολογιστής κουζίνας.

Αν και δεν τα κατάφερε πότε να βγει στη παραγωγή το ECHO IV ήταν η πρώτη έξυπνη συσκευή. Αυτή η ευφυέστατη συσκευή είχε την ικανότητα να βγάζει λίστες για ψώνια να ελέγχει τη θερμοκρασία του σπιτιού και να απενεργοποιεί/ενεργοποιεί άλλες συσκευές του σπιτιού. Ένα χρόνο μετά ήρθε ο υπολογιστής κουζίνας ο οποίος μπορούσε να αποθηκεύσει συνταγές, ωστόσο δεν κατάφερε να πουλήσει λόγο κακής φήμης.

1998 – Αρχές 2000 - Έξυπνα σπίτια.

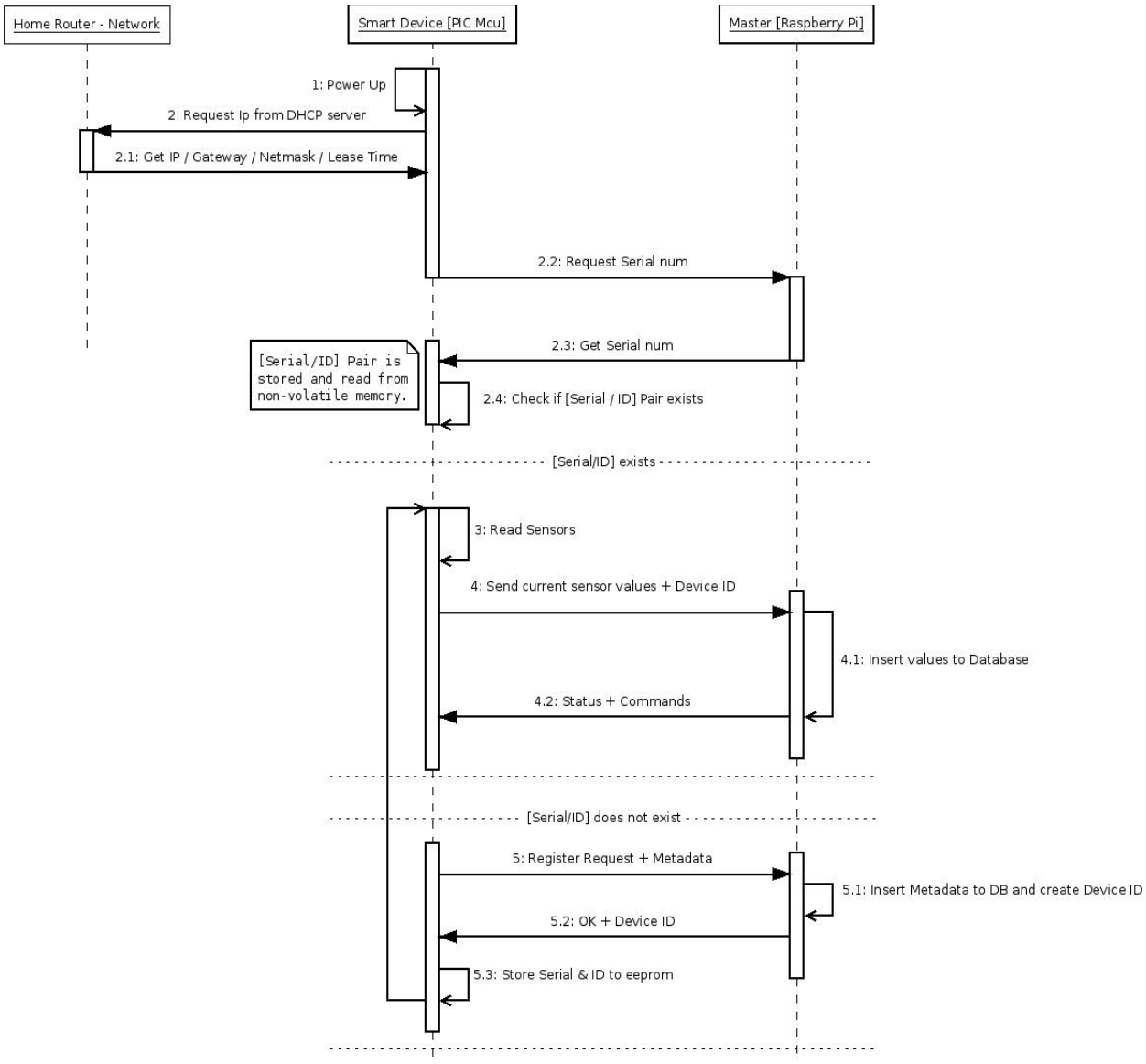
Προς τα τέλη του εικοστού αιώνα το έξυπνο σπίτι και ο οικιακός αυτοματισμός είχαν αρχίσει να γίνονται ραγδαία δημοφιλή. Λόγο της προόδου της τεχνολογίας και της επιστήμης, άρχισαν να εμφανίζονται διάφορες έξυπνες συσκευές προσιτές οικονομικά στους καταναλωτές, καθώς δημιουργήθηκαν τα πρώτα οικιακά δίκτυα, οικιακές τεχνολογίες και άλλα επινοήματα.

Τα έξυπνα σπίτια σήμερα.

Σήμερα το έξυπνο σπίτι βασίζεται περισσότερο στην ασφάλεια και την φιλικότητα προς το περιβάλλον. Είναι βιώσιμα και βοηθούν στο να ξοδεύεται όσο το δυνατόν λιγότερη ενέργεια. Μας ενημερώνουν για τυχών παραβάτες, είτε βρισκόμαστε εντός, είτε εκτός σπιτιού. Οι σημερινές τάσεις του οικιακού αυτοματισμού περιλαμβάνουν ασύρματο έλεγχο μέσω κινητού τηλεφώνου, αυτόματα φώτα, θερμοστάτες, προγραμματισμό ημέρας, ενημερώσεις μέσω διαδικτύου, και ασύρματη επιτήρηση μέσω μαγνητοσκόπησης. Η συνδεσιμότητα και διαδραστικότητα, οδηγούν το τρόπο ζωής και διαχείρισης των μοντέρνων οικογενειών. Ασχέτως με το πρόγραμμα του κάθε ανθρώπου, είτε είναι ταξίδια, σχολείο, δουλειά ή κοινωνικές δραστηριότητες, το σπίτι είναι πάντα διαθέσιμο. Ακόμη και τις ώρες που περνούν οι ιδιοκτήτες μέσα σε αυτό παρέχεται έλεγχος, ασφάλεια και άνεση ζωής. Πλέον μπορούμε να δούμε ολοκληρωμένα συστήματα οικιακού αυτοματισμού με τη χρήση συστημάτων όπως “Alexa” και “Amazon echo” και συμβατές με αυτά έξυπνες οικιακές συσκευές.

2. ΑΝΑΛΥΣΗ ΣΥΣΤΗΜΑΤΟΣ

2.1 Διάγραμμα ροής και αναλυτική επεξήγηση.



Σχήμα 2.1: Διάγραμμα ροής έξυπνου σπιτιού.

Παρακάτω αναλύεται πιο λεπτομερώς η λειτουργία του συστήματος βάση του δεδομένου διαγράμματος ροής.

1. Power UP

Κατά τη πρώτη (1) φάση, η έξυπνη (Μικροελεγκτής PIC) συσκευή ενεργοποιείται από τον χρήστη. Γίνονται οι πρώτες αρχικοποιήσεις της συσκευής οι οποίες μπορεί να περιλαμβάνουνε ενεργοποίηση υποσυστημάτων της συσκευής, αρχικοποιήσεις δεδομένων συσκευής και άλλες σχετικές λειτουργίες.

2. Σύνδεση στο τοπικό δίκτυο.

Κατά τη δεύτερη (2) φάση, η συσκευή λαμβάνει τη δική της διεύθυνση τοπικού δικτύου “IP” από το δρομολογητή του σπιτιού, καθώς και περαιτέρω πληροφορίες σχετικά με το δίκτυο (Lease Time, Netmask IP, Gateway IP). Έπειτα επιτυγχάνεται η σύνδεση με τον ελεγκτή σπιτιού (συσκευή τύπου αφέντη “Master”) (Raspberry Pi), ζητώντας το σειριακό του κωδικό. Ύστερα ελέγχει την εγκυρότητα του σειριακού κωδικού (Serial Number) και του προσωπικού του αριθμού αναγνώρισης (Device ID ή ID), (Το οποίο παρέχεται από τον ελεγκτή σπιτιού σε επόμενη φάση). Τέλος, εάν υπάρχει το ζευγάρι “Serial Num” & “ID” τότε η συσκευή προχωράει στη φάση ορθής λειτουργίας της στο δίκτυο (Φάση 3). Σε αντίθετη περίπτωση περνάει στη κατάσταση εγγραφής στο δίκτυο (Φάση 5).

3. Ανάγνωση αισθητήρων, αποστολή δεδομένων.

Σε αυτή τη φάση λειτουργίας η συσκευή κάνει ανάγνωση των απαραίτητων αισθητήρων. Επεξεργάζεται τα δεδομένα και τις μετρήσεις και τα μορφοποιεί καταλλήλως ώστε να αποσταλούν στο δίκτυο.

4. Αποστολή Δεδομένων – Λήψη εντολών.

Αμέσως μετά γίνεται η αποστολή των δεδομένων που έχει συλλέξει στον ελεγκτή σπιτιού, συνοδευόμενα από το προσωπικό της αριθμό αναγνώρισης. Τα δεδομένα αυτά αποθηκεύονται στη βάση δεδομένων από τον ελεγκτή σπιτιού “Master”. Σαν απάντηση στις μετρήσεις, ο ελεγκτής παρέχει πίσω στη συσκευή ένα χαρακτηριστικό κατάστασης “status”, ακολουθούμενο από της εντολές, που έχουν πιθανώς προκύψει, για την εκάστοτε συσκευή.

5. Εγγραφή συσκευής στο δίκτυο.

Κάθε καινούρια συσκευή που εισέρχεται στο δίκτυο, πρέπει να αποστείλει τα αρχικά της δεδομένα στον ελεγκτή σπιτιού. Μόνο τότε εμφανίζεται στη γραφική διεπαφή χρήστη και μπορεί να χρησιμοποιηθεί από τον ίδιο ή ακόμη και άλλες συσκευές ή τον ελεγκτή “Master” του σπιτιού. Κατά την εγγραφή η συσκευή, αποστέλλει ένα πίνακα δεδομένων με όλες τις εντολές που μπορεί να δεχτεί, καθώς και τύπους δεδομένων μετρήσεων που θα παρέχει στο δίκτυο. Κατά την επιτυχία αυτού του βήματος, η συσκευή λαμβάνει σαν απάντηση μια χαρακτηριστική λέξη “OK”, συνοδευόμενη με το προσωπικό αριθμό αναγνώρισης της συσκευής “Device ID”. Με τη λήψη ο αριθμός αυτός αποθηκεύεται στη μνήμη της συσκευής. Τέλος έχοντας ολοκληρώσει την διαδικασία εγγραφής στο δίκτυο του σπιτιού, η συσκευή περνάει στην κατάσταση ορθής λειτουργίας (Φάση 3).

2.2 Το Raspberry Pi σαν “Master”.

Στο προηγούμενο διάγραμμα αναφέρεται η οντότητα αφέντης (Master). Στη συγκεκριμένη υλοποίηση σαν Master, έχει χρησιμοποιηθεί ο μικροϋπολογιστής Raspberry Pi. Το Raspberry Pi ήταν χωρίς δεύτερη σκέψη μια από τις βέλτιστες λύσεις για τη δουλειά του Master, διότι προσφέρει πολλές δυνατότητες με πολύ χαμηλό κόστος.

Οι διεργασίες που αναλαμβάνει ο Master είναι στην ουσία ο γενικός έλεγχος των γύρω συσκευών. Γνωρίζει την κατάσταση κάθε συσκευής ανά πάσα ώρα και στιγμή και επεμβαίνει σε αυτές, εάν χρειάζεται ή εάν το ζητήσει ο χρήστης. Διατηρεί μια βάση δεδομένων με στατικά και δυναμικά στοιχεία των συσκευών. Παρέχει άμεσο έλεγχο σε κάθε συσκευή μέσω διαδικτύου χωρίς να απαιτείται κάποιο τηλεχειριστήριο ή εξειδηκευμένη συσκευή ελέγχου (Laptop, Tablet, Phone, Computer).

Οι Δυνατότητες που μας προσφέρει αυτή η πλατφόρμα είναι:

-Μικρό μέγεθος: ιδανικό για οικιακή χρήση καθώς μπορεί να τοποθετηθεί οπουδήποτε εντός του σπιτιού χωρίς πολλές συνδέσεις. Μπορεί να μετακινηθεί εύκολα ανά πάσα στιγμή, ακόμη και να ενσωματωθεί εντός του κυρίως πίνακα της οικίας.

-Μικρό κόστος: Από άποψη παραγωγής το κόστος είναι πάντα ένας από τους βασικότερους παράγοντες. Το Raspberry Pi είναι ότι καλύτερο σε θέμα κόστους διότι προσφέρει δυνατότητες ενός μικρού υπολογιστή, στο 1/10 της τιμής ενός συνηθισμένου υπολογιστή.

-Ανοικτό υλικό: Όλα τα σχηματικά του κυκλώματος και της πλακέτας για κάθε έκδοση RPi είναι ελεύθερα προσβάσιμα μέσω του διαδικτύου. Αυτό σημαίνει πως μπορεί η πλατφόρμα να ανασχεδιαστεί, ανάλογα με την εκάστοτε ανάγκη. Για παράδειγμα να αναλάβει το ρόλο ενός αυτοματοποιημένου πίνακα ελέγχου για το σπίτι.

-Πολύπλευρο λογισμικό: Το RPi τρέχει Linux στο πυρήνα του, λογισμικό ανοικτού κώδικα με ευκολία στη χρήση, το οποίο είναι προσαρμόσιμο σε ότι ανάγκες προκύψουν.

Είναι εφικτό να στηθεί ένα πολύπλοκο σύστημα, που να αναλαμβάνει πολλές εργασίες ταυτόχρονα σε μικρό χρονικό διάστημα, χωρίς να τρέχουν στο παρασκήνιο διεργασίες που δεν χρησιμεύουν στην υλοποίηση.

Πρακτικά το Rarspberry Pi λειτουργεί σαν ένας μικρός τοπικός server. Στο δυναμικό του έχει μια βάση δεδομένων για τη διαχείριση των συσκευών, καθώς και μερικά scripts που διεξάγουν κάποιες ενέργειες από και προς τη βάση, όταν κληθούν από την εκάστοτε συσκευή. Πιο συγκεκριμένα υπάρχουν 3(τρία) προγράμματα που αναλαμβάνουν τη λειτουργία του Master:

- get_serial: Κατά την εκτέλεσή του, παρέχει σαν έξοδο τον σειριακό κωδικό του Master. Ο σειριακός κωδικός αποτελείται από 14 αλφαριθμητικά ψηφία και είναι μοναδικός για κάθε σπίτι.

-register: Σε αυτό το πρόγραμμα γίνονται γνωστές όλες οι πληροφορίες της συσκευής που θέλει να εγγραφεί στη βάση (δηλαδή μια νέα συσκευή). Σαν είσοδο δέχεται έναν πίνακα με εντολές και χαρακτηριστικά της εκάστοτε συσκευής. Ο βασικός ελεγκτής του σπιτιού τότε, επιστρέφει σαν απάντηση ένα “OK” πως όλα ελήφθησαν καλώς, ακολουθούμενο από έναν αριθμό ταυτοποίησης της συσκευής.

-report: Μια εγγεγραμμένη στο δίκτυο συσκευή ανά τακτικά χρονικά διαστήματα στέλνει μετρήσεις και νέα στοιχεία στον Master, καλώντας στο report script. Κάθε φορά η συσκευή είναι αυτή που ξεκινάει την επικοινωνία. Με αυτό τον τρόπο επιτυγχάνεται ελαχιστοποίηση της χρήσης του δικτύου, καθώς οι επικοινωνίες γίνονται μόνο όταν χρειάζονται. Με αυτή την τεχνική ο βασικός ελεγκτής, μπορεί να διεξάγει άλλες λειτουργίες χωρίς να χρειάζεται να ψάχνει μια μία τις συσκευές. Αφού η συσκευή στείλει τα δεδομένα της στο Master, τότε λαμβάνει μια απάντηση με το χαρακτηριστικό status* και ύστερα πιθανές εντολές που πρέπει να εκτελέσει.

*status: το χαρακτηριστικό αυτό χρησιμοποιείτε ως έλεγχος λαθών, είναι ένας ακέραιος αριθμός που αλλάζει ανάλογα αν τα στοιχεία της συσκευής αναγνωρίζονται από τον

Master, αν πέτυχε η επικοινωνία, είτε αν κάποιο σφάλμα προέκυψε στη βάση. Ο Master απαντάει με τον αριθμό σφάλματος και έπειτα η εκάστοτε συσκευή πρέπει να λειτουργήσει κατάλληλα προς την επιδιόρθωση του λάθους.

2.3 PIC & AVR ως συσκευές.

Κάθε υλοποίηση διαμοιρασμένου ελέγχου, συνήθως απαιτεί περισσότερες από μια συσκευές για να επιτευχθεί ο έλεγχος άμεσα και έγκυρα. Είτε πρόκειται για ένα υψηλών προδιαγραφών χρηματοκιβώτιο, ένα εργοστάσιο, μια γραμμή παραγωγής, ένα δίκτυο σωληνώσεων, είτε ένα απλό σπίτι. Πολλές φορές απαιτείτε πολλές συσκευές να συνεργαστούν μεταξύ τους ως ένα ολοκληρωμένο σύστημα. Σε αυτές τις περιπτώσεις ήθηστε να χρησιμοποιείτε μια ή περισσότερες συσκευές “Master” (ένας βασικός ελεγκτής) όπως προαναφέραμε, καθώς και μικρότερες απλούστερες συσκευές “Slaves” (σαν υποσυστήματα ελέγχου), εδώ αναφερόμαστε σε αυτά τα υποσυστήματα ως συσκευές “Devices”.

Λόγο ελαχιστοποίησης κόστους, μεγέθους και απλοϊκής φύσης των οικιακών συσκευών, συνηθίζεται η χρήση μικροελεγκτών. Είναι αρκετά εύκολο να βασιστεί μια συσκευή γύρω από ένα απλό ολοκληρωμένο κύκλωμα με μερικά περιφερειακά ηλεκτρονικά στοιχεία για την οδήγηση κινούμενων μελών, θερμοστατών, ή οθονών και άλλα. Στη παρούσα πτυχιακή, έχουν χρησιμοποιηθεί 2 (δύο) διαφορετικά συστήματα μικροελεγκτών από διαφορετικούς κατασκευαστές σαν συσκευές. Πιο συγκεκριμένα, σαν παραδείγματα έχουν χρησιμοποιηθεί ένας μικροελεγκτής PIC της Microchip και ένας AVR της Atmel. Έκαστο σύστημα έχει μεγάλες διαφορές στην αρχιτεκτονική του, κατασκευαστικά αλλά και λειτουργικά (για παράδειγμα η μια συσκευή επικοινωνεί ασύρματα και η άλλη ενσύρματα), ωστόσο παρέχεται η δυνατότητα αυτές οι δυο διαφορετικές συσκευές να χρησιμοποιηθούν από τον Master ή ακόμη και να συνεργαστούν μεταξύ τους σαν να είχαν φτιαχτεί από το ίδιο κατασκευαστή.

Σαν λειτουργιά ως προς τη συνεργασία με το υπόλοιπο σύστημα, κάθε συσκευή είναι απλή. Έκαστη συσκευή διεξάγει κανονικά τη λειτουργία της και ανά τακτά χρονικά διαστήματα εξάγει δεδομένα σχετικά με τη κατάστασή της προς τον κύριο ελεγκτή

Master του σπιτιού. Έπειτα λαμβάνει μια απάντηση επιβεβαίωσης και πιθανές εντολές τις οποίες εκτελεί. Περισσότερο θα αναλύσουμε αυτές τις συσκευές σε επόμενες ενότητες.

(Κεφάλαιο: Etherpic Dev Board.)

(Κεφάλαιο: ESP3266 Temperature/Humidity board)

2.4 Home Plug

Το “Home Plug”, είναι ένας ορισμός που αναφέρεται σε προδιαγραφές, για διάφορες υλοποιήσεις επικοινωνίας μέσω γραμμών τροφοδοσίας. Κάποιες απ’ αυτές προσφέρουν επικοινωνία διαδικτύου, διαδικτυακή τηλεόραση, έλεγχο, βελτιστοποίηση χρήσης ενέργειας και επικοινωνία μεταξύ οικιακών συσκευών. Αναπτύχθηκε από την HomePlug Powerline Alliance ώστε να υπάρχουν σταθερές τεχνολογίες, κατά τις οποίες οι συσκευές θα μπορούσαν να επικοινωνούν μεταξύ τους και μέσω διαδικτύου, χρησιμοποιώντας την ήδη υπάρχουσα υποδομή παροχής ρεύματος στα σπίτια. Το 2010 αναγνωρίστηκε επίσημα ως “Standard”. Έπειτα το 2017, είχαν δημιουργηθεί τουλάχιστον 6 (έξι) πωλητές ολοκληρωμένων κυκλωμάτων για HomePlug, με υποστήριξη IEEE 1901 (Broadcom, Qualcomm Atheros, Sigma Designs, Intellon, SPiDCOM και Mstar).

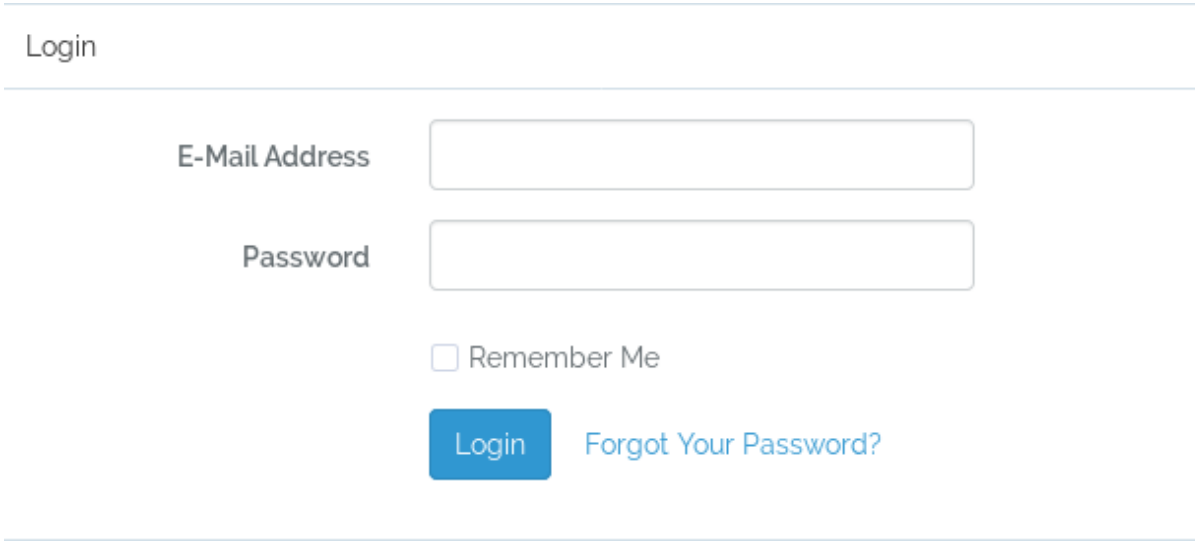
Στα μοντέρνα σπίτια πλέον, η πιο συνήθης χρήση είναι μέσω μεμονωμένων ηλεκτρονικών συσκευών, που κουμπώνουν στη πρίζα του τοίχου και προσφέρουν μια θύρα ehternet σαν έξοδο. Τα δεδομένα αποστέλλονται μεταξύ αυτών των συσκευών, με τη χρήση διαμόρφωσης συχνότητας (Frequency Modulation), με συχνότητες οι οποίες δεν επηρεάζουν την ορθή λειτουργία της γραμμής παροχής ρεύματος. Η ταχύτητα επικοινωνίας από 14Mbps [Mega bits per second, (Μέγα μπιτ ανα δευτερόλεπτο)] έως και 80-90Mbps σε πιο νέα συστήματα, μπορεί να υποστηρίξει περισσότερες χρήσεις συστημάτων επικοινωνιών. Επιπροσθέτως προσφέρει ασφάλεια δεδομένων με τη χρήση του πρότυπου κρυπτογράφησης AES-128.

2.5 PiCloud interface

Το “PiCloud interface” είναι μία web εφαρμογή η οποία αποτελεί τη διεπαφή χρήστη, η οποία προσφέρει την δυνατότητα προβολής των αναφορών και τον έλεγχο των διαφόρων συσκευών στο δίκτυο. Η web εφαρμογή τρέχει πάνω στον Master, ο οποίος είναι υπεύθυνος για την επικοινωνία μεταξύ των συσκευών. Στόχος της web εφαρμογής είναι να παρέχει ένα φιλικό προς το χρήστη τρόπο αλληλεπίδρασης με τις συσκευές, καθώς και την παροχή πληροφοριών σχετικά με τους διάφορους αισθητήρες που είναι διαθέσιμοι στο δίκτυο.

Η web εφαρμογή έχει αναπτυχθεί σε PHP με τη χρήση του Laravel framework. Στο front-end γίνετε χρήση του Bootstrap framework, το οποίο βοηθάει στη δημιουργία ενός απλού, αλλά και επίσης ευέλικτου UI που έχει σαν κύριο στόχο τη φιλικότητα προς το χρήστη. Η βιβλιοθήκη Chart.js χρησιμοποιείτε επίσης για τη δημιουργία δυναμικών γραφημάτων για τη παρουσίαση των δεδομένων που έχουν συλλεχθεί από την εκάστοτε συσκευή.

Κατά την πρώτη είσοδο του χρήστη στη εφαρμογή, του ζητείτε η δημιουργία ενός λογαριασμού χρήστη. Μετά τη δημιουργία του λογαριασμού, ο χρήστης έχει τη δυνατότητα πλέον, να χρησιμοποιήσει τα στοιχεία του για να συνδεθεί στο σύστημα και να αλληλεπιδράσει με την εφαρμογή.



Μετά τη είσοδο του χρήστη στο σύστημα, η εφαρμογή παρουσιάζει έναν πίνακα με όλες τις έξυπνες συσκευές οι οποίες έχουν εντοπισθεί στο δίκτυο, και έχουν καταχωρηθεί από το Master στη βάση. Εδώ ο χρήστης μπορεί να δει το όνομα της κάθε συσκευής, την κατάσταση της κάθε συσκευής, την τελευταία φορά που η συσκευή επικοινώνησε με το σύστημα, πότε η συσκευή εγγράφηκε στο σύστημα για πρώτη φορά, και τέλος πότε έγινε η τελευταία αλλαγή στην εγγραφή της συσκευής στη βάση.

Devices

Name	Status	Last Contact At	Registered At	Updated At
Fridge	On	2018-02-18 06:10:16	2018-02-18 06:07:16	2018-02-18 06:07:16
Heater	Off	2018-02-18 06:08:26	2018-02-18 06:08:26	2018-02-18 06:08:26
Furnance	Off	2018-02-18 06:09:05	2018-02-18 06:09:05	2018-02-18 06:09:05

Κάνοντας κλικ σε μία από τις γραμμές του πίνακα, μεταφερόμαστε στη σελίδα της αντίστοιχης συσκευής, από την οποία είναι δυνατή η προβολή των δεδομένων που έχουν συλλεχθεί από τους αισθητήρες τη συσκευής, καθώς και ο έλεγχος όποιων διαθέσιμων ενεργοποιητών.

Η σελίδα της συσκευής χωρίζετε σε έως τέσσερα διαφορετικά μέρη. Το πρώτο μέρος είναι ένα γράφημα στο οποίο εμφανίζονται οι τιμές των αισθητήρων μέσα στο τελευταίο εικοσιτετράωρο. Έπειτα ακολουθεί ο πίνακας με τις τρέχουσες τιμές των αισθητήρων καθώς και το πότε ενημερώθηκαν.

Attribute	Current Value	Last Update At
Temperature	-20 °C	2018-02-18 06:15:13
Door Status	Closed	2018-02-18 06:15:40
Humidity	13 %	2018-02-18 06:16:04

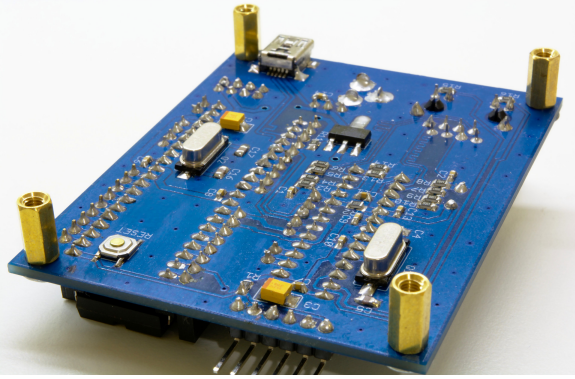
Στη συνέχεια έχουμε τον πίνακα με τις διαθέσιμες προς τη συσκευή εντολές, από τον οποίο ο χρήστης έχει τη δυνατότητα να ελέγξει τη συσκευή σε πραγματικό χρόνο, καθώς και να προγραμματίσει διάφορες εντολές έτσι ώστε να εκτελεστούν αυτόματα κάποια χρονική στιγμή στο μέλλον. Τέλος, είναι ο πίνακας με τις προγραμματισμένες εντολές από το χρήστη στη συσκευή, όπου μπορεί ο χρήστης, εύκολα να δει τις εντολές που έχει προγραμματίσει και πότε αυτές πρόκειται να εκτελεστούν.

Command	Set Value		Set At
Temperature	321 °C	Send	2018-03-26 00:48:45
Door	Open	Send	2018-03-26 00:47:15

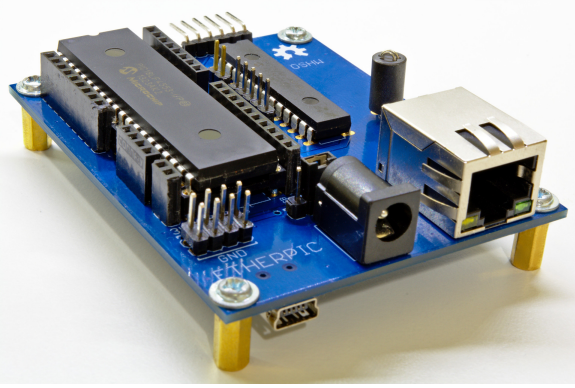
Scheduled Task	Scheduled For	Created At
Set the **temperature** to **150 °C**	2018-03-25 22:53:27	2018-03-25 21:06:21
Set the **temperature** to **120 °C**	2018-03-27 00:21:00	2018-03-25 23:21:32
Set the **temperature** to **300 °C**	2018-03-26 01:22:00	2018-03-25 23:22:28
Set the **temperature** to **111 °C**	2018-04-07 04:40:00	2018-03-26 00:48:17
Open the **door**	2018-03-25 22:38:25	2018-03-25 21:38:15
Close the **door**	2018-03-29 00:21:00	2018-03-25 23:21:46
Open the **door**	2018-04-07 00:26:00	2018-03-25 23:26:33
Close the **door**	2018-03-31 00:47:00	2018-03-26 00:47:24

3. ETHERPIC DEV BOARD

Στο ρόλο της έξυπνης συσκευής και στα πλαίσια της παρούσας πτυχιακής, αναπτύχθηκε ένα σύστημα το οποίο θα μπορούσε εύκολα να υποδυθεί το ρόλο οποιασδήποτε συσκευής. Η πλακέτα αποτελείται από μερικά βασικά στοιχεία, που επιτρέπουν την ενσύρματη επικοινωνία με το τοπικό δίκτυο, την άμεση χρήση απλών ενεργοποιητών (LEDs, μοτέρ τυπου μινιατούρας και ηχεία), καθώς και την ανταλλαγή και επεξεργασία δεδομένων από διάφορους αισθητήρες.



Εικόνα 3.1 Κατω επιφάνεια EtherPicBoard



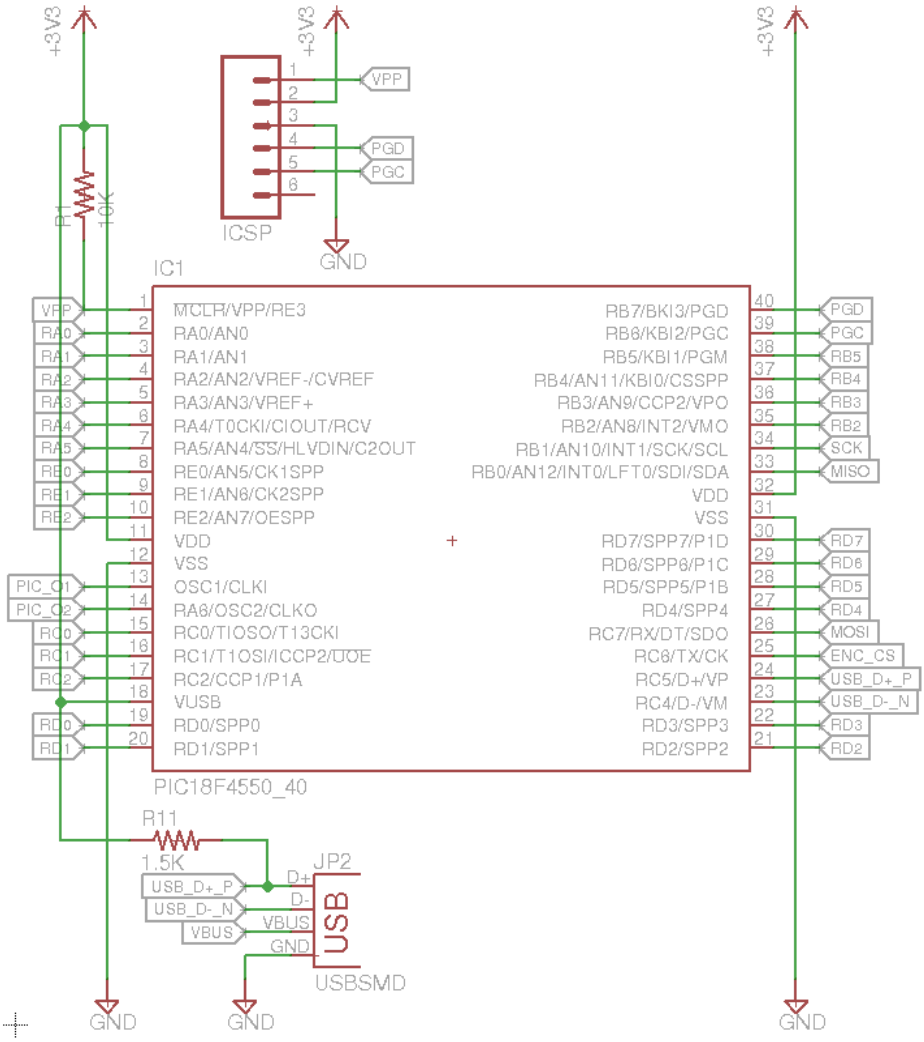
Εικόνα 3.2: Άνω επιφάνεια Etherpic Board

3.1 Μελέτη κυκλώματος

Το κύκλωμα αποτελείται από 2 (δύο) βασικούς μικροελεγκτές, τα παθητικά τους ηλεκτρονικά στοιχεία και κάποια άλλα περιφερειακά στοιχεία.

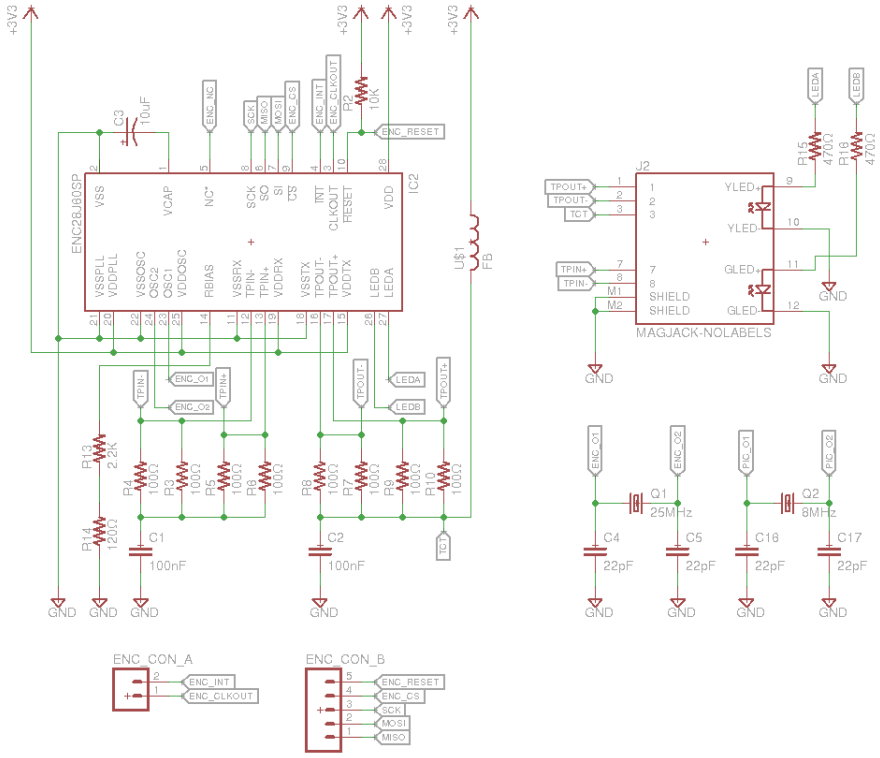
Τον εγκέφαλο του συστήματος αποτελεί ένας PIC μικροελεγκτής (PIC 18f4550) της Microchip. Οι περισσότερες πόρτες εισόδου/εξόδου του, είναι συνδεδεμένες με κονέκτορες για οποιαδήποτε πιθανή χρήση προκύψει. Πάνω στο μικροελεγκτή είναι συνδεδεμένος ένας διακόπτης, στο ρόλο του επαννεκινητή με μια “pull up” αντίσταση που κρατάει τον μικροελεγκτή ενεργοποιημένο μέχρι να πατηθεί ο διακόπτης όπου θα προκύψει επανεκκίνηση του συστήματος. Ένα βύσμα γενικού σειριακού δίαυλου, τύπου “USB mini” για τροφοδότηση αλλά και επικοινωνία προς τον έξω κόσμο. Στο βύσμα αυτό έχει τοποθετηθεί μια “pull up” αντίσταση στον ακροδέκτη “D+” του βύσματος, η οποία χαρακτηρίζει την συσκευή ως “Πλήρους ταχύτητας” “Full Speed”, η οποία επίσης χρησιμοποιείται για να εντοπίζουν εξωτερικές συσκευές την ύπαρξη συνδεσης. Επιπροσθέτως υπάρχει ένα κανάλι επικοινωνίας με τον δεύτερο μικροελεγκτή του κυκλώματος, ο οποίος καθιστά δυνατή την ενσύρματη επικοινωνία με το τοπικό δίκτυο. Τέλος έχει προβλεφθεί το σύστημα να μπορεί να κωδικοποιηθεί, μέσω ακροδεκτών σειριακού προγραμματισμού εντός κυκλώματος “ISCP (In-Circuit Serial Programming)”.

Στο σχήμα 3.1 παρατίθεται το σχηματικό του κυρίως μικροελεγκτή.



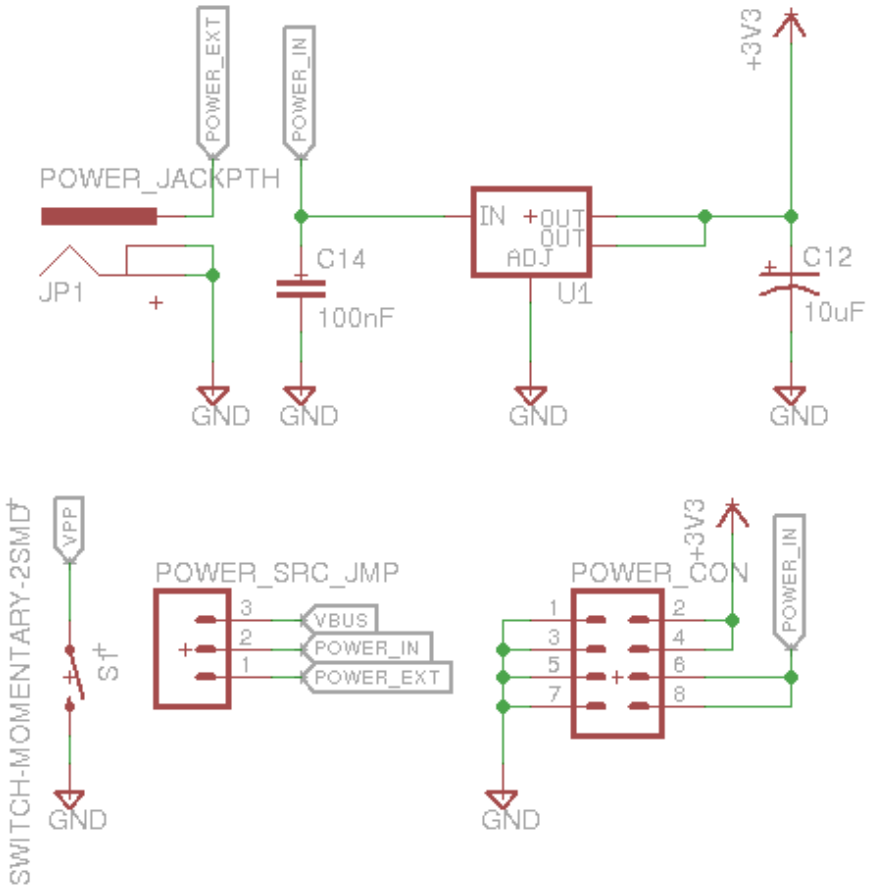
Σχήμα 3.1: Μικροελεγκτής PIC18f4550.

Το υποσύστημα ενσύρματης επικοινωνίας με το τοπικό δίκτυο, αποτελείται από ένα μικροελεγκτή ENC28J60SP της Microchip, ο οποίος σε συνεργασία με κάποια παθητικά ηλεκτρονικά στοιχεία του, είναι υπεύθυνος για την επικοινωνία με τον έξω κόσμο μέσω Ethernet. Ο PIC μπορεί να επικοινωνήσει με το ENC μέσω SPI πρωτοκόλλου, επεξεργάζεται μηνύματα και κάνει αποστολή/λήψη πακέτων προς εξωτερικά δίκτυα. Το βύσμα διεπαφής που χρησιμοποιείται εδώ είναι το MAGJACK με 2 ενδεικτικές διόδους εκπομπής φωτός “LED, Light Emmiting Diode”, για την ενημέρωση ανταλλαγής μηνυμάτων και σύνδεσης με το δίκτυο. Στην εικόνα 2, μπορούμε να παρατηρήσουμε τις συνδέσεις και τα παθητικά εξαρτήματα του μικροελεγκτή ENC28J60. Μπορούμε να προσέξουμε επίσης τις συνδέσεις που γίνονται με τους κρυστάλλους (ταλαντωτές) για κάθε ολοκληρωμένο κύκλωμα.



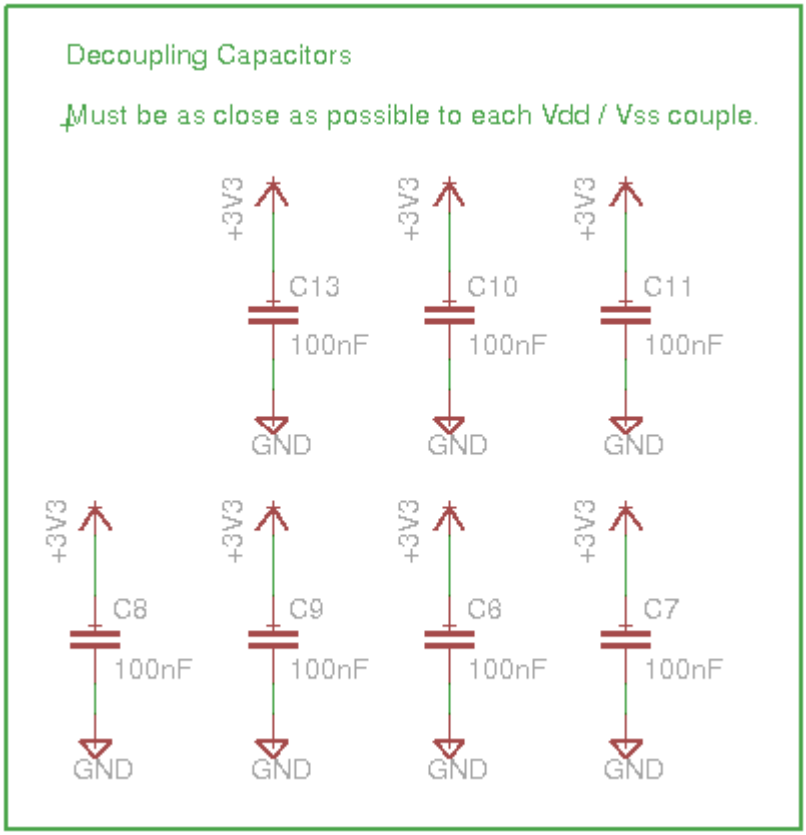
Σχήμα 3.2: Υποσύστημα ENC, Κρύσταλλοι, Εξωτερικές συνδέσεις

Όλο το σύστημα είναι δυνατό να τροφοδοτηθεί από σταθερή τάση 3.3 V απευθείας, ή από εξωτερικές τάσεις μεγαλύτερες των 4.5 V, χάρη στον σταθεροποιητή τάσης και μέσω σύνδεσης USB ή απλού διπολικού βύσματος. Στη πλακέτα έχουν τοποθετηθεί θηλυκά βύσματα με πρόσβαση στην τάση εισόδου, την τάση του σταθεροποιητή 3.3 V, και την “γείωση” του κυκλώματος. Για την επιλογή τροφοδοσίας μέσω USB ή διπολικού βύσματος, έχει τοποθετηθεί ένας βραχυκυκλωτήρας που συνδέει την είσοδο του σταθεροποιητή με τις τροφοδοσίες του βύσματος σειριακού δίαυλου ή του διπολικού βύσματος.



Σχήμα 3.3 Κύκλωμα τροφοδοσίας. Κουμπί επανεκκίνησης. Σταθεροποιητής τάσης.

Τέλος, έχουν τοποθετηθεί σε κάθε ζευγάρι τροφοδοσίας, πυκνωτές απόζευξης, ρόλος των οποίων είναι να φιλτράρουν πιθανούς θορύβους και ακίδες, της τροφοδοσίας που μπορούν να προκύψουν από εξωτερικούς παράγοντες. Οι πυκνωτές αυτοί, πρέπει να βρίσκονται όσο το δυνατόν κοντύτερα στους ακροδέκτες τροφοδοσίας κάθε εύθραυστου στα παραπάνω εξαρτήματος, όπως οι μικροελεγκτές για να μεγιστοποιηθεί η χρήση τους.

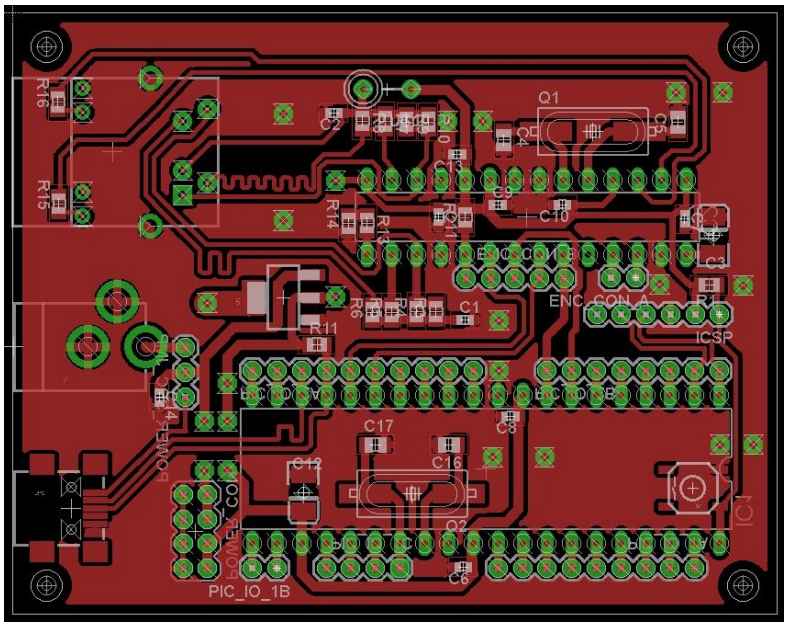


Σχήμα 3.4: Πυκνωτές απόζευξης.

3.2 Σχεδίαση κυκλώματος/πλακέτας.

Συχνά τυχαίνει να είναι απαραίτητη η εκτύπωση ενός κυκλώματος σε χάλκινη πλακέτα. Αυτό μπορεί να συμβεί επειδή υπάρχει ανάγκη παραγωγής μικρής ή μεγάλης κλίμακας του εκάστοτε κυκλώματος, είτε επειδή το σύστημα είναι πολύ περίπλοκο για να κολληθεί εξ ολοκλήρου στο χέρι. Άλλοι λόγοι που επιλέγουμε να τυπώσουμε ένα κύκλωμα, είναι επειδή μπορεί να υπάρχουν εντός αυτού, στοιχεία επιρρεπής στους θορύβους. Επιπλέων συνηθίζεται πολλές συνδέσεις που αποτελούν ζεύγος επικοινωνίας, να έχουν απαραίτητα ισαπέχουσες αποστάσεις από ακροδέκτη σε ακροδέκτη.

Έχοντας υπόψιν ότι πρέπει να σχεδιαστεί ένα κύκλωμα το οποίο εύκολα να αντικαθιστά τα βασικά ηλεκτρονικά οποιασδήποτε οικιακής συσκευής η ανάπτυξη του συστήματός μας προχώρησε στο σχεδιασμό και υλοποίηση ενός τυπωμένου κυκλώματος “PCB (Printed Circuit Board)”. Παρακάτω επισυνάπτεται η τελευταία έκδοση του σχεδίου του τυπωμένου κυκλώματος “EtherPic Dev Board”.



Σχήμα 3.5: Κάτω όψη σχεδίου τυπωμένου κυκλώματος.

Στην παραπάνω εικόνα μπορούμε να διακρίνουμε, πως έχουν προβλεφθεί αρκετά θέματα τα οποία λαμβάνονται ως σωστός σχεδιασμός για μια πλακέτα.

1. Διάδρομοι τροφοδοσίας.

Οι διάδρομοι τροφοδοσίας διαφέρουν απ’ τους υπόλοιπους, όντας πιο μεγάλοι στο πλάτος τους. Αυτό συμβαίνει για να μπορούν οι διάδρομοι τροφοδοσίας, να υποστηρίξουν όλα τα μέρη του συστήματος, χωρίς να υπάρχουν προβλήματα πτώσης τάσεων και παρασιτικής θερμότητας, λόγο υψηλής έντασης του ρεύματος σε συνδυασμό με μικρούς αγωγούς.

2. Ζεύγη επικοινωνίας.

Σε περιπτώσεις όπου έχουμε υψηλές ταχύτητες επικοινωνίας μέσω ζεύγους επικοινωνίας, είναι κρίσιμο οι αποστάσεις των δυο “διαδρομών” να είναι ιδανικά ίδιες. Με αυτόν τον τρόπο η αντίσταση των διαδρόμων του ζεύγους είναι ίδια, σαν αποτέλεσμα τα σήματα, φτάνουν από τον αποστολέα στο δέκτη στον ίδιο χρόνο. Κατά συνέπεια έχουμε μια πιο αξιόπιστη επικοινωνία που μπορεί να φτάσει πιο κοντά στο μέγιστο εύρος ταχυτήτων της.

3. Κρίσιμες αποστάσεις μεταξύ στοιχείων.

Γενικά όσο πιο κοντά στους μικροελεγκτές βρίσκονται τα παθητικά ηλεκτρονικά στοιχεία ή βασικότερα στοιχεία όπως κρύσταλλοι και πυκνωτές απόζευξης, τόσο καλύτερη λειτουργία θα έχει το σύστημα. Είναι καίριας σημασίας, ο κρύσταλλος ταλαντωτής “Clock” του μικροελεγκτή, να είναι όσο το δυνατόν πιο κοντά ώστε να διατηρηθεί σωστός ο χρονισμός. Το ίδιο ισχύει και για τους πυκνωτές απόζευξης “Decoupling Capacitors” οι οποίοι φιλτράρουν παρεμβολές και θορύβους που έρχονται από τη γραμμή τροφοδοσίας.

4. Πεδίο γείωσης “Ground Plain”

Το πεδίο γείωσης στα τυπωμένα κυκλώματα, είναι ένα μεγάλο κομμάτι χαλκού πάνω στη πλακέτα, που συνήθως περιτριγυρίζει το κύκλωμα εξ' ολοκλήρου. Αυτό βοηθάει στον εύκολο και “καθαρό” σχεδιασμό του κυκλώματος καθώς προστατεύει και από ηλεκτρονικούς θορύβους. Αποφορτίζει τη συσσώρευση ενέργειας σε διάφορα μέρη της πλακέτας καθώς αποτελεί μια εύκολη διαφυγή του ρεύματος προς τη “ΓΗ”. Τέλος σε περιπτώσεις που κάτι πάει στραβά και κάποιο εξάρτημα καεί ή γίνει κάποιο βραχυκύκλωμα, είναι αρκετές οι πιθανότητες να σωθούν τα περισσότερα εξαρτήματα για το λόγο που προαναφέρθηκε.

5. Αποσυμφόρηση θερμότητας.

Πολλές φορές κάποια ηλεκτρονικά εξαρτήματα πάσχουν από υψηλές θερμοκρασίες. Σε αυτή τη περίπτωση, μπορούμε να αποφύγουμε ενεργές ή παθητικές ψύκτρες, που δυσχεραίνουν την παράγωγη, ανεβάζοντας το κόστος και την πολυπλοκότητα κατασκευής. Αυτό επιτυγχάνεται με τον σχεδιασμό “copper pads”, τα οποία είναι μεγάλες περιοχές χαλκού πάνω στη πλακέτα στις οποίες μπορεί να δραπετεύσει η θερμότητα.

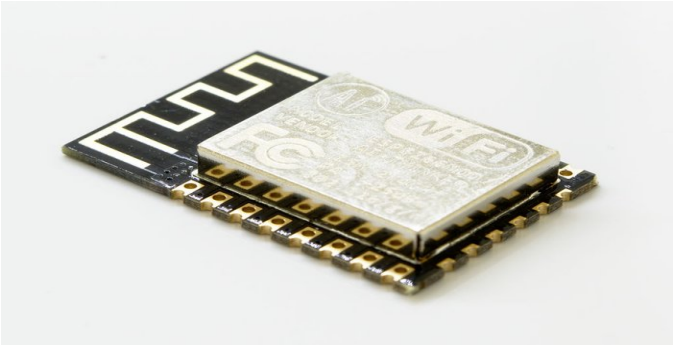
4. ESP 8266 TEMPERATURE/HUMIDITY BOARD.

Με την πρόοδο των ασύρματων επικοινωνιών, έχει αναπτυχθεί ένα πολύ μεγάλο πλήθος έξυπνων συσκευών, που χρησιμοποιεί στην καθημερινότητά του ο άνθρωπος. Τέτοιες συσκευές είναι εύκολο να επικοινωνήσουν και να συνεργαστούν με άλλες παρόμοιες συσκευές ή μεγαλύτερα συστήματα. Στα πλαίσια της παρούσας πτυχιακής, δημιουργήθηκε μια ακόμη συσκευή για να παρουσιάσει τις δυνατότητες ασύρματης επικοινωνίας που παρέχει το σύστημα μας.

Εικόνα 4.1: Μονάδα μέτρησης θερμοκρασίας & υγρασίας.

4.1 Μελέτη κυκλώματος.

Η συσκευή αυτή έχει σχεδιαστεί γύρω από μια πολύ δημοφιλής μονάδα, το ESP8266. Το ESP είναι ένα σύστημα μικροελεγκτή με δυνατότητες WiFi και ενσωματωμένο TCP/IP stack. Δημιουργήθηκε από την εταιρία Espressif Systems και άρχισε να γίνεται γνωστό από τον Αύγουστο του 2014 με τη μονάδα ESP-01. Πλέον το συναντάμε σε πολλές συσκευές μαζικής παραγωγής και πιο σπάνια σε πιο επαγγελματικού τύπου συσκευές.

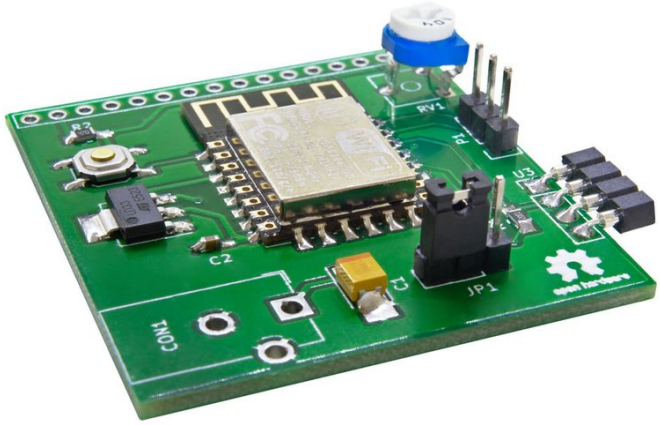


Εικόνα 4.2: ESP8266 module.

Η μόνη δυσκολία που παρουσιάστηκε με την συγκεκριμένη συσκευή, καθώς και με πάσης φύσεως ασύρματη συσκευή, είναι πως τέτοιες συσκευές δεν μπορούν να γνωρίζουν τα χαρακτηριστικά σύνδεσης του ασύρματου δρομολογητή κάθε πελάτη. Εκτός αυτού, δεν είναι εφικτό να ζητηθούν από οποιοδήποτε πελάτη ή εταιρία, αυτά τα χαρακτηριστικά, ώστε να προγραμματιστούν εντός της συσκευής. Αυτό το πρόβλημα ήρθε να λύση η ανάπτυξη ενός μικρού προγράμματος το οποίο θα αναλυθεί αργότερα σε αυτό το κεφάλαιο.

4.2 Σχεδίαση κυκλώματος/πλακέτας.

Δημιουργήθηκε μια πλακέτα βασισμένη στις δυνατότητες του ESP με μερικά απαραίτητα περιφερειακά εξαρτήματα. Σκοπός της συσκευής είναι να αναπαριστά οποιαδήποτε έξυπνη συσκευή μικρού μεγέθους ή ακόμη και φορητής οικιακής συσκευής. Στη συγκεκριμένη υλοποίηση είναι μια συσκευή μέτρησης θερμοκρασίας και υγρασίας του περιβάλλοντος.



Εικόνα 4.3: Μονάδα ESP με πλακέτα περιφερειακών.

Οι διαφορές των ασύρματων συσκευών απ’ τις ενσύρματες που είδαμε πριν είναι μικρές. Πέρα απ’ το γεγονός ότι η συσκευή έχει δυνατότητες ασύρματης επικοινωνίας, η βασική αλλαγή είναι πως η συσκευή απαιτεί κάποια δεδομένα σχετικά με το δίκτυο του χρήστη. Σε συνεργασία με το πρόγραμμα “Συνδετήρα Ασύρματων Συσκευών” (Wireless Device Connector) η συσκευή λαμβάνει τα απαραίτητα χαρακτηριστικά του ασύρματου δρομολογητή του χρήστη. Αφού αυτά τα στοιχεία αποθηκευτούν στη συσκευή, τότε μπορεί να επιτευχθεί η σύζευξή της στο δίκτυο και να εκκινήσει η ορθή λειτουργία της, μέσω του δικτύου του έξυπνου σπιτιού.

4.3 Συνδετήρας Ασύρματης Συσκευής (Wireless Device Connector).

Για να διευκολυνθεί η πρόσβαση μιας ασύρματης συσκευής στο δίκτυο του έξυπνου σπιτιού, δημιουργήθηκε ο “Συνδετήρας Ασύρματης Συσκευής”(Wireless Device Connector). Χάρη στο πρόγραμμα αυτό, δεν απαιτείται ο χρήστης να έχει τεχνικές γνώσεις και δεν χρειάζεται η ανταλλαγή ευαίσθητων δεδομένων του εκτός του προσωπικού του δικτύου. Κατά τη λειτουργία του ο ΣΑΣ, ζητάει από τον χρήστη τα διακριτικά στοιχεία του ασύρματου δρομολογητή, και τα στέλνει στη συσκευή που θέλει να εγγράψει ο χρήστης στο δίκτυο του. Ύστερα ο χρήστης επιλέγει τη συσκευή από μια λίστα επιλογών και εκτελεί την ανταλλαγή των στοιχείων με τη συσκευή που θέλει. Όλη αυτή η διαδικασία γίνεται μέσω ενός απλού γραφικού περιβάλλοντος με μερικά μόνο κλικ.

Εικόνα 4.4: Γραφική διεπαφή χρήστη του Συνδετήρα Ασύρματης Συσκευής

5. ΕΠΙΚΟΙΝΩΝΙΑ ΜΕΤΑΞΥ ΣΥΣΚΕΥΩΝ

5.1 Json String

Κάθε συσκευή στο δίκτυο επικοινωνεί ανά τακτά χρονικά διαστήματα με τον κεντρικό ελεγκτή, μέσω αιτημάτων Πρωτοκόλλου Μεταφοράς Υπερκείμενου (HyperText Transfer Protocol - HTTP). Με την αποστολή ενός αιτήματος, η συσκευή ενημερώνει τον ελεγκτή σχετικά με την κατάσταση της, συμπεριλαμβάνοντας τυχών μετρήσεις αισθητήρων της. Σε αυτό το αίτημα ο ελεγκτής απαντά παραθέτοντας τυχών νέες εντολές προς αυτή.

Τα μηνύματα που ανταλλάσσονται μεταξύ των συσκευών ακολουθούν ένα συγκεκριμένο πρότυπο σε μορφή JSON (JavaScript Object Notation). Το πρότυπο ενός μηνύματος το οποίο αποστέλλετε από το συσκευή στο κεντρικό ελεγκτή έχει την ακόλουθη μορφή:

```
{
    “deviceId”: <int>,
    “attributes”: [
          {
             “id” => <int>,
             “value” => <string>
          }
       ]
}
```

Όπου το “deviceId” είναι το αναγνωριστικό της συσκευής. Ενώ το “attributes” είναι ένας πινάκας ο οποίος περιέχει τις μετρήσεις αισθητήρων της συσκευής, συνοδευόμενες από το αναγνωριστικό τους.

Το πρότυπο ενός μηνύματος, το οποίο αποστέλλετε από το κεντρικό ελεγκτή στην συσκευή ως απάντηση στο προηγούμενο αίτημα έχει την ακόλουθη μορφή:

```
{
   “status”: <int>,
   “commands”: [
         {
            “id” => <int>,
            “value” => <string>
         }
      ]
}
```

Όπου το “status” είναι μία τιμή η οποία μπορεί να έχει τις τιμές 0, 1 ή 2 και έχει ως στόχο την ενημέρωση της συσκευής σχετικά με τη διαχείριση του προηγούμενού της αιτήματος. Εάν όλα πήγαν καλά και τα δεδομένα τα οποία απεστάλησαν εισήλθαν στη βάση με επιτυχία, τότε το “status” θα έχει την τιμή 0. Σε περίπτωση που το “deviceId” το οποίο δόθηκε δεν ήταν έγκυρο η τιμή θα είναι 1, τέλος αν κάποιο από τα “id” των “attributes” είναι άκυρο τότε παίρνει την τιμή 2.

Τέλος, το “commands” είναι ένας πινάκας ο οποίος περιέχει τις εντολές προς τη συσκευή, συνοδευόμενες από το αναγνωριστικό τους.

5.2 Πίνακας τύπων δεδομένων.

Τα δεδομένα τα οποία αποστέλλονται από τη συσκευή στο κεντρικό ελεγκτή, χωρίζονται σε τρεις κατηγορίες.

Τα αναλογικά (analog) τα οποία συνήθως αφορούν μετρήσεις από αισθητήρες, π.χ. η θερμοκρασία του φούρνου. Τα αναλογικά δεδομένα μπορούν να έχουν έναν από τους παρακάτω τύπους:

Τύπος	Μονάδα Μέτρησης	Κωδικός
Θερμοκρασία	C	100
Ποσοστό	%	101
Συγκέντρωση Αερίου	PPM	102
Πίεση	mBar	103
Ισχύς	Watts	104
Φωτεινότητα	Lumens	105
Ένταση ήχου	dB	106

Πίνακας 5.1: Κωδικοί τύπων δεδομένων.

Τις καταστάσεις (states), οι οποίες αφορούν διάφορες καταστάσεις στις οποίες μια συσκευή και τα μέρη τις μπορούν να βρίσκονται, π.χ. εάν η αντίσταση του φούρνου είναι ενεργή. Οι καταστάσεις μπορούν να έχουν έναν από τους παρακάτω τύπους:

Κατάσταση	Κωδικός
closed/open	200
active/inactive	201
very bad/bad/ok/good/excellent	202

Πίνακας 5.2: Κωδικοί καταστάσεων

Τέλος, το απλό κείμενο (plain text), το οποίο δίνει τη δυνατότητα στη συσκευή να αποστείλει μη προκαθορισμένα δεδομένα σε μορφή κειμένου. Ο τύπος απλού κείμενου έχει κωδικό 0.

Οι εντολές οι οποίες αποστέλλονται από το κεντρικό ελεγκτή στη συσκευή, χωρίζονται σε τρεις κατηγορίες. Τις αναλογικές (analog), οι οποίες συνήθως αφορούν τον ορισμό μιας τιμής που έχει να κάνει με ένα μέρος της συσκευής, π.χ. ο ορισμός της θερμοκρασία του φούρνου. Τα αναλογικά δεδομένα μπορούν να έχουν έναν από τους παρακάτω τύπους:

Τύπος	Μονάδα Μέτρησης	Κωδικός
Θερμοκρασία	C	100
Ποσοστό	%	101

Πίνακας 5.3: Τύποι εντολών

Τις καταστάσεις (states), οι οποίες αφορούν τον ορισμό μίας κατάστασης μιας συσκευής ή κάποιου από τους ενεργοποιητές της, π.χ. την ενεργοποίηση του θερμοσίφωνα. Οι καταστάσεις μπορούν να έχουν έναν από τους παρακάτω τύπους:

Κατάσταση	Κωδικός
close/open	200
enable/disable	201

Πίνακας 5.4: Καταστάσεις εντολών

Και τέλος το απλό κείμενο (plain text), το οποίο δίνει τη δυνατότητα στο κεντρικό ελεγκτή, να αποστείλει μη προκαθορισμένα δεδομένα σε μορφή κειμένου στη συσκευή, π.χ. για την εμφάνιση τους σε μία οθόνη στη συσκευή. Ο τύπος απλού κείμενου έχει κωδικό 0.

6. ΒΑΣΗ ΔΕΔΟΜΕΝΩΝ

6.1 Τρόπος λειτουργίας

Στον κεντρικό ελεγκτή διατηρείτε μία βάση δεδομένων MySQL, η οποία εμπεριέχει τα μετα-δεδομένα των συσκευών στο δίκτυο, τα δεδομένα από τις μετρήσεις των αισθητήρων που αποστέλλονται από τις συσκευές, καθώς και τις προς αποστολή εντολές σε αυτές.

Κατά τη διαδικασία εγγραφής μιας συσκευής στο σύστημα, δημιουργείτε μία νέα εγγραφή στη βάση με τα στοιχεία της συσκευής. Κάθε συσκευή διαθέτει επίσης μία λίστα από δεδομένα και εντολές που υποστηρίζει, όπου καταχωρούνται στη βάση. Κάνοντας χρήση αυτής της λίστας, γίνετε αυτόματη προσαρμογή της διεπαφής χρήστη ανάλογος την συσκευή.

Μέσω της διεπαφής χρήστη, ένας χρήστης είναι σε θέση να καταχωρίσει εντολές προς μία συσκευή στη βάση. Για την καλύτερη διαχείριση του εύρους ζώνης, οι εντολές προς μία συσκευή αποστέλλονται με τη μορφή απάντησης στο HTTP αίτημα της, το οποίο χρησιμοποιείτε για την αποστολή των μετρήσεων των αισθητήρων της, η οποία γίνετε περιοδικά ανά τακτά χρονικά διαστήματα.

6.2 Πίνακες και χρήση των στοιχείων τους

Η βάση δεδομένων που διατηρείτε από το κεντρικό ελεγκτή αποτελείτε από τους ακόλουθους πίνακες.

Τον πίνακα “users” ο οποίος περιέχει τις εγγραφές των χρηστών τις διεπαφής χρήστη.

Πεδίο	Τύπος	Null	Κλειδί	Προεπιλογή	Επιπλέον
id	int(10) unsigned	OXI	Πρωτεύων	NULL	auto_increment
name	varchar(255)	OXI		NULL	
email	varchar(255)	OXI	Μοναδικό	NULL	
password	varchar(255)	OXI		NULL	
remember_token	varchar(100)	NAI		NULL	
created_at	timestamp	NAI		NULL	
updated_at	timestamp	NAI		NULL	

Πίνακας 6.1: Πίνακας Χρηστών.

Τον πίνακα “password_resets” ο οποίος περιέχει το e-mail και την ένδειξη (token) για την επαναφορά του κωδικού πρόσβασης του χρήστη.

Πεδίο	Τύπος	Null	Κλειδί	Προεπιλογή	Επιπλέον
id	int(10) unsigned	OXI	Πρωτεύων	NULL	auto_increment
name	varchar(255)	OXI		NULL	
email	varchar(255)	OXI	Μοναδικό	NULL	

Πίνακας 6.2: Πίνακας επαναφοράς κωδικού.

Τον πίνακα “devices” ο οποίος περιέχει τις εγγραφές των συσκευών στο δίκτυο.

Πεδίο	Τύπος	Null	Κλειδί	Προεπιλογή	Επιπλέον
id	int(10) unsigned	OXI	Πρωτεύων	NULL	auto_increment
name	varchar(255)	OXI		NULL	
status	tinyint(1)	OXI		NULL	
last_contact_at	timestamp	OXI		CURRENT_TIMESTAMP	on update CURRENT_TIMESTAMP
created_at	timestamp	NAI		NULL	
updated_at	timestamp	NAI		NULL	

Πίνακας 6.3: Πίνακας Συσκευών

Τον πίνακα “device_attributes” ο οποίος περιέχει τους τύπους των δεδομένων που αποστέλλονται από τις συσκευές στον κεντρικό ελεγκτή.

Πεδίο	Τύπος	Null	Κλειδί	Προεπιλογή	Επιπλέον
id	int(10) unsigned	OXI	Πρωτεύων	NULL	auto_increment
device_id	int(10) unsigned	OXI	Ξένο	NULL	
name	varchar(255)	OXI		NULL	
type	smallint(5) unsigned	OXI		NULL	
hardcoded_id	int(11)	OXI		NULL	
created_at	timestamp	NAI		NULL	
updated_at	timestamp	NAI		NULL	

Πίνακας 6.4: Πίνακας στοιχείων συσκευών.

Τον πίνακα “device_attribute_values” ο οποίος περιέχει τα δεδομένα που αποστέλλονται από τις συσκευές στον κεντρικό ελεγκτή.

Πεδίο	Τύπος	Null	Κλειδί	Προεπιλογή	Επιπλέον
id	int(10) unsigned	OXI	Πρωτεύων	NULL	auto_increment
device_attribute_id	int(10) unsigned	OXI	Ξένο	NULL	
value	varchar(255)	OXI		NULL	
created_at	timestamp	NAI		NULL	
updated_at	timestamp	NAI		NULL	

Πίνακας 6.5: Πίνακας τιμών στοιχείων συσκευών.

Τον πίνακα “device_commands” ο οποίος περιέχει τους τύπους των εντολών που αποστέλλονται από τον κεντρικό ελεγκτή στις συσκευές.

Πεδίο	Τύπος	Null	Κλειδί	Προεπιλογή	Επιπλέον
id	int(10) unsigned	OXI	Πρωτεύων	NULL	auto_increment
device_id	int(10) unsigned	OXI	Ξένο	NULL	
name	varchar(255)	OXI		NULL	
type	smallint(5) unsigned	OXI		NULL	
hardcoded_id	int(11)	OXI		NULL	
created_at	timestamp	NAI		NULL	
updated_at	timestamp	NAI		NULL	

Πίνακας 6.6: Πίνακας εντολών συσκευών.

Τον πίνακα “device_command_values” ο οποίος περιέχει τις εντολές που αποστέλλονται από τον κεντρικό ελεγκτή στις συσκευές.

Πεδίο	Τύπος	Null	Κλειδί	Προεπιλογή	Επιπλέον
id	int(10) unsigned	OXI	Πρωτεύων	NULL	auto_increment
device_command_id	int(10) unsigned	OXI	Ξένο	NULL	
value	varchar(255)	OXI		NULL	
created_at	timestamp	NAI		NULL	
updated_at	timestamp	NAI		NULL	

Πίνακας 6.7: Πίνακας τιμών εντολών συσκευών.

Τέλος, υπάρχει ένας ακόμα πίνακας με το όνομα “migrations” ο οποίος χρησιμοποιείται από το framework της Laravel για το versioning της βάσης το οποίο γίνετε με τη χρήση database migrations.

Πεδίο	Τύπος	Null	Κλειδί	Προεπιλογή	Επιπλέον
id	int(10) unsigned	OXI	Πρωτεύων	NULL	auto_increment
migration	varchar(255)	OXI		NULL	
batch	int(11)	OXI		NULL	

Πίνακας 6.8: Πίνακας μεταναστεύσεων.

7. ΚΩΔΙΚΕΣ

7.1 Βασικοί κώδικες.

Παρακάτω παρατίθενται όλοι οι σχετικοί με την παρούσα πτυχιακή κώδικες πλήρως σχολιασμένοι.

Κώδικες λειτουργίας του Master (Raspberry Pi)

Όνομα προγράμματος: get_serial.php Γλώσσα προγραμματισμού: PHP Περίληψη: Παρέχει ως έξοδο ένα σειριακό αριθμό του Master μέσω αποθηκευμένου αρχείου.

#!/usr/bin/php
<?php
//clean first line for convenience towards device
@ob_end_clean();
 	
//get request from device
 	
//retrieve masters serial number from file & respond
$serial = file_get_contents('/var/lib/serialtoken/serial.file');
echo $serial;

Όνομα προγράμματος: register.php

Γλώσσα προγραμματισμού: PHP

Περίληψη: Εκτελεί την εγγραφή μιας καινούριας συσκευής στη βάση δεδομένων του Master (λόγο όγκου παρατίθενται τα βασικότερα κομμάτια του κώδικα)

```
//------------------------------REGISTER NEW DEVICE LOCALLY------------------------------
//get Metadata table from device
$metadata_log = file_get_contents('php://input');
$metadata_log = json_decode($metadata_log, true);

foreach ($metadata_log as $device_name => $device){
        //register new device name (automatically set device ID)
        $reg_dev_query = $db->prepare('         INSERT INTO `devices` (`name`)
                                                        VALUES (:device_name)');
        $reg_dev_query ->execute(array(':device_name' => $device_name));
        $device_id = $db->lastInsertId(); // get the registered device ID

        //register new device attributes (automatically set attributes IDs)
        $reg_att_query = $db->prepare(' INSERT          INTO          `device_attributes`
(`device_id`,`name`,`type`,`hardcoded_id)
VALUES (:device_id,:attribute_name,:attribute_type,:attribute_id)');
        foreach ($device['attributes'] as $attribute){
                $reg_att_query ->execute(array(  ':attribute_name' => $attribute['name'],
                ':attribute_type' => $attribute['type'],
                ':attribute_id' => $attribute['attribute_id'],
                ':device_id' => $device_id);
        }
```

```
        //register new device commands (automatically set commands IDs)
        $reg_com_query = $db->prepare('        INSERT     INTO     `device_commands`
(`device_id`,`name`,`type`,`hardcoded_id`)
                                                                VALUES
(:device_id,:command_name,:command_type,:command_id)');

        foreach ($device['commands'] as $command){
                $reg_com_query              ->execute(array(:command_name'              =>
$command['name'],':command_type'      =>      $command['type'],':command_id'      =>
$command['command_id'],':device_id' => $device_id));
        }
  }
  //------------------------------REGISTER NEW DEVICE LOCALLY END----------------------
```

Όνομα προγράμματος: report.php

Γλώσσα προγραμματισμού: PHP

Περίληψη: Εισάγει δεδομένα από μια συσκευή στη βάση δεδομένων. Ελέγχει εάν υπάρχουν εντολές για τη συγκεκριμένη συσκευή και τις αποστέλλει πίσω σαν απάντηση.

```
<?php
require_once '/var/www/appliances/init.php';

/* The following program is called whenever a device(PIC) wants to send its current sensor values to the master(RPi).
   Using the provided hard-coded attribute ID and device ID, RPi determines which attribute
  the device is referring to, inserts a new value into its DataBase.
   Then proceeds to send a command back to the device(PIC), but only if the command exists AND should be sent.
--this is what the incoming request array looks like--
$report_log = array (
    'deviceId' => 182,
    'attributes'=> array (
            array (
              'id' => 7,  //this is the hardcoded id which PIC knows
              'value' => -3
            )
        )
);
```

```
  answer must be
  $report_log = array ('status' => 0,
        'commands'=> array (array (
              'id' => 3,//this is the hardcoded id which PIC knows
              'value' => -5)));*/
  /*REPORT LOG INSERT ----------------------------------------
  This part of the scripts handles the incomming device(PIC) requests.
  *Based on the data provided by the device a SELECT query is executed.
  *If everything goes as planned the new values are inserted into the DataBase and the
statu is NOT SET (status=0).
  *Should a data-based error arrise then status is SET depending on the problem
        (status=1 |for Invalid Hardcoded ID)
        (status=2 |for Invalid Device ID)
  */
  $report_log = file_get_contents('php://input');
  //update incoming attribute values table for each device
        $report_log = json_decode($report_log, true);

              //get device_id and values of each device id from response table
              $device_id = $report_log['deviceId'];
              //check if Device ID exists, using count command, if result is 1 => ID is
valid.
              $check_dev_id = $db->prepare('SELECT COUNT(`id`) AS `IdExists`
FROM `devices` WHERE `id` =:device_id');
              $check_dev_id ->bindValue(':device_id', $device_id);
```

```
                $check_dev_id ->execute();
                $check_dev_id_res = $check_dev_id ->fetch();
                $check_dev_id ->closeCursor();

                if($check_dev_id_res['IdExists'] == 1){
                //find attribute ID on RPi-DB based on given hardcoded and device ID's
                $att_id_query = $db->prepare(    'SELECT `id` FROM `device_attributes`
                WHERE `device_id`=:device_id AND `hardcoded_id`=:hardcoded_id');
                  //If any values exist.
                  if( isset ($report_log['attributes'])){
                    foreach($report_log['attributes'] as $attribute){
                    //Bind values of query and execute it
                    $att_id_query ->bindValue(':device_id', $device_id);
                    $att_id_query ->bindValue(':hardcoded_id', $attribute['id']);
                    $att_id_query ->execute();
                    $attribute_id = $att_id_query ->fetch();
                     //if any errors occur during the attr ID (RPi-DB) is probably caused by
invalid HardID
                     $count = $att_id_query->rowCount();
                     $att_id_query ->closeCursor();
                          if($count!=0){ //row count=0 means invalid HardID
                          //update incoming commands table for each device
                          $ins_query = $db->prepare('       INSERT                    INTO
`device_attribute_values` (`device_attribute_id`,`value`)
                          VALUES(:attribute_id,:insert_value)');
                          //bind values of response table into query
```

```
                    $ins_query->execute(array(          ':attribute_id' => $attribute_id['id'],
                    ':insert_value' => $attribute['value']));
                    $status = 0;
                    //Inserting values into database
                    }else{
                    $status = 1;
                     //Invalid Hardcoded ID
                    }
                }
        }
  }else{
        $status = 2; //Invalid Device ID     }
  //REPORT LOG INSERT END ----------------------------------------
  /*SEND COMMAND LOG ----------------------------------------------
```

Thist part of the script sends the scheduled commands to pic wich are time dependend.

*If status variable is correct from the previus script parts then we can proceed to command the device.

*Selects a command (wich is to be executed "now") based on the incomming attribute. [$response_query]

*Deletes the command from database.

[$delete_sent_comm]

*Created a response array using values of response query.

[$command_log]

```
---- Answer to PIC with commands or status variable -----
response array is like so...
$report_log = array (
                'status' => 0,
        'commands'=> array (
            array (
              'id' => 3, //this is the hardcoded id which PIC knows
              'value' => -5
            )
        )
);
-----------------------------------------------------
*Sends the actual command.
*/
$command_log = array( 'status' => $status);
 if ( $status == 0 ){
        //select command values on the requested attribute
        $response_query = $db->prepare(         'SELECT
`device_command_id`,`hardcoded_id`,`value` FROM `device_command_values`
                                        INNER JOIN `device_commands`
                                        ON `device_command_id` = `device_commands`.`id`
                                        AND `device_id` = :device_id
                                        AND         (`scheduled_at`<         DATE_SUB(NOW(),
INTERVAL 1 MINUTE_SECOND) OR `scheduled_at` IS NULL)' );
        $response_query -> execute(array(':device_id' => $device_id));
```

```
        $res_quer_result = $response_query->fetchAll();

        //prepare to delete sent commands
        $delete_sent_comm = $db->prepare ('   DELETE                    FROM
`device_command_values`WHERE `device_command_id` = :command_id');
        //save command values
        foreach($res_quer_result as $command){
                $command_log['commands'][] = array(   'id' =>$command['hardcoded_id'],
                                                'value' => $command['value']);
                //delete each sent command
                $delete_sent_comm       ->      execute(array(':command_id'     =>
$command['device_command_id']));
        }}else{ $command_log['commands'][] = array(); }
  //send command_log
  header('content-type:application/json');
  echo $response = json_encode($command_log);
  //SEND COMMAN LOG END----------------------------------------------
```

Κώδικας ενσύρματης συσκευής PIC

Όνομα προγράμματος: Etherpic main

Γλώσσα προγραμματισμού: C

Περίληψη: Εκτελεί εγγραφή συσκευής στον Master, αποστέλλει δεδομένα από τους αισθητήρες της συσκευής, εκτελεί εντολές του χρήστη.

Παράλληλα προγράμματα: adc.c, aes.c, config.c, pxcommon.c, pxdhcp.c, pxenc.c, pxmac.c, pxnet.c, pxspi.c

Βιβλιοθήκες: adc.h, aes.h, pxcommon.h, pxdhcp.h, pxenc.h, pxmac.h, pxnet.h, pxspi.h, system.h

```
// MAC addresses starting with 80:80:80 are reserved and won't conflict with equipment manufacturers
TMacAddr gMyMAC = { 0x80, 0x80, 0x80, 0x00, 0x00, 0xA1 };

#ifndef USE_STATIC_IP
TIpAddr  gMyIP  = { 0, 0, 0, 0 };
#else
TIpAddr  gMyIP  = { 192, 168, 2, 111 };
#endif

#ifndef USE_STATIC_IP
TIpAddr  dhcp;
uint32_t leasetime;
TIpAddr  gateway;
TIpAddr  namesrv;
```

```
  TIpAddr  netmask;
  #else
  TIpAddr  gateway = {192, 168, 2, 1};
  TIpAddr  namesrv = {192, 168, 2, 1};
  TIpAddr  netmask = {255, 255, 255, 0};
  #endif

  //definitions used for comms between pic(device) & RPi (master)
  #define METADATA "{\"cooler\":{\"attributes\":
[{\"name\":\"temp\",\"type\":100,\"attribute_id\":1},
{\"name\":\"fan_speed\",\"type\":101,\"attribute_id\":2}],\"commands\":
[{\"name\":\"fan_speed\",\"type\":101,\"command_id\":2}]}}"
  #define ATTRIBUTE "{\"deviceId\":,\"attributes\":[{\"id\":,\"value\":},
{\"id\":,\"value\":}]}" //length of 0-45
  // helper line |12 |33 |42 |50 |59 //use these
numbers to properly insert data into ATTRIBUTE string
  #define ATTRIBUTETEMPID 1
  #define ATTRIBUTESPEEDID 2
  #define MASTER_SERIAL_SZ 13

  #define ATTRIBUTED "{\"deviceId\":198,\"attributes\":[{\"id\":5,\"value\":20}]}"

  //definitions used in steinhart's eq. (temp calculation)
  // resistance at 25 degrees C
  #define THERMISTORNOMINAL 1000
  // temp. for nominal resistance (almost always 25 C)
```

```
#define TEMPERATURENOMINAL 25
// The beta coefficient of the thermistor (usually 3000-4000)
#define BCOEFFICIENT 1053
// the value of the 'other' resistor
#define SERIESRESISTOR 10000



//variables used for comms between pic(device) & RPi (master)
uint8_t masterSerial[13];
uint16_t deviceId = 0;  //change back to 0 after testing
uint8_t postMode = 0;   //change back to 0 after testing
uint16_t attrValue = 0;
uint16_t commValue = 0;

//variable used for conversions and temp calculation
float tempValue = 0; //temporary value for use
float steinhart = 0; //steinhart equation helps us determine temperature

//delay variable
uint8_t waitTime = 5;

int8_t httpSessionStart ( TIpAddr ip_addr, uint16_t dstPort );
bool hasIpAddress ( void );
void onHttpConnect ( sock_t sock );
void onHttpResponse ( uint16_t status, char *data );
void initialize ( void );
```

```
void sendHttpRequest( sock_t sock );
void temperatureRead( void );
void requestDelay( void );


void main ( void ) {
    uint16_t seconds = 0;
    uint8_t i, count = 0;

    initialize();



    // Use noise from a floating ADC input to initialize rand()
    adcInit();
    srand(adcRead());
    adcClose();

    // Initialize and start networking
    netInit();
    netEnable();

    // Enable interrupts
    INTCONbits.GIE = 1;
    // The IP of the remote HTTP server
    TIpAddr  ip_addr  = { 0, 0, 0, 102 };
```

```
//ip_addr = (gMyIP[0], gMyIP[1], gMyIP[2], 102);
int8_t requestStatus = 0;
for ( ;; ) {
    // Every time TMR1 overflows
    if ( PIR1bits.TMR1IF ) {
        PIR1bits.TMR1IF = 0;
        count++;

        // If the HTTP request failed try again without waiting for 3 whole seconds
        if ( hasIpAddress() && requestStatus == -1 ) {
            requestStatus = httpSessionStart(ip_addr, 80);}

        // Approximately every 1s (1.048576s)
        if ( count >= T1PERSEC ) {
            count = 0;
            seconds++;
            if ( seconds > 999 ) {
                seconds = 0;}

            netTimer( seconds );

            // Every three seconds make an HTTP request to the web server
            if ( hasIpAddress() && seconds % 3 == 0 ) {
                requestStatus = httpSessionStart(ip_addr, 80);
            }
```

```
            #ifndef USE_STATIC_IP
            // Every 10 seconds to get an IP until the DHCP servers responds
            if ( !hasIpAddress() && (seconds == 1 || seconds % 10 == 0) ) {
               dhcpDiscover();
            }
            #endif}}
        netIdle();
        //search for Master's ip based on given IP via DHCP
        ip_addr[0] = gMyIP[0];
        ip_addr[1] = gMyIP[1];
        ip_addr[2] = gMyIP[2];}
}
void initialize ( void ) {
    uint8_t i;
    memset(masterSerial,0,13);
    TRISAbits.RA1 = 1; //Analog pin 1 as input
    TRISCbits.RC2 = 0; //PWM pin set as output
    OpenTimer2(T2_PS_1_16);
    OpenPWM1(500); //Open pwm at 2KHz
    SetDCPWM1(0);
#if defined(_18F4553)
    OSCCONbits.IRCF = 0b111; // TIMER 1 is 8MHz (INTOSC drives clock directly)
    ADCON1bits.PCFG = 0b1110; // All I/O pins are digital except AN0 (RA0)
#endif
```

```
    // wait 1 second for PICkit3 to use MCLR/PGC/PGD
    for ( i = 0; i < 100; i++ ) __delay_ms( 10 );
    T1CONbits.TMR1CS = 0; // Internal clock (FOSC/4)
    T1CONbits.T1CKPS = 0b11; // 1:8 Prescale value
    T1CONbits.TMR1ON = 1;
    // Timer 1 is going to overflow every 2^16 / ( FOSC / 4 / 8 ) = 262.144ms fpr FOSC
= 8Mhz}

  int8_t httpSessionStart ( TIpAddr ip_addr, uint16_t dstPort ) {
    sock_t sock;
      if ( sock = tcpEstablishConnection(ip_addr, dstPort, 0, TCP_PROTO_HTTP) ==
-1 ) {
        return -1;}
    return 0;}
  bool hasIpAddress ( void ) {
    return !(gMyIP[0] == 0 && gMyIP[1] == 0 && gMyIP[2] == 0 && gMyIP[3] == 0);}

  // Callbacks from pxnet TCP sockets
  //
  void onHttpResponse ( uint16_t status, char *data ) {
    uint8_t k =0 ;
    char *commheader;

    if ( status == 200 ) {
       switch(postMode){
```

```
        case 1: //get_serial requests provides back a masterSerial = RPi's serial
                strncpy(masterSerial, data, MASTER_SERIAL_SZ); //master serial is 12 chars and \0 anything else is invalid
            masterSerial[MASTER_SERIAL_SZ-1] = '\0';
            break;
        case 2: //report provides back commands by the user or "status:0:" string
              if ( strstr(data, "\"status\":0") && ( commheader = strstr(data, "\"value\"")) != NULL){
                commValue = 0;
                while ( !commValue ) {
                    if ( isdigit(commheader[ k ]) ) {
                        commValue = atoi(&commheader[ k ]);
                        break;
                    }
                    k++;
                }

                //kickstart fan
                SetDCPWM1(800);
                __delay_ms(50);

                //set "fan speed" according to command (sould be in range 0-100)
                if (commValue > 100) commValue = 100;
                //if (commValue < 0) commValue = 0;
                attrValue = commValue * 10 + 1024; //transform range of 0-100% to 0-1024 for PWM use
```

```
                SetDCPWM1(attrValue);   //set PWM according to given value
                attrValue = commValue;  //match attrValue as it was originally given for use as response to master (RPi)

                temperatureRead();     //reads temperature and stores it value in 'attrValue' variable



            }
            else if ( strstr(data, "\"status\":0") ){ temperatureRead(); }   //reads temperature and stores it value in 'attrValue' variable
            //else if ( strstr(data, "Invalid Hardcoded ID") ){}
            //else if ( strstr(data, "Invalid Device ID") ){}
            requestDelay();
            break;
        case 3: //register provides back a deviceID set from RPi
            if ( !strncmp(data,"OK",2 )){ //RPi response looks like "OK132" (OK+deviceID)
                deviceId = atoi(&data[2]);
            }
            break;
        }
    }
    else {
        //retry POST?
    }
```

```
}
//Lets assume we have a heater device

void onHttpConnect ( sock_t sock ) {
    //if serial number does not exist then we need to request it
      if ( !strcmp( masterSerial,NULL ) ) { //enable get serial request
         postMode = 1;
         sendHttpRequest( sock );
       }
      //if a deviceId is present it means that the device is registered, therefore can send
readings
       else if( deviceId ){    //enable report request read sensors.. send data
          postMode = 2;
          memset(tempAttrBuff,0,strlen(tempAttrBuff));
          sendHttpRequest( sock );
       }
      //if the deviceId does not exist then we have to register the device on smart home
and aquire a deviceId
       else{   //enable register request
          postMode = 3;
          sendHttpRequest( sock );
       }

}
```

```
void sendHttpRequest( sock_t sock ){
    // Send HTTP Request
    netPutBegin( sock );

    //soutpgm( sock, "POST /get_serial.php HTTP/1.0" CRLF );
    //soutpgm( sock, "User-Agent: PIC18LF4553" CRLF );
    //soutpgm( sock, "Host: 192.168.1.102" CRLF );
    //soutpgm( sock, "Accept: */*" CRLF );
    //soutpgm( sock, "Content-Type: application/x-www-form-urlencoded" CRLF );
    //soutpgm( sock, "Content-Length: 37" CRLF );
    //soutpgm( sock, CRLF );
    //soutpgm( sock, "data=");
    //soutpgm( sock, "data_to_be_sent");


    switch(postMode){   //based on postMode determine witch script is requested
        case 1:
            soutpgm( sock, "POST /get_serial.php HTTP/1.0" CRLF );
            break;
        case 2:
            soutpgm( sock, "POST /report.php HTTP/1.0" CRLF );
            break;
        case 3:
            soutpgm( sock, "POST /register.php HTTP/1.0" CRLF );
            break;
```

```
    }

    soutpgm( sock, "User-Agent: PIC18LF4553" CRLF );
    soutpgm( sock, "Host: 192.168.1.102" CRLF );
    soutpgm( sock, "Accept: */*" CRLF );
    soutpgm( sock, "Content-Type: application/x-www-form-urlencoded" CRLF );

    switch(postMode){  //based on postMode determine what is the data size
       case 1:
          soutpgm( sock, "Content-Length: 5" CRLF );
          break;
       case 2:
                  //sprintf(  buff,  "%d",  strlen(ATTRIBUTE)+sprintf(NULL,"%+d",deviceId)
+sprintf(NULL,"%+d",attrValue)+strlen(ATTRIBUTEID) );
          soutpgm( sock, "Content-Length: 70");
          //soutpgm( sock, buff );
          soutpgm( sock, CRLF );
          break;
       case 3:
          sprintf( buff, "%d", strlen(METADATA) );
          soutpgm( sock, "Content-Length: ");
          soutpgm( sock, buff );
          soutpgm( sock, CRLF );
          //}
          break;
    }
```

```
    soutpgm( sock, CRLF );
    //soutpgm( sock, "data=");
    switch(postMode){  //based on postMode determine what will be sent
       case 1:
          break;
       case 2:
          strncpy( buff, ATTRIBUTE , strlen(ATTRIBUTE) );
          strncpy( tempAttrBuff, ATTRIBUTE , strlen(ATTRIBUTE) );

          //place measured value into attribute table (for fan speed)
          sprintf( &buff[59] , "%d" , commValue );
          strcpy( &buff[59+sprintf(NULL,"%d",commValue)] , &tempAttrBuff[59]);
          strcpy( tempAttrBuff , buff);
          //place attributeId into attribute table (for fan speed)
          sprintf( &buff[50], "%d" , ATTRIBUTESPEEDID );
                              strcpy(   &buff[50+sprintf(NULL,"%d",ATTRIBUTESPEEDID)],
&tempAttrBuff[50]);
          strcpy( tempAttrBuff , buff);

          //place measured value into attribute table (for temperature)
          sprintf( &buff[42] , "%d" , attrValue );
          strcpy( &buff[42+sprintf(NULL,"%d",attrValue)] , &tempAttrBuff[42]);
          strcpy( tempAttrBuff , buff);

          //place attributeId into attribute table (for temperature)
```

```
            sprintf( &buff[33], "%d" , ATTRIBUTETEMPID );
                                    strcpy(    &buff[33+sprintf(NULL,"%d",ATTRIBUTETEMPID)],
&tempAttrBuff[33]);
            strcpy( tempAttrBuff , buff);

            //place deviceId into attribute table
            sprintf( &buff[12],"%d", deviceId );
            strcpy( &buff[12+sprintf(NULL,"%d",deviceId)], &tempAttrBuff[12]);
            strcpy( tempAttrBuff , buff);

            soutpgm( sock, buff );
            soutpgm( sock, CRLF );
            break;
        case 3:
            soutpgm( sock, METADATA CRLF );
            break;
    }

    netPutFinalize( sock );
 }


 void temperatureRead (void){

    //take a temperature reading
                CloseADC();     //close any previous ADC enables
```

```
                    OpenADC(ADC_FOSC_2 & ADC_RIGHT_JUST & ADC_2_TAD,
ADC_CH1 & ADC_INT_OFF & ADC_REF_VDD_VSS, ADC_2ANA); //set ADC to read
from AN1
            ConvertADC();
            while ( BusyADC() );
              tempValue = ReadADC();  //acquire actual temp
            CloseADC();
            //math math math
            //tempValue = (tempValue * 3.3) / 4092;
            //convert to resistance
            tempValue = 1000 / ((4092 / tempValue)-1) ;
            //tempValue = 1000 / tempValue;

            //convert to temperature
            steinhart = tempValue / 1000;     // (R/Ro)
            steinhart = log(steinhart);              // ln(R/Ro)
            steinhart /= 1090;               // 1/B * ln(R/Ro)
            steinhart += 1.0 / (21 + 273.15); // + (1/To)
            steinhart = 1.0 / steinhart;              // Invert
            steinhart -= 273.15;                     // convert to C
            attrValue = (int)steinhart;

 }

 void requestDelay(){
    //variable to delay each PIC_device_request time
```

```
    uint8_t waitTime = 1;
    uint16_t i,j;

    //create delay 1 second for each interation
        while(waitTime > 0){
            for(i=25000;i>0;i--){ //1 second delay
                for(j=100;j>0;j--){
                    Nop();
                }
            }
            waitTime--;
        }
        //waitTime = 5;
}
```

Κώδικας ασύρματης συσκευής AVR(ESP)

Όνομα προγράμματος: esp_slave_device.ino

Γλώσσα προγραμματισμού: C

Περίληψη: Εκτελεί εγγραφή συσκευής στον Master, αποστέλει δεδομένα από τους αισθητήρες της συσκευής, εκτελεί εντολές του χρήστη. Λαμβάνει στοιχεία ασύρματης σύνδεσης.

Σχετιζόμενα προγράμματα: WirelessConnector.jar

```
#define DEBUG
#define MASTER_IP_BYTE 102
#define SSID_SZ 32
#define PASSWORD_SZ 128
#define POST_DATA_SZ 255
#define REQUEST_URL_SZ 255
#define SERIAL_SZ 32
#define RESPONSE_SZ 255

const int rs = 4, en = 2, d4 = 13, d5 = 12, d6 = 14, d7 = 16;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

IPAddress ipSelf(0, 0, 0, 0);
IPAddress ipMaster(0, 0, 0, 0);

byte deviceId = 0;
char masterSerial[SERIAL_SZ];
```

```
int DHT11 = 5;
SimpleDHT11 dht;

bool uart_read_wifi_credentials ( void );

void epprom_write_wifi_credentials ( char* ssid, char* password );
void epprom_read_wifi_credentials ( char* ssid, char* password );
void epprom_write_serial_id_pair  ( char* serial, byte deviceId );
void epprom_read_serial_id_pair ( char* serial, byte* deviceId  );

int http_serial ( char* serial );
int http_register ( byte* deviceId );
int http_report ( byte temperature, byte humidity );

void lcd_clear_line ( byte line );

void setup() {

  char ssid[SSID_SZ];
  char password[PASSWORD_SZ];

  // Configure Serial
  Serial.begin(9600);

  // Configure EEPROM
  EEPROM.begin(512);
```

```
  // Configure LCD
  lcd.begin(16, 4);

  // Read WiFi credentials from EEPROM
  epprom_read_wifi_credentials(ssid, password);

  // If SSID and password were read successfully from EEPROM attempt to connect to to WiFi
  if ( strlen(ssid) > 0 && strlen(password) > 0 ) {

    // Configure WiFi
    WiFi.begin(ssid, password);

    lcd.setCursor(0, 0);
    lcd.print("Attempting to");
    lcd.setCursor(0, 1);
    lcd.print("connect...");

    // Wait to connect to WiFi
    uint8_t i = 0;
    while ( WiFi.status() != WL_CONNECTED && i++ < 100 ) delay(100);

    lcd.clear();

    if ( WiFi.status() != WL_CONNECTED ) {
```

```
    lcd.setCursor(0, 0);
    lcd.print("WiFi connection");
    lcd.setCursor(0, 1);
    lcd.print("has failed!");
    lcd.setCursor(0, 2);

  }
  else {

    // Find self and master IP
    ipSelf = ipMaster = WiFi.localIP();
    ipMaster[3] = MASTER_IP_BYTE;

    #ifdef DEBUG
    lcd.setCursor(0, 0);
    lcd.print("WiFi connection");
    lcd.setCursor(0, 1);
    lcd.print("was successful!");
    lcd.setCursor(0, 3);
    lcd.print(ipSelf);
    delay(2000);
    #endif

  }
```

```
  }

}

void loop() {

  char receivedSerial[SERIAL_SZ];
  int requestStatus;

  if ( uart_read_wifi_credentials() == true ) {

    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("WiFi credentials");
    lcd.setCursor(0, 1);
    lcd.print("received");
    lcd.setCursor(0, 3);
    lcd.print("Please reset");

    // Enter deep sleep until reset
    ESP.deepSleep(0);

  }

  if ( WiFi.status() == WL_CONNECTED ) {
```

```
  // If the device is not initialized
  if ( deviceId == 0 ) {

    // Fetch serial from server
    if ( (requestStatus = http_serial(receivedSerial)) == 0 ) {

      #ifdef DEBUG
      lcd_clear_line(0);
      lcd_clear_line(1);
      lcd.setCursor(0, 0);
      lcd.print("SN: ");
      lcd.print(receivedSerial);
      delay(2000);
      #endif

      // Read serial and device id stored in EEPROM
      epprom_read_serial_id_pair(masterSerial, &deviceId);

      // Compare with received serial
      if ( strcmp(receivedSerial, masterSerial) == 0 ) {

        #ifdef DEBUG
        lcd_clear_line(1);
        lcd.setCursor(0, 1);
        lcd.print("Stored ID: ");
        lcd.print(deviceId);
```

```
  delay(2000);
  #endif

}
else {

  // If they don't match send register request to server
  if ( (requestStatus = http_register(&deviceId)) == 0 ) {

    // Write master serial and device id to EEPROM
    epprom_write_serial_id_pair(receivedSerial, deviceId);
    strncpy(masterSerial, receivedSerial, SERIAL_SZ);

    #ifdef DEBUG
    lcd_clear_line(1);
    lcd.setCursor(0, 1);
    lcd.print("Assigned ID: ");
    lcd.print(deviceId);
    delay(2000);
     #endif

  }
  else {

    #ifdef DEBUG
    lcd_clear_line(2);
```

```
            lcd.setCursor(0, 2);
            lcd.print("Error R");
            lcd.print(requestStatus);
            delay(2000);
            #endif

          }

        }

      }
      else {

        #ifdef DEBUG
        lcd_clear_line(2);
        lcd.setCursor(0, 2);
        lcd.print("Error S");
        lcd.print(requestStatus);
        delay(2000);
        #endif

      }

    }

    // Read temperature sensor
```

```
  byte temperature = 0;
  byte humidity = 0;

  if ( dht.read(DHT11, &temperature, &humidity, NULL) != SimpleDHTErrSuccess ) {
    lcd.setCursor(0, 2);
    lcd.print("Sensor Failure"); // In case of error output sensor failure and return
  }
  else {

    lcd.setCursor(0, 2);
    lcd.print("Temperature: ");
    lcd.print(temperature);
    lcd.print("C");

    lcd.setCursor(0, 3);
    lcd.print("Humidity: ");
    lcd.print(humidity);
    lcd.print("%");

    #ifdef DEBUG
    delay(2000);
    #endif

    // Report temperature value to master
    if ( (requestStatus = http_report(temperature, humidity)) == 0 ) {
```

```
        #ifdef DEBUG
        lcd_clear_line(2);
        lcd.setCursor(0, 2);
        lcd.print("Values sent");
        lcd_clear_line(3);
        lcd.setCursor(0, 3);
        lcd.print("successfully!");
        #endif

      }
      else {

        #ifdef DEBUG
        lcd_clear_line(2);
        lcd.setCursor(0, 2);
        lcd.print("Error L");
        lcd.print(requestStatus);
        delay(2000);
        #endif

      }

    }

  }
```

```
  delay(3000);

}

/**
 * Clear the specified line of the LCD
 */
void lcd_clear_line ( byte line ) {

  if ( line >= 0 && line <= 3 ) {
    lcd.setCursor(0, line);
    lcd.print("                ");
  }

}

/**
 * Sends the temperature attribute value to server
 */
int http_report ( byte temperature, byte humidity ) {

  char postData[POST_DATA_SZ];
  char requestUrl[REQUEST_URL_SZ];
  char response[RESPONSE_SZ];
  String res;
```

```
    HTTPClient http;

    // Set request URL
    sprintf(requestUrl, "http://%s/report.php", ipMaster.toString().c_str());
    http.begin(requestUrl);

                sprintf(postData,       "{\"deviceId\":%d,\"attributes\":[{\"id\":1,\"value\":%d},
{\"id\":2,\"value\":%d}]}", deviceId, temperature, humidity);

    // Start connection and send HTTP headers
    http.addHeader("Content-Type", "application/json");
    http.addHeader("Accept", "application/json");
    int httpCode = http.POST(postData);

    if( httpCode > 0 && httpCode == HTTP_CODE_OK ) {

      res = http.getString();
      http.end();

      res.trim();
      strncpy(response, res.c_str(), RESPONSE_SZ);
      response[RESPONSE_SZ - 1] = '\0';

      return 0;
    }
```

```
  String error = http.errorToString(httpCode).c_str();
  http.end();

  return httpCode;

}

/**
 * Sends registration request to server
 */
int http_register ( byte* deviceId ) {

  char postData[POST_DATA_SZ];
  char requestUrl[REQUEST_URL_SZ];
  char response[RESPONSE_SZ];
  String res;

  HTTPClient http;

  // Set request URL
  sprintf(requestUrl, "http://%s/register.php", ipMaster.toString().c_str());
  http.begin(requestUrl);

                sprintf(postData, "{\"temperature monitor\":{\"attributes\":
[{\"name\":\"temperature\",\"type\":100,\"attribute_id\":1},
{\"name\":\"humidity\",\"type\":101,\"attribute_id\":2}],\"commands\":[]}}");
```

```
  // Start connection and send HTTP headers
  http.addHeader("Content-Type", "application/json");
  http.addHeader("Accept", "application/json");
  int httpCode = http.POST(postData);

  if( httpCode > 0 && httpCode == HTTP_CODE_OK ) {

    res = http.getString();
    http.end();

    res.trim();
    strncpy(response, res.c_str(), RESPONSE_SZ);
    response[RESPONSE_SZ - 1] = '\0';

     if ( response[0] == 'O' && response[1] == 'K' || response[0] == 'o' && response[1]
== 'k' ) {

      *deviceId = (byte) atoi(&response[2]);
      return 0;

    }

    *deviceId = 0;
    return 1;

  }
```

```
  String error = http.errorToString(httpCode).c_str();
  http.end();

  *deviceId = 0;
  return httpCode;

}

/**
 * Fetches the master serial number from server
 */
int http_serial ( char* serial ) {

  char postData[POST_DATA_SZ];
  char requestUrl[REQUEST_URL_SZ];
  String res;

  HTTPClient http;

  // Set request URL
  sprintf(requestUrl, "http://%s/get_serial.php", ipMaster.toString().c_str());
  http.begin(requestUrl);

  postData[0] = '\0';
```

```
  // Start connection and send HTTP headers
  http.addHeader("Content-Type", "application/json");
  http.addHeader("Accept", "application/json");
  int httpCode = http.POST(postData);

  if( httpCode > 0 && httpCode == HTTP_CODE_OK ) {

    res = http.getString();
    http.end();

    res.trim();
    strncpy(serial, res.c_str(), SERIAL_SZ);
    serial[SERIAL_SZ - 1] = '\0';

    return 0;

  }

  String error = http.errorToString(httpCode).c_str();
  http.end();

  return httpCode;

}

/**
```

```
 * Reads the SSID and WiFi password from UART
 */
bool uart_read_wifi_credentials ( void ) {

  int bytesRead = 0;
  char ssid[SSID_SZ];
  char password[PASSWORD_SZ];

  ssid[0] = '\0';
  password[0] = '\0';

  // Read SSID from serial
  if ( Serial.available() > 0 ) {
    bytesRead = Serial.readBytesUntil(char(10), ssid, SSID_SZ);
    ssid[bytesRead] = '\0';
  }

  // Make sure that the SSID read is smaller than the buffer size
  if ( strlen(ssid) >= SSID_SZ - 1 ) {
    return false;
  }

  // Read password from serial
  if ( Serial.available() > 0 ) {
    bytesRead = Serial.readBytesUntil(char(10), password, PASSWORD_SZ);
    password[bytesRead] = '\0';
```

```
  }

  // Make sure that the password read is smaller than the buffer size
  if ( strlen(password) >= PASSWORD_SZ - 1 ) {
    return false;
  }

  // If both SSID and password are being read successfully
  if ( strlen(ssid) > 0 && strlen(password) > 0 ) {

    // Write SSID and password to EEPROM
    epprom_write_wifi_credentials(ssid, password);

    // Send ok message back to client
    Serial.write("ok");

    return true;

  }

  return false;

}

/**
 * Writes the SSID and WiFi password to EEPROM
```

```
 */
void epprom_write_wifi_credentials ( char* ssid, char* password ) {

  int offset;

  // Write SSID length to EEPROM
  offset = 0;
  EEPROM.write(offset++, (byte) strlen(ssid));

  // Write SSID to EEPROM
  for ( int i = 0; ssid[i] != '\0' && i < SSID_SZ; ++i ) {
    EEPROM.write(offset + i, ssid[i]);
  }

  // Write password length to EEPROM
  offset += strlen(ssid);
  EEPROM.write(offset++, (byte) strlen(password));

  // Write password to EEPROM
  for ( int i = 0; password[i] != '\0' && i < PASSWORD_SZ; ++i ) {
    EEPROM.write(offset + i, password[i]);
  }

  // Write changes to EEPROM
  EEPROM.commit();
```

```
}

/**
 * Reads the SSID and WiFi password from EEPROM
 */
void epprom_read_wifi_credentials ( char* ssid, char* password ) {

  int ssidLength = 0;
  int passwordLength = 0;
  int offset;

  ssid[0] = '\0';
  password[0] = '\0';

  // Read SSID from EEPROM
  offset = 0;
  ssidLength = (int) EEPROM.read(offset++);

  if ( ssidLength > 0 ) {

    int i;

    for ( i = 0; i < ssidLength && i < SSID_SZ; ++i ) {
      ssid[i] = (char) EEPROM.read(offset+i);
    }
```

```
    ssid[i] = '\0';

  }

  // Read password from EEPROM
  offset += ssidLength;
  passwordLength = (int) EEPROM.read(offset++);

  if ( passwordLength > 0 ) {

    int i;

    for ( i = 0; i < passwordLength && i < PASSWORD_SZ; ++i ) {
      password[i] = (char) EEPROM.read(offset+i);
    }

    password[i] = '\0';

  }

}

/**
 * Reads the master serial number from EEPROM
 */
void epprom_read_serial_id_pair ( char* serial, byte* deviceId ) {
```

```
  int serialLength = 0;
  int offset;

  serial[0] = '\0';

  // Read serial
  offset = SSID_SZ + PASSWORD_SZ;
  serialLength = (int) EEPROM.read(offset++);

  if ( serialLength > 0 ) {

    int i;

    for ( i = 0; i < serialLength && i < SERIAL_SZ; ++i ) {
      serial[i] = (char) EEPROM.read(offset+i);
    }

    serial[i] = '\0';

  }

  // Read device id
  offset += serialLength;
  *deviceId = (byte) EEPROM.read(offset++);
```

```
}

/**
 * Writes the master serial number to EEPROM
 */
void epprom_write_serial_id_pair  ( char* serial, byte deviceId ) {

  int offset;

  // Write serial length to EEPROM
  offset = SSID_SZ + PASSWORD_SZ;
  EEPROM.write(offset++, (byte) strlen(serial));

  // Write serial to EEPROM
  for ( int i = 0; serial[i] != '\0' && i < SSID_SZ; ++i ) {
    EEPROM.write(offset + i, serial[i]);
  }

  // Write device id length to EEPROM
  offset += strlen(serial);
  EEPROM.write(offset++, (byte) deviceId);

  // Write changes to EEPROM
  EEPROM.commit();

}
```

Κώδικας προσωπικού υπολογιστή χρήστη.

Όνομα προγράμματος: WirelessConnector.jar

Γλώσσα προγραμματισμού: JAVA

Περίληψη: Επικοινωνεί με την ασύρματη συσκευή μέσω του υπολογιστή του χρήστη για να λάβει πληροφορίες ασύρματης σύνδεσης του οικιακού δικτύου.

```
// Using AWT container and component classes
// Using AWT event classes and listener interfaces
//An AWT program inherits from the top-level container java.awt.Frame
public class GUI_main extends Frame implements ActionListener, WindowListener {
	private static final long serialVersionUID = 1L;
	private Label lbl_SSID, lbl_Pass, lbl_SerDevices, lbl_Info;
	private TextField tf_SSID, tf_Pass;
	private Choice ch_SerDevices;
	private Button btn_OK;
	// Constructor to setup GUI components and event handlers
	public GUI_main() {
		setLayout(new FlowLayout());
	// "super" Frame (container) sets layout to FlowLayout, which arranges
	// the components from left-to-right, and flow to next row from top-to-bottom.
		lbl_SSID = new Label("Enter WLan name (SSID): ");  // Construct Label
		add(lbl_SSID);// "super" Frame container adds Label component
		tf_SSID = new TextField("SSID",20);		//Construct TextField
		add(tf_SSID);
		//tf_SSID.addActionListener(this);
		// "tfInput" is the source object that fires an ActionEvent upon entered.
	// The source add "this" instance as an ActionEvent listener, which provides
```

```
        //  an ActionEvent handler called actionPerformed().
        // Hitting "enter" on tfInput invokes actionPerformed().

                lbl_Pass = new Label("Enter WLan Password: ");
                add(lbl_Pass);

                tf_Pass = new TextField("Pass",20);
                add(tf_Pass);

                lbl_SerDevices = new Label("Device to connect: ");
                add(lbl_SerDevices);

  //---------------Create combo box with available devices--------------------------------
  ch_SerDevices = new Choice();//This will list all available serial port devices
  SerialPort[] PortList = SerialPort.getCommPorts(); /retruns a list of all ports
                for(int i = 0; i < PortList.length; i++){
                        ch_SerDevices.add(PortList[i].getPortDescription());   //adds    each
serial devices description
                }
                add(ch_SerDevices);
                //-------------------------------------------------------------------------------------------



                btn_OK = new Button("OK");
                add(btn_OK);
```

```
        btn_OK.addActionListener(this);
        // "btn_OK" is the source object that fires an ActionEvent when clicked.
        // The source add "this" instance as an ActionEvent listener, which provides
        // an ActionEvent handler called actionPerformed().
        // Clicking "btn_OK" invokes actionPerformed().

        addWindowListener(this);
        // "super" Frame (source object) fires WindowEvent.
        // "super" Frame adds "this" object as a WindowEvent listener.

        lbl_Info = new Label("Press OK to establish connection");
        add(lbl_Info);

            setTitle("Wireless device connector");
            setSize(350, 200);
            setVisible(true);
    }

    /**
     * @param args
     */
    public static void main(String[] args) {
            // TODO Auto-generated method stub

            new GUI_main();
```

```
        }

        // ActionEvent handler - Called back upon button-click.
        @Override
        public void actionPerformed(ActionEvent evt) {
                //disable "ok" button for safety
                btn_OK.setEnabled(false);

                //let user know that the program is working.
                lbl_Info.setText("Configuring selected device");
                lbl_Info.setVisible(true);

                SerialPort SelectedDevice = SerialPort.getCommPorts()
[ch_SerDevices.getSelectedIndex()];     //get user selected device to communicate
                SelectedDevice.setBaudRate(9600);     //set baud rate (this could be
omitted since the default baud rate is 9600)
                SelectedDevice.openPort();                    //initialize the connection
to the port
                OutputStream outputStream = SelectedDevice.getOutputStream();

                //this part of the code outputs a string through the chosen port.--
                PrintStream writer = new PrintStream(outputStream);

                writer.print(tf_SSID.getText() + "\n");       //write SSID to device
                writer.print(tf_Pass.getText() + "\n");       //write pass to device
                //-----------------------------------------------------------------
```

```
                //receive response from device----------------------------------

        SelectedDevice.setComPortTimeouts(SerialPort.TIMEOUT_READ_BLOCKING,
1000, 0);       //waits 1 second for response.
                InputStream inStream = SelectedDevice.getInputStream();



                Scanner s = new Scanner(inStream, "UTF-8"); //scans  the  input  stream
and connects every byte into a string format
                s.useDelimiter("\\A");
                String response = s.hasNext() ? s.next(): "";

                /* ******* WARNING *******
                 * The proper response should be "ok" however even if the device
responds with "ok" response variable does not contain just "ok"
                 * using compareToIgnoreCase we created a temporary workaround but
this needs to be resolved.
                 */
            if(!response.isEmpty()){ //if response is ok then proceed to new device or
"wait"
                lbl_Info.setText("Device Response: " + response);
                        btn_OK.setEnabled(true);
            }
            else{                                                       //error
handling
```

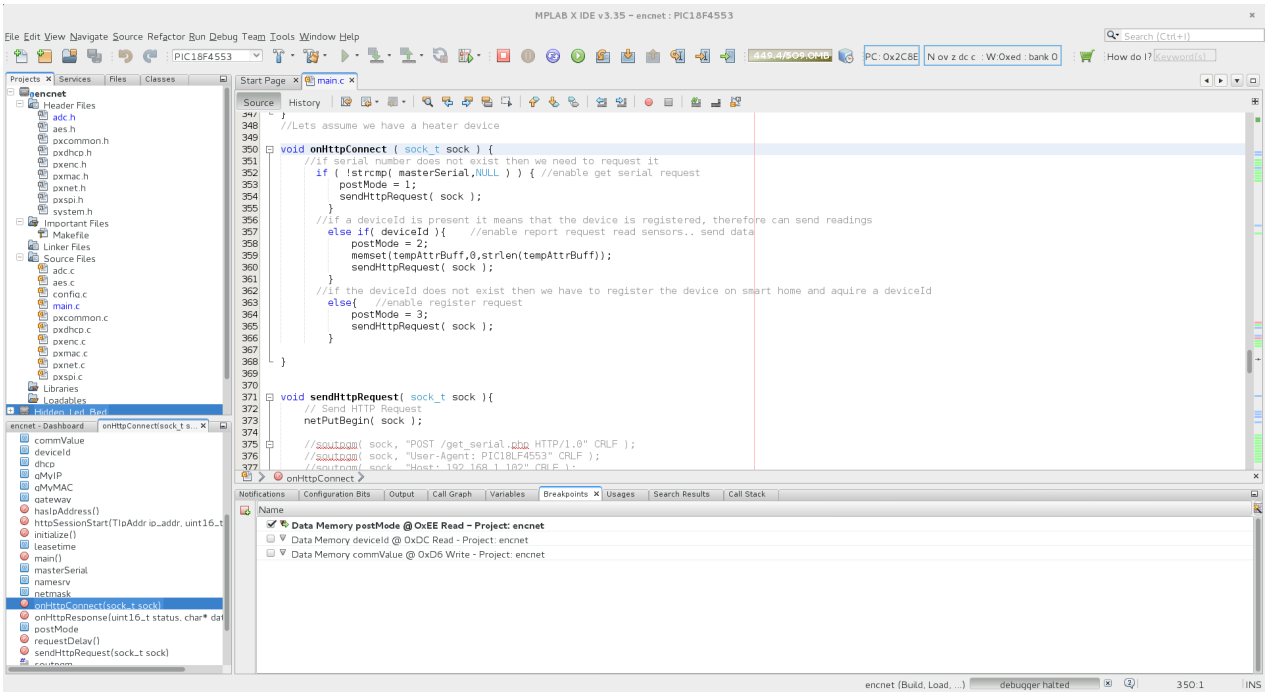
```
                lbl_Info.setText("Invalid Device or Device error");
                btn_OK.setEnabled(true);
            }
            s.close();   //free up variables
                //-----------------------------------------------------------------
                SelectedDevice.closePort();                    //free up the port
        }

        @Override
        public void windowClosing(WindowEvent evt) {
                System.exit(0);  // Terminate the program
        }
        // Not Used, BUT need to provide an empty body to compile.
        @Override public void windowOpened(WindowEvent evt) { }
        @Override public void windowClosed(WindowEvent evt) { }
        // For Debugging
        @Override public void windowIconified(WindowEvent evt)
{ System.out.println("Window Iconified"); }
        @Override public void windowDeiconified(WindowEvent evt)
{ System.out.println("Window Deiconified"); }
        @Override public void windowActivated(WindowEvent evt)
{ System.out.println("Window Activated"); }
        @Override public void windowDeactivated(WindowEvent evt)
{ System.out.println("Window Deactivated"); }
  }
```

7.2 Αποσφαλμάτωση, έλεγχος και χρήση συστήματος.

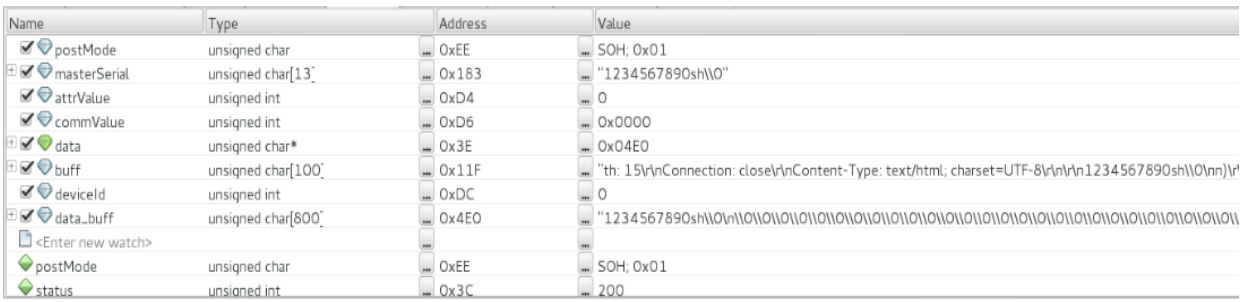
Επικοινωνία συσκευής – κεντρικού ελεγκτή.

Παρακάτω φαίνεται εικονογραφημένη η διαδικασία που περνάει μια καινούρια συσκευή στο δίκτυο. Έχουμε θέσει κάποια breakpoints για να γίνει πιο εύκολα η αποσφαλμάτωση.



Εικόνα 7.1: Breakpoints

Αρχικά η συσκευή κάνει post στο get-serial.php και λαμβάνει το σειριακό κωδικό από το Raspberry Pi (Master). Κρατάει τον αριθμό αυτό σε μια μεταβλητή “masterSerial” (2η εικονιζόμενη μεταβλητή στο παράθυρο Variables).

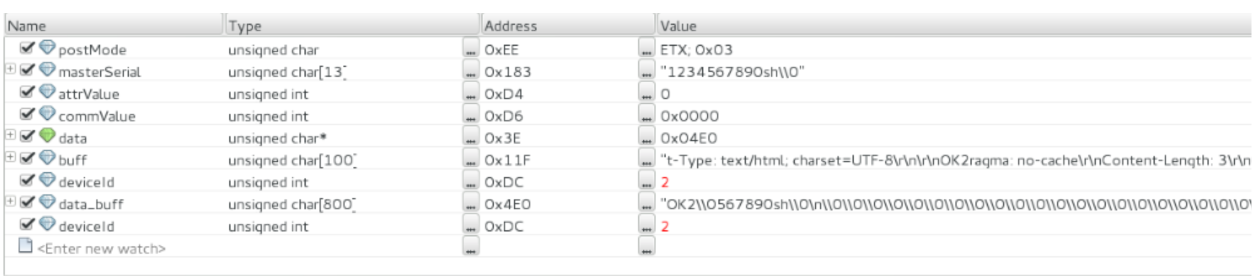


Name	Type	Address	Value
postMode	unsigned char	0xEE	SOH; 0x01
masterSerial	unsigned char[13]	0x183	"1234567890sh\\0"
attrValue	unsigned int	0xD4	0
commValue	unsigned int	0xD6	0x0000
data	unsigned char*	0x3E	0x04E0
buff	unsigned char[100]	0x11F	"th: 15\r\nConnection: close\r\nContent-Type: text/html; charset=UTF-8\r\n\r\n1234567890sh\\0\nn)\r
deviceId	unsigned int	0xDC	0
data_buff	unsigned char[800]	0x4E0	"1234567890sh\\0\n\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\
<Enter new watch>			
postMode	unsigned char	0xEE	SOH; 0x01
status	unsigned int	0x3C	200

Εικόνα 7.2: Σειριακός κωδικός

Ύστερα η συσκευή κάνει αίτημα στο register.php και στέλνει τα metadata της στον Master. Η απάντηση που λαμβάνει είναι OK και ακολούθως το ID που της δίνεται. Στην επόμενη εικόνα μπορούμε να παρατηρήσουμε το ID που έλαβε η συσκευή καθώς και την απάντηση του Master “OK2” (2 τελευταίες μεταβλητές στο παράθυρο Variables).

Το δεδομένο ID αποθηκεύεται στη μνήμη της συσκευής ως “deviceId”.

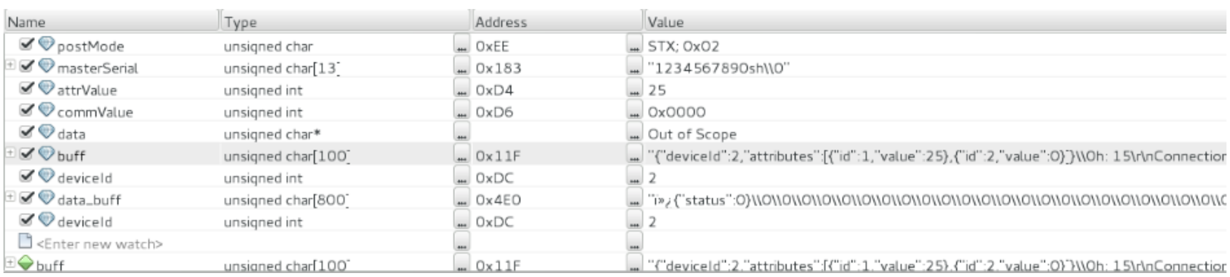


Name	Type	Address	Value
postMode	unsigned char	0xEE	ETX; 0x03
masterSerial	unsigned char[13]	0x183	"1234567890sh\\0"
attrValue	unsigned int	0xD4	0
commValue	unsigned int	0xD6	0x0000
data	unsigned char*	0x3E	0x04E0
buff	unsigned char[100]	0x11F	"t-Type: text/html; charset=UTF-8\r\n\r\nOK2ragma: no-cache\r\nContent-Length: 3\r\n
deviceId	unsigned int	0xDC	2
data_buff	unsigned char[800]	0x4E0	"OK2\\0567890sh\\0\n\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0
deviceId	unsigned int	0xDC	2
<Enter new watch>			

Εικόνα 7.3: Αριθμός ταυτοποίησης συσκευής.

Αφού έχει γίνει η εγγραφή στο δίκτυο, η συσκευή είναι έτοιμη να χρησιμοποιηθεί. Αρχικά η συσκευή παίρνει μετρήσεις από το περιβάλλον ή από τον εαυτό της ανάλογα το είδος συσκευής, εδώ μετράμε θερμοκρασία περιβάλλοντως και ενεργοποιούμε ένα ανεμιστήρα.

Στο επόμενο βήμα η συσκευή εισάγει τα δεδομένα των αισθητήρων σε ένα κείμενο για να αποσταλεί στον Master. Μπορούμε να διακρίνουμε αυτό το κείμενο στη μεταβλητή buff (σημειωμένη μεταβλητή).



Name	Type	Address	Value
postMode	unsigned char	0xEE	STX; 0x02
masterSerial	unsigned char[13	0x183	"1234567890sh\\0"
attrValue	unsigned int	0xD4	25
commValue	unsigned int	0xD6	0x0000
data	unsigned char*		Out of Scope
buff	unsigned char[100	0x11F	"{"deviceId":2,"attributes":[{"id":1,"value":25},{"id":2,"value":0}]}\\0h: 15\r\nConnection
deviceId	unsigned int	0xDC	2
data_buff	unsigned char[800	0x4E0	"ï»¿{"status":0}\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\
deviceId	unsigned int	0xDC	2
<Enter new watch>			
buff	unsigned char[100	0x11F	"{"deviceId":2,"attributes":[{"id":1,"value":25},{"id":2,"value":0}]}\\0h: 15\r\nConnection

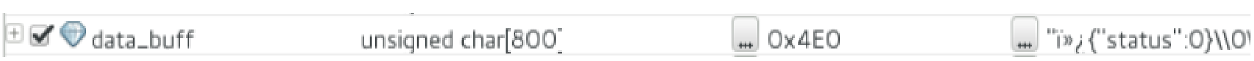
Εικόνα 7.4: Αποστολή μετρήσεων.

Σαν απάντηση η συσκευή θα λάβει πίσω ένα χαρακτηριστικό status και πιθανές εντολές που έχουν προκύψει από τον χρήστη.

data_buff	unsigned char[800	0x4E0	"ï»¿{"status":0,"commands":[{"id":2,"value":"80"}]}\\0

Εικόνα 7.5: Μεταβλητή δεδομένων με εντολές.

Σε περίπτωση που δεν υπάρχουν εντολές, η απάντηση περιέχει μόνο το status.

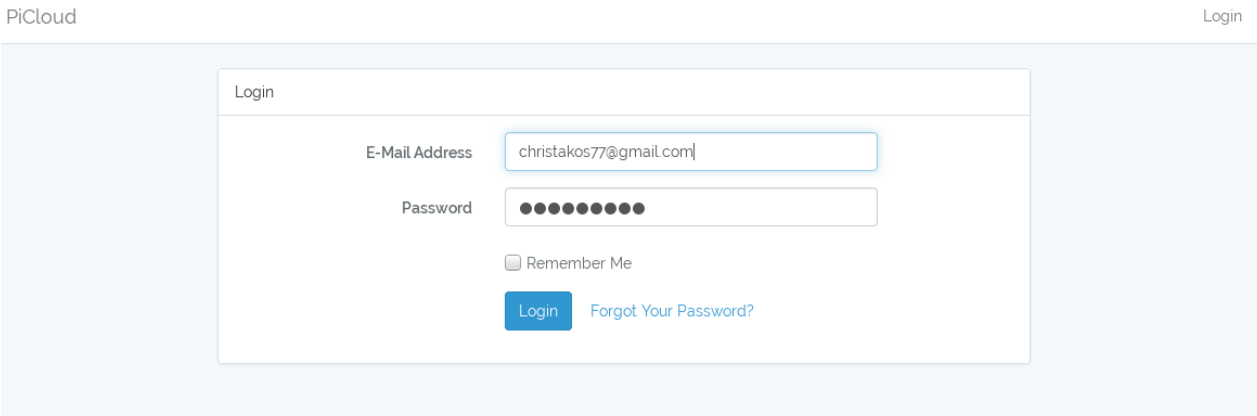


data_buff	unsigned char[800	0x4E0	"ï»¿{"status":0}\\0

Εικόνα 7.6: Μεταβλητή δεδομένων χωρίς εντολές.

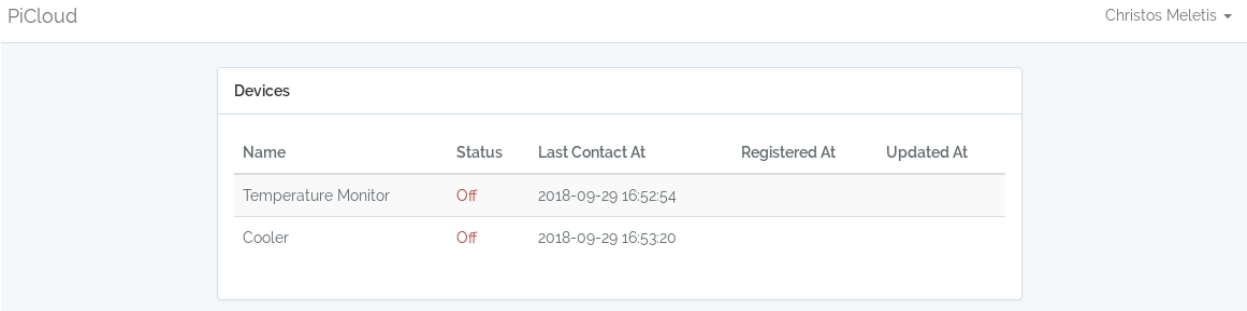
Διεπαφή χρήστη.

Παράλληλα με τις διεργασίες των συσκευών ο χρήστης μπορεί να εγγραφεί στο σύστημα με τα στοιχεία του. Να δει την κατάσταση των συσκευών που έχει στο δίκτυό του και να τους δώσει εντολές.



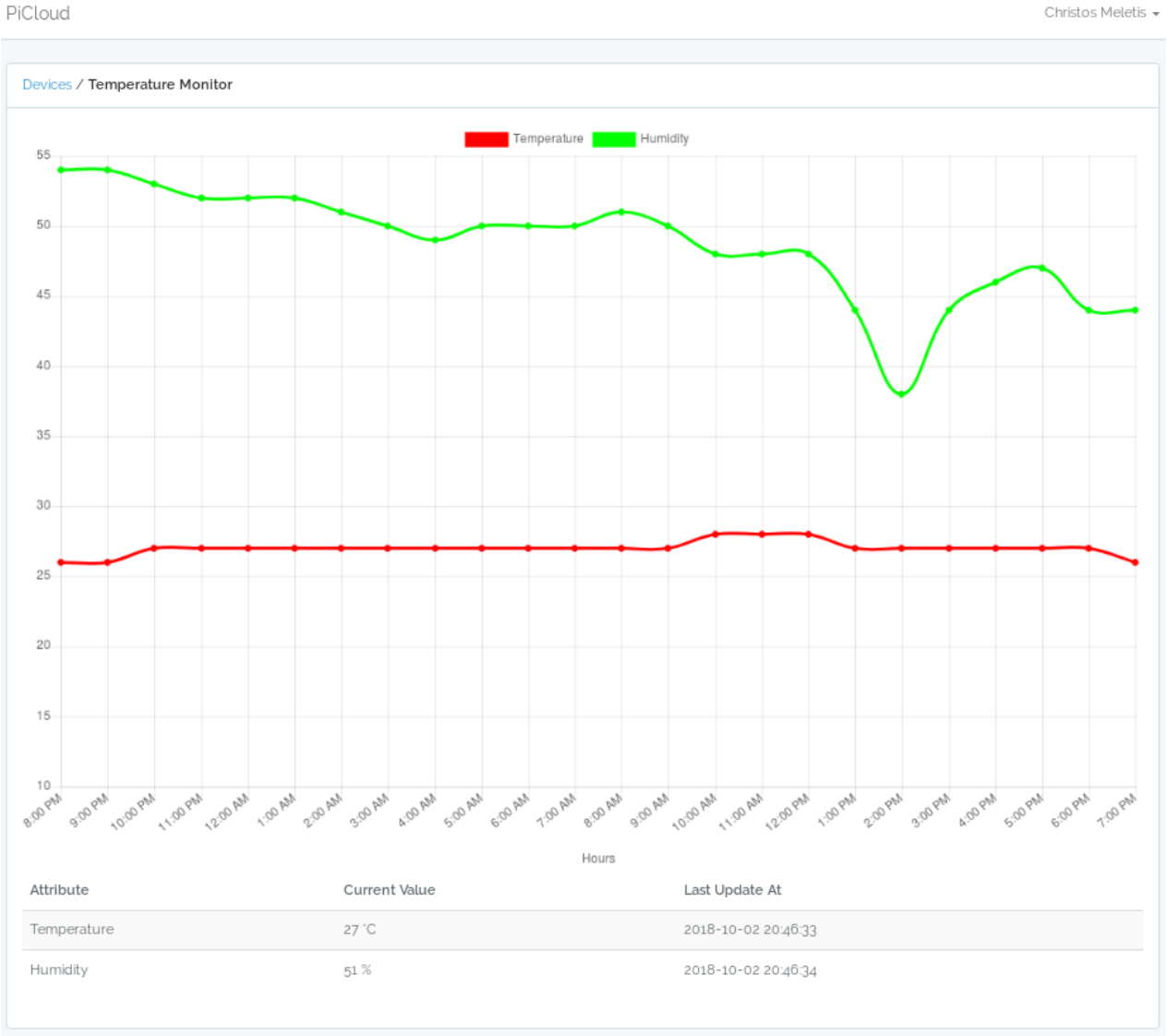
Εικόνα 7.7: Φόρμα σύνδεσης χρήστη.

Στην αρχική καρτέλα αμέσως μετά τη σύνδεση, δημιουργείται μια λίστα με τις διαθέσιμες συσκευές του σπιτιού. Κάθε συσκευή φαίνεται με το όνομά της. Εδώ ο χρήστης μπορεί να δει ποιες συσκευές είναι ενεργές και πότε έγινε η τελευταία επικοινωνία με κάθε συσκευή.



Εικόνα 7.8: Λίστα συσκευών.

Τέλος, επιλέγοντας μια συσκευή από τη λίστα, ο χρήστης παραπέμπεται στη καρτέλα ελέγχου της συσκευής. Εδώ ο χρήστης μπορεί να δει γραφήματα λειτουργίας της συσκευής καθώς και να την ελέγξει.

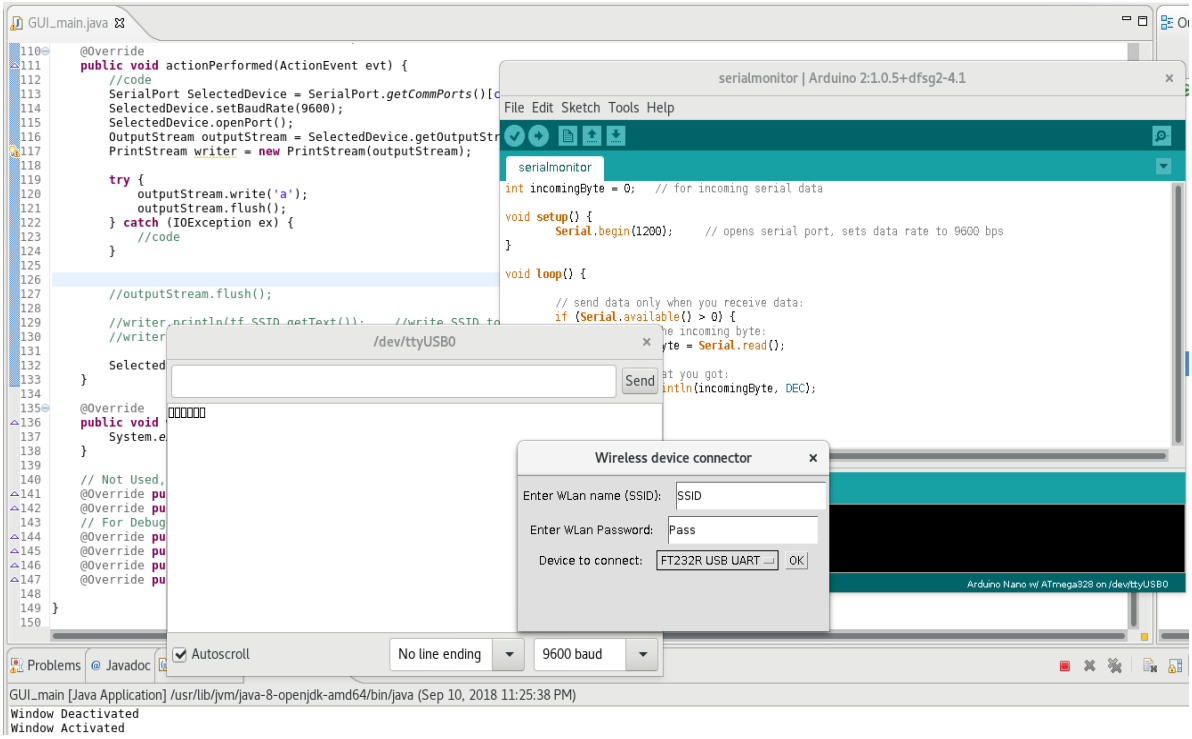


Εικόνα 7.9:Διάγραμμα συσκευής.

Σύνδεση ασύρματης συσκευής στο δίκτυο.

Σε περίπτωση που πρέπει να συνδεθεί στο δίκτυο του έξυπνου σπιτιού μια συσκευή με ασύρματες δυνατότητες, θα πρέπει να είναι γνωστά σε αυτήν τα διακριτικά σύνδεσης

στον ασύρματο δρομολογητή του χρήστη. Η ανταλλαγή των διακριτικών επιτυγχάνεται με το πρόγραμμα “Σύνδεσης Ασύρματης Συσκευής” (Wireless Device Connector).



Εικόνα 7.10: Ανταλλαγή στοιχείων ασύρματου δρομολογητή.

Η συγκεκριμένη εικόνα εξήχθει κατά τη διάρκεια αποσφαλμάτωσης. Στο πίσω μέρος φαίνονται κομμάτια από τους κώδικες που τρέχει ο προσωπικός υπολογιστής του χρήστη (αριστερά πίσω) και κομμάτι του κώδικα που τρέχει η συσκευή (δεξιά πίσω). Ύστερα μπορούμε να διακρίνουμε τη γραφική διεπαφή χρήστη, στην οποία δίνονται τα στοιχεία του ασύρματου δρομολογητή και επιλέγεται η συσκευή που πρέπει να συνδεθεί στο δίκτυο του έξυπνου σπιτιού. Ακριβώς αριστερά αυτού βρίσκονται τα δεδομένα που έλαβε η συσκευή. Τα δεδομένα έχουν τη μορφή “κουτιών” και είναι λανθασμένα, το πρόβλημα αυτό προέκυψε από μη ταυτόσημες ταχύτητες σειριακής επικοινωνίας μεταξύ του προσωπικού υπολογιστή του χρήστη και της ασύρματης συσκευής.

ΠΑΡΑΡΤΗΜΑΤΑ

Αφέντης(Master): Ο ορισμός αυτός συνήθως αποδίδεται σε συσκευές ή προγράμματα που παίζουν το ρόλο του εντολέα σε ένα δίκτυο. Συνήθως αυτές οι

συσκευές είναι υπεύθυνες για τον έλεγχο άλλων συσκευών ή προγραμμάτων, την αποστολή εντολών σε αυτά και τη διαχείριση λαθών που προκύπτουν στο υποδίκτυο, του εκάστοτε “Master”.

Σκλάβος(Slave): Στην επιστήμη των υπολογιστών, σκλάβο συνήθως ονομάζουμε μια συσκευή ή ένα πρόγραμμα που εκτελέι κάποιες λειτουργίες τις οποίες όμως επιβλέπει και ελέγχει ένα άλλο σύστημα.

Αποσφαλμάτωση: Ορίζεται η διαδικασία κατά την οποία οι δημιουργοί ενός προγράμματος ή συστήματος, ελέγχουν τη διαδικασία ορθής λειτουργίας του αντίστοιχου προγράμματος/συστήματος για κάθε πιθανό αποτέλεσμα.

Διεπαφή Χρήστη (web interface): Είναι το πρόγραμμα το οποίο αναλαμβάνει την επικοινωνία μεταξύ του χρήστη και του συστήματος. Λειτουργεί σαν μεσάζοντας καθιστόντας ευκολότερη και ασφαλέστερη τη χρήση ενός συστήματος.

Φυλλομετρητής (web browser): Είναι το πρόγραμμα με το οποίο ένας χρήστης μπορεί να έχει πρόσβαση στις διάφορες σελίδες του διαδικτύου.

Διεύθυνση IP (Identification Protocol): Αποτελείται από μια σειρά τεσσάρων, τριψήφιων αριθμών. Χρησιμοποιείται για την αναγνώριση των διάφορων συσκευών σε ένα δίκτυο.

Δρομολογητής (Router): Συσκευή υπεύθυνη για τη διασύνδεση τοπικών συσκευών στο διαδίκτυο.

Front-End: Κλάδος της επιστήμης των υπολογιστών που ασχολείται με τη μπροστινή όψη των συστημάτων. Τέτοια συστήματα μπορεί να είναι, μια ιστοσελίδα, μια γραφική διεπαφή χρήστη, μια εφαρμογή.

Framework: Δομή προγράμματος γενικής πρακτικότητας, που μπορεί να αλλάξει με επιπρόσθετο κώδικα από τον χρήστη για την εξατομίκευση των εφαρμογών.

Pull-Up αντίσταση: Μια αντίσταση όπου ο ένας εκ των δυο ακροδεκτών της είναι

συνδεδεμένος στην τροφοδοσία του συστήματος.

Alexa & Amazon Echo: Συστήματα πολαπλών χρήσεων με αναγνώριση φωνητικών εντολών.

ΒΙΒΛΙΟΓΡΑΦΙΑ

Η ιστορία των έξυπνων σπιτιών - IoT Evolution
http://www.iotevolutionworld.com/m2m/articles/376816-history-smart-homes.html
Ιστορία του Οικιακού Αυτοματισμού - The Ambient
https://www.the-ambient.com/features/visions-through-the-ages-history-of-home-automation-178
Home Plug - Wikipedia