ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

Σχεδιασμός και Υλοποίηση Ασύρματου Μετρητικού Συστήματος Ενέργειας

Εφαρμογή σε Κατανεμημένα Συστήματα ΑΠΕ Δήμων

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Αποστολάκη Ευγενία

Επιβλέπων: Συκάς Ευστάθιος
Καθηγητής ΕΜΠ

Αθήνα, Οκτώβριος 2015

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

Σχεδιασμός και Υλοποίηση Ασύρματου Μετρητικού Συστήματος Ενέργειας

Εφαρμογή σε Κατανεμημένα Συστήματα ΑΠΕ Δήμων

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Αποστολάκη Ευγενία

Επιβλέπων: Συκάς Ευστάθιος
Καθηγητής ΕΜΠ

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 4η Νοεμβρίου 2015

.....	.....	.....
Συκάς Ευστάθιος	Θεολόγου Μιχαήλ	Στασινόπουλος Γεώργιος
Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.

Αθήνα, Οκτώβριος 2015

.....

ΑΠΟΣΤΟΛΑΚΗ ΕΥΓΕΝΙΑ

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

Περίληψη

Ο στόχος της Διπλωματικής Εργασίας είναι η διερεύνηση μιας πλήρους ασύρματης λύσης η οποία θα απευθύνεται σε δήμους που επιθυμούν να εγκαταστήσουν ή έχουν ήδη εγκατεστημένα φωτοβολταϊκά πάνελ (Φ/Β) σε διάφορα σημεία (π.χ., σε κολώνες φωτισμού). Η εν λόγω λύση επιτρέπει τη συλλογή, την αποθήκευση, την επεξεργασία και την παρουσίαση μέσω γραφικού περιβάλλοντος μετρήσεων που αφορούν στην πλεονάζουσα παραγόμενη ενέργεια.

Αρχικά, διερευνήθηκαν διαφορετικές ασύρματες, μικρής εμβέλειας, τεχνολογίες και αξιολογήθηκε η επίδοσή τους αναφορικά με τη συχνότητα λειτουργίας, την απόσταση αποστολής δεδομένων και την ταχύτητα μεταφοράς δεδομένων. Στη συνέχεια, σχεδιάστηκε και υλοποιήθηκε ένας κόμβος (“measurement kit”) μέτρησης και αποστολής της παραγόμενης/καταναλισκόμενης ενέργειας από τα Φ/Β, σε πραγματικό χρόνο, με ασύρματο τρόπο. Επίσης, υλοποιήθηκε μια πύλη (“gateway”) η οποία θα λαμβάνει τα δεδομένα από τα “measurement kits”, θα τα επεξεργάζεται και θα τα αποστέλλει στο cloud. Τέλος, αναπτύχθηκε ένα γραφικό περιβάλλον, μέσω του οποίου ο χρήστης θα μπορεί να ενημερώνεται σχετικά με τις τρέχουσες μετρήσεις ενέργειας αλλά και να έχει πρόσβαση σε στατιστικά/ιστορικά δεδομένα ενέργειας.

Η εργασία πραγματοποιήθηκε σε συνεργασία με την COSMOTE.

Λέξεις κλειδιά: Arduino Uno, Arduino Nano, Raspberry Pi 2, ασύρματο δίκτυο αισθητήρων, διαδίκτυο των αντικειμένων, γραφικό περιβάλλον, MQTT, RFM69W, nRF24L01+, DHT11, μετρητής ισχύος, μετρητής φωτεινότητας, συλλογή δεδομένων, δίκτυο RF.

Abstract

The scope of this thesis was the investigation of a complete wireless solution addressed to municipalities that wish to install or have already installed photovoltaic panels (PVs) in various spots (e.g. lamp posts). The proposed solution allows for the collection, storage, process and presentation through a graphical interface of the excess energy measurements.

First, various wireless, short range, technologies were investigated and their performance was assessed concerning the frequency, the distance and the bitrates than can be achieved. Then, a “measurement kit” was designed in order to transmit the produced/consumed energy from the PVs to a “gateway” wirelessly in real time. Moreover, a “gateway” was implemented, which receives the data from the “measurement kits”, processes them and uploads them to the cloud. Finally, a graphical user interface (WebGUI) was developed, through which the user can be informed about the current energy measurements and access to statistics/historical energy-related data.

The thesis was conducted in collaboration with COSMOTE.

Key words: Arduino Uno, Arduino Nano, Raspberry Pi 2, wireless sensor network, Internet of things, graphical user interface, MQTT, RFM69W, nRF24L01+, DHT11, power measurement, photocell, data collection, RF network.

Ευχαριστίες

Θα ήθελα να εκφράσω τις θερμές μου ευχαριστίες στον κ. Γιώργο Λυμπερόπουλο, Υποδιευθυντή Εξέλιξης Δικτύου Σταθερής και Κινητής της COSMOTE, για τις υποδείξεις του και την καθοδήγησή του σε όλο το διάστημα εκπόνησης της διπλωματικής μου εργασίας.

Ακόμη, θα ήθελα να ευχαριστήσω επίσης τον καθηγητή της σχολής Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών κ. Ευστάθιο Συκά για τη δυνατότητα που μου προσέφερε να ασχοληθώ με ένα τόσο ενδιαφέρον αντικείμενο.

Τέλος, θα ήθελα να ευχαριστήσω την οικογένειά μου για την αμέριστη υποστήριξη τους καθ' όλη τη διάρκεια των σπουδών μου.

Περιεχόμενα

Κεφάλαιο 1. Εισαγωγή

Η βιώσιμη παραγωγή ενέργειας γίνεται ολοένα και πιο επιτακτική σε μια κοινωνία που έχει ανάγκη τη συνεχή ροή της, χωρίς τις αρνητικές συνέπειες της παραγωγής της μέσω συμβατικών μέσων που επιβαρύνουν το περιβάλλον. Οι δήμοι, ως δημόσιοι φορείς, μπορούν να εγκαταστήσουν συστήματα ανανεώσιμων πηγών ενέργειας στην περιοχή τους συνεισφέροντας με αυτό τον τρόπο είτε στη μείωση των λειτουργικών τους εξόδων είτε στον περιορισμό τους, και μπορούν έτσι να εκμεταλλευτούν τα ανταποδοτικά οφέλη για κοινοφελείς σκοπούς, προβάλλοντας παράλληλα ένα οικολογικό πρόσωπο. Η φύση αυτών των συστημάτων τα θέλει διεσπαρμένα στα διαθέσιμα σημεία του εκάστοτε δήμου. Η τεχνολογία των ασύρματων δικτύων αισθητήρων μπορεί να δώσει λύση στο πρόβλημα παρακολούθησης τέτοιου είδους συστημάτων, σε ό,τι αφορά την παραγωγή ενέργειας και όχι μόνο.

Η διείσδυση των ανανεώσιμων πηγών ενέργειας στην Ελλάδα

Με τον όρο ανανεώσιμες ή ήπιες μορφές ενέργειας αναφερόμαστε σε μορφές εκμεταλλεύσιμης ενέργειας που προέρχονται από διάφορες φυσικές διαδικασίες όπως ο άνεμος, η γεωθερμία η κυκλοφορία του νερού και άλλες. Ο όρος ήπιες χρησιμοποιείται για 2 λόγους. Πρώτον επειδή για την εκμετάλλευση τους δεν απαιτείται κάποια ενεργητική παρέμβαση όπως για παράδειγμα εξόρυξη, άντληση ή καύση αλλά απλώς η εκμετάλλευση της ήδη υπάρχουσας ροής ενέργειας στη φύση. Δεύτερον, αναφερόμαστε σε «καθαρές» μορφές, φιλικές προς το περιβάλλον, οι οποίες δεν αποδεσμεύουν υδρογονάνθρακες, διοξείδιο του άνθρακα ή τοξικά και ραδιενεργά απόβλητα, όπως ακριβώς συμβαίνει με τις υπόλοιπες πηγές ενέργειας που χρησιμοποιούνται σε μεγάλη κλίμακα.

Ως ανανεώσιμες πηγές ενέργειας θεωρούνται γενικά οι εναλλακτικές των παραδοσιακών πηγών ενέργειας όπως η ηλιακή και η αιολική και μπορούν να θεωρηθούν η απαρχή για την επίλυση των οικολογικών προβλημάτων που αντιμετωπίζει σήμερα η γη. Μέχρι το τέλος του 2012 το 19% της παγκόσμιας κατανάλωσης ενέργειας προήλθε από ανανεώσιμες πηγές. [1]

Με τον όρο ηλιακή ενέργεια χαρακτηρίζεται το σύνολο των διαφόρων μορφών ενέργειας που προέρχονται από τον ήλιο. Τέτοιες είναι το φως, η θερμότητα όπως και διάφορες ακτινοβολίες. Είναι η μεγαλύτερη ενεργειακή είσοδος πάνω στην γη, η οποία όμως μπορεί να συνεχιστεί για δισεκατομμύρια χρόνια στο μέλλον με τον ίδιο ρυθμό. Τα φωτοβολταϊκά συστήματα εκμεταλλεύονται το φωτοβολταϊκό φαινόμενο που συνίσταται στη μετατροπή μέρους του ορατού φάσματος της ηλιακής ακτινοβολίας σε ηλιακή ενέργεια.

Παρακάτω παρουσιάζονται συνοπτικά τα εξής πλεονεκτήματα των Φ/Β συστημάτων [2]:

- Η ηλιακή ενεργεία είναι ανεξάντλητη ενεργειακή πηγή, με μηδενικό κόστος και εύκολη διαθεσιμότητα.
- Απευθείας μετατροπή της ηλιακής ενέργειας σε ηλεκτρική χωρίς θερμικό ή μηχανικό ενδιάμεσο στάδιο.
- Αποτελούν τεχνολογία φιλική προς το περιβάλλον καθώς δεν το μολύνουν, λειτουργούν αθόρυβα και εφαρμόζοντας κατάλληλο αρχιτεκτονικό σχεδιασμό δεν αλλοιώνουν αισθητικά το περιβάλλον.
- Επεκτείνονται πολύ εύκολα προκειμένου να καλύψουν αυξημένες μελλοντικές απαιτήσεις του φορτίου.
- Είναι πολύ αξιόπιστα με διάρκεια ζωής πάνω από 25 χρόνια.
- Μικρές ανάγκες συντήρησης καθώς δεν έχουν κινούμενα μέρη και επομένως δεν χρειάζονται συνεχή παρακολούθηση.
- Αποτελούν ιδανική λύση για την κάλυψη μικρών φορτίων (μέχρι και 10 kW) που απαιτούνται σε απομακρυσμένες περιοχές. Σε τέτοιες εφαρμογές η κάλυψη του φορτίου από το φωτοβολταϊκό σύστημα είναι συνήθως οικονομικότερη από την επέκταση του δικτύου διανομής ή την εγκατάσταση μιας ηλεκτρογεννήτριας.

Αντίθετα ως μειονεκτήματα θα μπορούσαν να αναφερθούν τα εξής:

- Υψηλό κόστος εγκατάστασης[1]
- Η ανάγκη ύπαρξης συσκευών αποθήκευσης ενέργειας για την κάλυψη του φορτίου όταν δεν υπάρχει ηλιακή ακτινοβολία και αλλά και την εξισορρόπηση των εποχιακών διαφορών της ηλιακής ενέργειας.
- Παράγουν συνεχή τάση η οποία πρέπει να μετατραπεί σε εναλλασσόμενη (με τη χρήση αντιστροφέα). Αυτό έχει ως αποτέλεσμα την απώλεια ενέργειας κατά 4-12%.
- Δεν φαίνονται ικανά να λύσουν το ενεργειακό πρόβλημα των ανεπτυγμένων χωρών από μόνα τους, αναμένεται όμως ότι θα διαδραματίζουν δευτερεύοντα ρόλο στα δίκτυα που θα βασίζονται σε άλλες μορφές ενέργειας.

Ο τομέας της αξιοποίησης της ηλιακής ενέργειας στην Ελλάδα μετράει ήδη 3 δεκαετίες δραστηριότητας. Η έναρξη του ως παραγωγικός κλάδος εντοπίζεται στο 1978 . Ως και το 1987 η ανάπτυξη της αγοράς είναι ραγδαία ενώ στην συνέχεια σταθεροποιείται (2001) ως αποτέλεσμα της μείωσης των τιμών του πετρελαίου και γενικότερα της παραμονής των τιμών του ηλεκτρικού ρεύματος σε χαμηλά επίπεδα. Βασικός στόχος της Ευρωπαϊκής Ένωσης είναι μέχρι το 2020 το 20% της κατανάλωσης ενέργειας να προέρχεται από ανανεώσιμες πηγές. Σύμφωνα με τα στοιχεία του 2014, η Ελλάδα κατατάσσεται στη τέταρτη θέση παγκοσμίως στην εγκατεστημένη φωτοβολταϊκή ισχύ κατά κεφαλή μετά τη Γερμανία, την Ιταλία και το Βέλγιο και παράγει πλέον το 7,6% του ηλεκτρισμού της από φωτοβολταϊκά, τη στιγμή που σε παγκόσμιο επίπεδο το ποσοστό διείσδυσης δεν ξεπερνά το 1%. [3]

[1] Μια ενδεικτική τιμή είναι 2700 ευρώ ανά εγκατεστημένο Κιλοβάτ (kW) ηλεκτρικής ισχύος.

Αυτή η διείσδυση ενισχύει και την εγχώρια προστιθέμενη αξία, καθώς στην Ελλάδα παράγονται [4]:

- Φωτοβολταϊκά πλαίσια
- Βάσεις στήριξης (σταθερές ή κινούμενες)
- Μετασχηματιστές
- Καλώδια
- Πίνακες και άλλο ηλεκτρολογικό υλικό
- Επικουρικός εξοπλισμός φωτοβολταϊκών σταθμών (οικίσκοι, υλικό περιφράξεων, ιστοί, σωληνώσεις, κλπ)
- Λογισμικό τηλεμετρίας και εξοπλισμός τηλεπικοινωνιών

Δεδομένης, λοιπόν, της μέχρι τώρα διείσδυσης των ΑΠΕ, τις προοπτικές ανάπτυξής τους και τη στοχαστικότητα των φυσικών φαινομένων, η παρακολούθηση και ο έλεγχος της παραγωγής είναι απαραίτητος για την καλύτερη αξιοποίηση και πρόβλεψή της.

Ασύρματα δίκτυα αισθητήρων

Τα "Ασύρματα Δίκτυα Αισθητήρων" αποτελούν απόγονο των "Ασύρματων Ad-Hoc Δικτύων" και στην εποχή μας έχουν βρει πληθώρα εφαρμογών στο πλαίσιο Συστημάτων Παρακολούθησης και Εποπτείας, Οικιακών και Βιομηχανικών Αυτοματισμών αλλά και Μηχανισμών Άμεσης Επέμβασης. [5]

Τα Ασύρματα Δίκτυα Αισθητήρων (Wireless Sensor Networks) είναι ολοκληρωμένα συστήματα συλλογής, μετάδοσης και επεξεργασίας πληροφοριών. Βασικό χαρακτηριστικό τους γνώρισμα είναι ότι τα δομικά τους στοιχεία είναι λειτουργικά ανεξάρτητοι και ενεργειακά αυτόνομοι αισθητήρες, οι οποίοι τοποθετούνται (διασπείρονται) σε μια γεωγραφική περιοχή (περιορισμένης ή ευρείας έκτασης) με στόχο να μεταδώσουν πληροφορίες προς μια Κεντρική Μονάδα Επεξεργασίας. Ο σχεδιασμός ενός Ασύρματου Δικτύου Αισθητήρων στηρίζεται στις δύο βασικές αρχές λειτουργίας του: τη δυνατότητα για περιοδική λήψη δεδομένων και τη δυνατότητα για εντοπισμό συμβάντων που χρίζουν άμεσης αντίδρασης/ αντιμετώπισης.

Η σημαντικότητα και η προτεραιοποίηση των αρχών αυτών καθορίζεται από το είδος και τις ιδιαίτερες ανάγκες της εκάστοτε εφαρμογής. Όταν η εφαρμογή απαιτεί την περιοδική λήψη δεδομένων, πιθανές έκτακτες καθυστερήσεις στην αποστολή των στοιχείων δεν παίζουν σημαντικό ρόλο, άρα το δίκτυο μπορεί να σχεδιαστεί με προτεραιότητα την αύξηση του χρόνου ζωής. Όταν η εφαρμογή είναι προσανατολισμένη στον εντοπισμό επειγόντων συμβάντων, το δίκτυο οφείλει να στοχεύει στην αποστολή δεδομένων αμέσως χωρίς να λαμβάνει σημαντικά υπόψη το κόστος λειτουργίας.

Παρακάτω ακολουθούν τα πιο σημαντικά χαρακτηριστικά γνωρίσματα των Ασύρματων Δικτύων Αισθητήρων (τα οποία ταυτοχρόνως αποτελούν και τα συγκριτικά τους πλεονεκτήματα απέναντι στα Ad-Hoc Δίκτυα): [6]

- **Προσανατολισμός στην Εφαρμογή (Application Specific):** Τα Ασύρματα Δίκτυα Αισθητήρων σχεδιάζονται με βάση τις εξειδικευμένες ανάγκες και απαιτήσεις της εκάστοτε εφαρμογής. Σπανίως ένα Δίκτυο το οποίο έχει υλοποιηθεί για μία συγκεκριμένη εφαρμογή μπορεί να χρησιμοποιηθεί (χωρίς προσαρμογές) σε κάποια άλλη εφαρμογή. Για παράδειγμα, ένα Δίκτυο από αισθητήρες που έχουν ενταχθεί στην κεντρική θέρμανση ενός κτιρίου ρυθμίζοντας τη μέγιστη επιμέρους θερμοκρασία των ορόφων, στηρίζεται σε εντελώς διαφορετικές αρχές σχεδιασμού από ένα στατικό Δίκτυο αισθητήρων που μετρούν το ύψος βροχόπτωσης μιας γεωγραφικής περιοχής.
- **Κλίμακα Μεγέθους:** Τα Ασύρματα Δίκτυα Αισθητήρων μπορούν να είναι όσο εκτενή ή όσο περιορισμένα απαιτεί η εφαρμογή, αποτελούμενα από μερικές δεκάδες έως και αρκετές εκατοντάδες αισθητήρες. Τόσο η αρχιτεκτονική όσο και τα δικτυακά πρωτόκολλα θα πρέπει να είναι σε θέση να διαχειριστούν τέτοια μεγέθη
- **Αυτορρύθμιση (Self-Configurability) & Ανοχή Σφαλμάτων (Fault Tolerance):** Τα Δίκτυα λειτουργούν με αλγορίθμους ικανούς να ρυθμίσουν την τοπολογία τους τη στιγμή της εγκατάστασης αλλά και να διαχειριστούν τυχόν μελλοντικές ανάγκες για μεταβολές σε αυτή: χρησιμοποιούν αλγορίθμους ικανούς να αντιμετωπίσουν το φαινόμενο της απώλειας αισθητήρων (π.χ. λόγω φθοράς ή και κλοπής) με τρόπο ώστε να μη διαταράσσεται η συνολική σταθερότητα λειτουργίας τους. Η συνολική κατάσταση του Δικτύου ελέγχεται ανά τακτά χρονικά διαστήματα.
- **Χρόνος Ζωής:** Όταν πρόκειται για Δίκτυα αποτελούμενα από αισθητήρες οι οποίοι τροφοδοτούνται από ηλεκτρικά στοιχεία (π.χ. μπαταρίες), ο χρόνος ζωής τους είναι άμεσα συνυφασμένος με το χρόνο ζωής των πηγών ενέργειας των Αισθητήρων. Σημαντικό ρόλο στη διατήρηση της βιωσιμότητας ενός Δικτύου για μεγάλο χρονικό διάστημα παίζει ο προσεκτικός σχεδιασμός γύρω από τη διαχείριση της ενέργειας η οποία απαιτείται για τη λειτουργία του κάθε αισθητήρα. Η σχέση ανάμεσα στην ποιότητα λειτουργίας και την απαιτούμενη τροφοδοσία των αισθητήρων είναι αντιστρόφως ανάλογη: ξοδεύοντας περισσότερη ενέργεια, επιτυγχάνουμε μεν καλύτερη απόδοση αλλά το Δίκτυο έχει μικρότερο συνολικό χρόνο ζωής. Ακριβώς επειδή τα Δίκτυα αυτά σχεδιάζονται με τρόπο ώστε να εξυπηρετήσουν συγκεκριμένες ανάγκες, ο τρόπος εξισορρόπησης της σχέσης ποιότητας-ενέργειας εξαρτάται από το είδος της εφαρμογής.
- **Αυτόνομη Λειτουργία & Προγραμματισμός:** Κάθε ένας από τους αισθητήρες πρέπει να είναι σε θέση να λάβει αποφάσεις για τη διατήρηση της εύρυθμης λειτουργίας του Δικτύου (και) χωρίς την παρέμβαση του χρήστη, (π.χ. αυξομειώνοντας τη συχνότητα δειγματοληψίας) αλλά και να δεχθεί επαναπρογραμματισμό (π.χ. σε περίπτωση που αλλάξει η στρατηγική χρήσης του Δικτύου).
- **Απλότητα Σχεδιασμού:** Με δεδομένο ότι οι αισθητήρες διαθέτουν περιορισμένους πόρους λειτουργίας, δεν υπάρχει η δυνατότητα να υποστηριχθεί ιδιαιτέρως υψηλή πολυπλοκότητα, ούτε σε επίπεδο λειτουργικού συστήματος ούτε σε επίπεδο αλγορίθμου λειτουργίας του Δικτύου.
- **Ποιότητα Υπηρεσιών (Quality of Service):** Σε τέτοια Δίκτυα η έννοια της ποιότητας υπηρεσιών μπορεί να διαφέρει σημαντικά από τα συνήθη πρότυπα των “Ad Hoc” Δικτύων. Για παράδειγμα, σε ένα Ασύρματο Δίκτυο αισθητήρων, θα μπορούσε να

μην παίζει τόσο σημαντικό ρόλο η ταχύτητα μετάδοσης των δεδομένων όσο η αξιόπιστη μετάδοση του συνόλου χωρίς απώλειες, αποφεύγοντας τις επανεκπομπές.

Για να καλυφθούν οι παραπάνω απαιτήσεις έχουν αναπτυχθεί νέοι τρόποι ασύρματης επικοινωνίας μεταξύ των δομικών στοιχείων (κόμβων) του Δικτύου. Οι κυριότεροι από αυτούς είναι: [7]

- **Επικοινωνία Πολλαπλών Βημάτων (Multi-Hop):** Οι ενδιάμεσοι κόμβοι του Δικτύου λειτουργούν ως «προωθητές» μηνυμάτων. Ένα τέτοιο είδος επικοινωνίας είναι χρήσιμο όταν η απευθείας σύνδεση δύο κόμβων είναι ανέφικτη (ή κρίνεται ασύμφορη, λόγω του υψηλού κόστους της ενέργειας μετάδοσης).
- **Δεδομενοκεντρική Επικοινωνία (Data-Centric):** Αισθητήρες οι οποίοι πραγματοποιούν μετρήσεις του ίδιου φυσικού μεγέθους, μπορούν να ομαδοποιηθούν από την Κεντρική Μονάδα Παρακολούθησης βάσει του φαινομένου παρακολούθησης (και όχι π.χ. βάσει της διεύθυνσης IP).
- **Συνάθροιση Δεδομένων (Data Aggregation):** Αισθητήρες οι οποίοι πραγματοποιούν μετρήσεις του ίδιου φυσικού μεγέθους, υπάρχει η δυνατότητα να μεταδώσουν προς την Κεντρική Μονάδα Παρακολούθησης πλεονάζοντα (ή επικαλυπτόμενα) στοιχεία. Η «περιττή» πληροφορία εξαλείφεται ώστε να εξοικονομηθεί ενέργεια και να βελτιωθεί ο συνολικός ρυθμός απόκρισης του Δικτύου.

Οργάνωση κειμένου

Η εργασία έχει οργανωθεί ως εξής:

Στο 1^{ο} κεφάλαιο έγινε μια εισαγωγή στη διείσδυση των ΑΠΕ στην Ελλάδα και την τεχνολογία των ασύρματων δικτύων αισθητήρων.

Στο 2^{ο} κεφάλαιο τίθενται οι προδιαγραφές του προς σχεδίαση συστήματος.

Στο 3^{ο} κεφάλαιο συγκρίνονται διαφορετικές τεχνολογίες hardware, και ασύρματης επικοινωνίας τόσο από λειτουργική πλευρά αλλά και από πλευράς κόστους.

Στο 4^{ο} κεφάλαιο παρουσιάζεται η αρχιτεκτονική του συστήματος, από τη μετρητική συσκευή ως τον web server, το πρωτόκολλο ασύρματης επικοινωνίας μεταξύ μετρητικών συσκευών και πυλών και τα πρωτόκολλα μεταφοράς δεδομένων στο cloud.

Στο 5^{ο} κεφάλαιο παρουσιάζεται το πρωτότυπο που κατασκευάστηκε με λεπτομερείς οδηγίες για τη διασύνδεση του hardware και την εγκατάσταση του software.

Στο 6^{ο} κεφάλαιο αναλύονται τα αποτελέσματα των δοκιμών με δύο διαφορετικές τεχνολογίες ασύρματης επικοινωνίας.

Στο 7^{ο} κεφάλαιο παρουσιάζονται τα συμπεράσματα που προέκυψαν κατά την εκπόνηση της εργασίας και μελλοντικές επεκτάσεις.

Κεφάλαιο 2. Προδιαγραφές συστήματος

Σε αυτό το κεφάλαιο θα θέσουμε τις προδιαγραφές ενός ασύρματου μετρητικού συστήματος ενέργειας χαμηλού κόστους που θα απευθύνεται σε δήμους με εγκατεστημένα φωτοβολταϊκά πάνελ σε διάφορα τυχαία αλλά σταθερά σημεία όπως π.χ. σε κολώνες φωτισμού. Αυτό το σύστημα αποτελείται από τις μετρητικές συσκευές, τις πύλες και τη διαδικτυακή εφαρμογή που επεξεργάζεται τα δεδομένα.

Πιο συγκεκριμένα οι λειτουργικές απαιτήσεις του προς σχεδίαση συστήματος είναι:

- Χαμηλό κόστος (εξοπλισμός, διασύνδεση, εγκατάσταση/συντήρηση)
    - Μικρού μέγεθος και χαμηλού κόστους μετρητικές συσκευές στα σημεία παραγωγής ενέργειας (Φ/Β)
    - Οι μετρητικές συσκευές δε θα έχουν απ' ευθείας σύνδεση με το Διαδίκτυο
    - Αποστολή δεδομένων από τις συσκευές αυτές σε ένα κεντρικό σημείο (gateway) το οποίο θα τα επεξεργάζεται και θα τα αποστέλλει στο cloud (xDSL, 3G/4G) | Θα προτιμηθούν σημεία τα οποία έχουν ήδη πρόσβαση στο διαδίκτυο μέσω xDSL.
- Το gateway θα πρέπει να υποστηρίζει ταυτόχρονα πολλαπλές ασύρματες τεχνολογίες χαμηλής ισχύος έτσι ώστε να μπορεί να ικανοποιήσει τις ανάγκες Δήμων με διαφορετικά χαρακτηριστικά/απαιτήσεις π.χ. urban/suburban/rural περιβάλλον
- Επεκτασιμότητα ώστε να υποστηρίζονται περισσότερες μετρητικές συσκευές ανά gateway
- Δυνατότητα απομακρυσμένου ελέγχου σε επίπεδο gateway
- Υποστήριξη αυτοματοποιημένων διαδικασιών ελέγχου φωτισμού προς μείωση κατανάλωσης ενέργειας
- Φιλικό προς το χρήστη γραφικό περιβάλλον: παρουσίαση των μετρήσεων ανά σημείο ή συγκεντρωτικά, σε πραγματικό χρόνο, ανά ώρα/ημέρα/εβδομάδα/μήνα/έτος/lifetime, απεικόνιση του δικτύου αισθητήρων σε χάρτη, ένδειξη βλαβών

Μετρητική συσκευή

Η μετρητική συσκευή θα πρέπει να εγκατασταθεί σε πολλά σημεία οπότε θέλουμε να είναι χαμηλού κόστους, μικρή σε μέγεθος, με επαρκή αριθμό I/O pins για δυνατότητα επέκτασης της λειτουργικότητάς της, και με χαμηλή κατανάλωση ενέργειας.

Αποτελείται από αισθητήρες και σχετικές διατάξεις και ένα μικροϋπολογιστή. Ο μικροϋπολογιστής δέχεται ως είσοδο τα δεδομένα/μετρήσεις από τους αισθητήρες, τα οποία στη συνέχεια τα προωθεί στην πύλη με ασύρματο τρόπο.

Η μετρητική συσκευή υποστηρίζει μέσω αισθητήρων και διατάξεων τις παρακάτω λειτουργίες:

- Μέτρηση παραγόμενης ισχύος από το Φ/Β
- Μέτρηση καταναλισκόμενης ισχύος από το λαμπτήρα
- Μέτρηση φωτεινότητας
- Ένδειξη μπαταρίας της μετρητικής συσκευής
- Μέτρηση θερμοκρασίας και σχετικής υγρασίας
- Αισθητήρα βροχής
- Μέτρηση υπεριώδους ακτινοβολίας
- Αισθητήρες φωτιάς ή καπνού
- Ασύρματη αποστολή δεδομένων στην πύλη του συστήματος

Τα δεδομένα ισχύος των Φ/Β αφορούν άμεσα το Δήμο, αφού μπορεί ανά πάσα στιγμή να γνωρίζει πόση ενέργεια επιστρέφει στο δίκτυο και να υπολογίζει σε επίπεδο μήνα ή έτους πόση εξοικονομεί. Επίσης, είναι ένας τρόπος επισκόπησης της υγείας του Φ/Β καθώς η σύγκριση των δεδομένων από διαφορετικά πάνελ μπορεί να δείξει αν έχει πέσει η απόδοσή του ή αν έχει πάθει βλάβη.

Το υποσύστημα μέτρησης της ισχύος που καταναλώνει ο λαμπτήρας σε συνδυασμό με τη φωτεινότητα μπορεί να ενσωματωθεί στο σύστημα «έξυπνου» φωτισμού του Δήμου. Οι ενδείξεις της φωτεινότητας μπορούν να σημάνουν την ανάγκη για φωτισμό στην περιοχές που είναι εγκατεστημένη η συσκευή. Με αυτό τον τρόπο μειώνεται η κατανάλωση, αφού με αυτό τον τρόπο διασφαλίζεται ότι ο φωτισμός θα ενεργοποιείται μόνο όταν υπάρχει ανάγκη. Επιπρόσθετα, δεδομένα μηδενικής κατανάλωσης κατά τη διάρκεια της νύχτας μπορούν να σημάνουν ειδοποίηση προς την τεχνική υπηρεσία του Δήμου για την άμεση αντικατάστασή της.

Επειδή η ισχύς που απαιτείται από τη μετρητική συσκευή θα είναι μικρή, δε συμφέρει να τη συνδέσουμε στο υπάρχον δίκτυο. Η μπαταρία αυτή θα μπορεί να επαναφορτίζεται κατά τη διάρκεια της μέρας από το Φ/Β αλλά είναι χρήσιμο να παρακολουθούμε την κατάστασή της.

Το σύστημα προαιρετικά θα περιλαμβάνει ένα πακέτο μετεωρολογικών μεγεθών όπως θερμοκρασία, σχετική υγρασία και αισθητήρα βροχής. Τα δεδομένα αυτών των μετρήσεων αν και δεν μπορούν να αξιοποιηθούν από ένα Δήμο άμεσα, μπορούν να πωληθούν σε αφενός σε υπηρεσίες μετεωρολογικών προβλέψεων αλλά και σε διαφημιστικές εταιρίες που μπορούν να τα εκμεταλλευτούν.

Από τη μια μεριά μια μετεωρολογική υπηρεσία μπορεί να αξιοποιήσει αυτά τα δεδομένα για να παρακολουθεί το μικροκλίμα μιας περιοχής και στη συνέχεια να παρέχει στοχευμένες προβλέψεις. Αν και στα αστικά κέντρα δεν είναι απαραίτητο, στις αγροτικές περιοχές η πρόγνωση της βροχής και των απότομων μεταβολών θερμοκρασιών ή υγρασίας μπορούν να συμβάλλουν σε αποδοτικότερη παραγωγή.

Από την άλλη μεριά, αυτά τα δεδομένα είναι χρήσιμα σε όρους δυναμικού marketing διότι οι καιρικές συνθήκες είναι ενδεικτικές του τι προϊόντα ή υπηρεσίες θα πωληθούν. Σε περιοχές με έντονη χιονόπτωση, για παράδειγμα, βρίσκονται σε ζήτηση αντιολισθητικές αλυσίδες. Οι επαγγελματίες της διαφήμισης δεν έχουν παρά να προβλέψουν και να βελτιώσουν την αγορασιμότητα των προϊόντων τους. Ανάλογα με τις καιρικές συνθήκες, την τοποθεσία και το καιρικό ιστορικό, είναι δυνατή η στρατηγική προώθηση σε συγκεκριμένο κοινό ή η σχετική τοποθέτηση σχετικών προϊόντων σε κατάλληλο σημείο του μαγαζιού.

Με τον ίδιο τρόπο λοιπόν, τα καιρικά δεδομένα μπορούν να χρησιμοποιηθούν και για την αύξηση της ζήτησης όταν πέφτει σε χαμηλά επίπεδα. Για παράδειγμα, η πελατεία των εστιατορίων πέφτει τις βροχερές μέρες. Για να ενισχυθεί αυτή η πελατεία, ένα κατάλληλο σύστημα θα μπορούσε να ενεργοποιεί ειδικές προσφορές για τη συγκεκριμένη μέρα ώστε να αντιστραφεί αυτή η τάση. Αντίστοιχα, η διαδικτυακές διαφημίσεις μπορούν να γίνουν περισσότερο δυναμικές ανάλογα με τον καιρό. Ένα αυτόματο εργαλείο μπορεί να προωθεί πακέτα διακοπών σε παραθαλάσσιους προορισμούς όταν η θερμοκρασία ανέβει πάνω από τους 30 ^{o}C, ή αδιάβροχα ρούχα μια μέρα που βρέχει. Αυτή η αυτοματοποίηση μπορεί να αυξήσει τις πωλήσεις ενός προϊόντος ή μιας υπηρεσίας και να διατηρήσει το καταναλωτικό ενδιαφέρον.

Επίσης προαιρετικά υποστηρίζεται και ένα υποσύστημα πρόληψης και ειδοποίησης πυρκαγιάς το οποίο θα βασίζεται στην ανίχνευση υπεριώδους ακτινοβολίας σχετικής με φωτιά και στον αισθητήρα καπνού ή πυρκαγιάς. Ένα τέτοιο σύστημα μπορεί εγκατασταθεί σε σημεία κοντά σε δασικές εκτάσεις, και σε σημεία που η ανθρώπινη πρόσβαση είναι δύσκολη πχ εγκαταστάσεις Φ/Β σε αγροτική περιοχή. Αν ο αισθητήρας φωτιάς ενεργοποιηθεί ειδοποιείται η πυροσβεστική υπηρεσία του Δήμου.

Η μετρητική συσκευή είναι χαμηλού κόστους και δεν έχει την κατάλληλη υπολογιστική ισχύ και μνήμη ώστε να διαχειριστεί τα δεδομένα τοπικά. Γι’ αυτό και η διασύνδεσή της με μια πύλη είναι απαραίτητη και θα γίνει μέσω μιας ασύρματης ζεύξης χαμηλής ισχύος στα 433MHz/868MHz/2.4GHz, που εξασφαλίζει και την επεκτασιμότητα αυτού δικτύου. Η επιλογή της κατάλληλης τοπολογίας εξαρτάται από τη φυσική θέση μεταξύ όλων αυτών των κόμβων καθώς επίσης και από την εμβέλεια των κεραιών.

Διασύνδεση με πύλη και συλλογή δεδομένων

Η πύλη ως αυτόνομο σύστημα πρέπει να έχει περισσότερη υπολογιστική ισχύ σε σχέση με τις μετρητικές συσκευές καθώς επίσης και εύκολη σύνδεση στο διαδίκτυο. Επειδή τα σημεία που αυτό είναι δυνατό είναι περιορισμένα, θέλουμε να εγκαταστήσουμε τον ελάχιστο αριθμό πυλών ώστε να καλυφθούν επαρκώς όλα τα σημεία ενδιαφέροντος μειώνοντας έτσι το λειτουργικό κόστος, άρα πρέπει να υποστηρίζει ταυτόχρονα πολλαπλές ασύρματες τεχνολογίες χαμηλής ισχύος. Τέλος, η πύλη πρέπει να παρέχει στο χρήστη τη δυνατότητα απομακρυσμένου ελέγχου και πρόσβασης σε αυτή.

Η πύλη έχει ρόλο «μεσάζοντα» στην επικοινωνία μεταξύ του χρήστη και των μετρητικών συσκευών και αποτελείται από μια συσκευή η οποία:

- Συλλέγει ασύρματα τα δεδομένα των μετρήσεων από τις μετρητικές συσκευές μέσω διαφορετικών τεχνολογιών.
- Τα επεξεργάζεται κατάλληλα.
- Τα προωθεί στο cloud για περεταίρω επεξεργασία.

Η πύλη ορίζει ένα υποδίκτυο και συνδέεται με τις μετρητικές συσκευές εντός του, από τις οποίες συλλέγει τις μετρήσεις σε τακτά χρονικά διαστήματα (κάθε 10sec). Η πύλη στη συνέχεια τις ομαδοποιεί κατάλληλα, ανά συσκευή ή ανά φυσικό μέγεθος και τις αποστέλλει προς ένα MQTT broker και μια βάση δεδομένων.

Η σύνδεση με το cloud γίνεται ενσύρματα (μέσω xDSL) ή ασύρματα (μέσω τεχνολογιών 2G/3G/4G). Η επιλογή της κατάλληλης σύνδεσης εξαρτάται από τη θέση της πύλης, αλλά είναι προτιμητέο να βρίσκεται σε ένα κτήριο του Δήμου με υπάρχουσα σύνδεση xDSL ώστε το μηνιαίο κόστος να μην επιβαρύνεται από συνδρομή στο δίκτυο κινητής τηλεφωνίας.

Γραφικό περιβάλλον απεικόνισης δεδομένων και ελέγχου

Το τελευταίο στάδιο του συστήματος περιλαμβάνει ένα γραφικό περιβάλλον για την προβολή των δεδομένων και τον απομακρυσμένο έλεγχο είτε από συμβατικό υπολογιστή είτε από smartphone/tablet. Μπορεί να φιλοξενηθεί σε εξυπηρετητή (server) που ο Δήμος ήδη νοικιάζει ή έχει στην κατοχή του (πχ για φιλοξενία της ιστοσελίδας του).

Αυτό το περιβάλλον συγκεκριμένα:

- Καταγράφει τις μετρήσεις που λαμβάνει από τις πύλες στη βάση δεδομένων.
- Απεικονίζει τα δεδομένα συγκεντρωτικά ή/και ανά αισθητήρα σε πραγματικό χρόνο, αντλώντας δεδομένα που φτάνουν στον MQTT broker, ή ιστορικά σε επίπεδο ώρας, ημέρας, εβδομάδας, μήνα, έτους κλπ από τα δεδομένα που έχουν αποθηκευτεί στη βάση δεδομένων.
- Απεικονίζει το δίκτυο αισθητήρων σε χάρτη.
- Κάνει εκτίμηση της ενέργειας που παρήγαγαν τα Φ/Β και της ενέργειας που κατανάλωσαν τα φώτα. Ο υπολογισμός της παραγόμενης ενέργειας από τα Φ/Β γίνεται θεωρώντας ότι οι μετρήσεις ισχύος καταφτάνουν ανά 10 δευτερόλεπτα.
- Εμφανίζει ειδοποιήσεις σχετικά με δυσλειτουργίες/ βλάβες του συστήματος. Για παράδειγμα από τη σύγκριση δεδομένων δεδομένα μεταξύ διαφορετικών πάνελ παρατηρηθεί ότι ένα Φ/Β παράγει σημαντικά μικρότερη ενέργεια από κάποιο άλλο κοντά του τότε στέλνει ειδοποίηση ότι ίσως χρειάζεται συντήρηση. Αν εντοπιστεί ότι κάποιος λαμπτήρας έχει μηδενική κατανάλωση κατά τη διάρκεια της νύχτας τότε αυτό σημαίνει ότι έχει χαλάσει και στέλνει σχετική ειδοποίηση στην τεχνική υπηρεσία του Δήμου. Ένα σύστημα ειδοποιήσεων έχει προβλεφθεί επίσης αν η στάθμη της μπαταρίας της μετρητικής συσκευής πέσει πολύ χαμηλά, αντίστοιχα και όταν το σύστημα αντιληφθεί ότι δεν έχει λάβει δεδομένα από κάποια μετρητική συσκευή για μεγάλο χρονικό διάστημα, που σημαίνει είτε ότι έχει βλάβη.

Κεφάλαιο 3. Υποψήφιες Τεχνολογίες και Εξοπλισμός

Σε αυτό το κεφάλαιο θα εξετάσουμε εναλλακτικές τεχνολογίες που αφορούν στην επικοινωνία μετρητικής συσκευής-πύλης καθώς και τον απαραίτητο εξοπλισμό ώστε να καλυφθούν οι ανάγκες του προς σχεδίαση συστήματος. Σχετικά με το κόστος του εξοπλισμού ανατρέξαμε σε ιστοσελίδες όπως το ebay.com και amazon.com .

Υπολογιστικά συστήματα

Για τις ανάγκες υλοποίησης των μετρητικών συσκευών και των πυλών εξετάστηκαν διάφορα υπολογιστικά συστήματα που κυκλοφορούν στην αγορά, είτε με τη μορφή Single-board microcontrollers (SBM) είτε ως single-board computers (SBC). Παρακάτω θα ακολουθήσει σύγκριση μεταξύ των δημοφιλέστερων συστημάτων που χρησιμοποιούνται σε εφαρμογές ασύρματων δικτύων αισθητήρων.

Single-board microcontrollers

Ένας Single-board microcontroller (SBM) είναι ένας μικροϋπολογιστής συνδεδεμένος πάνω σε μια πλακέτα τυπωμένου κυκλώματος (PCB). Αυτή η πλακέτα περιλαμβάνει όλα τα επιμέρους απαραίτητα κυκλώματα που είναι απαραίτητα για εργασίες ελέγχου δηλαδή το μικροϋπολογιστή, κυκλώματα εισόδου/εξόδου, ρολόι, μνήμη RAM, μνήμη για αποθήκευση προγραμμάτων, και άλλα ολοκληρωμένα κυκλώματα αν είναι απαραίτητο. [8]

Σήμερα οι μικροϋπολογιστές είναι φθηνοί και ο σχεδιασμός ενός PCB απλός, καθώς η ανάπτυξη των συστημάτων γίνεται με λογισμικό ανοιχτού κώδικα. Οι προγραμματιστικές γλώσσες ανωτέρου επιπέδου απλοποιούν τις διαφορές μεταξύ μικροϋπολογιστών στον προγραμματιστή της εφαρμογής. Τα προγράμματα πλέον αποθηκεύονται σε μνήμες flash, και η ανάπτυξη μπορεί να γίνει από κοινούς προσωπικούς υπολογιστές και το εκτελέσιμο περνάει στον μικροελεγκτή μέσω καλωδίου USB.

Αυτά τα συστήματα μικροϋπολογιστών υποστηρίζουν πολλές μορφές σημάτων εισόδου/εξόδου που επιτρέπουν στην εφαρμογή τον έλεγχο ενός συστήματος του «πραγματικού κόσμου». Πάνω στα SBM μπορούμε να βρούμε ψηφιακές εισόδους/εξόδους που δίνουν ένα bit δεδομένων (on/off), αναλογικές εισόδους με ένα αναλογικό πολυπλέκτη και κοινούς μετατροπείς από αναλογικό σε ψηφιακό (analog to digital converter, ADC). Σε άλλα συστήματα υπάρχουν αναλογικές έξοδοι που μπορεί να χρησιμοποιούν μετατροπείς από ψηφιακό σε αναλογικό (DAC) ή να ελέγχονται από διαμόρφωση πλάτους παλμού (Pulse-width modulation, PWM).

Καθώς κατασκευάζονται από χαμηλού κόστους υλικό, και έχουν μικρό κόστος ανάπτυξης, οι SBM συνήθως απευθύνονται σε κοινό μη εξοικειωμένο με τον προγραμματισμό όπως από χομπίστες και καλλιτέχνες, αλλά και από επαγγελματίες για να αποκτήσουν εμπειρία πάνω σε μια οικογένεια επεξεργαστών.

Πίνακας 1 Σύγκριση βασικών χαρακτηριστικών μεταξύ κοινων SBM

Name	Processor	CPU Speed	Analog In/Out	Digital IO/PWM	EEPROM [KB]	RAM	Flash [KB]	Cost (€)
Arduino Due [9]	ATSAM3X8E	84 MHz	12/2	54/12	-	96	512	36
Arduino Mega 2560	ATmega2560	16 MHz	16/0	54/15	4	8	256	35
Arduino Micro	ATmega32U4	16 MHz	12/0	20/7	1	2.5	32	16
Arduino Nano	ATmega328P	16 MHz	8/0	14/6	1	2	32	8
Arduino Uno	ATmega328P	16 MHz	6/0	14/6	1	2	32	20
Arduino Yùn	ATmega32U4	16 MHz	12/0	20/7	1	2.5	32	52
Arduino Zero	ATSAMD21G18	48 MHz	6/1	14/10	-	32	256	45
Intel Edison [10]	Intel Atom "Tangier"	1 GHz	6/0	20/4	-	1 GB	4 GB	44
STM NUCLEO [11]	STM32F103RB T6	72 MHz	6/0	42169	-	20	128	14
TI MSP430 LaunchPad [12]	ARM Cortex M4F	48 MHz	8	16/7	-	64	256	14

Αξίζει να αναφέρουμε ότι τα σχέδια του Arduino είναι δημοσιευμένα με άδεια Creative Commons, και οι παραπάνω τιμές αντιστοιχούν στα μοντέλα που κυκλοφορούν επίσημα από την εταιρία. Στο εμπόριο βρίσκουμε απομιμήσεις των Arduino Uno και Nano που κοστίζουν 1-2€, τα οποία όμως απόλυτα συμβατά με το Arduino IDE, και επιλέχθηκαν για το πρωτότυπο που κατασκευάστηκε.

Πλακέτες Arduino Nano & Uno

Το Arduino είναι μία υπολογιστική πλατφόρμα βασισμένη σε μία απλή μητρική πλακέτα ανοικτού κώδικα με ενσωματωμένο μικροελεγκτή και αναπτυσσόμενο περιβάλλον για την εύκολη και απλή συγγραφή λογισμικού, που μπορεί να χρησιμοποιηθεί για την κατασκευή ψηφιακών συσκευών και διαδραστικών αντικειμένων. [13]

Εικόνα 1 Arduino Uno και Arduino Nano

Πίνακας 2 Τεχνικες προδιαγραφές Arduino Uno και Arduino Nano

Name	Uno	Nano
Processor	ATmega328P	ATmega328P
Operating/ Input Voltage	5 V / 7-12 V	5 V / 7-9 V
CPU Speed	16 MHz	16 MHz
Analog In/Out	6/0	8/0
Digital IO/PWM	14/6	14/6
EEPROM [KB]	1	1
SRAM [KB]	2	2
Flash [KB]	32	32
USB	Regular	Mini
UART	1	1

Οι πλακέτες χρησιμοποιούν διάφορες εκδόσεις 8-bit Atmel AVR μικροελεγκτών ή 32-bit επεξεργαστών ARM Atmel. Αυτά τα συστήματα παρέχουν σύνολα ψηφιακών και αναλογικών ακροδέκτες I/O στους οποίους μπορούν να διασυνδεθούν διάφορες πλακέτες επέκτασης και άλλα κυκλώματα. Το Arduino διαθέτει σειριακό interface και ο μικροελεγκτής ATmega υποστηρίζει τη σειριακή επικοινωνία, την οποία το Arduino προωθεί μέσα από έναν ελεγκτή Serial-over-USB ώστε να συνδέεται με τον υπολογιστή μέσω USB. Η

σύνδεση αυτή χρησιμοποιείται για την μεταφορά των προγραμμάτων που σχεδιάζονται από τον υπολογιστή στο Arduino αλλά και για αμφίδρομη επικοινωνία του Arduino με τον υπολογιστή μέσα από το πρόγραμμα την ώρα που εκτελείται.

Για τον προγραμματισμό των μικροελεγκτών, η πλατφόρμα Arduino παρέχει ένα ολοκληρωμένο περιβάλλον ανάπτυξης (IDE) σύμφωνα με το πρόγραμμα Processing, το οποίο μπορεί να υποστηριχθεί από τις γλώσσες προγραμματισμού C και C++.

Ο μικροελεγκτής («microcontroller») είναι ένας τύπος επεξεργαστή, μια παραλλαγή ουσιαστικά του μικροεπεξεργαστή, ο οποίος μπορεί να λειτουργήσει με ελάχιστα εξωτερικά εξαρτήματα, λόγω των πολλών ενσωματωμένων υποσυστημάτων που διαθέτει. Χρησιμοποιείται ευρύτατα σε όλα τα ενσωματωμένα συστήματα ελέγχου χαμηλού και μεσαίου κόστους, όπως αυτά που χρησιμοποιούνται σε αυτοματισμούς, ηλεκτρονικά καταναλωτικά προϊόντα (από ψηφιακές φωτογραφικές μηχανές έως παιχνίδια), ηλεκτρικές συσκευές και κάθε είδους αυτοκινούμενα τροχοφόρα οχήματα.

Η υπολογιστική πλατφόρμα Arduino αποτελείται από ένα μικροελεγκτή Atmel AVR και συμπληρωματικά εξαρτήματα για την διευκόλυνση του χρήστη στον προγραμματισμό και την ενσωμάτωση του σε άλλα κυκλώματα. Όλες οι πλατφόρμες περιλαμβάνουν ένα γραμμικό ρυθμιστή τάσης 5V και έναν κρυσταλλικό ταλαντωτή 16MHz (ή κεραμικό αντηχητή σε κάποιες παραλλαγές).

Ο μικροελεγκτής είναι από κατασκευής προγραμματισμένος με ένα «bootloader» δηλαδή ένα εσωτερικό κώδικα, έτσι ώστε να μην χρειάζεται εξωτερικός προγραμματιστής. Το Arduino διατίθεται σε πολλές εκδόσεις ανάλογα με τις ανάγκες και τις δυνατότητες του κάθε προγραμματιστή γενικά όμως όλες οι πλακέτες είναι προγραμματισμένες μέσω μιας σειριακής σύνδεσης RS-232, αλλά ο τρόπος με τον οποίο αυτό υλοποιείται ποικίλλει ανάλογα με την έκδοση. Οι σειριακές πλακέτες Arduino περιέχουν ένα απλό κύκλωμα αντιστροφής για την μετατροπή ανάμεσα στα σήματα των επιπέδων «RS-232» και «TTL».

Τα πιο σημαντικά πλεονεκτήματα των Arduino είναι: [14]

- *Ευέλικτο*: Μπορεί να συνεργαστεί με όλα τα γνωστά λειτουργικά συστήματα, όπως Windows, Macintosh OSX και Linux, σε αντίθεση με άλλους γνωστούς μικροελεγκτές που περιορίζονται σε κλειστού λογισμικού IDEs.
- *Απλό και σαφές προγραμματιστικό περιβάλλον*: Το προγραμματιστικό περιβάλλον του Arduino είναι εύκολο στην χρήση από αρχάριους, αλλά και αρκετά ευέλικτο για έμπειρους προγραμματιστές.
- *Επεκτάσιμο και ανοιχτού λογισμικού software*: Το λογισμικό του Arduino είναι ανοιχτού κώδικα και διαθέσιμο για επέκταση από έμπειρους προγραμματιστές. Η γλώσσα προγραμματισμού μπορεί να επεκταθεί μέσα από βιβλιοθήκες της C++, και όσοι επιθυμούν να διεισδύσουν σε τεχνικές λεπτομέρειες μπορούν να ανατρέξουν στην γλώσσα προγραμματισμού AVR, καθώς και να προσθέσουν κώδικα AVR-C άμεσα στο λειτουργικό του Arduino.
- *Υποστήριξη από την κοινότητα:* Υπάρχει μεγάλη συλλογή από έτοιμα projects, tutorials και οδηγίες που παρέχονται απ' ευθείας από τους χρήστες των Arduino.

Single-board computers

Ένας Single-board computer(SBC) είναι ένας υπολογιστής συνδεδεμένος πάνω σε μια πλακέτα μαζί με μικροϋπολογιστές, μνήμη, εισόδους/εξόδους και άλλα χαρακτηριστικά απαραίτητα ώστε να αποτελούν ένας πλήρως λειτουργικός υπολογιστής. Χρησιμοποιούνται στην εκπαίδευση, για ανάπτυξη συστημάτων ή για χρήση σε ενσωματωμένα συστήματα. Σε αντίθεση με τα SBM, οι SBC έχουν μεγαλύτερη υπολογιστική ισχύ, τρέχουν ένα λειτουργικό σύστημα, συνήθως Linux, και μπορούν να συνδεθούν σε οθόνη. [15]

Πίνακας 3 Σύγκριση βασικών χαρακτηριστικών μεταξύ κοινων SBC [16]

Name	Processor	Input Voltage	CPU Core	CPU Speed	RAM size	USB ports	Operating System	€
Banana Pi M1	Allwinner A20	5 V	2	1GHz	1 GB	2	Android 4.4, Raspbian, Lubuntu, Debian	38
Banana Pi M2	Allwinner A31s	5 V	4	1GHz		4	Android and Linux	44
Beagle Bone	TI Sitara AM335x	5 V	1	720MHz	256 MB	1	Linux, Android, OpenBSD	50
Beagle Bone Black	TI Sitara AM335x	5 V	1	1GHz	512MB	1	Linux, Android, OpenBSD	45
Cubieboard	Allwinner A10	5 V	1	1GHz	1	2	Android, Lubuntu	56
Cubieboard 2	Allwinner A20	5 V	2	1GHz	1	2	Android, Lubuntu	61
Cubieboard 3	Allwinner A20	5 V	2	1GHz	2	3	Android, Lubuntu	70
Intel Galileo Gen 2	Intel Quark SoC X1000	7-15 V	1	400MHz	256 MB	1	Android, Lubuntu	49
Raspberry Pi Model A	Broadcom BCM 2835	5 V	1	700MHz	256 MB	1	Raspbian,other Linux distros	22
Raspberry Pi Model A+	Broadcom BCM 2835	5 V	1	700MHz	256 MB	1	Raspbian, other Linux distros	18
Raspberry Pi Model B	Broadcom BCM 2835	5 V	1	700MHz	512MB	2	Raspbian, other Linux distros	30
Raspberry Pi Model B+	Broadcom BCM 2835	5 V	1	700MHz	512MB	4	Raspbian, other Linux distros	22
Raspberry Pi 2 Model B	Broadcom BCM 2836	5 V	4	900MHz	1	4	Raspbian, other Linux distros,Windows10	30

Στο πρωτότυπο που υλοποιήθηκε επιλέξαμε ένα Raspberry Pi2 για να χρησιμοποιηθεί ως πύλη.

Raspberry Pi 2

Το Raspberry Pi είναι ένας χαμηλού κόστους και μικρού μεγέθους υπολογιστής που συνδέεται σε μια οθόνη υπολογιστή ή στην τηλεόραση και χρησιμοποιεί ένα τυπικό πληκτρολόγιο και ποντίκι. Πρόκειται για μια ικανή μικρή συσκευή που επιτρέπει στους ανθρώπους όλων των ηλικιών να εξερευνήσουν το υπολογιστικό σύστημα και να μάθουν πώς να προγραμματίζουν σε γλώσσες όπως Scratch και Python. Είναι ικανοί να κάνουν ό,τι θα περίμενε κανείς από έναν επιτραπέζιο υπολογιστή, δηλαδή περιήγηση στο διαδίκτυο και αναπαραγωγή βίντεο υψηλής ευκρίνειας, αλλά και σύνταξη υπολογιστικών προγραμμάτων, επεξεργασία κειμένου και παιχνίδια. [17]

Εικόνα 2 Raspberry Pi 2

Στο Raspberry Pi χρησιμοποιείται η μέθοδος SoC (System on a Chip), κατά την οποία τοποθετούνται όλα τα απαραίτητα ηλεκτρονικά, συμπιεσμένα σε ένα πολύ μικρό πακέτο, για την λειτουργία ενός υπολογιστή σε ένα μόνο chip. Αντίθετα οι κοινοί υπολογιστές έχουν ξεχωριστά chips για CPU, GPU, USB controller, RAM και άλλα.

Για τη λειτουργία του παρέχονται πολλά πακέτα λογισμικού, ένα από τα οποία καλείται ο χρήστης να εγκαταστήσει. Η επιλογή βασίζεται στην εξοικείωση του εκάστοτε χρήστη, αλλά και στην εφαρμογή την οποία θέλει να υλοποιήσει, ακόμη διατίθεται και ειδική έκδοση για αρχάριους. Γενικά προτιμάται το λογισμικό Linux και η εταιρία προτείνει την Python σαν γλώσσα προγραμματισμού, χωρίς όμως αυτό να είναι απόλυτο, αφού συνεργάζεται εξαιρετικά με πολλές άλλες γλώσσες όπως C, C++, Java, Scratch και Ruby, οι οποίες είναι προεγκατεστημένες και ακόμα περισσότερες που μπορεί να εγκαταστήσει ο χρήστης. [18]

Τα πιο σημαντικά πλεονεκτήματά του είναι:

- *Προσιτό οικονομικά*: Σε σύγκριση με άλλες παρόμοιες εναλλακτικές λύσεις, το Pi2 προσφέρει τις καλύτερες προδιαγραφές για την τιμή του. Είναι μία από τις λίγες συσκευές της ίδιας κατηγορίας, που προσφέρει 1GB μνήμης RAM. Άρα είναι προσιτό οικονομικά, με κόστος αγοράς περίπου 30€.
- *Υποστήριξη από την κοινότητα*: Το Raspberry Pi έχει τεράστια υποστήριξη από την κοινότητα. Υπάρχει λογισμικό σχεδιασμένο για για αυτό κυρίως σε forum χρηστών.

- *Δυνατότητες overclocking*: Είναι ένας τρόπος ώστε να εξαντληθούν οι δυνατότητες του hardware του Raspberry Pi2 στο μέγιστο βαθμό. Οι κατασκευαστές συσκευών τροποποιούν πολλές φορές τα συστατικά του hardware για να λειτουργούν σε χαμηλότερες ταχύτητες βελτιώνοντας την αξιοπιστία και τη διάρκεια ζωής του καθώς και ταυτόχρονα εξασφαλίζουν επαρκή απόδοση. Αλλά μπορεί να υπάρξουν προβλήματα απόδοσης ανάλογα με με την εφαρμογή που χρησιμοποιείται.
- *Μεγάλη ποικιλία σε χρήσεις*: Υπάρχει η δυνατότητα αλλαγής κάρτας SD, εύκολα και γρήγορα, για διαφορετικές λειτουργίες από αυτές της υπάρχουσας συνδεδεμένης στο Raspberry Pi2, επειδή κάθε κάρτα μπορεί να αποθηκεύει τα δεδομένα. Επίσης σε μια κάρτα SD μπορεί να δημιουργηθεί αντίγραφο ασφαλείας και να χρησιμοποιηθεί αργότερα αν χρειαστεί. Άρα δεν είναι μια συσκευή που προορίζεται αποκλειστικά για μια χρήση.

Απ’ την άλλη πλευρά το Raspberry Pi παρουσιάζει μερικές αδυναμίες, όπως το γεγονός ότι δεν εκτελέσιμα τα αρχεία x86 στο Pi, επειδή δεν υποστηρίζει αυτή την αρχιτεκτονική. Ακόμη η μνήμη RAM δεν μπορεί να επεκταθεί. Παρ’ όλα αυτά γίνονται προσπάθειες βελτίωσης των δυνατοτήτων του Raspberry Pi διατηρώντας στο ίδιο επίπεδο το κόστος αγοράς και το μικρό μέγεθός του.

Αισθητήρες και διατάξεις μετρήσεων

Παρακάτω θα συγκρίνουμε διάφορους αισθητήρες που μπορούν να χρησιμοποιηθούν ώστε να λάβουμε μετρήσεις στα φυσικά μεγέθη που παρακολουθούμε.

Αισθητήρας ρεύματος

Στην αγορά κυκλοφορούν διάφοροι αισθητήρες ρεύματος. Στον παρακάτω πίνακα συγκρίνουμε τους πιο κοινούς.

Πίνακας 4 Σύγκριση χαρακτηριστικών αισθητήρων ρεύματος

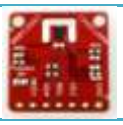
Μοντέλο	Μέγιστο ρεύμα εισόδου	Τιμή	Εικόνα
ACS712 [19]	5A,20A ή 30A	0.30-1€	
SCT-013-000 [20]	100A	5 €	
SCT-013-030	30A	5 €	
max471 [21] (με ενσωματωμένο αισθητήρα τάσης)	3A	1 €	
TA12-100 [22]	5A	4-5€	

Δεομένου ότι η μέγιστη ισχύς που καταναλώνουν οι παραδοσιακές λάμπες φωτισμού δρόμου είναι 580W με τάση δικτύου, ένα εύρος μέτρησης 0-5A είναι υπεραρκετό για την εφαρμογή μας. Στο πρωτότυπο που κατασκευάστηκε επιλέχθηκε ο SCT-013-000 λόγω διαθεσιμότητας στην ελληνική αγορά.

Αισθητήρας θερμοκρασίας και σχετικής υγρασίας

Υπάρχουν διάφοροι οικονομικοί αισθητήρες θερμοκρασίας και σχετικής υγρασίας. Παρακάτω θα συγκρίνουμε κάποια χαρακτηριστικά τους.

Πίνακας 5 Σύγκριση κοινών αισθητήρων θερμοκρασίας-υγασίας

Αισθητήρας	Τάση λειτουργίας	Εύρος υγρασίας, ακρίβεια	Εύρος θερμοκρασίας, ακρίβεια	Κόστος	Εικόνα
DHT11 [23]	3-5V	20-80%, ±5%	0-50°C ±2°C	<1€	
DHT22 [24]	3-5V	0-100%, ±2-5%	-40-125°C ±0.5°C	2 €	
SHT71 [25]	2.4-5.5 V	0-100%, ±3.0	-40°C -125°C ±0.4	15 €	
SHT75	2.4-5.5 V	0-100%, ±1.8	-40°C -125°C ±0.3	25 €	
HDC1008 [26]	3-5V	0-100%, 4%	-20°C-85°C, ±0.2°C	7 €	

Δεδομένου ότι η εφαρμογή που σχεδιάζουμε θα μετράει εξωτερικές θερμοκρασίες περιβάλλοντος, πρέπει να κάνουμε πρόβλεψη και για θερμοκρασίες υπό το μηδέν, προδιαγραφή που καθιστά τον DHT22 τον καταλληλότερο αισθητήρα με το μικρότερο κόστος.

Σε αυτό το σημείο σημειώνεται ότι στο πρωτότυπο που υλοποιήθηκε χρησιμοποιήθηκε ο DHT11 λόγω διαθεσιμότητας στην ελληνική αγορά.

Άλλοι αισθητήρες

Πίνακας 6 Παρουσίαση λοιπών αισθητήρων

αισθητήρας	κόστος	σχόλια	εικόνα
βροχής	1-2€	Ενδεικτικά αναφέρουμε ένα αισθητήρα βροχής που βασίζεται στον τελεστικό ενισχυτή LM393 και σε ένα τυπωμένο κύκλωμα που «συλλέγει» τις σταγόνες της βροχής. Αυτές δημιουργούν παράλληλων αντιστάσεων που μετρούνται από τον τελεστικό. Όσο περισσότερο νερό συγκεντρωθεί, η συνολική αντίσταση του τυπωμένου μειώνεται.	
πυρκαγιάς/IR	<1€	Αυτό το module έχει στην άκρη του ένα αισθητήρα υπέρυθρης ακτινοβολίας και η άλλη άκρη συνδέεται με μια αναλογική είσοδο του Arduino. Όσο μεγαλύτερη ακτινοβολία τότε τόσο μικρότερη τιμή θα διαβάζει στην έξοδο.	
UV ML8511	3 €	Αρχή λειτουργίας όμοια με παραπάνω	

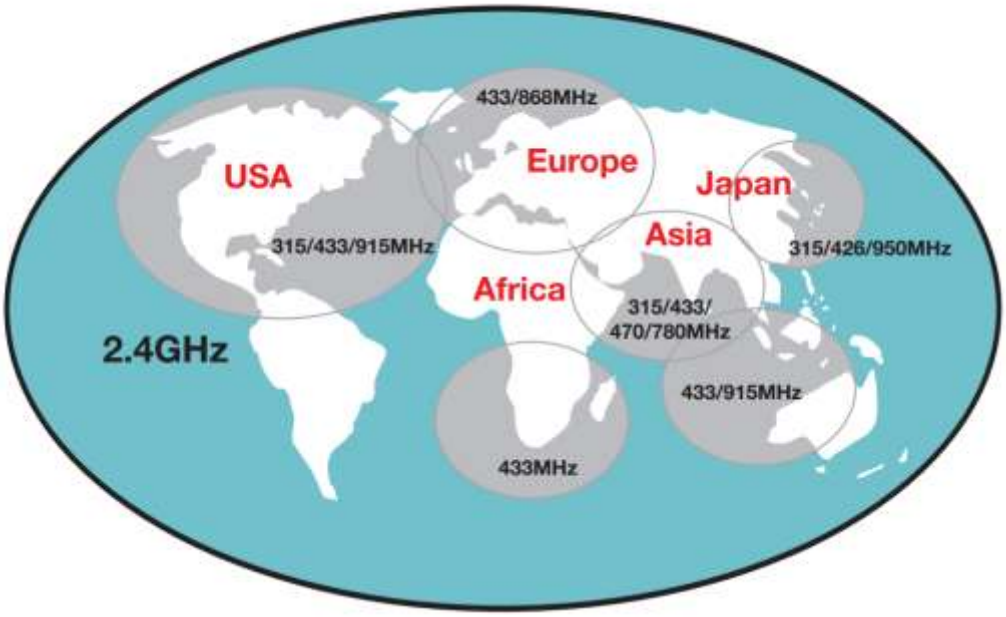
Ασύρματη επικοινωνία κόμβων-πύλης

Στο γρήγορα αναπτυσσόμενο Διαδίκτυο των Πραγμάτων (Internet of Things – IoT), κανένα ασύρματο πρότυπο δεν μπορεί να υπερισχύσει. Με διάφορα πρότυπα στην αγορά, που εκτείνονται πάνω από πολλαπλές συχνότητες και χρησιμοποιούν διαφορετικά πρωτόκολλα επικοινωνίας, η επιλογή της κατάλληλης τεχνολογίας ασύρματης συνδεσιμότητας για μια εφαρμογή IoT είναι σημαντικό βήμα στην ανάπτυξή της. [27]

Η απαίτηση για χαμηλή ισχύ και χαμηλή κατανάλωση ενέργειας γίνεται ένας ολοένα και πιο σημαντικός παράγοντας στο σχεδιασμό και υλοποίηση των ασύρματων δικτύων αισθητήρων. Αυτή η απαίτηση επεκτείνεται και στην επικοινωνία μεταξύ κόμβων και πύλης, είτε μειώνοντας την ισχύ εκπομπής των πομποδεκτών ή χρησιμοποιώντας μια τοπολογία χαμηλής ισχύος όπως η mesh. Υπάρχουν επίσης διάφορα πρωτόκολλα που μπορούν να χρησιμοποιηθούν ανάλογα με το ρυθμό μετάδοσης δεδομένων, την καθυστέρηση διάδοσης και την απόσταση μεταξύ των κόμβων του δικτύου. [28]

ISM συχνότητες (433/ 868/915 MHz και 2.4 GHz)

Οι ασύρματες μεταδόσεις ρυθμίζονται από παγκόσμιους οργανισμούς που ορίζουν μπάντες συχνοτήτων για συγκεκριμένες χρήσεις και ορίζουν πρότυπα για αυτές. Για το μεγαλύτερο μέρος του φάσματος στις περισσότερες περιοχές δίνονται άδειες χρήσης, δηλαδή για συγκεκριμένο κανάλι πρέπει να αγοραστεί άδεια από τον αρμόδιο ρυθμιστή πχ για τις συχνότητες της κινητής τηλεφωνίας. Η ITU-R, ο τομέας της ITU που συντονίζει την παγκόσμια χρήση του φάσματος, έχει εξασφαλίσει κάποιες μπάντες για εφαρμογές ISM (industrial, scientific and medical), βιομηχανικές, επιστημονικές ή ιατρικές, η κάθε μια με τα δικά της πλεονεκτήματα και προκλήσεις. Αυτές χρησιμοποιούνται για βιομηχανικούς, επιστημονικούς ή ιατρικούς σκοπούς εκτός των τηλεπικοινωνιών. Τα τελευταία χρόνια όμως χρησιμοποιούνται επίσης και για τηλεπικοινωνιακά συστήματα μικρής εμβέλειας και ισχύος όπως τηλεχειριστήρια, ασύρματα τηλέφωνα και ασύρματη σύνδεση στο Internet (Wi-Fi). Κάποιες συχνότητες είναι διαθέσιμες παγκοσμίως, ενώ άλλες σε συγκεκριμένες περιοχές.



Εικόνα 3 Ελεύθερες συχνότητες ανά τον κόσμο

915MHz

Χρησιμοποιείται στις ΗΠΑ, τον Καναδά και περιορισμένα στην Αυστραλία και Νέα Ζηλανδία. Έχει εύρος ζώνης 26MHz με αυστηρούς κανονισμούς λειτουργίας και εμβέλεια μέχρι 1χλμ.

868MHz

Αυτή η μπάντα χρησιμοποιείται στην Ευρώπη. Συγκεκριμένες περιοχές του φάσματός της έχουν οριστεί για συγκεκριμένες εφαρμογές (συναγερμοί πυρκαγιάς, ασφάλεια) και σε κάποιες υπομπάντες υπάρχουν περιορισμοί στη μετάδοση. Η ισχύς μετάδοσης κυμαίνεται από τα 5mW ως τα 500mW. Τα διαθέσιμα modules κυκλοφορούν σε ποικιλία, έχουν μικρές κεραίες, μπορούν να είναι μεγάλης εμβέλειας των 500mW ή μικρής εμβέλειας του 1mW. Η συμφόρηση είναι σχετικά μικρή. Από την άλλη πλευρά λόγω των περιορισμών το ZigBee μπορεί να λειτουργήσει μόνο σε ένα κανάλι σε αυτή τη μπάντα.

433MHz

Χρησιμοποιείται παγκοσμίως εκτός από τις ΗΠΑ. Μεγάλη ποικιλία πομποδεκτών από διάφορους κατασκευαστές, ακόμα και με ολοκληρωμένο κύκλωμα για ακόμα μικρότερο κόστος. Μικρότερες απώλειες διάδοσης από τη μπάντα των 868MHz, άρα απαιτείται μικρότερη ισχύς για την επίτευξη της ίδιας εμβέλειας. Συνήθως απαιτείται κεραία.

2.4GHz

Η πιο δημοφιλής μπάντα είναι αυτή των 2,4GHz επειδή είναι διαθέσιμη παγκοσμίως, με διαφορετικά πρωτόκολλα όπως το Zigbee, το Wi-Fi και το Bluetooth και άλλα παρόμοια (Wibree, Zwave, Nanotron). Το εύρος ζώνης που έχει ανατεθεί σε αυτή τη μπάντα είναι παραπάνω από 80MHz που δίνει τη δυνατότητα για μεγάλους ρυθμούς δεδομένων. Παρόλα αυτά, αυτή η μπάντα του φάσματος είναι επιρρεπής σε παρεμβολές λόγω συμφόρησης. Είναι μικρής εμβέλειας λόγω απωλειών σε σχέση με μικρότερες συχνότητες και επειδή η διείσδυση μέσα από τοίχους είναι δύσκολη. Επίσης πολλά από τα πρωτόκολλα

δεν είναι βελτιστοποιημένα για δεδομένα πραγματικού χρόνου, χρησιμοποιώντας ένα αλγόριθμο καλύτερης περίπτωσης, και ίσως αυξήσουν την κατανάλωση ενέργειας αν χρειάζεται να ξαναστέλνουν πακέτα.

Πρότυπα πρωτόκολλα ασύρματης επικοινωνίας

ZigBee

Το πρότυπο IEEE 802.15.4 αναπτύχθηκε με σκοπό τη παροχή υπηρεσιών φυσικού επιπέδου και επιπέδου ζεύξης σε δίκτυα χαμηλού ρυθμού μετάδοσης δεδομένων και περιορισμένης κατανάλωσης ενέργειας, και τα δύο θεμελιώδη χαρακτηριστικά ενός δικτύου WSN.

Το πρότυπο ZigBee αποτελεί την επέκταση του IEEE 802.15.4 στα ανώτερα επίπεδα του δικτύου (επίπεδα δικτύου και εφαρμογής) και συνεργατικά υλοποιούν μια ενιαία εμπορική πλατφόρμα, η οποία καλείται τεχνολογία ZigBee και βρίσκει εφαρμογή σε δίκτυα χαμηλού ρυθμού μετάδοσης (LPAN – Low-rate Personal Area Networks). Η τεχνολογία ZigBee χρησιμοποιεί τις ελεύθερες μπάντες συχνοτήτων περί τα 2.4GHz (παγκόσμια χρήση), 915MHz (Αμερική) και 868MHz (Ευρώπη). Ο ρυθμός μετάδοσης της υπηρεσίας στη τεχνολογία είναι 250kbps στα 2.4GHz, 40kbps στα 915MHz και 100kbps στα 868MHz. Είναι πολύ δημοφιλές σε εφαρμογές οικιακού αυτοματισμού.

Εικόνα 4 Πομποδέκτης ZigBee

Οι συσκευές ZigBee είναι χαμηλής ισχύος και γι' αυτό έχουν εμβέλεια 10-100 μέτρα και εξαρτώνται από τις περιβαλλοντικές συνθήκες. Μπορούν όμως να μεταδώσουν δεδομένα σε μεγάλες αποστάσεις χρησιμοποιώντας άλλες συσκευές ως ενδιάμεσους. Η χρήση τους είναι πιο απλή σε σχέση με αυτή των φθηνών πομποδεκτών ραδιοσυχνοτήτων, και η επικοινωνία μεταξύ διαφορετικών modules αρκετά αξιόπιστη, ενώ η τιμή των πομποδεκτών ZigBee ξεκινούν από τα 15€.

Bluetooth & Bluetooth Low Energy

Το πρότυπο Bluetooth αναπτύχθηκε σε εφαρμογές διασύνδεσης συσκευών (WPAN), στις οποίες η χαμηλή κατανάλωση ενέργειας είναι πρωταρχικής σημασίας. Το πρότυπο βασίζεται στο πρωτόκολλο IEEE 802.15.1 και χρησιμοποιεί την ελεύθερη ζώνη συχνοτήτων των 2.4GHz. Ο ρυθμός μετάδοσης που υποστηρίζει είναι μέχρι 2Mbps και χρησιμοποιείται

κυρίως σε τοπολογία σημείου προς σημείο ή αστέρα. Το Bluetooth Low Energy (γνωστό και ως Bluetooth Smart) είναι μια πιο πρόσφατη προσθήκη στο πρότυπο. Έχει σχεδιαστεί για χαμηλότερο βαθμό εξυπηρέτησης, και γι' αυτό έχει μειωμένη κατανάλωση. Το κλασικό Bluetooth μπορεί να έχει μέχρι 8 ενεργές συσκευές σε ένα δίκτυο αστέρα, ενώ το Bluetooth LE 10-20.

Η εμβέλεια συσκευών του προτύπου είναι 50 m, με ισχύ εκπομπής 10 mW. Η τιμή τους ξεκινά από <1€.

Εικόνα 5 Πομποδέκτης Bluetooth

Σύγκριση hardware ασύρματης επικοινωνίας

nRF24L01+

Το RF module nRF24L01+ της εταιρείας Nordic semiconductors είναι RF συσκευή χαμηλής κατανάλωσης και μικρής εμβέλειας (περίπου 100 μέτρα). Συνδυάζει RF πομποδέκτη, RF synthesizer και λειτουργεί στα 2.4 GHz. Υποστηρίζει διασύνδεση SPI υψηλής ταχύτητας μετάδοσης και αποτελεί ένα πολύ αποδοτικό μέσο ασύρματης επικοινωνίας μεταξύ των μικροελεγκτών που χρησιμοποιήσαμε. [29]

Επίσης υπάρχουν βιβλιοθήκες ανοιχτού κώδικα που το αξιοποιούν για συνδεσμολογία αστέρα αλλά και mesh [30], [31], [32]

Το πιο σημαντικό χαρακτηριστικό του είναι η χαμηλή του τιμή και η υψηλή του απόδοση. Κυκλοφορεί στην πιο απλή του χωρίς κεραία ή με κεραία για μεγαλύτερο κέρδος, με κόστος <1€.

Εικόνα 6 Το nRF24L01+

Εικόνα 7 Το nRF24L01+ με κεραία

RFM69W, RFM69HW

Ο RFM69W είναι ένας πομποδέκτης που μπορεί κυκλοφορεί για τις ελεύθερες συχνότητες των 315, 433, 868 και 915MHz. Όλες οι κύριες παράμετροί του προγραμματίζονται και οι περισσότερες μπορούν να οριστούν δυναμικά. Έχει το μοναδικό πλεονέκτημα του προγραμματισμού επικοινωνιών σε στενή ή ευρεία ζώνη. Είναι βελτιστοποιημένο για χαμηλή κατανάλωση ισχύος ενώ δίνει ισχύ εξόδου 13dBm και έχει εμβέλεια περίπου 500 μέτρα [33]. Επίσης υπάρχουν βιβλιοθήκες ανοιχτού κώδικα που το αξιοποιούνγια δίκτυα σε τοπολογία αστέρα [34] ή mesh [35]

Η τιμή του ξεκινάει από τα 3.50€. Υπάρχει και η παραλλαγή του, ο πομποδέκτης RFM69HW με ισχύ εξόδου 20dBm και εμβέλεια περίπου 1χλμ. Η τιμή του ξεκινάει από τα 4€.

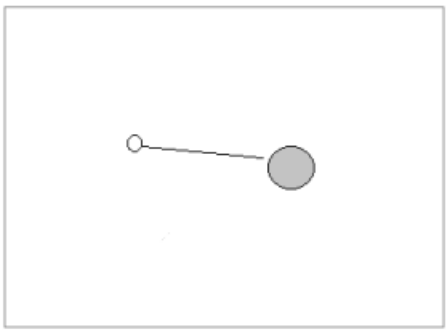
Εικόνα 8 Εμπρόσθια και πίσω όψη των πομποδεκτών RFM69W/ RFM69HW

Τοπολογίες ασύρματων δικτύων

Τα σύρματα δίκτυα μπορούν να χαρακτηριστούν και από την τοπολογία τους. Η ανάπτυξη των ασύρματων δικτύων αισθητήρων εκμεταλλεύεται ήδη υπάρχουσες τοπολογίες και τις εξελίξει, ώστε να τις προσαρμόσει στις ανάγκες για μειωμένο κόστος και πολυπλοκότητα των δικτύων αυτών, βελτιώνοντας την αξιοπιστία τους. [36]

Τοπολογία Point to Point (σημείου προς σημείο)

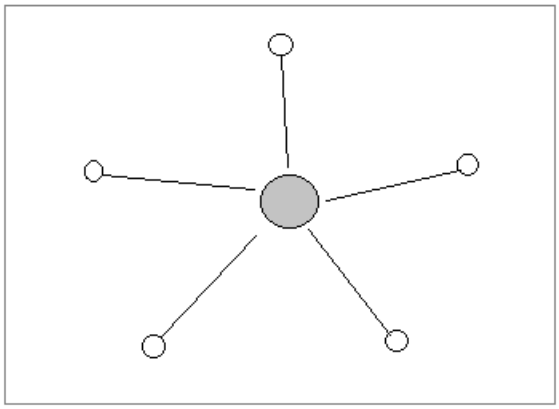
Είναι η απλούστερη δυνατή τοπολογία. Σε αυτήν την τοπολογία συμμετέχουν μόνο 2 κόμβοι από τους οποίους ο ένας πρέπει να είναι η τελική συσκευή και ο άλλος η πύλη.



Εικόνα 9 Τοπολογία σημείου προς σημείο

Τοπολογία Star (αστέρα)

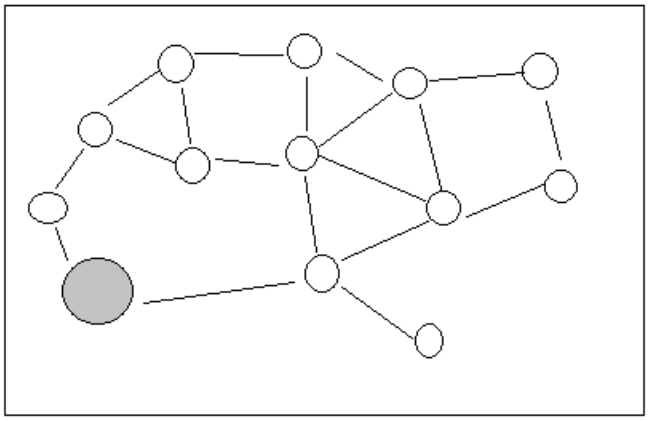
Η τοπολογία αστέρα είναι η απλούστερη στα δίκτυα ασύρματων αισθητήρων, όπου κάθε κόμβος επικοινωνεί απευθείας με την πύλη. Λόγω ελάχιστων δικτυακών απαιτήσεων, αυτή η τοπολογία απλοποιεί το δίκτυο όπου είναι δυνατό. Παρόλα αυτά η επεκτασιμότητα του δικτύου είναι περιορισμένη. Οι κόμβοι που βρίσκονται σε σχετικά μεγάλες αποστάσεις από την πύλη θα έχουν χαμηλής ποιότητας σύνδεση. Επομένως, η τοπολογία αστέρα καλό είναι να χρησιμοποιείται μόνο όταν ο αριθμός των κόμβων είναι σχετικά μικρός και βρίσκονται στην περιοχή κάλυψης της κεραίας που χρησιμοποιείται.



Εικόνα 10 Τοπολογία αστέρα

Τοπολογία mesh (πλέγματος)

Σε αυτή την τοπολογία κάθε κόμβος μπορεί να συνδεθεί σε διάφορους άλλους αλλά μόνο κάποιοι από αυτούς αποτελούν πύλες. Το σήμα πηγαίνει από τον ένα κόμβο στον άλλο μέχρι να φτάσει στην πύλη. Η διαδρομή του σήματος καθορίζεται από πρωτόκολλο δρομολόγησης. Είναι πιο περίπλοκα στη σχεδίαση και μπορεί να παρουσιάσουν μεγαλύτερες καθυστερήσεις στη δρομολόγηση μηνυμάτων σε σχέση με τα δίκτυα αστέρα. Δίνουν όμως τη δυνατότητα για μεγαλύτερη εμβέλεια δικτύου μέσω των μεταπηδήσεων διατηρώντας χαμηλή την ισχύ μετάδοσης.



Εικόνα 11 Τοπολογία mesh

Επικοινωνία gateway-cloud

Ο τελικός προορισμός της πληροφορίας είναι το cloud και αυτό μπορεί να επιτευχθεί με δύο τρόπους. Ο ένας είναι απλώς συνδέοντας τις μετρητικές συσκευές απευθείας στο Internet. Αλλά αυτή η μεθοδολογία επιφέρει ένα κόστος σύνδεσης ανάλογο με τον αριθμό των μετρητικών συσκευών που χρησιμοποιούνται. Ο άλλος είναι χρησιμοποιώντας συσκευές-πύλες προς το cloud οι οποίες αναλαμβάνουν την ασύρματη συλλογή των δεδομένων των μετρητικών συσκευών και στη συνέχεια τα προωθούν είτε μέσω DSL, αν φυσικά η πύλη βρίσκεται κοντά σε ένα τέτοιο σημείο, είτε μέσω δικτύου κινητής τηλεφωνίας.

Ενδεικτικά αναφέρουμε ότι ένα καλώδιο Ethernet 45 μέτρων κοστολογείται από 8-9€, το κόστος ενός wifi dongle ξεκινάει από 2€ για ταχύτητα μεχρι 100Mbps, και σε αυτό το κόστος προστίθεται επίσης η σύνδεση DSL. Το κόστος των 3G dongles ξεκινάει από τα 7€ και πάλι πρέπει να προστεθεί το κόστος του πακέτου δεδομένων που θα πρέπει να χρησιμοποιηθεί.

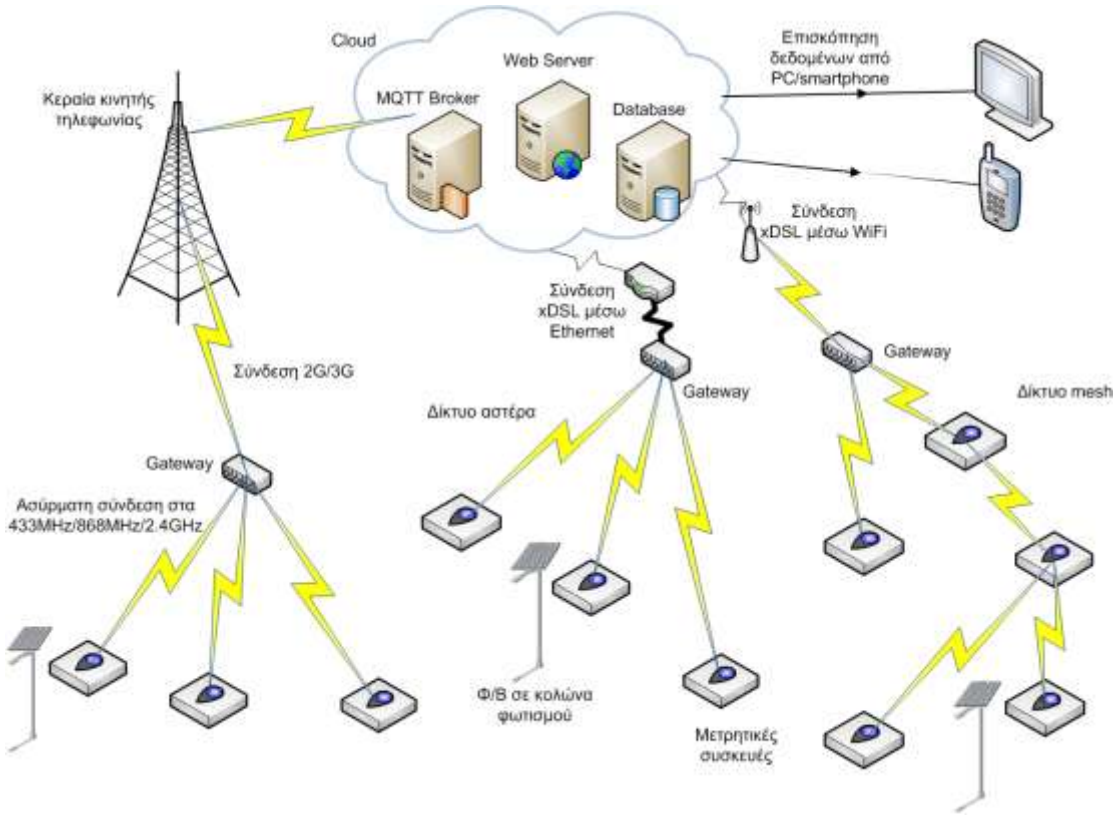
Για λόγους πληρότητας αναφέρουμε ότι μια πρόσθετη πλακέτα Ethernet για το Arduino Uno κοστίζει 15€-30€ , ενώ μια αντίστοιχη με δυνατότητα σύνδεσης στο δίκτυο 3G για Arduino Yun κοστίζει 30€.

Κεφάλαιο 4. Σχεδιασμός συστήματος

Σε αυτό το κεφάλαιο παρουσιάζεται αναλυτικά η ολοκληρωμένη σχεδίαση του συστήματος που προτείνεται.

Αρχιτεκτονική του συστήματος

Στο παρακάτω σχήμα παρουσιάζεται η αρχιτεκτονική του ασύρματου μετρητικού συστήματος ενέργειας. Ο ρόλος των μετρητικών συσκευών είναι να συλλέγουν τα επιθυμητά δεδομένα από το περιβάλλον και να τα αποστέλλουν ασύρματα στις πύλες μέσω των συχνοτήτων των 433MHz/868MHz/2.4GHz. Τα υποδίκτυα που αποτελούνται από πύλες και μετρητικές συσκευές βρίσκονται σε τοπολογία αστέρα ή πλέγματος. Στη συνέχεια οι πύλες στέλνουν τα δεδομένα στο cloud μέσω σύνδεσης xDSL, η οποία γίνεται είτε ενσύρματα είτε μέσω WiFi, ή μέσω του δικτύου κινητής τηλεφωνίας. Αυτά τα δεδομένα αποστέλλονται σε έναν MQTT broker για ενημέρωση του χρήστη σε πραγματικό χρόνο και αποθηκεύονται σε μια βάση δεδομένων για μελλοντική επεξεργασία/αξιοποίηση. Διατίθεται επίσης γραφικό περιβάλλον μέσω του οποίου ο χρήστης μπορεί να έχει πρόσβαση στα δεδομένα του συστήματος μέσω προσωπικού υπολογιστή ή smartphone/tablet.



Εικόνα 12 Αρχιτεκτονική του ασύρματου μετρητικού συστήματος ενέργειας

Σχεδίαση ασύρματης επικοινωνίας μετρητικών συσκευών-πυλών

Όπως φαίνεται στο παραπάνω σχήμα, η διασύνδεση των μετρητικών συσκευών με τις πύλες γίνεται ασύρματα στις συχνότητες 433MHz/866MHz/2.4GHz σε συνδεσμολογία αστέρα με τα modules nRF24L01+και RFM69HW, για τα οποία θα σχεδιαστεί ένα κοινό μήνυμα, το οποίο αναλύεται παρακάτω, έτσι ώστε οι μετρητικές συσκευές να το κατασκευάζουν και να το στέλνουν και οι πύλες που το λαμβάνουν να επεξεργάζονται τις πληροφορίες του με τον ίδιο τρόπο ανεξάρτητα από το πώς αποστέλλεται/λαμβάνεται. Εναλλακτικά, προτείνεται ένα δίκτυο mesh (αξιοποιώντας για παράδειγμα το πρωτόκολλο Zigbee), το οποίο όμως δεν μελετήθηκε στα πλαίσια της διπλωματικής εργασίας.

Οι μετρητικές συσκευές αποτελούνται από συσκευές Arduino στις οποίες είναι συνδεδεμένοι διάφοροι αισθητήρες καθώς και μία ασύρματη μονάδα αποστολής δεδομένων (nRF24L01+ ή RFM69HW). Κάθε μετρητική συσκευή χαρακτηρίζεται από μια μοναδική ταυτότητα κόμβου nodeID, η οποία την ξεχωρίζει από κάθε άλλη στο ίδιο δίκτυο. Τα σημεία που τοποθετούνται είναι σταθερά -εγκαθίστανται σε θέσεις που ο Δήμος έχει εγκαταστήσει Φ/Β (π.χ. κολώνες φωτισμού). Κατά την εγκατάσταση είναι επιθυμητό να γνωρίζουμε τις γεωγραφικές συντεταγμένες τους. Τόσο αυτές όσο και το nodeID αποθηκεύονται στη βάση δεδομένων του συστήματος και τοπικά στην πύλη με την οποία επικοινωνούν τα δεδομένα τους.

Επίσης κάθε μία από αυτές τις συσκευές έχει τη δυνατότητα να παρακολουθεί την καταναλισκόμενη και παραγόμενη ενέργεια, να παρακολουθεί τη φωτεινότητα του περιβάλλοντος και την υπολειπόμενη ενέργεια της μπαταρίας. Προαιρετικά θα συμπεριλαμβάνει και αισθητήρες που θα παρακολουθούν μετεωρολογικές μεταβολές ή/και ένα σύστημα ανίχνευσης πυρκαγιάς. Ενδεικτικά αναφέρουμε τέσσερις (4) διαφορετικούς τύπους συσκευών (deviceType) που αναλαμβάνουν διαφορετικά σετ μετρήσεων:

deviceType 0:

- Ένδειξη μπαταρίας
- παραγόμενη ισχύς από το Φ/Β
- καταναλισκόμενη ισχύς από το λαμπτήρα
- φωτεινότητα

deviceType 1:

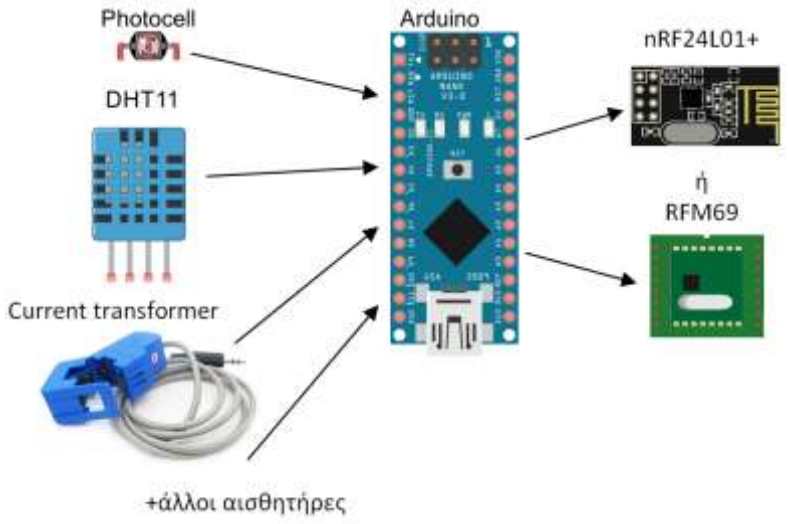
- Ένδειξη μπαταρίας
- παραγόμενη ισχύς από το Φ/Β
- καταναλισκόμενη ισχύς από το λαμπτήρα
- φωτεινότητα
- Θερμοκρασία
- σχετική υγρασία
- Αισθητήρας βροχής

deviceType 2:

- Ένδειξη μπαταρίας
- παραγόμενη ισχύς από το Φ/Β
- καταναλισκόμενη ισχύς από το λαμπτήρα
- φωτεινότητα
- υπεριώδης ακτινοβολία
- υπέρυθρη ακτινοβολία

deviceType 3:

- Ένδειξη μπαταρίας
- παραγόμενη ισχύς από το Φ/Β
- καταναλισκόμενη ισχύς από το λαμπτήρα
- φωτεινότητα
- Θερμοκρασία
- σχετική υγρασία
- Αισθητήρας βροχής
- υπεριώδης ακτινοβολία
- υπέρυθρη ακτινοβολία



Εικόνα 13 Σχεδίαση μετρητικής συσκευής

Η ισχύς στη μεριά του Arduino δίνεται ως float αριθμός δηλαδή αναπαρίσταται από 4 bytes. Για να αποφύγουμε να στείλουμε ένα πραγματικό αριθμό 4 bytes είναι προτιμότερο να στείλουμε τη στιγμιαία ισχύ πολλαπλασιασμένη επί 100 ώστε να προκύψει ακέραιος αριθμός 2 bytes. Έπειτα η πύλη θα τον διαιρέσει με 100 και θα στείλει στο cloud την πραγματική τιμή με ακρίβεια δύο δεκαδικών. Καθώς αναμένουμε μέγιστη μετρούμενη ισχύ περίπου στα 600W, οπότε ένα εύρος τιμών 0-60000 μπορεί να μετρηθεί με 2 bytes.

Ως float αριθμοί περιγράφονται και η θερμοκρασία και η σχετική υγρασία. Για τη μεν θερμοκρασία αναμένουμε να έχει εύρος από -40-125°C ενώ η υγρασία από 0-100%. Με την ίδια λογική θα στείλουμε αυτές τις ενδείξεις πολλαπλασιασμένες επί 100 για να επιτύχουμε μετέπειτα ακρίβεια εκατοστού.

Η φωτεινότητα, η ένδειξη βροχής και οι ακτινοβολίες UV και IR μετρούνται από την αναλογική είσοδο που μπορεί να έχει έως 1023 διαφορετικές στάθμες.

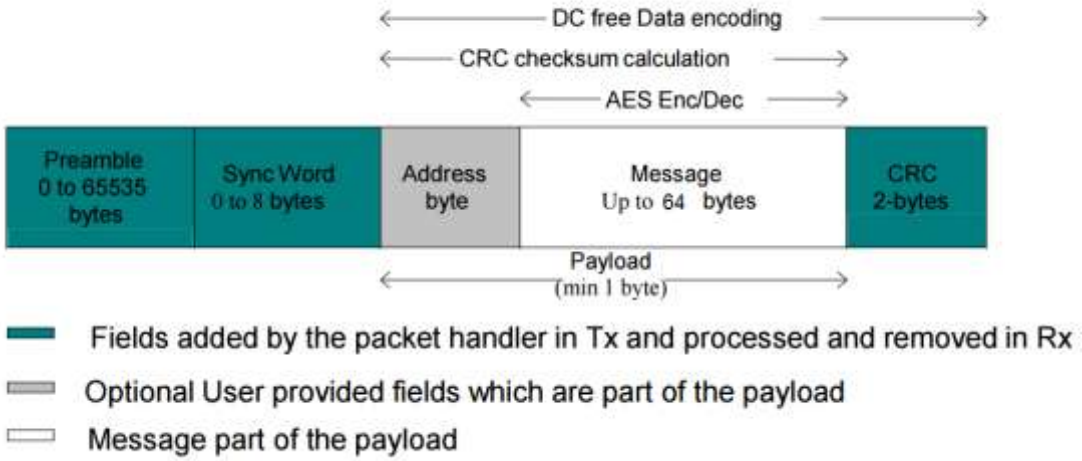
Η ένδειξη της μπαταρίας θα είναι ένα ακέραιο ποσοστό επί τοις εκατό.

Οι παραπάνω απαιτήσεις συνοψίζονται στον παρακάτω πίνακα.

	Εύρος	Αριθμός bytes	Arduino type	C/C++ type
nodeID	0 - 255	1	byte	uint8_t
deviceType	0 - 255	1	byte	uint8_t
Battery	0 - 100	1	byte	uint8_t
Production (x100)	0 - 60000	2	unsigned int	uint16_t
Consumption (x100)	0 - 60000	2	unsigned int	uint16_t
Luminosity	0 - 1024	2	unsigned int	uint16_t
Temperature (x100)	-4000 - 12500	2	int	int16_t
Humidity (x100)	0 - 10000	2	unsigned int	uint16_t
Rain	0 - 1024	2	unsigned int	uint16_t
UV	0 - 1024	2	unsigned int	uint16_t
IR	0 - 1024	2	unsigned int	uint16_t

Η επικοινωνία μεταξύ μετρητικών συσκευών και πυλών γίνεται αξιοποιώντας τις ελεύθερες συχνότητες των 434/868MHz και 2.4GHz, μέσω των πομποδεκτών RFM69W και το nRF24L01+. Θέλουμε το μήνυμα που θα χρησιμοποιηθεί να είναι συμβατό και με τις δύο αυτές τεχνολογίες ασύρματης επικοινωνίας. Αυτό το μήνυμα θα ενσωματωθεί στα προβλεπόμενα bytes δεδομένων που βρίσκονται στα πακέτα που χρησιμοποιούν τα δύο modules για την μεταξύ τους επικοινωνία. Σύμφωνα με τα datasheets τους [29] [33] το μεν μπορεί να υποστηρίξει payload μέχρι 64bytes και το δε μέχρι 32 bytes. Γι' αυτό το λόγο, το μήνυμα απαιτείται να σχεδιαστεί με τέτοιο τρόπο έτσι ώστε να μην ξεπερνά τα 32 bytes.

Εικόνα 14 Packet format του nRF24L01+



Εικόνα 15 Packet format του RFM69w

Σύμφωνα με τις παραπάνω προϋποθέσεις κατασκευάστηκε η δομή του μηνύματος το οποίο αποστέλλεται από τις μετρητικές συσκευές (Arduino), το οποίο ενσωματώνεται στο message/payload μερος των πακέτων των RF modules.. Αποτελείται από το nodeID της συσκευής, το deviceType που τη χαρακτηρίζει και έπειτα ακολουθούν οι επιμέρους μετρήσεις. Για τα διαφορετικά device types που ορίστηκαν παραπάνω θα έχουμε διαφορετική μορφή μηνύματος. Η πύλη από τη μεριά της ξέρει τι δεδομένα θα ακολυθήσουν αφότου διαβάσει το device type.

nodeID 1byte	deviceType 1byte	Battery 1byte	Production 2bytes	Consumption 2bytes	Luminosity 2bytes

Εικόνα 16 Δομή μηνύματος για το device type 0

nodeID 1byte	deviceType 1byte	Battery 1byte	Production 2bytes	Consumption 2bytes	Luminosity 2bytes	Temperature 2bytes	Humidity 2bytes	Rain 2bytes

Εικόνα 17 Δομή μηνύματος για το device type 1

nodeID 1byte	deviceType 1 byte	Battery 1 byte	Production 2 bytes	Consumption 2 bytes	Luminosity 2 bytes	UV 2 bytes	IR 2 bytes

Εικόνα 18 Δομή μηνύματος για το device type 2

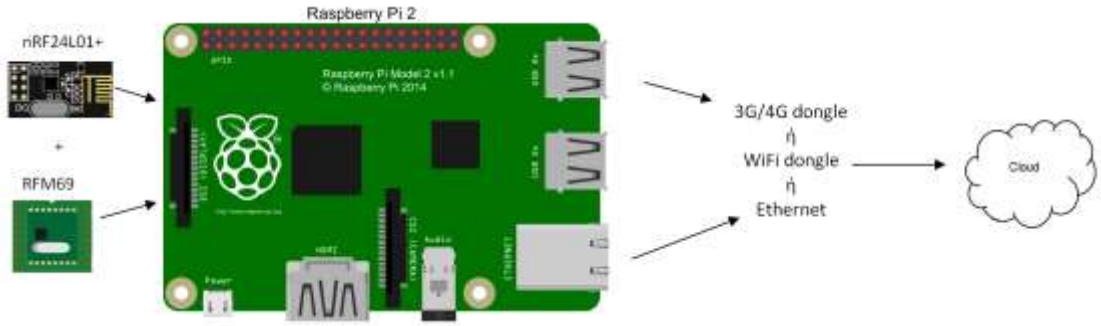
nodeID 1byte	deviceType 1byte	Battery 1byte	Production 2bytes	Consumption 2bytes	Luminosity 2bytes	Temperature 2bytes	Humidity 2bytes	Rain 2bytes	UV 2bytes	IR 2bytes

Εικόνα 19 Δομή μηνύματος για το device type 3

Αυτή η σχεδίαση επιτρέπει στο σύστημα να υποστηρίξει πάνω από 10 αισθητηρες ταυτόχρονα.

Αποστολή δεδομένων στο cloud

Η σύνδεση της πύλης με το Internet γίνεται είτε μέσω σύνδεσης DSL (μέσω Ethernet ή WiFi) είτε μέσω δικτύου κινητής τηλεφωνίας. Το ποια μέθοδο θα διαλέξουμε εξαρτάται από το αν στο σημείο στο οποίο θα τοποθετηθεί η πύλη υπάρχει υφιστάμενη σύνδεση xDSL (αφού η συγκεκριμένη επιλογή δεν θα επιβαρύνει το κόστος της λύσης).



Εικόνα 20 Σχεδίαση πύλης

Κάθε πύλη του δήμου έχει μια δική της ταυτότητα (gatewayID) η οποία ονοματίζει και το υποδίκτυο στο οποίο υπάγονται συγκεκριμένες μετρητικές συσκευές. Συγκεκριμένα η πύλη γνωρίζει τις συντεταγμένες κάθε συσκευής και όταν παραλαμβάνει δεδομένα τα συσχετίζει σύμφωνα με το nodeID του πακέτου που έχει ληφθεί. Διευκρινίζεται επίσης ότι μια συσκευή πύλη μπορεί να έχει αισθητήρες πάνω της και να έχει διπλό ρόλο, αλλά προτιμούμε να μην αποτελεί καθολική λύση για μείωση του κόστους.

Έπειτα, αναλόγως με το είδος των μετρήσεων που έχει λάβει τα επεξεργάζεται κατάλληλα δηλαδή φροντίζει να διαιρέσει τη θερμοκρασία, την υγρασία, την παραγόμενη και την καταναλισκόμενη ισχύ διά 100 ώστε να προκύψει η πραγματική μέτρηση. Στη συνέχεια αυτές οι πληροφορίες μαζί με την ταυτότητα της πύλης αποστέλλονται σε έναν MQTT broker και αποθηκεύονται σε μια βάση δεδομένων για περαιτέρω επεξεργασία ή/και παρουσίαση των μετρήσεων.

Το MQTT (Message Queue Telemetry Transport) είναι ένα πρωτόκολλο ανταλλαγής μηνυμάτων πάνω από TCP/IP. Χρησιμοποιείται κυρίως για απομακρυσμένες συνδέσεις όπου το εύρος ζώνης του δικτύου είναι περιορισμένο. Η προεπιλεγμένη θύρα είναι η 1883. Το πρωτόκολλο αυτό στηρίζεται στο publish/subscribe. Μία συσκευή κάνει publish τα δεδομένα σε κάποιο topic και όσες συσκευές θέλουν να λαμβάνουν αυτά τα δεδομένα κάνουν subscribe στο συγκεκριμένο topic. Για να γίνει αυτό χρειάζεται ένας broker, όπως είναι το RSMB, WebSphere MQ, HiveMQ, Apache Apollo, Apache ActiveMQ, RabbitMQ, MQTT.js, mosca, 2lemetry, GnatMQ, ενώ γνωστότερο είναι το MOSQUITTO, που είναι υπεύθυνο για την μεταφορά αυτών των μηνυμάτων στις εγγεγραμμένες συσκευές, το οποίο χρησιμοποιούμε κι εμείς. Η γνωστότερη εφαρμογή που χρησιμοποιεί το MQTT είναι το Facebook Messenger. [38]

Το MQTT ορίζει τρία επίπεδα του Quality of Service (QoS). Το QoS καθορίζει πόσο πολύ ο broker / client θα προσπαθήσει να εξασφαλίσει ότι το μήνυμα έχει ληφθεί. Τα μηνύματα

μπορούν να αποσταλούν σε οποιοδήποτε επίπεδο QoS, και οι πελάτες μπορούν να προσπαθήσουν να εγγραφούν σε topic σε κάθε επίπεδο QoS. Αυτό σημαίνει ότι ο πελάτης επιλέγει το μέγιστο QoS που θα λάβει. Για παράδειγμα, εάν ένα μήνυμα που δημοσιεύεται στο QoS 2 και ένας πελάτης είναι συνδρομητής με QoS 0, το μήνυμα θα πρέπει να παραδοθεί σε αυτό τον πελάτη με QoS 0. Αν ένας δεύτερος πελάτης έχει επίσης εγγραφεί στο ίδιο θέμα, αλλά και με QoS 2 θα λάβει το ίδιο μήνυμα αλλά με QoS 2. Για ένα δεύτερο παράδειγμα, αν ένας πελάτης είναι συνδρομητής με QoS 2 και ένα μήνυμα δημοσιεύτηκε στις QoS 0, ο πελάτης θα λάβει για QoS 0. [39]

Τα υψηλότερα επίπεδα QoS είναι πιο αξιόπιστα, αλλά έχουν μεγαλύτερη καθυστέρηση και χρειάζονται μεγαλύτερο εύρος ζώνης.

- 0: Ο broker / client θα παραδώσει το μήνυμα μία φορά, χωρίς καμία επιβεβαίωση.
- 1: Ο broker / client θα παραδώσει το μήνυμα, τουλάχιστον μία φορά, με την επιβεβαίωση που απαιτείται.
- 2: Ο broker / client θα παραδώσει το μήνυμα ακριβώς μια φορά, χρησιμοποιώντας μια χειραψία τεσσάρων σταδίων.

Στη συνέχεια μέσω Javascript δημιουργούμε στοιχειώδης clients που θα «ακούν» τα topic στα οποία δημοσιεύονται μηνύματα, και αναπαριστούμε τα δεδομένα σε διαγράμματα πραγματικού χρόνου.

Για τις ανάγκες των εφαρμογής χρειάστηκε να υλοποιήσουμε και να σχεδιάσουμε μια σχεσιακή βάση δεδομένων η οποία θα αποθηκεύει όλες αυτές τις πληροφορίες που συλλέγονται, για μελλοντική πρόσβαση σε αυτές. Υπήρχαν πολλές λύσεις που μπορούσαμε να διαλέξουμε όπως MICROSOFT SQL SERVER, ORACLE, POSTGRESQL. Αλλά εμείς επιλέξαμε MySQL λόγω της ευχρηστίας και της δυναμικότητας που προσφέρει. Μη ξεχνάμε ότι η MySQL είναι δημοφιλής βάση δεδομένων για διαδικτυακά προγράμματα και ιστοσελίδες. Χρησιμοποιείται σε κάποιες από τις πιο διαδεδομένες διαδικτυακές υπηρεσίες, όπως το YouTube, η Wikipedia, το Google, το Facebook και το Twitter.

Η βάση δεδομένων έχει καταγεγραμμένες όλες τις μετρητικές συσκευές, σε ποιο δήμο και ποιο υποδίκτυο ανήκουν, τις συντεταγμένες τους και ποιες μετρήσεις αποστέλλουν. Στη συνέχεια μέσω php scripts ο server επικοινωνεί με τη βάση δεδομένων και ανασύρει δεδομένα ώστε αυτά να αναπαρασταθούν γραφικά στο χρήστη. Συγκεκριμένα μέσω αυτών των scripts μπορούμε να «διαβάσουμε» δεδομένα ανά δήμο, υποδίκτυο, μετρητική συσκευή σε επίπεδο ώρας, ημέρας, εβδομάδας, μήνα, κοκ.

Κεφάλαιο 5. Υλοποίηση πρωτοτύπου

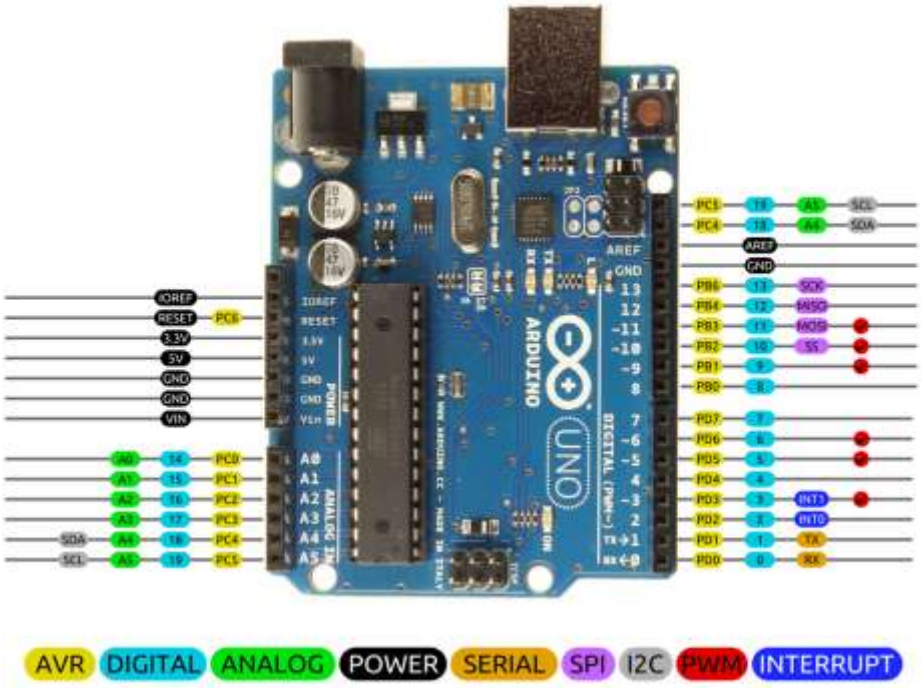
Στα πλαίσια της αυτής της διπλωματικής εργασίας κατασκευάστηκε ένα πρωτότυπο, τα χαρακτηριστικά του οποίου, ο τρόπος λειτουργίας του και η μεθοδολογία κατασκευής του παρουσιάζεται παρακάτω.

Μετρητικές συσκευές

Στο πρωτότυπο κατασκευάστηκαν δύο μετρητικές συσκευές οι οποίες βασίστηκαν πάνω στα συστήματα μικροελεγκτών Arduino Uno και Arduino Nano. Στο Arduino Uno τοποθετήθηκε ο αισθητήρας ρεύματος και μια φωτοαντίσταση, ενώ στο Arduino Nano τοποθετήθηκε ο αισθητήρας θερμοκρασίας/υγρασίας DHT11 και μια φωτοαντίσταση. Παρακάτω παρουσιάζονται αυτές οι δύο συσκευές, καθώς και η διασύνδεσή τους με τους επιμέρους αισθητήρες και διατάξεις.

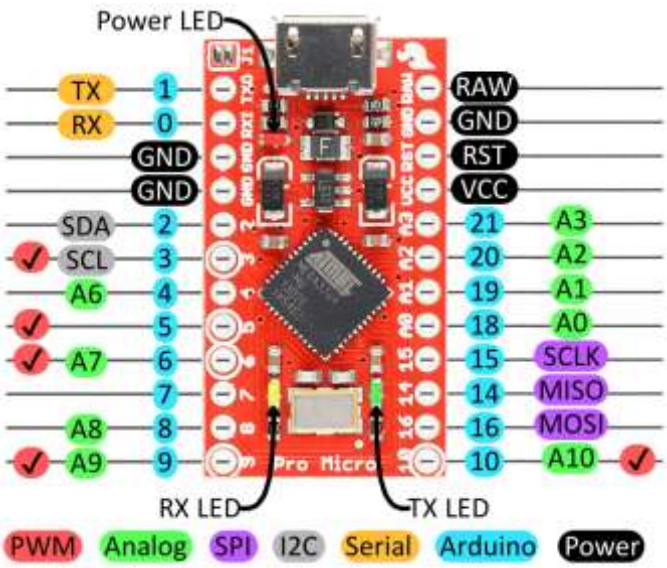
Arduino Uno & Nano

Το Arduino Uno είναι μια πλατφόρμα βασισμένη στον ATmega 328. Περιέχει 14 ψηφιακούς ακροδέκτες εισόδου-εξόδου («I/O»), 6 αναλογικές εισόδους και 6 ψηφιακές εξόδους και έναν ταλαντωτή των 16 MHz. Η σύνδεση γίνεται με «USB» καλώδιο, ενώ υπάρχει και υποδοχή σύνδεσης με ρεύμα, καθώς και δυνατότητα εντός του κυκλώματος, σειριακού προγραμματισμού-ICSP («In-Circuit Serial Programming») και ένα κουμπί επανεκκίνησης (reset) σε περίπτωση που βραχυκυκλώσει η πλατφόρμα. [40]



Εικόνα 21 Χάρτης ακροδεκτών Arduino Uno Rev.3

Το Arduino Nano διαφέρει από το Arduino Uno ως προς το μέγεθος, που το καθιστά βολικό για χρήση σε breadboard για την ανάπτυξη πρωτοτύπων. Και αυτό βασίζεται στον ATmega 328. Περιέχει 14 ψηφιακούς ακροδέκτες εισόδου-εξόδου («I/O»), 8 αναλογικές εισόδους και 6 ψηφιακές εξόδους και έναν ταλαντωτή των 16 MHz. Η σύνδεση γίνεται με «Mini-B USB» καλώδιο, ενώ υπάρχει και υποδοχή σύνδεσης με ρεύμα, καθώς και δυνατότητα εντός του κυκλώματος, σειριακού προγραμματισμού-ICSP («In-Circuit Serial Programming») και ένα κουμπί επανεκκίνησης (reset) σε περίπτωση που βραχυκυκλώσει η πλατφόρμα. [41]



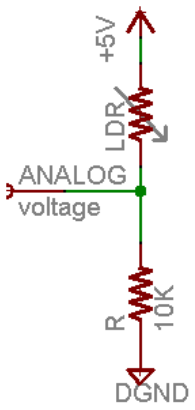
Εικόνα 22 Χάρτης ακροδεκτών Arduino Nano

Διασύνδεση αισθητήρων και διατάξεων

Διάταξη μέτρησης φωτεινότητας

Για να μετρήσουμε φωτεινότητα θα χρησιμοποιήσουμε μια φωτοαντίσταση (photocell), συνδέοντάς τη από τη μία μεριά με την τροφοδοσία των 5V και από την άλλη μεριά με τη γείωση μέσω μιας αντίστασης 10kΩ. Τότε το σημείο μεταξύ της σταθερής αντίστασης και της μεταβλητής αντίστασης συνδέεται με την αναλογική είσοδο A0 του Arduino. [42]

Ο διαιρέτης τάσης που κατασκευάστηκε θα δίνει στην αναλογική είσοδο τάση ίση με $V_{in} = V_{cc} * \frac{R}{R+R_{photocell}}$.



Σχήμα 1 Συνδεσμολογία διάταξης μέτρησης φωτεινότητας

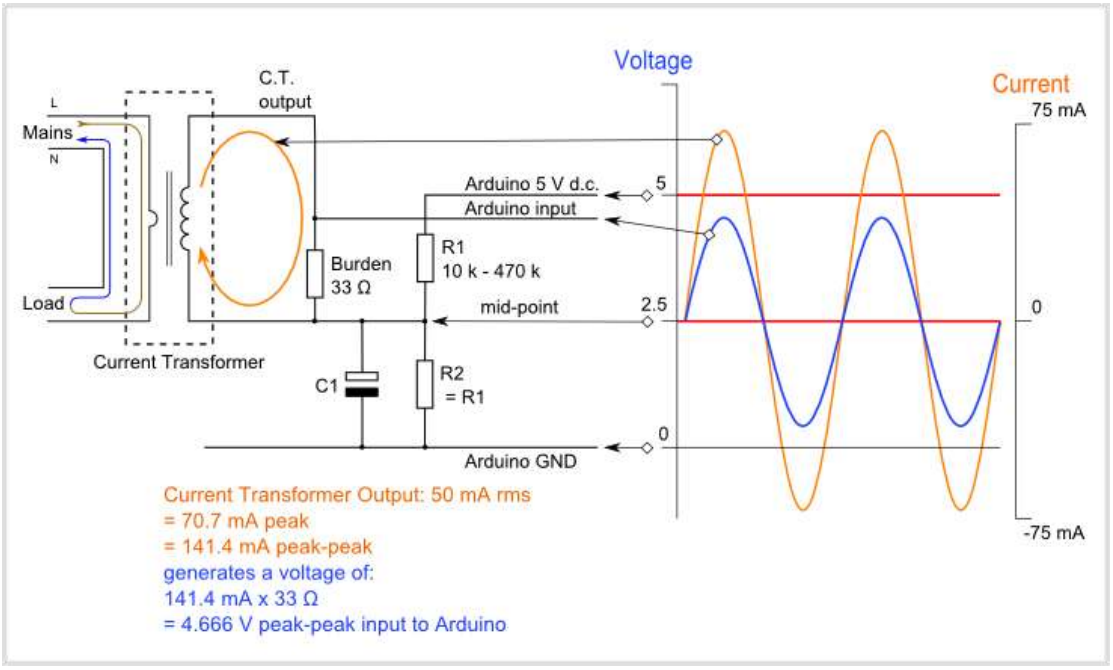
Διάταξη μέτρησης κατανάλωσης ηλεκτρικής ενέργειας

Ο αισθητήρας ρεύματος που χρησιμοποιήθηκε σε αυτή τη διπλωματική είναι ο μετασχηματιστής ρεύματος SCT-013 της εταιρίας YHDC.

Ο πυρήνας του χωρίζεται και με αυτό τον τρόπο ο αισθητήρας “αγκαλιάζει” το καλώδιο που θέλουμε να μετρήσουμε το ρεύμα που το διαπερνά. Τέτοιου τύπου αισθητήρες ονομάζονται non-invasive και είναι ιδανικοί για απλές εφαρμογές. [43]

Εικόνα 23 Ο αισθητήρας ρεύματος SCT-013

Όπως κάθε μετασχηματιστής έχει το πρωτεύον τύλιγμα, πυρήνα από φερρίτη και το δευτερεύον τύλιγμα. Το πρωτεύον πηνίο θα είναι η φάση ή ο ουδέτερος του καλωδίου που περνάει από την τρύπα του αισθητήρα, αλλά όχι και τα δύο μαζί διότι τα διαρρέουν ίσα ρεύματα αλλά σε διαφορετικές κατευθύνσεις, οπότε θα δημιουργηθούν δύο ίσα και αντίθετα μαγνητικά πεδία και θα ακυρώσουν το ένα το άλλο οπότε δεν θα έχουμε έξοδο στη δευτερεύον πηνίο. Για κάθε 100A στο πρωτεύον πηνίο επάγουν 50mA στο δευτερεύον, άρα το δευτερεύον πηνίο αποτελείται από 100/0.050=2000 σπείρες



Σχήμα 2 Συνδεσμολογία αισθητήρα ρεύματος με Arduino

Παράλληλα με το δευτερεύον πηνίο θα τοποθετήσουμε μια αντίσταση φορτίου ώστε να δώσει ανάλογη τάση, μέγεθος το οποίο μπορεί να μετρηθεί από τον Analog to Digital Converter του Arduino. Η αντίσταση φορτίου επιλέγεται σύμφωνα με το επιθυμητό current range όπως περιγράφεται παρακάτω.

Επιλέγουμε τη μέγιστη ενεργό τιμή του ρεύματος που θέλουμε να μετρήσουμε $I_{in,RMS}$.

Αυτή είναι η rms τιμή οπότε για το στιγμιαίο μέγιστο ρεύμα

$$I_{max,peak} = I_{in,RMS} * \sqrt{2}$$

Στην έξοδο του δευτερεύοντος πηνίου θα έχουμε

$$I_{out,max} = \frac{I_{max,peak}}{2000} = I_{in,RMS} * \frac{\sqrt{2}}{2000}$$

Ο ADC του Arduino δεχεται είσοδο από 0 ως 5V άρα η τάση αναφοράς για εμάς θα είναι $V_{ref} = \frac{5V}{2} = 2{,}5V$.

Επομένως, η ιδανική αντίσταση φορτίου θα είναι $R_{burden,ideal} = \frac{V_{ref}}{Iout,max}$

Συγκεντρωτικά η αντίσταση υπολογίζεται από τον παρακάτω τύπο

$$R_{burden,ideal} = \frac{V_{ref}}{Iout,max} = \frac{2.5 * 2000}{I_{in,RMS} * \sqrt{2}} = \frac{5000}{I_{in,RMS} * \sqrt{2}}$$

Άρα για να μετρήσουμε I=100A χρειαζόμαστε αντίσταση 35Ω που δεν υπάρχει οπότε επιλέγουμε την αμέσως μικρότερη που είναι 33Ω.

Αν και το σήμα του ρεύματος ταλαντώνεται γύρω από το 0, το Arduino δέχεται ως είσοδο μόνο θετική τάση από 0 ως 5 V. Γι αυτό θα φροντίσουμε η ταλάντωση της τάσης της αντίστασης φορτίου να γίνεται γύρω από τα 2.5V τοποθετώντας ένα διαιρέτη τάσης ανάμεσα στα 5V που δίνει το Arduino και τη γείωσή του. Ο διαιρέτης τάσης θα αποτελείται από 2 αντιστάσεις 10kΩ. Αν διαλέξουμε μεγαλύτερες αντιστάσεις τότε μειώνεται η καταναλισκόμενη ενέργεια, ιδιότητα χρήσιμη αν η συσκευή μέτρησης τροφοδοτείται από μπαταρίες.

Επίσης συνδέουμε ένα πυκνωτή 10 μF ανάμεσα στη γη και το σημείο αναφοράς που θα σταθεροποιεί την τάση.

Άλλες διατάξεις

Οι αισθητήρες βροχής, UV ακτινοβολίας και πυρκαγιάς συνδέονται σε αναλογικές εισόδους του Arduino.

Διάταξη ασύρματης επικοινωνίας μέσω του RFM69

RFM69 →	*Arduino Uno*
NSS →	D10 (μέσω διαιρέτη τάσης 2.2K/1K)
MOSI →	D11 (μέσω διαιρέτη τάσης 2.2K/1K)
MISO →	D12 (απευθείας)
SCK →	D13 (μέσω διαιρέτη τάσης 2.2K/1K)
GND →	GND
3.3 →	3.3V
ANA →	σύρμα 17.3cm στον αέρα (ως κεραία λ/2)

Διάταξη ασύρματης επικοινωνίας μέσω του nRF24L01+

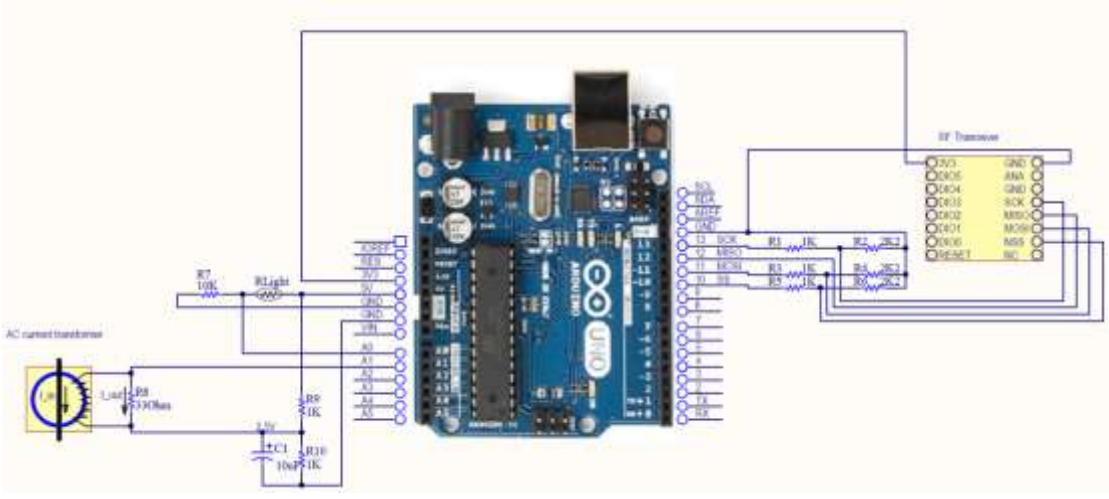
Το nRF24L01+ διαθέτει 8 pins για αλληλεπίδραση με το σύστημα, και αυτοί είναι οι εξής: Vcc, GND (ground), IRQ (interrupt), CE (chip enable) και οι 4 pins για την διεπαφή με το SPI (CSN, SCK, MISO, MOSI). Τα I/O pins είναι σχεδιασμένα ώστε να αντέχουν τάση ως 5V. Παρόλα αυτά στο datasheet αναφέρεται ότι το εύρος καλής λειτουργίας για το Vcc είναι 1.9V ως 3.6V.

Η διεπαφή SPI χρησιμοποιεί 4 pins, CSN, SCK (serial clock), MISO (master in-slave out) και MOSI (master out-slave in) για αποστολή και λήψη δεδομένων. Το CSN (chip select not) pin είναι ενεργό-χαμηλά, και συνήθως διατηρείται ψηλά. Όταν το pin αυτό γίνεται χαμηλό, το 24L01 ξεκινάει να παρακολουθεί την SPI θύρα του για δεδομένα και τα επεξεργάζεται κατάλληλα. Τα υπόλοιπα pins συνδέονται σύμφωνα με τα ονόματά τους με τα αντίστοιχα pins της διεπαφής SPI.

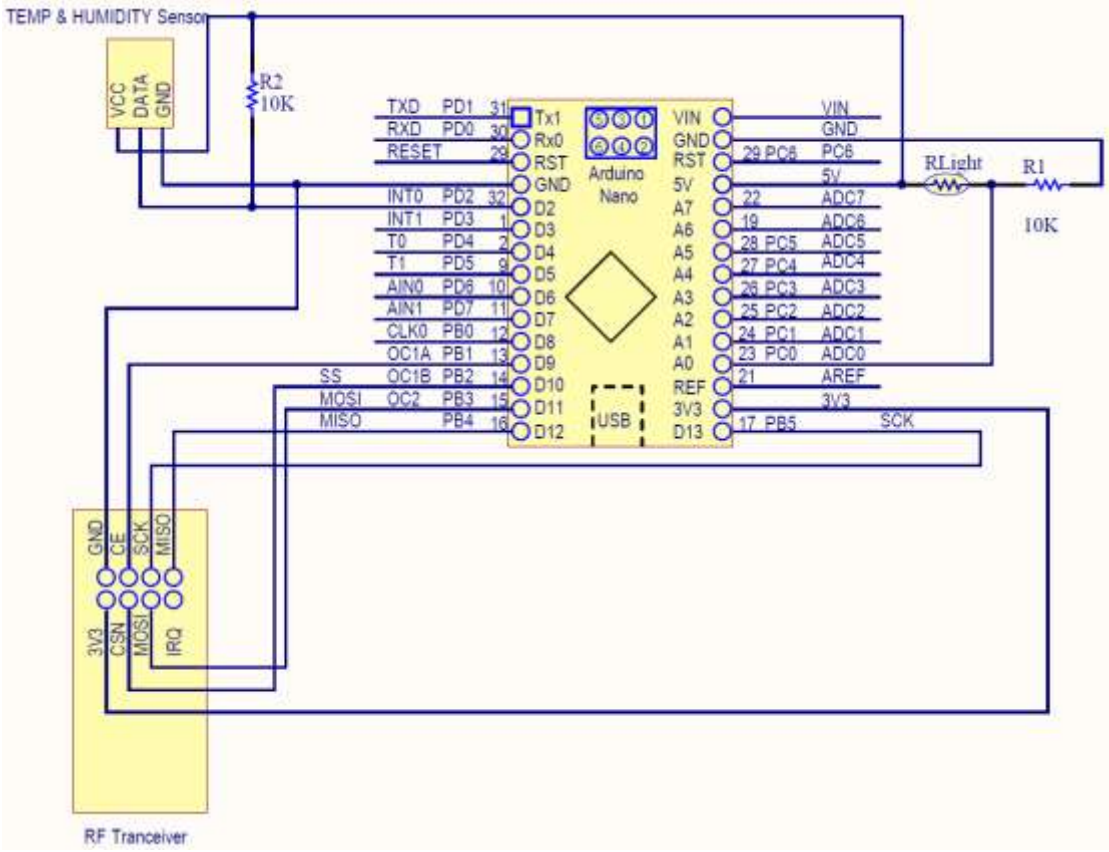
Τα δύο εναπομείναντα pins είναι το CE και το IRQ. Το CE χρησιμοποιείται για τον έλεγχο της αποστολής και της λήψης δεδομένων όταν το 24L01 βρίσκεται σε λειτουργία πομπού (TX mode) και σε λειτουργία δέκτη (RX mode) αντίστοιχα. Το IRQ είναι το pin διακοπής και είναι ενεργό χαμηλά. Υπάρχουν τρεις εξωτερικές διακοπές που μπορούν να κάνουν αυτό το pin χαμηλό όταν ενεργοποιούνται.

nRF24L01+	→	*Arduino Nano*
CE	→	D9
CSN	→	D10
MOSI	→	D11
MISO	→	D12
SCK	→	D13
GND	→	GND
VCC	→	3.3V

Τελική διασύνδεση hardware



Σχήμα 3 Σχηματικό με τις περιφερειακές συνδέσεις του Arduino Nano



Σχήμα 4 Σχηματικό με τις περιφερειακές συνδέσεις του Arduino Nano

Περιβάλλον ανάπτυξης

Ο προγραμματισμός των Arduino έγινε σε περιβάλλον Windows7 μέσω του Arduino IDE, μία πολυπλατφορμική εφαρμογή γραμμένη σε Java και βασίζεται στο περιβάλλον της γλώσσας προγραμματισμού Processing. Είναι ένα ολοκληρωμένο περιβάλλον ανάπτυξης και είναι απλούστερο και φιλικότερο προς το χρήστη, σε αντίθεση με παρόμοια περιβάλλοντα ανάπτυξης όπως το Eclipse και το Visual Studio. Περιέχει έναν επεξεργαστή πηγαίου κώδικα (editor), μεταγλωτιστή (compiler) και serial monitor που παρακολουθεί τις επικοινωνίες της συσκευής (USB) και αναλαμβάνει να στείλει αλφαριθμητικά της επιλογής μας στο Arduino μέσω αυτής.

Ακολουθώντας τα παρακάτω βήματα προγραμματίζουμε τις συσκευές [44]:

1. Συνδέουμε το Arduino με τον υπολογιστή μέσω ενός καλωδίου USB τύπου A/B. Τα Arduino Nano και Uno τροφοδοτούνται μέσω αυτού του καλωδίου. Ένα πράσινο LED με την ένδειξη PWR θα ανάψει. Με αυτή τη σύνδεση τα Windows7 κατεβάζουν τους απαραίτητους drivers αυτόματα.

2. Κατεβάζουμε το Arduino IDE από το επίσημο website του Arduino https://www.**arduino**.cc/en/Main/Software . Το εγκαθιστούμε και έπειτα το εκτελούμε.
3. Κατεβάζουμε τις βιβλιοθήκες DHT11, EmonLib, RFM69 από το Tools > Libraries > Manage Libraries. Αυτές οι βιβλιοθήκες είναι απαραίτητες για να δουλέψουμε με τα περιφερειακά modules DHT11, YHDC SCT013 και RFM69 αντίστοιχα. Για το nRF24L01+ εγκαθιστούμε χειροκίνητα τη βιβλιοθήκη που βρίσκεται σε αυτό το link https://github.com/TMRh20/RF24 . Την κατεβάζουμε κατεβάζοντάς τη και κάνοντας extract το περιεχόμενο του zip στο My Documents\Arduino\libraries .
4. Κάνουμε copy-paste τους πηγαίους κώδικες του Παραρτήματος Β, και τους αποθηκεύουμε σε φάκελο της επιλογής μας. Κάνουμε κλικ στο κουμπί «Verify» ώστε να μεταφραστεί το πρόγραμμα.
5. Επιλέγουμε την επιθυμητή συσκευή από το μενού Tools > Board και την εικονική σειριακή θύρα(COM#) στην οποία την έχουμε συνδέσει. Αν δε γνωρίζουμε ποια είναι αυτή, αποσυνδέουμε τη συσκευή, ξανανοίγουμε το μενού, το λήμμα που λείπει αντιστοιχεί στη θύρα του Arduino μας, και επανασυνδέουμε.
6. Κάνουμε κλικ στο κουμπί «Upload». Περιμένουμε μερικά δευτερόλεπτα. Τα LEDs Rx και Tx του Arduino θα αναβοσβήνουν. Αν το ανέβασμα του προγράμματος έγινε σωστά θα εμφανιστεί ένα μήνυμα «Done uploading».
7. Ανοίγουμε το Serial Monitor πατώντα Ctrl+Shift+M ώστε να διαβάσουμε τα μηνύματα που τυπώνονται από το πρόγραμμα.

Αφότου φορτώσουμε το sketch nrf24_endpoint.ino (σελ.89) στο Arduino Nano το Serial Monitor περιμένουμε μηνύματα σαν τα παρακάτω σε περίπτωση επιτυχούς ή ανεπιτυχούς αποστολής αντίστοιχα.

```
nrf24_endpoint

Message (20 bytes) | nodeID: 1, devID: 1
luminosity: 428
temp(x100): 2600, humi(x100): 3700
Now sending theData
Got blank response. round-trip delay: 728 microseconds

Message (20 bytes) | nodeID: 1, devID: 1
luminosity: 445
temp(x100): 2600, humi(x100): 3700
Now sending theData
Sending failed.
```

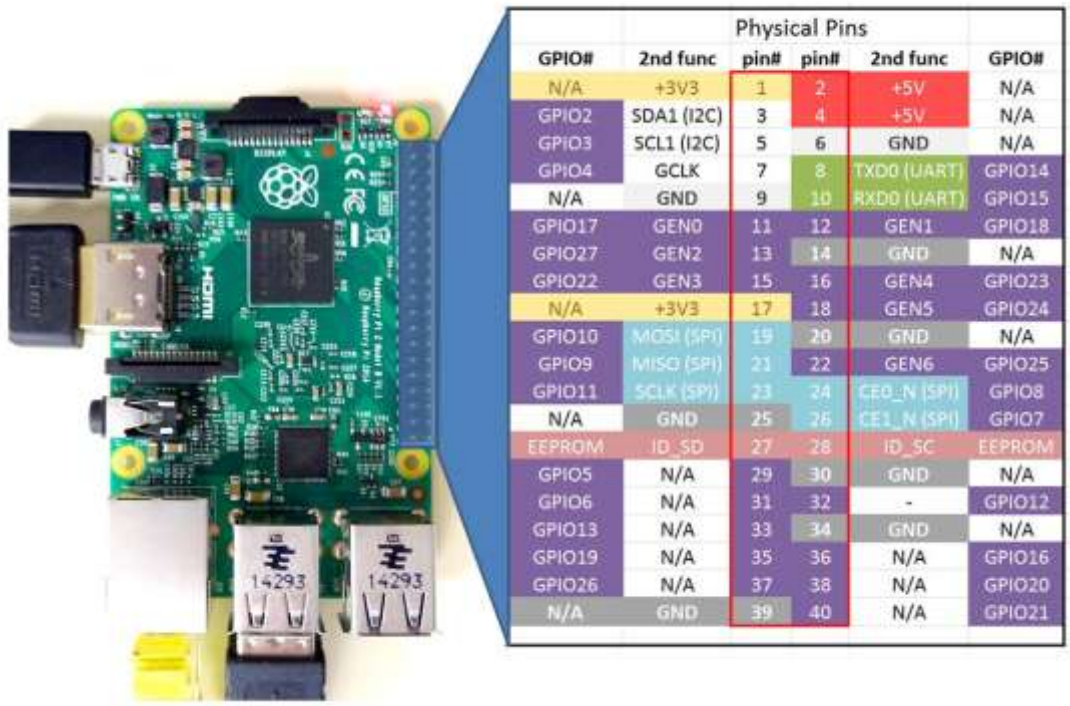
Αφότου φορτώσουμε το sketch rfm69_endpoint.ino (σελ. 85) στο Arduino Nano το Serial Monitor περιμένουμε μηνύματα όπως παρακάτω.

```
rfm69_endpoint

Initializing Radio...OK!!

Transmitting at 868 Mhz...
Message (20 bytes) | nodeID: 2, devID: 0
consumption(x100): 8258, luminosity: 814
```

Πύλη

Raspberry 2 Model B

Το Raspberry Pi 2 Model B διαθέτει 1GB μνήμη RAM, CPU στα 900MHz, τέσσερις θύρες USB και μία 100mb θύρα Ethernet. Επίσης υπάρχει η δυνατότητα σύνδεσης οθόνης μέσω HDMI ή RCA καλωδίου, ενώ η έξοδος του ήχου επιτυγχάνεται μέσω HDMI ή αναλογικά από την θύρα 3.5mm. Για την τροφοδοσία του απαιτείται παροχή ενέργειας 5V, ενώ η κατανάλωσή του είναι πολύ μικρή, περίπου 800mA (4W). Αυτή μπορεί να επιτευχθεί απλά συνδέοντας το Raspberry Pi σε κάποια παροχή ρεύματος USB στη θύρα micro-USB και θα εκκινήσει αμέσως. Τέλος, συνδέεται μια SD CARD για την εκκίνηση και μακροχρόνια αποθήκευση, στην οποία υπάρχει εγκατεστημένο όλο το απαραίτητο λογισμικό για την λειτουργία του Raspberry Pi.



Εικόνα 24 Χάρτης ακροδεκτών Raspberry Pi 2

Στήσιμο Raspberry Pi 2

Παρακάτω ακολουθεί η διαδικασία εκκίνησης του Raspberry Pi 2 και η εγκατάσταση λειτουργικού συστήματος σε αυτό. [45]

Το λειτουργικό που επιλέχθηκε είναι το Rasbian καθώς είναι πολύ κοντά στην έκδοση Debian για σταθερούς υπολογιστές και έτσι μπορεί χρησιμοποιηθεί εύκολα και χωρίς περαιτέρω μελέτη. Για να εγκαταστήσουμε το λειτουργικό Raspbian στο Raspberry Pi θα πρέπει αρχικά να αντιγράψουμε το αρχείο wheezy-raspbian.img, που είναι διαθέσιμο στην επίσημη ιστοσελίδα του raspberry pi, στην κάρτα SD που διαθέτουμε.

Βάζουμε την κάρτα σε έναν υπολογιστή που στην συγκεκριμένη περίπτωση τρέχει Windows 7 και με το SDFormatter κάνουμε format στην κάρτα έχοντας την επιλογή size adjustment στο ON έτσι να γίνει προσαρμογή στο μέγεθος της κάρτας. Έπειτα, με το Win32DiskImager αντιγράφουμε το αρχείο image στην κάρτα. Οι δύο παραπάνω διαδικασίες μπορούν να γίνουν και με άλλα αντίστοιχα προγράμματα που χρησιμοποιούνται για την διαμόρφωση καρτών SD.

Στη συνέχεια, βάζουμε την κάρτα SD στο Raspberry pi και αφού το έχουμε συνδέσει στο τοπικό μας δίκτυο το ανάβουμε. Οι αρχικές ρυθμίσεις του Raspberry Pi είναι η απόκτηση IP να γίνεται μέσω dhcp client και η δυνατότητα σύνδεσης ssh να είναι ενεργοποιημένη. Έτσι, βρίσκουμε την IP address του Raspberry pi, από τις συνδεδεμένες συσκευές στο Router, που από εδώ και στο εξής θα την αναφέρουμε ως <Pi IP>.

Συνδεόμαστε στην <Pi IP> με ssh με ένα πρόγραμμα όπως το Remote Desktop Connection. Αν όλα έχουν πάει καλά η σύνδεση είναι επιτυχής και εμφανίζεται στον υπολογιστή μας ένα παράθυρο με την λέξη "login as:". Το προκαθορισμένο username και password είναι pi και raspberry αντίστοιχα.

Για να εισέλθουμε στις ρυθμίσεις του Raspberry Pi εκτελούμε την εντολή sudo raspi-config (Επισημαίνεται ότι για τον χρήστη pi δεν χρειάζεται κωδικός για να εκτελεστεί μία εντολή με sudo).

Στη συνέχεια εκτελούμε την εντολή **`sudo apt-get update && sudo apt-get upgrade`** (απαντάμε με YES στο ερώτημα αν θέλουμε να συνεχίσουμε) έτσι ώστε να γίνουν όλες οι τελευταίες αναβαθμίσεις που είναι διαθέσιμες για το Raspberry pi καθώς επίσης και την εντολή **`sudo apt-get clean`** για να διαγραφούν τα προσωρινά αρχεία της εγκατάστασης.

Διασύνδεση hardware

Τα δύο RF modules συνδέονται παράλληλα στο RPi.

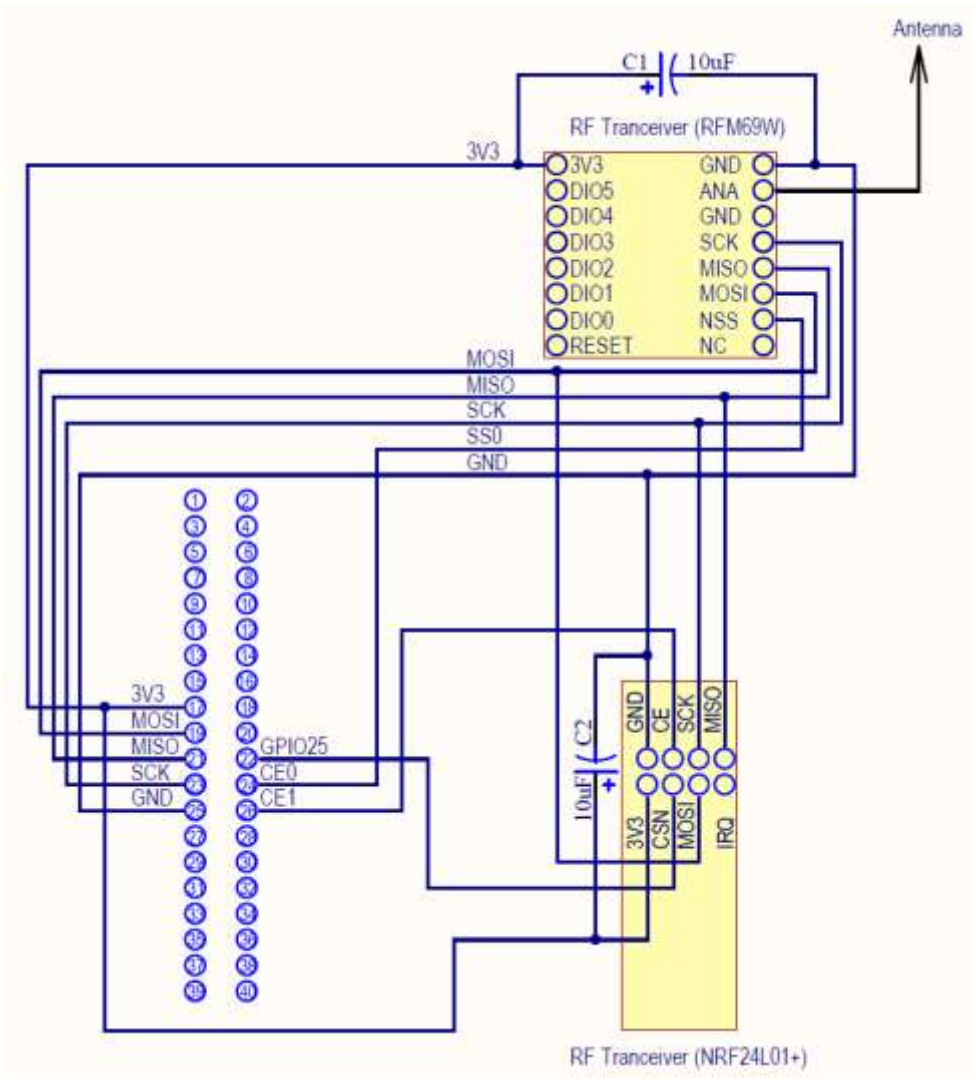
Παρακάτω ακολουθεί η επιμέρους διασύνδεση, ενώ ακολουθεί σχηματικό με το ολοκληρωμένο κύκλωμα.

RFM69w

RFM69 →	*Raspberry Pi2*
3.3V →	Pin 17 (3.3V)
GND →	Pin 25 (Ground)
SLCK →	Pin 23 (SPI clock)
MISO →	Pin 21 (SPI MISO)
MOSI →	Pin 19 (SPI MOSI)
NSS →	Pin 24 (SPI CE0)
ANA →	σύρμα 17.3cm στον αέρα (ως κεραία λ/2)

nRF24L01+

nRF24L01+ →	*Raspberry Pi2*
Pin 1 (GND) →	Pin 25 (Ground)
Pin 2 (VCC) →	Pin 17 (3.3V)
Pin 3 (CE) →	Pin 26 (SPI CE1)
Pin 4 (CSN) →	Pin 22 (GPIO 25)
Pin 5 (SCK) →	Pin 23 (SPI clock)
Pin 6 (MOSI) →	Pin 19 (SPI MOSI)
Pin 7 (MISO) →	Pin 21 (SPI MISO)



Σχήμα 5 Σχηματικό με τις περιφερειακές συνδέσεις του Raspberry Pi 2

Εγκατάσταση βιβλιοθηκών

Η εγκατάσταση των επιμέρους βιβλιοθηκών γίνεται εκτελώντας τις παρακάτω εντολές.

Εγκατάσταση βιβλιοθήκης RFM69:

```
sudo apt-get install git-core
git clone git://git.drogon.net/wiringPi
cd wiringPi
./build
gpio load spi
wget
http://rdepablos.merlitec.com/sites/default/files/rfm69lib.tar.gz
tar -zxvf rfm69lib.tar.gz
```

Εγκατάσταση βιβλιοθήκης MQTT.c

```
git clone
http://git.eclipse.org/gitroot/paho/org.eclipse.paho.mqtt.c.git
cd org.eclipse.paho.mqtt.c.git
make
sudo make install
```

Εγκατάσταση βιβλιοθήκης NRF24

```
wget http://tmrh20.github.io/RF24Installer/RPi/install.sh
chmod +x install.sh
./install.sh
```

Ανάπτυξη προγραμμάτων

Θα τοποθετήσουμε το πρόγραμμα που θα χειρίζεται την ασύρματη επικοινωνία μέσω του RFM69 σε ένα directory με όνομα `rfm69_gateway`, το μεταγλωττίζουμε και το τρέχουμε. Τα αρχεία της βιβλιοθήκης είναι απαραίτητο να βρίσκονται και αυτά στο ίδιο directory. Ο πηγαίος κώδικας βρίσκεται στη σελίδα 93.

```
cp –a rfm69lib rfm69_gateway
cd rfm69_gateway
rm test_rfm69 testrfm69.c
```

αντιγράφουμε το rfm69_gateway.c και το Makefile στο directory

```
make
sudo ./rfm69_gateway
```

Κατά την εκτέλεση του παραπάνω προγράμματος περιμένουμε στην οθόνη του terminal μηνύματα με την παρακάτω μορφή

```
rfm69_gateway.c

New packet received!
Length: 20
RSSI: -7
node = 2
deviceID = 0
cons_x100 = 5202
lumi = 818
Waiting for up to 10 sec to publish 52.02 on topic
diploma/Ilioup/GW_1/node_2/dev_0/cons/ | client: RFM69w1
Message with delivery token 1 delivered
Waiting for up to 10 sec to publish 818 on topic
diploma/Ilioup/GW_1/node_2/dev_0/lumi/ | client: RFM69w1
Message with delivery token 2 delivered
Database connection successful.
Disconnected from database.
```

Το αντίστοιχο πρόγραμμα που χειρίζεται την επικοινωνία μέσω του nRF25L01+ και το Makefile για τη μεταγλώττισή του τα αποθηκεύουμε σε directory με όνομα nrf24_gateway. Ο πηγαίος κώδικας βρίσκεται στη σελίδα 103.

mkdir nrf24_gateway
αντιγράφουμε το nrf24_gateway.c και το Makefile στο directory
make
sudo ./nrf24_gateway.c

Κατά την εκτέλεση του παραπάνω προγράμματος περιμένουμε στην οθόνη του terminal μηνύματα με την παρακάτω μορφή

```
I received 20 bytes!
nodeID:1
deviceID:1
lumi: 428
temp:26.00
humi:37.00
Waiting for up to 10 sec to publish 428 on topic
diploma/Ilioupoli/GW_1/node_1/dev_1/lumi/ | client: GWID1-nRF24
Message with delivery token 1 delivered
Waiting for up to 10 sec to publish 26.00 on topic
diploma/Ilioupoli/GW_1/node_1/dev_1/temp/ | client: GWID1-nRF24
Message with delivery token 2 delivered
Waiting for up to 10 sec to publish 37.00 on topic
diploma/Ilioupoli/GW_1/node_1/dev_1/humi/ | client: GWID1-nRF24
Message with delivery token 3 delivered
Database connection successful.
Disconnected from database.
```

Web Server

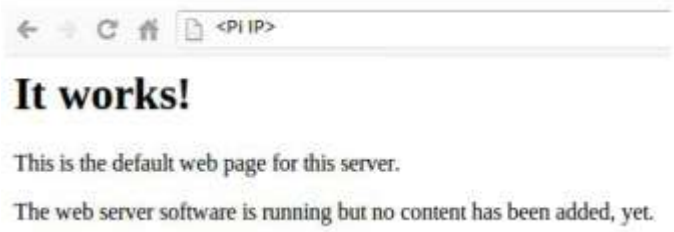
Δημιουργία Virtual Machine

Το Virtual Machine θα χρησιμοποιηθεί για την αποθήκευση των δεδομένων σε βάση δεδομένων και για την παρουσίαση τους σε μια ιστοσελίδα με τη μορφή γραφημάτων. Η δημιουργία του VM γίνεται με τη βοήθεια της πλατφόρμας ~okeanos.

1. Κάνουμε κλικ στο κουμπί “New Machine +”
2. Επιλέγουμε το επιθυμητό λειτουργικό σύστημα. Για της ανάγκες της εργασίας επιλέχθηκε το Ubuntu Server (LTS)
3. Στη συνέχεια επιλέγουμε τα χαρακτηριστικά του εικονικού υπολογιστή όπως μέγεθος RAM και δίσκου.
4. Στη συνέχεια επιλέγουμε IP διεύθυνση.
5. Δίνουμε όνομα στο VM
6. Επιβεβαιώνουμε τις επιλογές μας
7. Συνδεόμαστε στο VM μεσω ssh

Εγκατάσταση Apache2

Για την εγκατάσταση του Apache2 εκτελούμε την εντολή **`sudo apt-get install apache 2 –y`**. Για να ελέγξουμε αν η εγκατάσταση έγινε σωστά πληκτρολογούμε σε ένα browser την ip διεύθυνση του server και θα πρέπει να εμφανιστεί η παρακάτω σελίδα.



Εγκατάσταση php5

Για την εγκατάσταση της php5 εκτελούμε την εντολή **`sudo apt-get install php5 libapache2-mod-php5 –y`**

Εγκατάσταση mySQL

Για την εγκατάσταση της mySQL εκτελούμε την εντολή **mysql-server libapache2-mod-auth-mysql php5-mysql** . Για να έχουμε πρόσβαση στη βάση δεδομένων θέτουμε password πληκτρολογώντας **mysql -u root** και στη συνέχεια **SET PASSWORD FOR 'root'@'localhost' = PASSWORD('yourpassword');**

Εγκατάσταση phpMyAdmin

Για την εγκατάσταση του phpmyadmin εκτελούμε την εντολή **sudo apt-get install phpmyadmin**

Εγκατάσταση MQTT Broker

Για την εγκατάσταση του Mosquitto MQTT Broker εκτελούμε την εντολή **sudo apt-get install mosquitto mosquitto-clients** . Για να επιτρέψουμε τα tcp πακέτα να περνάνε από το port 1883, προκαθορισμένο για το MQTT εκτελούμε την εντολή **sudo iptables -A INPUT -p tcp -m tcp --dport 1883 -j ACCEPT** . Το Mosquitto ήδη τρέχει στο background ως deamon.

Για να ενεργοποιήσουμε τα web sockets έτσι ώστε να «ακούμε» τα δεδομένα του MQTT από τη θύρα 8000 θα χρειαστούμε το αρχείο mosquitto.conf.diploma (σελίδα το οποίο μεταφέρουμε στο home directory. Στη συνέχεια σταματάμε το mosquitto **sudo service mosquitto stop** , και το επανεκκινούμε με τις ρυθμίσεις που έχει το παραπάνω αρχείο **sudo mosquitto -c mosquitto.conf.evgenia –d**.

Στήσιμο Βάσης Δεδομένων

Έχοντας εγκαταστήσει την PHP και το phpMyAdmin κάνουμε login στο <serverip>/ phpmyadmin με username root και password αυτό που ορίσαμε κατά την εγκατάσταση.

Για τη δημιουργία της βάσης δεδομένων της εφαρμογής μας πατάμε “Databases” και στο πεδίο **Create database** συμπληρώνουμε το όνομα που θέλουμε να της δώσουμε πχ diplomadb και κάνουμε κλικ στο ”Create”. Η νεα βάση δεδομένων εμφανίζεται στη λίστα παρακάτω και κάνουμε κλικ στο όνομά της για να προσθέσουμε πίνακες.Στο πεδίο “Name” συμπληρώνουμε το όνομα του πίνακα πχ measurements, και στο πεδίο “Number of columns” τον αριθμό των στηλών, στη δική μας περίπτωση συμπληρώνουμε 19, και κάνουμε κλικ στο “Go”.

Στη συνέχεια συμπληρώνουμε τα στοιχεία των στηλών ένα ένα δίνοντάς τους ονόματα, τύπο δεδομένων κλπ. Τελικώς φτάνουμε στην παρακάτω διάταξη:

#	Name	Type	Collation	Attributes	Null	Default	Extra
1	id	int(11)			No	None	AUTO_INCREMENT
2	datetime	timestamp		on update CURRENT_TIMESTAMP	No	CURRENT_TIMESTAMP	ON UPDATE CURRENT_TIMESTAMP
3	country	tinytext	utf8_general_ci		No	None	
4	region	tinytext	utf8_general_ci		No	None	
5	municipality	tinytext	utf8_general_ci		No	None	
6	gwid	int(11)			No	None	
7	nodeid	int(11)			No	None	
8	deviceid	int(11)			No	None	
9	lat	decimal(13,10)			No	None	
10	lng	decimal(13,10)			No	None	
11	batt	int(11)			Yes	NULL	
12	prod	decimal(6,2)			Yes	NULL	
13	cons	decimal(6,2)			Yes	NULL	
14	lumi	int(11)			Yes	NULL	
15	temp	decimal(5,2)			Yes	NULL	
16	humi	decimal(5,2)			Yes	NULL	
17	rain	int(11)			Yes	NULL	
18	uv	int(11)			Yes	NULL	
19	ir	int(11)			Yes	NULL	

Εικόνα 25 Δομη βάσης δεδομένων

Στη στήλη id κάναμε tick στο A_I ή Auto increment έτσι ώστε κάθε νέα καταχώρηση να έχει μια μοναδική ταυτότητα. Στη στήλη datetime δώσαμε τύπο δεδομένων timestamp και default τιμή κάθε καταχώρησης τη χρονική στιγμή που καταχωρείται. Στις στήλες που θα αποθηκεύσουμε τα επιμέρους δεδομένα δώσαμε default τιμή NULL.

Στη συνέχεια θα δημιουργήσουμε έναν χρήστη αυτής της βάσης δεδομένων, τα αναγνωριστικά του οποίου θα χρησιμοποιούν τα προγράμματα των gateways για να συνδέονται σε αυτή. Αυτός ο χρήστης θα έχει περιορισμένα δικαιώματα για λόγους ασφαλείας. Από την αρχική σελίδα κάνουμε κλικ στο “Users” και μετά στο “Add Users”. Συμπληρώνουμε τα πεδία: στο πεδίο “User name” δίνουμε το όνομα του χρήστη πχ diploma_user, password 1111, και στο πεδίο “host” το αφήνουμε “Any host” ως έχει, και πατάμε ”Go”.Θα εμφανιστεί η λίστα με όλους τους χρήστες. Πατάμε το link “Edit Privileges” δίπλα από το όνομα χρήστη που δημιουργήσαμε. Στο πεδίο **Database-specific privileges** επιλέγουμε τη diplomadb. Θα μεταφερθούμε σε μια σελίδα όπου θα επιλέξουμε privileges (προνόμια) για τον diploma_user. Θα επιλέξουμε μόνο τα SELECT και INSERT και πατάμε Go.

WebGUI

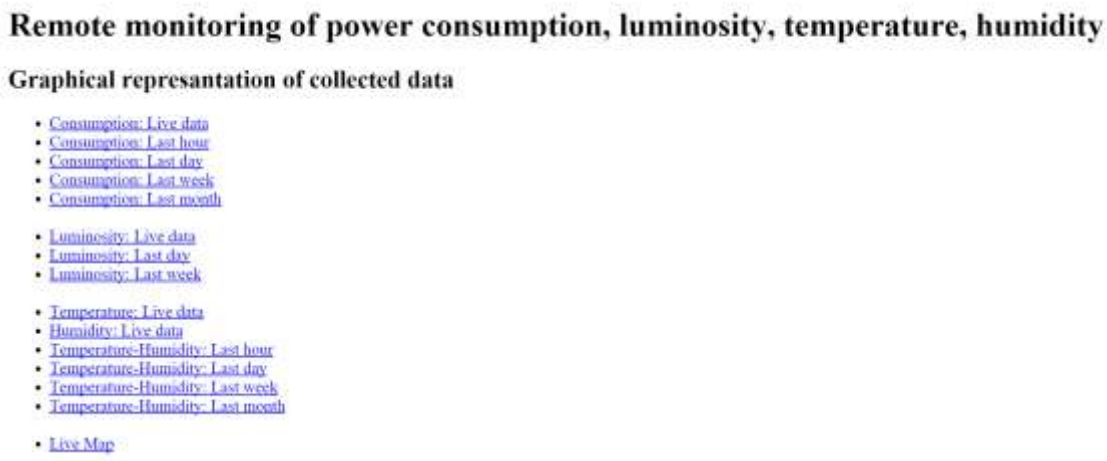
Στο home directory δημιουργούμε ένα φάκελο diploma. Εκει αντιγράφουμε τα αρχεία που αποτελούν το web GUI. Για να μεταφέρουμε όλα αυτά τα αρχεία από το home directory στο /var/www/html/ εκτελούμε την εντολή **`sudo cp -r ~/diploma/* /var/www/html/diploma/`**

Ο φάκελος diploma περιέχει τα παρακάτω αρχεία και directories:

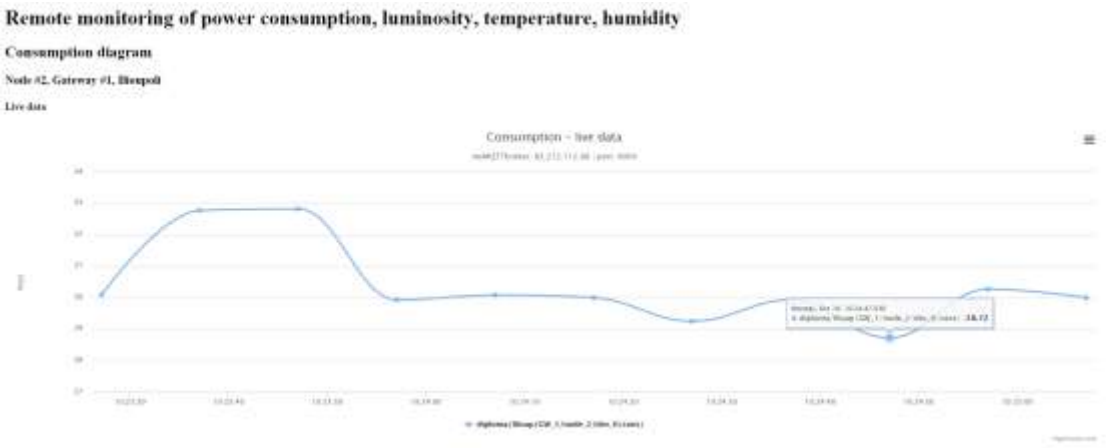
```
cons-lastday.php
cons-lasthour.php
cons-lastmonth.php
cons-lastweek.php
cons-live.html
humi-live.html
index.html
lumi-lastday.php
lumi-lastweek.php
lumi-live.html
temp.html
temp-humi-lastday.php
temp-humi-lasthour.php
temp-humi-lastmonth.php
temp-humi-lastweek.php
```

Οι παραπάνω κώδικες βρίσκονται στο παράρτημα Β, σελίδα 114.

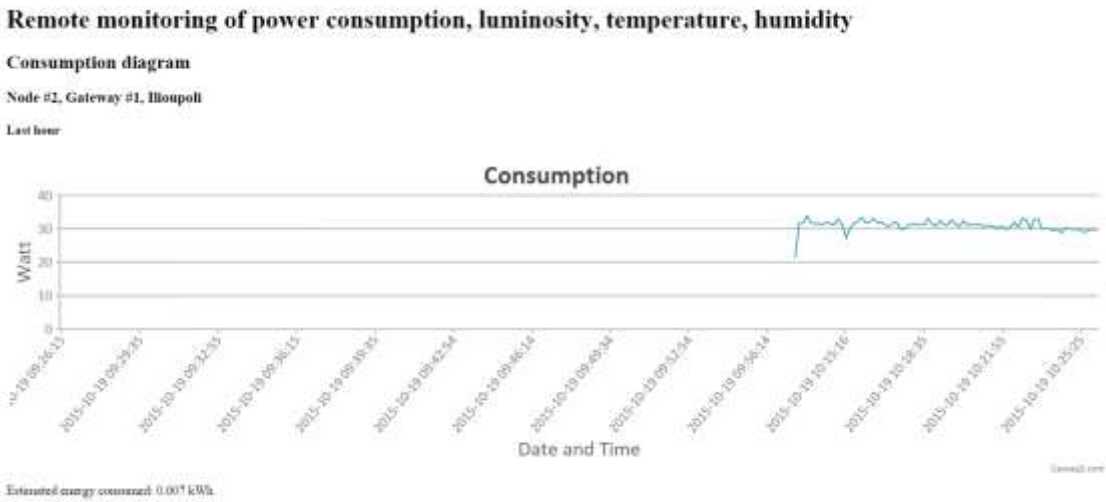
Screenshots



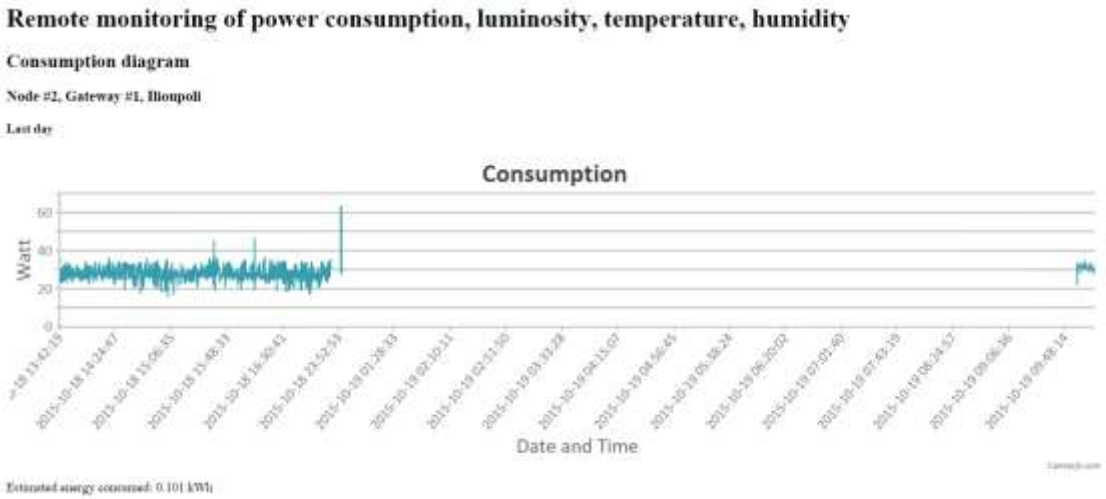
Εικόνα 26 index.html



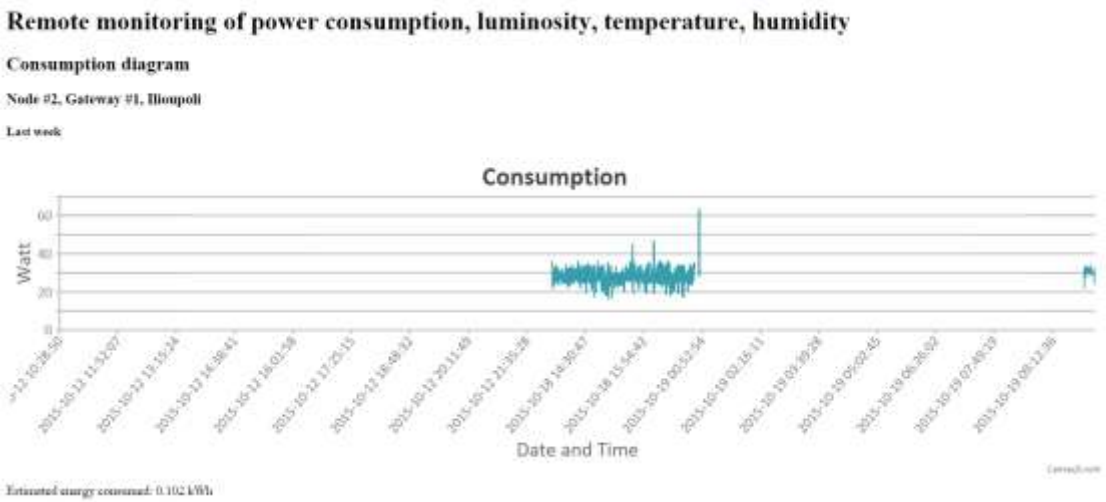
Εικόνα 27 cons-live.html



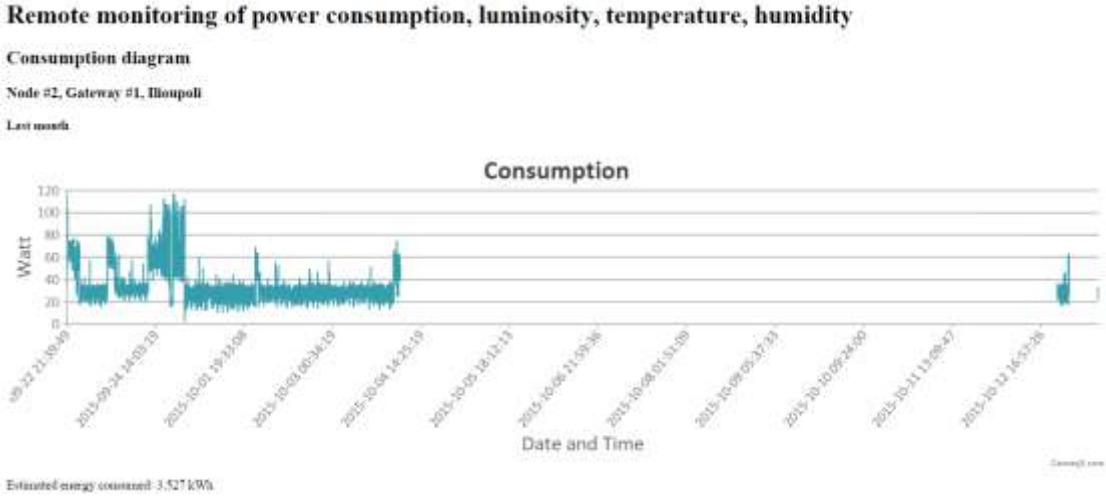
Εικόνα 28 cons-lasthour.php



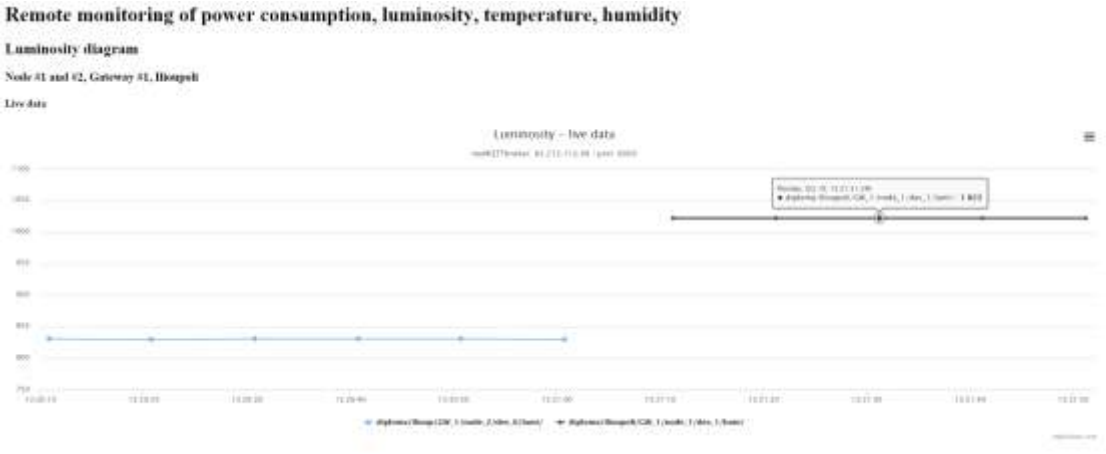
Εικόνα 29 cons-lastday.php



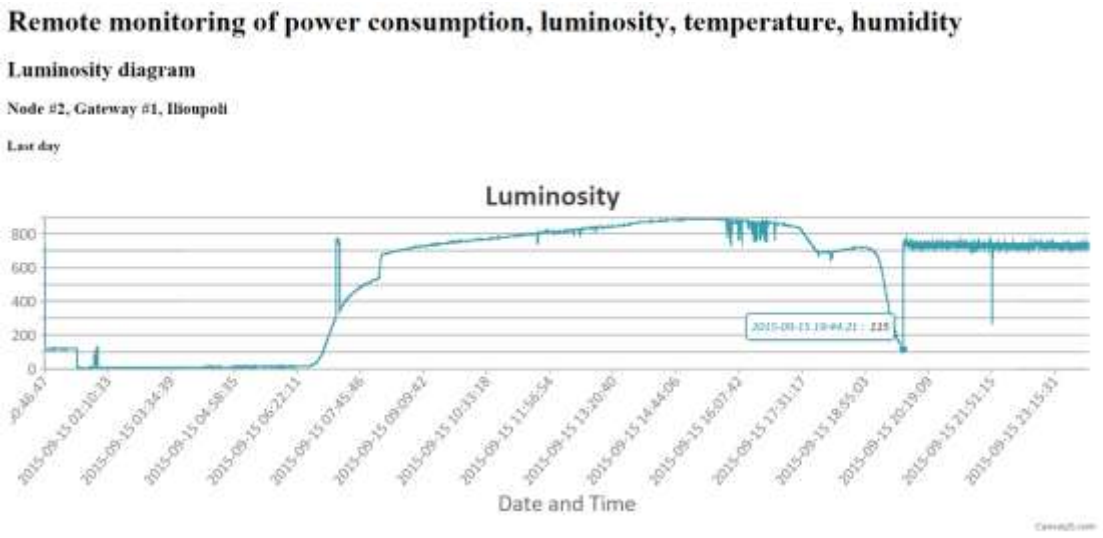
Εικόνα 30 cons-lastweek.php



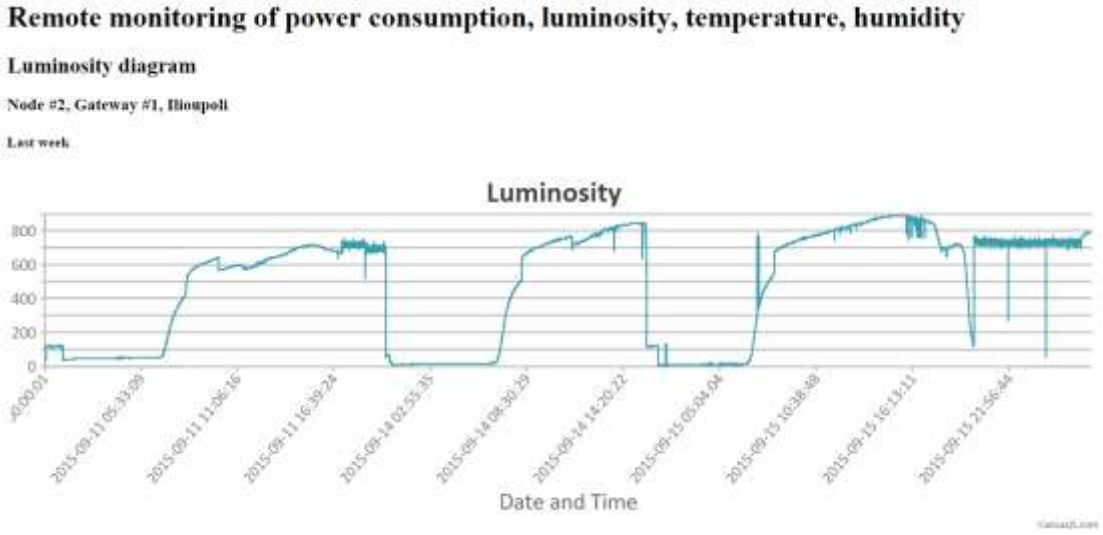
Εικόνα 31 cons-lastmonth.php



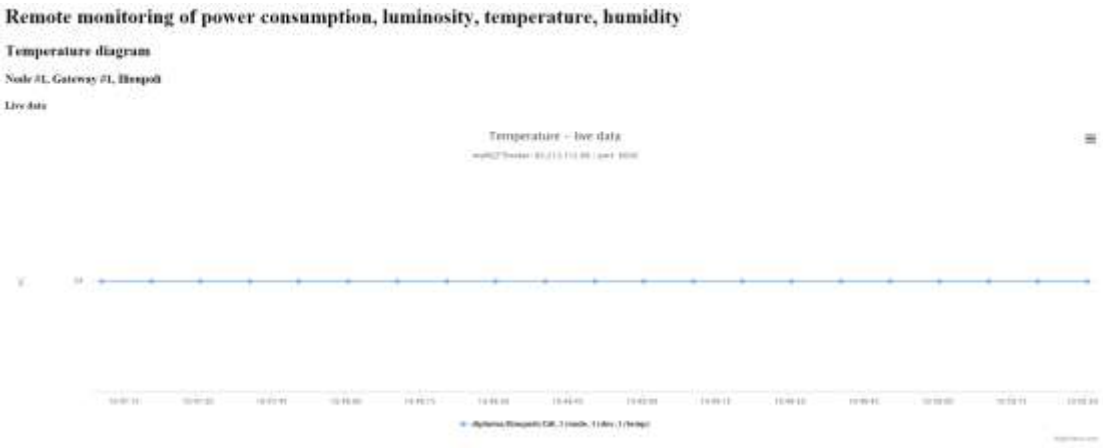
Εικόνα 32 lumi.-live.html



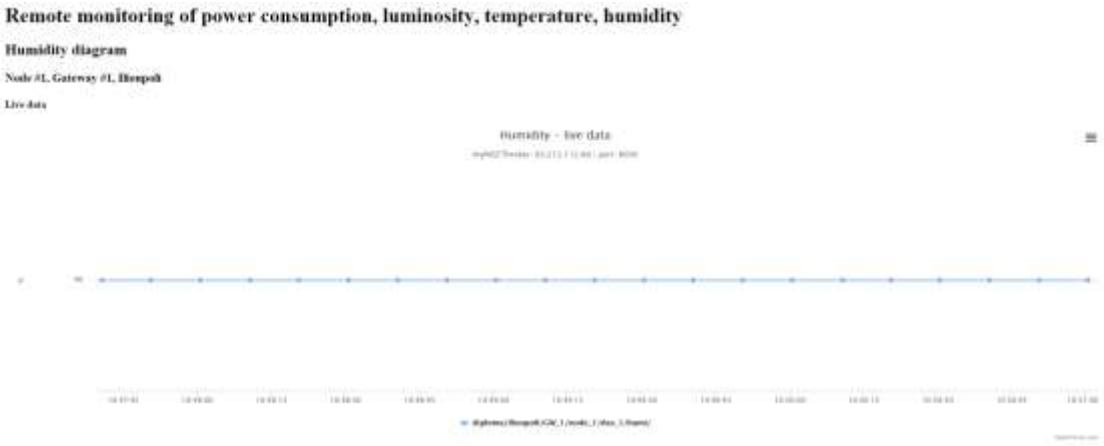
Εικόνα 33 lumi-lastday.php



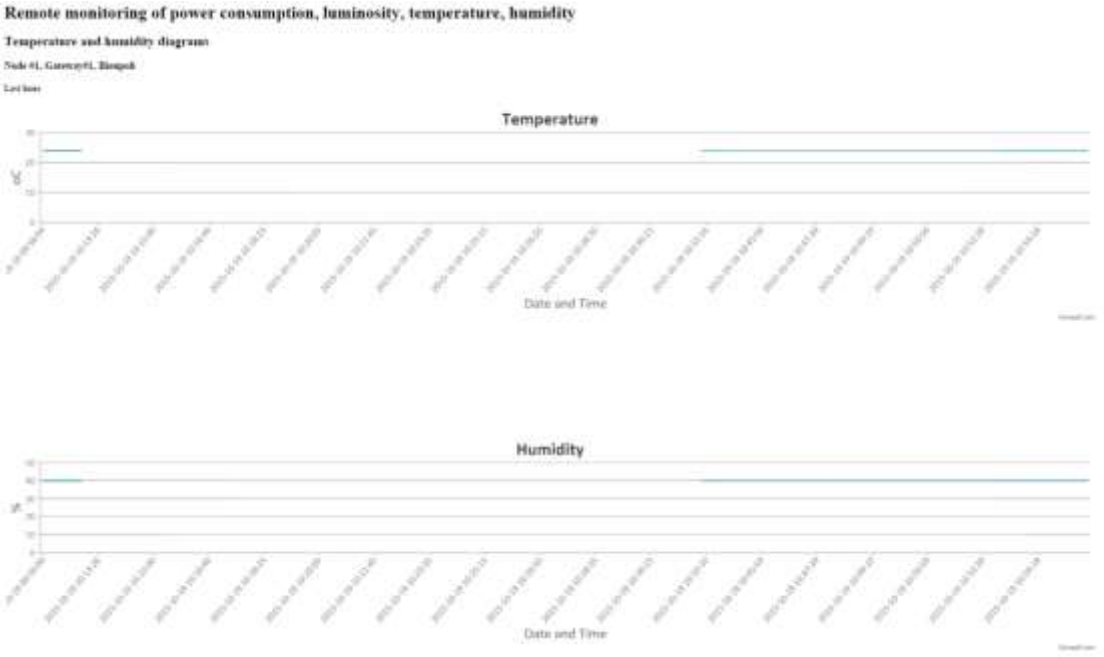
Εικόνα 34 lumi-lastweek.php



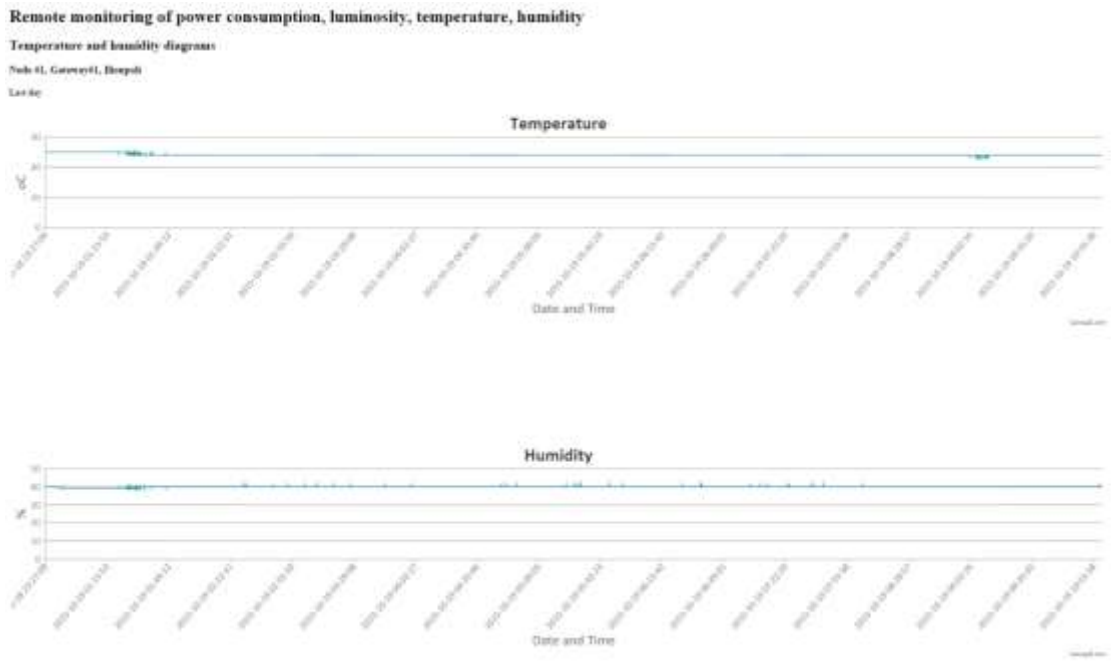
Εικόνα 35 temp-live.html



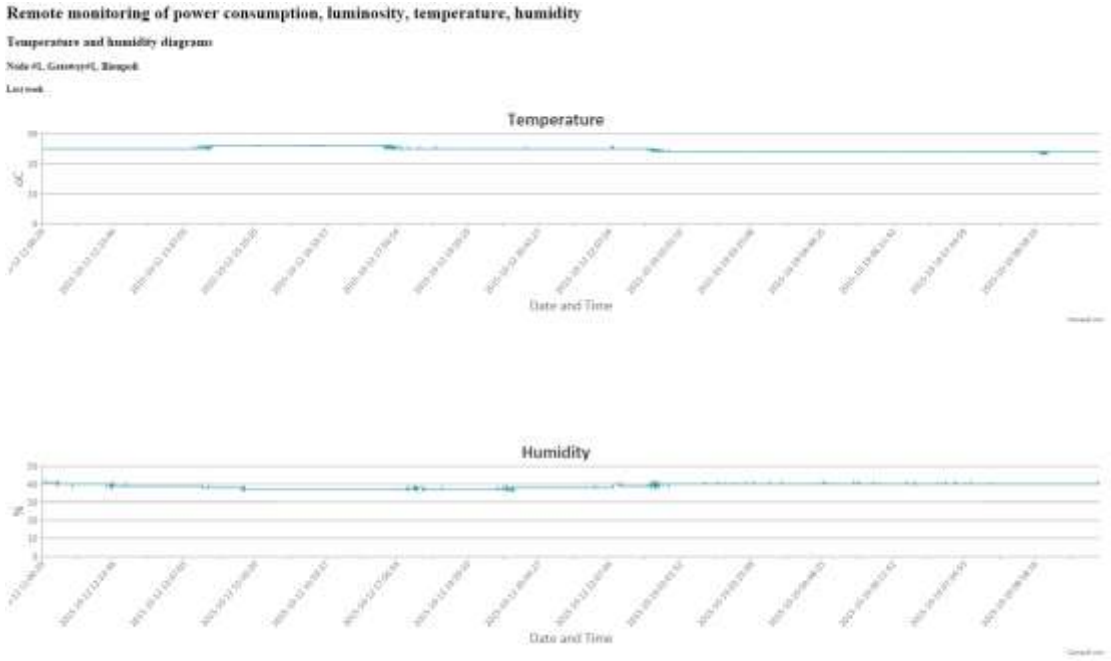
Εικόνα 36 humi-live.html



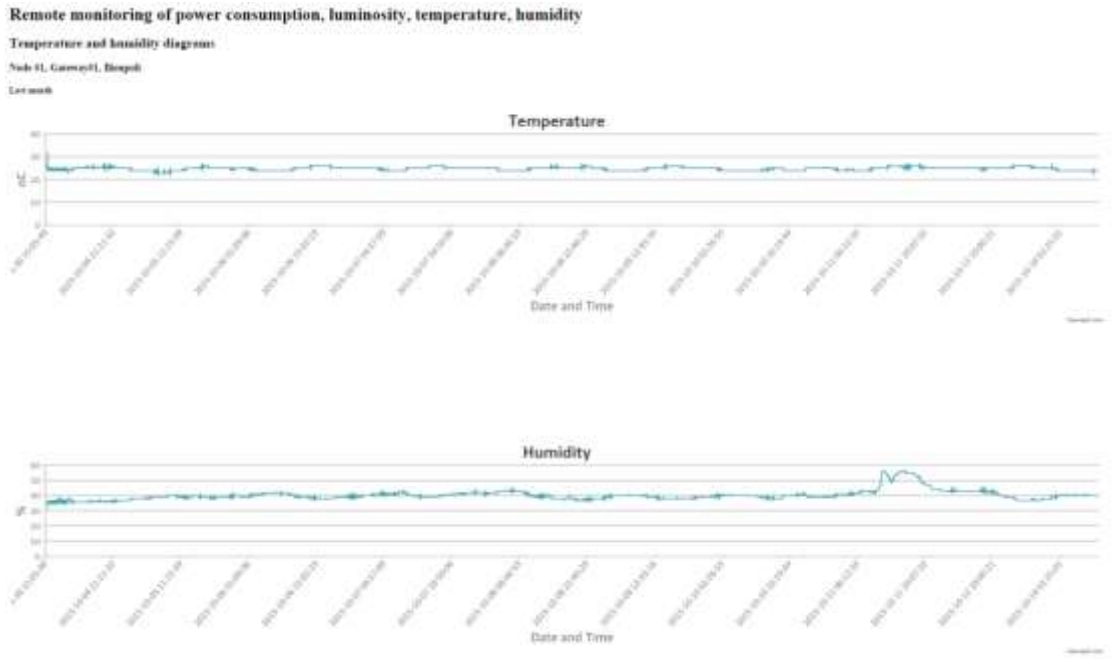
Εικόνα 37 temp-humi-lasthour.php



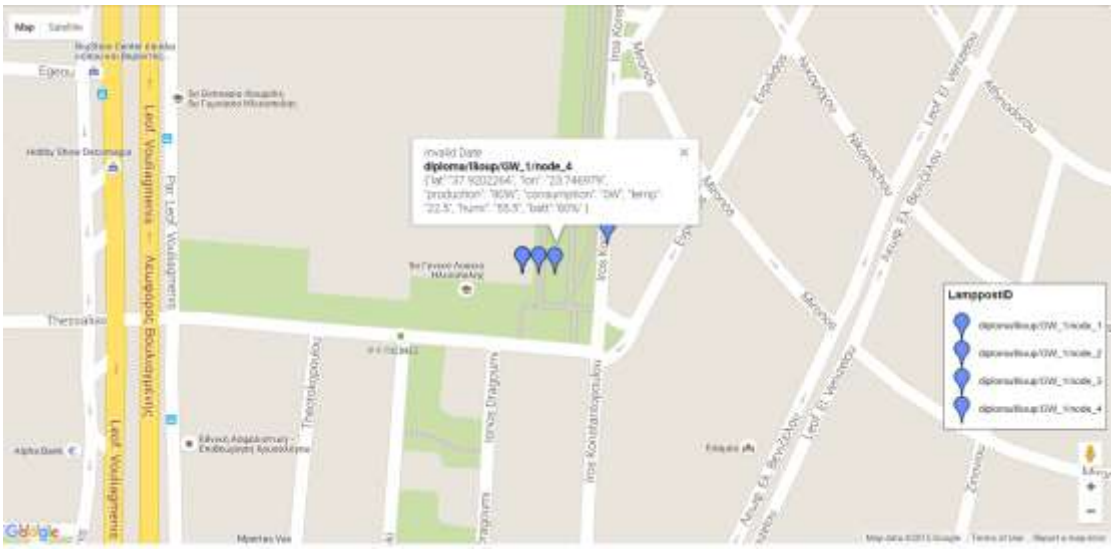
Εικόνα 38 temp-humi-lastday.php



Εικόνα 39 temp-humi-lastweek.php



Εικόνα 40 temp-humi-lastmonth.p



Εικόνα 41 map.html

Κεφάλαιο 6. Δοκιμές / Αποτελέσματα

Κατά την εκπόνηση αυτής της εργασίας ελέγξαμε την εμβέλεια των 2 RF modules που χρησιμοποιήσαμε.

Εμβέλεια RFM69w

Στο συγκεκριμένο πομποδέκτη έχουμε τη δυνατότητα να διαβάσουμε ψηφιακά το RSSI, δηλαδή την ισχύ του εισερχόμενου σήματος. Η μία συσκευή-πομπός (Arduino) τρέχει το sketch rfm69_rangetest.ino (σελ. 139) στέλνει ακέραιους αριθμούς από το 1-100, ενώ η άλλη συσκευή-δέκτης (Raspberry Pi), που τρέχει το rfm69_rangetest.c (σελ. 142), λαμβάνει το μέσο όρο των RSSI των 10 πρώτων πακέτων που θα λάβει. Η διαδικασία αυτή επαναλαμβάνεται μετακινώντας το δέκτη σε διαδοχικές αποστάσεις από τον πομπό για διαφορετικά bitrates.

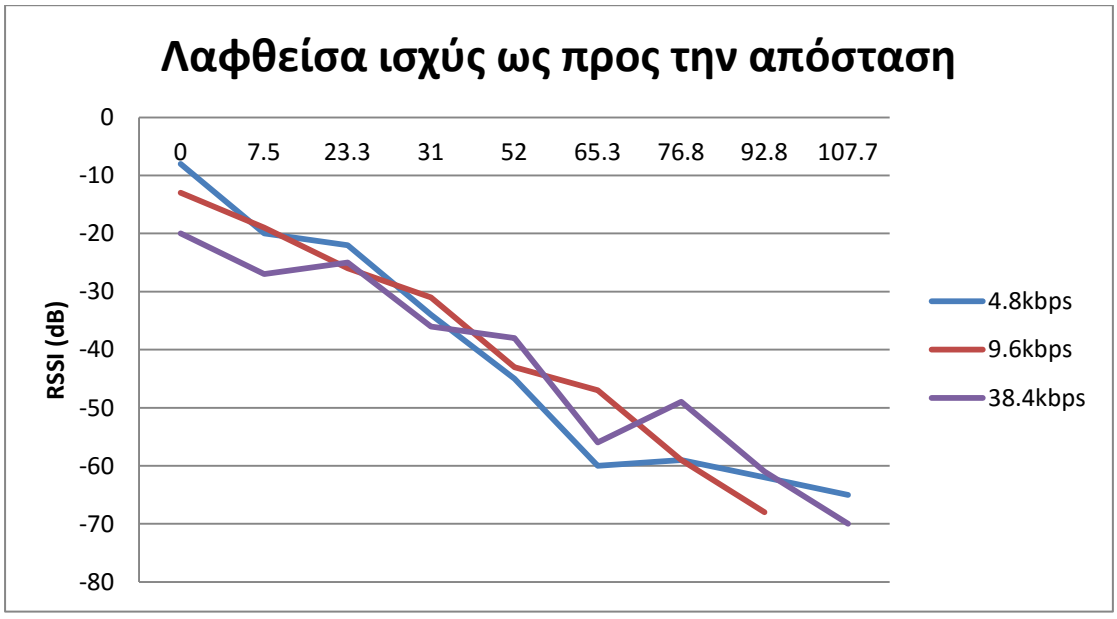
Στο serial monitor του πομπού βλέπουμε:

```
Range testing RFM69W
Initializing Radio...OK!!

Transmitting 0-100 at 868 Mhz...
Message (1 bytes) | number: 0
Message (1 bytes) | number: 1
Message (1 bytes) | number: 2
Message (1 bytes) | number: 3
Message (1 bytes) | number: 4
Message (1 bytes) | number: 5
```

Ενώ στη μεριά του δέκτη βλέπουμε:

```
Strarting rangetest.c for RFM69W ...

RSSI: -8    Received: = 39
RSSI: -8    Received: = 40
RSSI: -8    Received: = 41
RSSI: -8    Received: = 42
RSSI: -8    Received: = 43
RSSI: -8    Received: = 44
RSSI: -8    Received: = 45
RSSI: -8    Received: = 46
RSSI: -8    Received: = 47
RSSI: -8    Received: = 48

============
RSSI average of 10 packets: -8.0
============
```

Ακολουθώντας την παραπάνω διαδικασία προέκυψαν τα παρακάτω αποτελέσματα. Παρατηρούμε ότι η εμβέλεια είναι περίπου 100m αν και θεωρητικά θα μπορούσε να είναι 500m, κάτι που εξηγείται όμως από το γεγονός ότι η κεραία των πομποδεκτών ήταν ένα απλό σύρμα μήκους 17.3cm (λ/2)



Εμβέλεια nRF24L01+

Το nRF24L01+ δεν υποστηρίζει την ανάγνωση του RSSI, οπότε ο πομπός στέλνει ακέραιους αριθμούς από το 1-100 και μετράει σε τι ποσοστό λαμβάνει ACKs από το δέκτη. Αυτές οι μετρήσεις έγιναν σε διαδοχικές αποστάσεις από τον πομπό και για διαφορετικά bitrates. Ο πομπός βρίσκεται πάνω σε ένα Arduino Nano, που τρέχει το sketch nrf24_send.ino (σελ. 143), ενώ ο δέκτης σε ένα Arduino Uno, που τρέχει το sketch nrf24_receive.ino (σελ. 145).

Στο serial monitor του πομπού βλέπουμε:

```
RF24 rangetest: rfm24_send
DataRate set ok!
Now sending 0
Got blank response.  round-trip delay: 1116 microseconds
Now sending 1
Got blank response.  round-trip delay: 1120 microseconds
Now sending 2
Got blank response.  round-trip delay: 1084 microseconds
Now sending 3
Got blank response.  round-trip delay: 1084 microseconds
Now sending 4
Got blank response.  round-trip delay: 1120 microseconds
Now sending 5
Got blank response.  round-trip delay: 1080 microseconds
[…]
Now sending 98
Got blank response.  round-trip delay: 1088 microseconds
Now sending 99
Got blank response.  round-trip delay: 1088 microseconds

success: 100
fails: 0
```

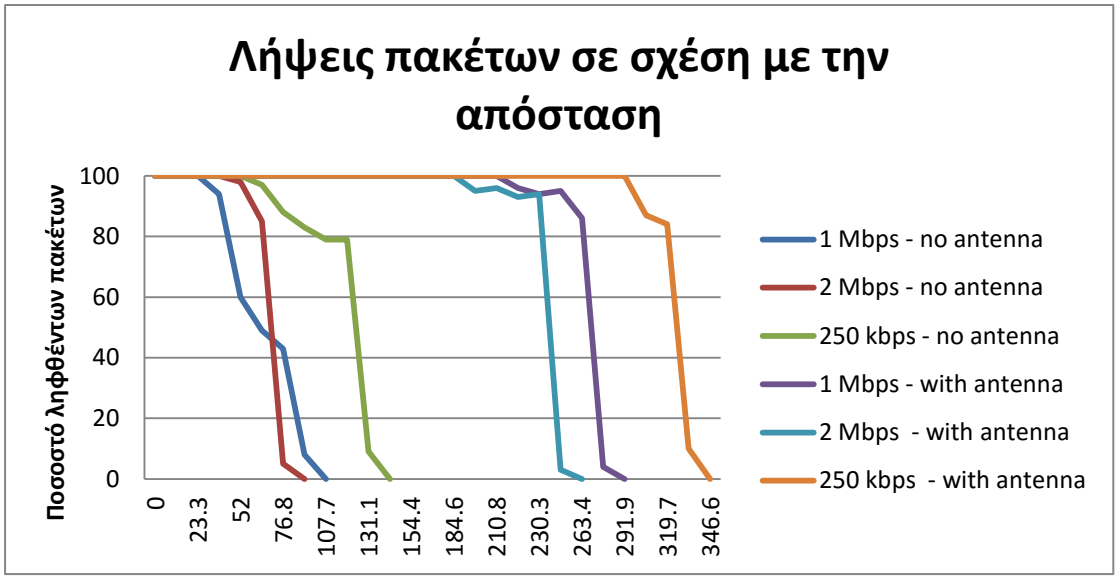
Ενώ στη μεριά του δέκτη βλέπουμε:

```
RF24 rangetest: rfm24_receive
DataRate set ok!
Received 0
Received 1
Received 2
Received 3
Received 4
Received 5
[…]
Received 98
Received 99
```

Για αυτό τον πομποδέκτη πραγματοποιήθηκαν μετρήσεις με και χωρίς κεραία. Όπως ήταν αναμενόμενο, τα modules με κεραία έχουν μεγαλύτερο κέρδος και άρα επιτυγχάνουν

μεγαλύτερη εμβέλεια σε σχέση με τα modules με την τυπωμένη κεραία. Επίσης όσο μικρότερο το bitrate τόσο μεγαλύτερη και η εμβέλεια που επιτυγχάνεται.

Συγκεκριμένα, χωρίς κεραία η επικοινωνία διακόπτεται στα 75m με ρυθμό μετάδοσης 1Mbps, στα 65m με ρυθμό μετάδοσης 2Mbps και στα 120m με ρυθμό μετάδοσης 250kbps. Με κεραία η εμβέλεια σημειώνει αύξηση ως και 200m δηλαδή για αποστολή δεδομένων στο 1Mbps λάβαμε ικανοποιητικό ποσοστό πακέτων μέχρι τα 230m, στα 2Mbps η επικοινωνία ήταν ικανοποιητική μέχρι τα 230m ενώ με ρυθμό μετάδοσης 250kbps η εμβέλεια άγγιξε τα 320m.



Κεφάλαιο 7. Σύνοψη

Συμπεράσματα

Στο πλαίσιο της διπλωματικής εργασίας, στόχος της οποίας ήταν ο σχεδιασμός και υλοποίηση ενός ασύρματου μετρητικού συστήματος ενέργειας για εφαρμογή σε κατανεμημένα συστήματα ανενεώσιμων πηγών ενέργειας, αναπτύχθηκε ένα σύστημα που αποτελείται από δύο μετρητικές συσκευές, μια πύλη και ένα βασικό γραφικό περιβάλλον εποπτείας των δεδομένων.

Η μια μετρητική συσκευή αποτελείται από ένα Arduino Uno με συνδεδεμένες μια διάταξη μέτρησης στιγμιαίας καταναλούμενης ισχύος και μια διάταξη μέτρησης φωτεινότητας με τη βοήθεια μιας αμπεροτσιμπίδας και ενός φωτοκυττάρου αντίστοιχα, καθώς επίσης και ένα module ασύρματης επικοινωνίας RFM69W που εκπέμπει στα 868MHz.

Η δεύτερη μετρητική συσκευή αποτελείται από ένα Arduino Nano και μια διάταξη μέτρησης της φωτεινότητας μαζί με ένα αισθητήρα θερμοκρασίας και σχετικής υγρασίας και ένα module ασύρματης επικοινωνίας nRF24L01+ που λειτουργεί στη συχνότητα των 2.4GHz.

Οι μετρητικές αυτές συσκευές επεξεργάζονται και αποστέλουν τις μετρήσεις που λαμβάνουν από τους αισθητήρες ασύρματα σε μία «κεντρική» πύλη (gateway), η οποία υποστηρίζει ασύρματη σύνδεση με τις μετρητικές συσκευές και στα 2.4 GHz και στα 868 MHz.

Η πύλη αποστέλει με τη σειρά της αυτές τις μετρήσεις σε έναν MQTT broker και σε μια βάση δεδομένων που φιλοξενείται σε ένα server. Στον ίδιο server φιλοξενείται και ένα γραφικό περιβάλλον το οποίο μπορεί να απεικονίσει αυτά τα δεδομένα σε πραγματικό χρόνο, σε επίπεδο ώρας, μέρας, εβδομάδα και μήνα.

Προτεινόμενες Επεκτάσεις

Κατά την τριβή με το αντικείμενο στα πλαίσια της εργασίας, βρεθήκαν πολλά σημεία τα οποία θα μπορούσαν να βελτιώσουν σημαντικά την εφαρμογή, και αρκετές ιδέες και προτάσεις, οι οποίες μπορούν να υλοποιηθούν για προέκταση και πειραματισμό.

Ένα από τα μειονεκτήματα του συστήματος που περιγράφηκε είναι ότι είναι στατικό όσον αφορά την προσθήκη νέων μετρητικών συσκευών στο δίκτυο. Κάθε φορά που επιθυμεί ο χρήστης να τοποθετήσει μία νέα πρέπει να επεμβαίνει στον κώδικα της πύλης και να τον

προσθέτει. Έτσι θα ήταν προτιμότερο η πύλη να την αναγνώριζε μόνη της και να ενημέρωνε τον χρήστη για τη λειτουργία της.

Θα είχε ενδιαφέρον να γίνει η υλοποίηση του παραπάνω δικτύου χρησιμοποιώντας τοπολογία mesh.

Σημαντική προσθήκη θα ήταν η ανάπτυξη και υλοποίηση ενός έξυπνου συστήματος φωτισμού, ενός συστήματος ενημέρωσης των πολιτών για τις καιρικές συνθήκες και ενός συστήματος πυρασφάλειας και η ενσωμάτωσή τους στο υπάρχον σύστημα, με τη δυνατότητα να σταλούν εντολές προς το σύστημα απομακρυσμένα.

Επιπλέον, προτείνεται η ανάπτυξη ενός εργαλείου σχεδίασης στο οποίο να εισάγονται δεδομένα σχετικά με τις θέσεις εγκατεστημένων Φ/Β, τις θέσεις που υπάρχει υφιστάμενη σύνδεση xDSL, κτλ. και αυτό να επιλέγει τις κατάλληλες τεχνολογίες (ανά περίπτωση), τις θέσεις που θα πρέπει να τοποθετηθούν οι πύλες (gateways) ώστε να ελαχιστοποιείται το συνολικό κόστος εγκατάστασης και λειτουργίας της λύσης.

Επίσης καλό θα ήταν να διερευνηθούν ζητήματα ασφαλείας του δικτύου ως προς θέματα δομής, δυνατοτήτων συσκευών-κόμβων, μεθόδων δρομολόγησης κλπ., και να εξεταστούν οι απειλές οι οποίες είναι πιθανό να θέσουν υπό αμφισβήτηση την ασφάλειά του και μηχανισμοί αντιμετώπισης τους.

Τέλος, για την παρουσίαση των μετρήσεων χρησιμοποιήθηκαν κυρίως html, javascript και απλουστευμένα γραφικά στοιχεία. Προτείνεται, λοιπόν, η το γραφικό περιβάλλον να γίνει πιο «φιλικό» στο χρήστη και να είναι προσβάσιμο και μέσω smartphone/tablet.

Βιβλιογραφία

[1] REN 21 - Renewable Energy Policy Network for the 21st Century, "RENEWABLES 2014 GLOBAL STATUS REPORT," 2014. [Online]. http://www.ren21.net/Portals/0/documents/Resources/GSR/2014/GSR2014_full%20report_low%20res.pdf

[2] Renewable Energy World. Solar energy pros and cons: Photovoltaic PV systems. [Online]. http://www.renewablegreenenergypower.com/solar-energy-pros-and-cons-pv-systems/#.UFtfyVEfo_d

[3] econews.gr. (2015, Ιούνιος) Τα φωτοβολταϊκά καλύπτουν το 7% της ζήτησης στην Ελλάδα – Οι παγκόσμιες τάσεις. [Online]. http://www.econews.gr/2015/06/10/fotovoltaika-ellada-solarpowereurope-122866/

[4] Σύνδεσμος εταιριών φωτοβολταϊκών. (2013) [Online]. www.helapco.gr

[5] Στέλλα Ιωαννίδου, "Τεχνο-Οικονομική Προσέγγιση Ενεργειακής Αυτονομίας σε Εφαρμογές Δικτύων Αισθητήρων" Μεταπτυχιακή Εργασία, ΔΠΜΣ «Τεχνο-Οικονομικά Συστήματα», 2011.

[6] D. Minoli, T. Znati K. Sohraby, *Wireless Sensor Networks*.: Wiley, 2007. [Online]. http://image.sciencenet.cn/olddata/kexue.com.cn/bbs/upload/12615WSN-2007.pdf

[7] A. Willig H. Karl, *Protocols and Architectures for Wireless Sensor Networks*.: Wiley, 2005.

[8] Wikipedia. Single-board microcontroller. [Online]. https://en.wikipedia.org/wiki/Single-board_microcontroller

[9] Arduino. Compare board specs. [Online]. https://www.arduino.cc/en/Products.Compare

[10] Intel. Intel® Edison Development Platform. [Online]. https://communities.intel.com/servlet/JiveServlet/downloadBody/23139-102-3-27268/edison_PB_331179-001.pdf

[11] STMicroelectronics. STM32 Nucleo board. [Online]. http://www.st.com/web/catalog/tools/FM116/SC959/SS1532/LN1847/PF260001

[12] Texas Instruments. MSP Launchpads. [Online]. http://www.ti.com/ww/en/launchpad/launchpads-msp430-msp-exp432p401r.html#tabs

[13] Wikipedia. Arduino. [Online]. https://en.wikipedia.org/wiki/Arduino

[14] Arduino. What is Arduino? [Online]. https://www.arduino.cc/en/Guide/Introduction

[15] Wikipedia. Single-board computer. [Online]. https://en.wikipedia.org/wiki/Single-board_computer

[16] Wikipedia. Comparison of single-board computers. [Online]. https://en.wikipedia.org/wiki/Comparison_of_single-board_computers

[17] Wikipedia. Raspberry Pi. [Online]. https://en.wikipedia.org/wiki/Raspberry_Pi

[18] Raspberry Pi Foundation. What Is A Raspberry Pi? [Online]. https://www.raspberrypi.org/help/what-is-a-raspberry-pi/

[19] Allegro Microsystems LLC. ACS712: Fully Integrated, Hall-Effect-Based Linear Current Sensor IC with 2.1 kVRMS Voltage Isolation and a Low-Resistance Current Conductor. [Online]. http://www.allegromicro.com/en/Products/Current-Sensor-ICs/Zero-To-Fifty-Amp-Integrated-Conductor-Sensor-ICs/ACS712.aspx

[20] Split-Core Current Transformer SCT-013 Series. [Online]. https://nicegear.co.nz/obj/pdf/SCT-013-datasheet.pdf

[21] MAX471 datasheet. [Online]. http://pdfserv.maximintegrated.com/en/ds/MAX471-MAX472.pdf

[22] Itead Studio. Electronic Brick of Electricity Meter. [Online]. ftp://imall.iteadstudio.com/Electronic_Brick/IM120710018/DS_IM120710018.pdf

[23] D-Robotics. DHT11 Humidity & Temperature Sensor. [Online]. http://www.micropik.com/PDF/dht11.pdf

[24] Aosong Electronics Co.,Ltd. Digital-output relative humidity & temperature sensor/module DHT22. [Online]. https://www.sparkfun.com/datasheets/Sensors/Temperature/DHT22.pdf

[25] Sensirion. Datasheet SHT7x (SHT71, SHT75). [Online]. http://www.sensirion.com/fileadmin/user_upload/customers/sensirion/Dokumente/Humidity/Sensirion_Humidity_SHT7x_Datasheet_V5.pdf

[26] Texas Instruments. HDC1008 Low Power, High Accuracy Digital Humidity Sensor with Temperature Sensor. [Online]. http://www.ti.com/general/docs/lit/getliterature.tsp?genericPartNumber=hdc1008&fileType=pdf

[27] Texas Instruments. (2014) Wireless connectivity for the Internet of Things. [Online]. http://www.ti.com/lit/wp/swry010/swry010.pdf?DCMP=ep-con-wcs-cmtech&HQS=ep-con-wcs-cmtech-hpb-whip-en

[28] Publitek European Editors. (2012) Evaluating the Different Protocols for Low Power Wireless Networks in Industrial Automation. [Online]. http://www.digikey.com/en/articles/techzone/2012/jan/evaluating-the-different-protocols-for-low-power-wireless-networks-in-industrial-automation

[29] Nordic Semiconductor. nRF24L01+ Single Chip 2.4GHz Tranceiver. [Online]. https://www.sparkfun.com/datasheets/Components/SMD/nRF24L01Pluss_Preliminary_Product_Specification_v1_0.pdf

[30] GitHub. Optimized High Speed Driver for nRF24L01(+) 2.4GHz Wireless Transceiver. [Online]. http://tmrh20.github.io/RF24/

[31] GitHub. MeshNet. [Online]. https://github.com/mattibal/meshnet

[32] GitHub. Mesh Networking Layer for RF24 Radios. [Online]. http://tmrh20.github.io/RF24Mesh/index.html

[33] HopeRF Electronic. RFM69W ISM TRANSCEIVER MODULE V 1. 3. [Online]. http://www.hoperf.com/upload/rf/RFM69W-V1.3.pdf

[34] GitHub. RFM69 Library. [Online]. https://github.com/LowPowerLab/RFM69

[35] Airspayce. RadioHead Packet Radio library for embedded microprocessors. [Online]. http://www.airspayce.com/mikem/arduino/RadioHead/

[36] Gurwinder Kaur and Rachit Mohan Garg, "Energy efficient topologies for wireless sensor networks," *International Journal of Distributed and Parallel Systems*, 2012. [Online]. http://www.airccse.org/journal/ijdps/papers/3512ijdps16.pdf

[37] Sparkfun. Data Types in Arduino. [Online]. https://learn.sparkfun.com/tutorials/data-types-in-arduino

[38] Wikipedia. MQTT. [Online]. https://en.wikipedia.org/wiki/MQTT

[39] IBM Internet of Things Foundation. MQTT. [Online]. https://docs.internetofthings.ibmcloud.com/messaging/mqtt.html

[40] Arduino. Arduino UNO & Genuino UNO. [Online]. https://www.arduino.cc/en/Main/ArduinoBoardUno

[41] Arduino. Arduino Nano. [Online]. https://www.arduino.cc/en/Main/ArduinoBoardNano

[42] Adafruit Industries. Using a Photocell. [Online]. https://learn.adafruit.com/photocells/using-a-photocell

[43] OpenEnergyMonitor. CT sensors - An introduction. [Online]. http://openenergymonitor.org/emon/buildingblocks/ct-sensors-introduction

[44] Arduino. Getting Started with Arduino on Windows. [Online]. https://www.arduino.cc/en/Guide/Windows

[45] Raspberry Pi Foundation. Quick Start Guide. [Online]. https://www.raspberrypi.org/help/quick-start-guide/

[46] Arduino Official Website. [Online]. www.arduino.cc

[47] Raspberry Pi Foundation Official Website. [Online]. https://www.raspberrypi.org

[48] Jan-Piet Mens. [Online]. http://jpmens.net/2013/03/10/visualizing-energy-consumption-with-mqtt/

[49] Mosquitto - An Open Source MQTT v3.1/v3.1.1 Broker. [Online]. http://mosquitto.org/

[50] MQTT.org. [Online]. http://mqtt.org/

[51] Instructables. Uber Home Automation w/ Arduino & Pi. [Online]. http://www.instructables.com/id/Uber-Home-Automation-w-Arduino-Pi/?ALLSTEPS

Παράρτημα – Πηγαίος Κώδικας

End-device (measurement-kit @Lamp-post)

Arduino Uno sketch

rfm69_endpoint.ino

```
#include <RFM69.h>
#include <RFM69registers.h>
#include <SPI.h>
#include <DHT11.h>
#include <Wire.h>
#include <stdlib.h>
#include "EmonLib.h"                   // Include Emon Library
#include "printf.h"

#define NODEID        2    //unique for each node on same network
#define DEVICEID      0
#define NETWORKID     100  //the same on all nodes that talk to each other
#define GATEWAYID     1

//Match frequency to the hardware version of the radio on your Arduino
#define FREQUENCY   RF69_868MHZ // other options: RF69_433MHZ  RF69_868MHZ RF69_915MHZ
#define ENCRYPTKEY    "0123456789012345" //exactly the same 16 characters/bytes on all nodes!
//#define IS_RFM69HW    //uncomment only for RFM69HW! Leave out if you have RFM69W!
#define ACK_TIME      50 // max # of ms to wait for an ack
#define SERIAL_BAUD   115200  //must be 9600 for GPS, use whatever if no GPS


/*  DeviceID=0 Standard package
 *  node ID, deviceID, battery, production, consumption, luminosity
 *  DeviceID=1 Standard + Extra features (meteo)
 *  temperature, humidity, rain
 *  DeviceID=2 Standard + Extra features (fire)
 *  UV, IR
 *  DeviceID=2 Standard + Extra features (meteo+fire)
 *  all of the above
*/

typedef struct {
  byte  nodeID;
  byte  deviceID;
  byte  batt;
  byte  pad[1];
  unsigned int prod_x100;
  unsigned int cons_x100;
  unsigned int lumi;
  int temp_x100;
```

```
42.   unsigned int humi_x100;
43.   unsigned int rain;
44.   unsigned int UV;
45.   unsigned int IR;
46. }Payload;
47. Payload theData;
48.
49. //byte sendSize = 0;
50. boolean requestACK = false;
51. unsigned long loop_start_time;
52.
53. RFM69 radio;
54.
55. EnergyMonitor emon1;                   // Create an instance
56. int photocellPin = 0;     // the cell and 10K pulldown are connected to a0
57.
58. #if (DEVICEID==1 || DEVICEID==3)
59.   #define DHT11PIN 2 // temp_humi sensor signal pin connected to D2
60.   DHT11 dht11(DHT11PIN);
61.   float temp, humi;
62. #endif
63.
64. void setup()
65. {
66.   Serial.begin(SERIAL_BAUD);          //  setup serial
67.   Serial.println(F("rfm69_endpoint "));
68.   Serial.println(F(" "));
69.   Wire.begin();
70.   delay(1000);
71.   Serial.print("Initializing Radio...");
72.   radio.initialize(FREQUENCY, NODEID, NETWORKID);
73.   Serial.println("OK!!");
74.
75.   radio.encrypt(ENCRYPTKEY);
76.   char buff[50];
77.   sprintf(buff, "\nTransmitting                  at                  %d                  Mhz...",
    FREQUENCY == RF69_433MHZ ? 433 : FREQUENCY == RF69_868MHZ ? 868 : 915);
78.   Serial.println(buff);
79.
80.     //Initialise theData
81.   theData.nodeID      = NODEID;
82.   theData.deviceID    = DEVICEID;
83.   theData.batt        = 0;
84.   theData.pad[0]      = 0;
85.   theData.prod_x100   = 0;
86.   theData.cons_x100   = 0;
87.   theData.lumi        = 0;
88.   theData.temp_x100   = 0;
89.   theData.humi_x100   = 0;
```

```
90.   theData.rain        = 0;
91.   theData.UV          = 0;
92.   theData.IR          = 0;
93.
94.    emon1.current(1, 60.6);                  //  Current(input  pin,  calibration=  I1/I2/Burden
    Resistance=100A/0.05A/33Ω)
95.
96. }
97.
98. void loop() {
99.
100.  loop_start_time = millis();
101.
102.  // calculate battery voltage
103.  // ...
104.
105.  // calculate production
106.  // ...
107.
108.  // calculate consumption
109.  theData.cons_x100 = 230.0*emon1.calcIrms(1480)*100;
110.
111.  // calculate luminosity
112.  theData.lumi = analogRead(photocellPin);
113.
114.  //calculate temperature and humidity
115.  #if (DEVICEID==1 || DEVICEID==3)
116.    get_temp_humi();  //read temp, humi from sensor
117.    theData.temp_x100 = round(temp)*100;
118.    theData.humi_x100 = round(humi)*100;
119.  #endif
120.
121.  // calculate rain
122.  // ...
123.
124.  // calculate UV
125.  // ...
126.
127.  // calculate rain
128.  // ...
129.
130.  printOutgoingPacket();//print in serial for debugging
131.
132.  radio.send(GATEWAYID, (const void*)(&theData), sizeof(theData), requestACK);
133.
134.  //delay(10000);  // wait 10sec
135.  delay(10000-millis()+loop_start_time); //thelw synoliko delay 10sec
136.         //millis()-loop_start_time=posh wra ekane na metrhsei klp
137.         //prepei na perimenei parapanw gia 10sec-(millis()-loop_start_time)
```

```
138.
139.}//end loop
140.
141.void printOutgoingPacket(){
142.
143.    Serial.print("Message (");
144.    Serial.print(sizeof(theData));
145.    Serial.print(" bytes) | ");
146.    Serial.print("nodeID: ");
147.    Serial.print(theData.nodeID);
148.    Serial.print(", devID: ");
149.    Serial.println(theData.deviceID);
150.
151.//  Not yet supported
152.//    Serial.print("battery: ");
153.//    Serial.print(theData.batt);
154.//    Serial.print(", production(x100): ");
155.//    Serial.print(theData.prod_x100);
156.    Serial.print("consumption(x100): ");
157.    Serial.print(theData.cons_x100);
158.
159.    Serial.print(", luminosity: ");
160.    Serial.println(theData.lumi);
161.
162.  #if (DEVICEID==1 || DEVICEID==3)
163.    Serial.print("temp(x100): ");
164.    Serial.print(theData.temp_x100);
165.    Serial.print(", humi(x100): ");
166.    Serial.println(theData.humi_x100);
167.//    Serial.print(", rain: ");
168.//    Serial.println(theData.rain);
169.  #endif
170.
171.  #if (DEVICEID==2 || DEVICEID==3)
172.//  Not yet supported
173.//    Serial.print("UV: ");
174.//    Serial.print(theData.UV);
175.//    Serial.print(", IR: ");
176.//    Serial.println(theData.IR);
177.  #endif
178.}
179.
180.#if (DEVICEID==1 || DEVICEID==3)
181.void get_temp_humi(){
182.    int err;
183.    if((err=dht11.read(humi, temp))==0){
184.  //    Serial.print("temperature:");
185.  //    Serial.print(temp);
186.  //    Serial.print(" humidity:");
```

```
187.  //    Serial.print(humi);
188.  //    Serial.println();
189.    }else{
190.  //    Serial.println();
191.  //    Serial.print("Error No :");
192.  //    Serial.print(err);
193.  //    Serial.println();
194.    }
195.    delay(DHT11_RETRY_DELAY); //delay for reread
196.  }
197.#endif
```

Arduino Nano sketch

nrf24_endpoint.ino

```
1.  #include <SPI.h>
2.  #include "RF24.h"
3.  #include "EmonLib.h"
4.  #include <DHT11.h>
5.
6.  #define NODEID    1
7.  #define DEVICEID  1
8.  #define SERIAL_BAUD   115200
9.
10. /*  DeviceID=0 Standard package
11.  *  node ID, deviceID, battery, production, consumption, luminosity
12.  *  DeviceID=1 Standard + Extra features (meteo)
13.  *  temperature, humidity, rain
14.  *  DeviceID=2 Standard + Extra features (fire)
15.  *  UV, IR
16.  *  DeviceID=2 Standard + Extra features (meteo+fire)
17.  *  all of the above
18. */
19.
20. typedef struct {
21.   byte  nodeID;
22.   byte  deviceID;
23.   byte  batt;
24.   byte  pad[1];
25.   unsigned int prod_x100;
26.   unsigned int cons_x100;
27.   unsigned int lumi;
28.   int temp_x100;
29.   unsigned int humi_x100;
30.   unsigned int rain;
31.   unsigned int UV;
32.   unsigned int IR;
```

```
}Payload;
Payload theData;

/* Hardware configuration: Set up nRF24L01+ radio on SPI bus plus pins 9 & 10 */
RF24 radio(9,10);
byte addresses[][6] = {"1Node","2Node"}; // Radio pipe addresses for the 2 nodes to communicate.

//unsigned long timer;
EnergyMonitor emon1;                   // Create an instance
int photocellPin = 0;     // the cell and 10K pulldown are connected to a0

#if (DEVICEID==1 || DEVICEID==3)
  #define DHT11PIN 2 // temp_humi sensor signal pin connected to D2
  DHT11 dht11(DHT11PIN);
  float temp, humi;
#endif

void setup(){

  Serial.begin(SERIAL_BAUD);          //  setup serial
  Serial.println(F("nrf24_endpoint "));
  Serial.println(F(" "));

  // Setup and configure radio
  radio.begin();
  radio.enableAckPayload();                     // Allow optional ack payloads

  // This opens two pipes for these two nodes to communicate back and forth.
   radio.openWritingPipe(addresses[0]);            // Both radios listen on the same pipes by default, but opposite addresses
  radio.openReadingPipe(1,addresses[1]);        // Open a reading pipe on address 1, pipe 0

  radio.startListening();                       // Start listening
  //radio.printDetails();

  //Initialise theData
  theData.nodeID      = NODEID;
  theData.deviceID    = DEVICEID;
  theData.batt        = 0;
  theData.pad[0]      = 0;
  theData.prod_x100   = 0;
  theData.cons_x100   = 0;
  theData.lumi        = 0;
  theData.temp_x100   = 0;
  theData.humi_x100   = 0;
  theData.rain        = 0;
  theData.UV          = 0;
  theData.IR          = 0;
```

```
80.
81.   //emon1.current(1,  60.6);                 //  Current(input  pin,  calibration=  I1/I2/Burden
    Resistance=100A/0.05A/33Ω)
82.
83. }
84.
85. void loop(void) {
86.
87.   unsigned long loop_start_time = millis();
88.
89.   // calculate battery voltage
90.   // ...
91.
92.   // calculate production
93.   // ...
94.
95.   // calculate consumption - not supported on Nano
96.   //theData.cons_x100 = 230.0*emon1.calcIrms(1480)*100;
97.
98.   // calculate luminosity
99.   theData.lumi = analogRead(photocellPin);
100.
101.  //calculate temperature and humidity
102.  #if (DEVICEID==1 || DEVICEID==3)
103.    get_temp_humi();  //read temp, humi from sensor
104.    theData.temp_x100 = round(temp)*100;
105.    theData.humi_x100 = round(humi)*100;
106.  #endif
107.
108.  // calculate rain
109.  // ...
110.
111.  // calculate UV
112.  // ...
113.
114.  // calculate rain
115.  // ...
116.
117.  printOutgoingPacket(); //print in serial for debugging
118.
119.   radio.stopListening();                                  // First, stop listening so we can
    talk.
120.  Serial.println(F("Now sending theData "));
121.
122.  unsigned long time = micros();                          // Record the current microsecond
    count
123.
124.  if ( radio.write(&theData,sizeof(theData)) ){           // Send the counter variable to the
    other radio
```

```
125.    if(!radio.available()){                             // If nothing in the buffer, we got an
    ack but it is blank
126.      Serial.print(F("Got blank response. round-trip delay: "));
127.      Serial.print(micros()-time);
128.      Serial.println(F(" microseconds"));
129.    }
130.    else{
131.      while(radio.available() ){                      // If an ack with payload was received
132.//        radio.read( &gotData, sizeof(gotData) );    // Read it, and display the response time
133.//        unsigned long timer = micros();
134.//        Serial.print(F("Got response "));
135.//        Serial.print(gotData);
136.//        Serial.print(F(" round-trip delay: "));
137.//        Serial.print(timer-time);
138.//        Serial.println(F(" microseconds"));
139.//        counter++;                                    // Increment the counter variable
140.      }
141.    }
142.  }
143.  else{
144.    Serial.println(F("Sending failed."));           // If no ack response, sending failed
145.  }
146.
147.  Serial.println(F(" "));
148.  //delay(10000);  // wait 10sec
149.  delay(10000-millis()+loop_start_time); //thelw synoliko delay 10sec
150.                //millis()-loop_start_time=posh wra ekane na metrhsei klp
151.                //prepei na perimenei parapanw gia 10sec-(millis()-loop_start_time)
152.}
153.
154.void printOutgoingPacket(){
155.
156.    Serial.print("Message (");
157.    Serial.print(sizeof(theData));
158.    Serial.print(" bytes) | ");
159.    Serial.print("nodeID: ");
160.    Serial.print(theData.nodeID);
161.    Serial.print(", devID: ");
162.    Serial.println(theData.deviceID);
163.
164.//  Not yet supported
165.//    Serial.print("battery: ");
166.//    Serial.print(theData.batt);
167.//    Serial.print(", production(x100): ");
168.//    Serial.print(theData.prod_x100);
169.//    Serial.print(", consumption(x100): ");
170.//    Serial.print(theData.cons_x100);
171.
172.    Serial.print("luminosity: ");
```

```
173.    Serial.println(theData.lumi);
174.
175.  #if (DEVICEID==1 || DEVICEID==3)
176.    Serial.print("temp(x100): ");
177.    Serial.print(theData.temp_x100);
178.    Serial.print(", humi(x100): ");
179.    Serial.println(theData.humi_x100);
180.//    Serial.print(", rain: ");
181.//    Serial.println(theData.rain);
182.  #endif
183.
184.  #if (DEVICEID==2 || DEVICEID==3)
185.//  Not yet supported
186.//    Serial.print("UV: ");
187.//    Serial.print(theData.UV);
188.//    Serial.print(", IR: ");
189.//    Serial.println(theData.IR);
190.  #endif
191.}
192.
193.
194.#if (DEVICEID==1 || DEVICEID==3)
195.void get_temp_humi(){
196.    int err;
197.    if((err=dht11.read(humi, temp))==0){
198.  //    Serial.print("temperature:");
199.  //    Serial.print(temp);
200.  //    Serial.print(" humidity:");
201.  //    Serial.print(humi);
202.  //    Serial.println();
203.    }else{
204.  //    Serial.println();
205.  //    Serial.print("Error No :");
206.  //    Serial.print(err);
207.  //    Serial.println();
208.    }
209.    delay(DHT11_RETRY_DELAY); //delay for reread
210.  }
211.#endif
```

Gateway

Raspberry Pi 2 + RFM69

Στο directory rfm69_gateway:

rfm69_gateway.c

```
#include "rfm69.h"
#include <wiringPi.h>
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
#include <time.h>
#include <unistd.h>
#include <fcntl.h>
#include <string.h>
#include <pthread.h>
#include <errno.h>
#include "MQTTClient.h"
#include <mysql/mysql.h>

#define COUNTRY         "Greece"
#define REGION          "Attica"
#define GWID        "1"  // Gateway ID
#define MUNID       "Ilioup" // municipality ID
#define ADDRESS     "tcp://83.212.112.98:1883"
#define CLIENTID    "RFM69w1"
#define PAYLOAD     ""
#define QOS         1
#define TIMEOUT     10000L

#define FREQUENCY RF69_868MHZ
#define NODEID 1
#define NETWORKID 100
#define TXPOWER 31
// A 16 bit password
#define CRYPTPASS "0123456789012345"

#define SERVER          "83.212.112.98"
#define USER            "diploma_user"
#define PASSWORD        "1111"
#define DATABASE        "diplomadb"

#define LATITUDE        37.924926
#define LONGITUDE       23.7454326

#define NUMBEROFNODES 2 //how many nodes in this network?
#define LAT 0
#define LNG 1
float coordinates[NUMBEROFNODES][2] = { {37.924926,23.7454326},
                                                                {37.924927,23.7
454325}};
// coordinates[NODEID-1][0] == LAT
// coordinates[NODEID-1][1] == LNG
```

```
47.
48. /*
49. typedef struct {
50.         uint8_t nodeID;
51.         uint8_t deviceID;
52.         uint8_t batt;
53.         uint8_t pad[1];
54.         uint16_t prod_x100;
55.         uint16_t cons_x100;
56.         uint16_t lumi;
57.         int16_t temp_x100;
58.         uint16_t humi_x100;
59.         uint16_t rain;
60.         uint16_t UV;
61.         uint16_t IR;
62. }Payload;
63. */
64.
65. char received[63];
66. int converted[12];
67. int rssi;
68. char datalen;
69. char senderId;
70. char *TOPIC;
71.
72. uint8_t nodeID;
73. uint8_t deviceID;
74. uint8_t batt;
75. //uint8_t pad[1];
76. uint16_t prod_x100;
77. uint16_t cons_x100;
78. uint16_t lumi;
79. int16_t temp_x100;
80. uint16_t humi_x100;
81. uint16_t rain;
82. uint16_t UV;
83. uint16_t IR;
84.
85. uint8_t nodeID, deviceID;
86. //uint8_t batt;
87. //float prod;
88. float cons;
89. uint16_t lumi;
90. float temp, humi;
91. //uint16_t rain, UV, IR;
92.
93. void send2mySQL(void);
94. void send2MQTT(void);
95.
```

```
96. int main(int argc, char* argv[]) {
97.
98.         int i;
99.
100.        printf("rfm69_gateway.c\n");
101.        rfm69_initialize(FREQUENCY, NODEID, NETWORKID);
102.
103.        rfm69_encrypt(CRYPTPASS);
104.        rfm69_setPowerLevel(TXPOWER); //Set Power level: Max Power
105.
106.        while(1)
107.        {
108.                rfm69_receive();
109.                datalen = rfm69_getDataLen();
110.                if(datalen > 0)
111.                {
112.                        rssi = rfm69_getRssi();
113.                        rfm69_getData(received);
114.                        senderId = rfm69_getSenderId();
115.
116.                        printf("\nNew packet received!\n\r");
117.                        //printf("From: %i\n\r", senderId);
118.                        printf("Length: %i\n\r", datalen);
119.                        printf("RSSI: %i\n\r", rssi);
120.                        //printf("Data:\n\r");
121.                        for(i = 0; i < datalen; i++) {
122.                                //printf("%02x ", received[i]);
123.                                converted[i]=(int)received[i];
124.                        }
125.
126.                        /*** Data processing*****************************/
127.
128.                        nodeID  = converted[0];
129.                        printf("node = %i\n",nodeID);
130.
131.                        deviceID = converted[1];
132.                        printf("deviceID = %i\n",deviceID);
133.
134.                        //battery not yet supported
135.                        //batt = converted[2];
136.
137.                        //pad[0] = converted[3];
138.
139.                        //production not yet supported
140.                        //prod_x100 = 0x10000+converted[5]*0x100+converted[4];
141.
142.                        cons_x100 = converted[7]*0x100+converted[6];
143.                        printf("cons_x100 = %d\n",cons_x100);
144.                        cons = (float)cons_x100/100;
```

```
145.
146.                         lumi = converted[9]*0x100+converted[8];
147.                         printf("lumi = %d\n",lumi);
148.
149.                         if (deviceID == 1 || deviceID == 3){
150.                                 temp_x100 = converted[11]*0x100+converted[10];
151.                                 printf("temp x100 = %i\n",temp_x100);
152.                                 temp = (float)temp_x100/100;
153.
154.                                 humi_x100 = converted[13]*0x100+converted[12];
155.                                 printf("humi x100 = %i\n",humi_x100);
156.                                 humi = (float)humi_x100/100;
157.
158.                                 //rain not yet supported
159.                                 //rain = converted[15]*0x100+converted[14];
160.                         }
161.
162.                         /* UV, IR not yet supported
163.                         if (deviceID == 2 || deviceID == 3){
164.                                 UV = converted[17]*0x100+converted[16];
165.                                 IR = converted[19]*0x100+converted[18];
166.                         }
167.                         */
168.
169.                         /*********************************/
170.
171.                         send2MQTT();
172.                         send2mySQL();
173.                 }
174.         }
175.}
176.
177.void send2MQTT(){
178.
179.         MQTTClient client;
180.         MQTTClient_connectOptions conn_opts = MQTTClient_connectOptions_initializer;
181.         MQTTClient_message pubmsg = MQTTClient_message_initializer;
182.         MQTTClient_deliveryToken token;
183.         int rc;
184.
185.         MQTTClient_create(&client, ADDRESS, CLIENTID, MQTTCLIENT_PERSISTENCE_NONE, NULL);
186.         conn_opts.keepAliveInterval = 20;
187.         //  conn_opts.cleansession = 1;
188.
189.         if ((rc = MQTTClient_connect(client, &conn_opts)) != MQTTCLIENT_SUCCESS){
190.                 printf("Failed to connect, return code %d\n", rc);
191.                 //exit(-1);
192.         }
193.
```

```
194.        char topic_buffer[80];
195.        char measure_buffer[7];
196.
197.        //Send consumption
198.         sprintf(topic_buffer,"diploma/%s/GW_%s/node_%i/dev_%i/cons/",MUNID,GWID,nodeID,deviceID
    );
199.        //printf("%s\n",topic_buffer); // e.g. "diploma/Ilioup/GW_1/node_2/dev_1/cons/"
200.        TOPIC=topic_buffer; //printf("%s\n",TOPIC);
201.        sprintf(measure_buffer,"%.2f", cons);  //float to string
202.        //printf("topic_buffer cons = %s\n", measure_buffer);
203.        pubmsg.payload = measure_buffer;
204.                    pubmsg.payloadlen = strlen(measure_buffer); //printf("pubmsg.payloadlen    =
    %i\n",pubmsg.payloadlen);
205.        pubmsg.qos = QOS;
206.        pubmsg.retained = 0;
207.        MQTTClient_publishMessage(client, TOPIC, &pubmsg, &token);
208.           printf("Waiting  for  up  to  %d  sec  to  publish  %.2f  on  topic  %s  |  client:
    %s\n", (int)(TIMEOUT/1000), cons, TOPIC, CLIENTID);
209.        rc = MQTTClient_waitForCompletion(client, token, TIMEOUT);
210.        printf("Message with delivery token %d delivered\n", token);
211.
212.        /* Production, battery not yet supported
213.        //Send production
214.         sprintf(topic_buffer,"diploma/%s/GW_%s/node_%i/dev_%i/prod/",MUNID,GWID,nodeID,deviceID
    );
215.        //printf("%s\n",topic_buffer); // e.g. "diploma/Ilioup/GW_1/node_2/dev_1/prod/"
216.        TOPIC=topic_buffer; //printf("%s\n",TOPIC);
217.        sprintf(measure_buffer,"%.2f", prod);  //float to string
218.        //printf("topic_buffer prod = %s\n", topic_buffer);
219.        pubmsg.payload = measure_buffer;
220.               pubmsg.payloadlen   =   strlen(measure_buffer);   //printf("pubmsg.payloadlen    =
    %i\n",pubmsg.payloadlen);
221.        pubmsg.qos = QOS;
222.        pubmsg.retained = 0;
223.        MQTTClient_publishMessage(client, TOPIC, &pubmsg, &token);
224.          printf("Waiting  for  up  to  %d  sec  to  publish  %.2f  on  topic  %s  |  client:  %s\n",
    (int)(TIMEOUT/1000), prod, TOPIC, CLIENTID);
225.        rc = MQTTClient_waitForCompletion(client, token, TIMEOUT);
226.        printf("Message with delivery token %d delivered\n", token);
227.
228.        //Send battery voltage
229.         sprintf(topic_buffer,"diploma/%s/GW_%s/node_%i/dev_%i/batt/",MUNID,GWID,nodeID,deviceID
    );
230.        //printf("%s\n",topic_buffer); // e.g. "diploma/Ilioup/GW_1/node_2/dev_1/batt/"
231.        TOPIC=topic_buffer; //printf("%s\n",TOPIC);
232.        sprintf(measure_buffer,"%if", batt);  //float to string
233.        //printf("topic_buffer batt = %s\n", topic_buffer);
234.        pubmsg.payload = measure_buffer;
```

```
235.              pubmsg.payloadlen = strlen(measure_buffer);  //printf("pubmsg.payloadlen =
    %i\n",pubmsg.payloadlen);
236.        pubmsg.qos = QOS;
237.        pubmsg.retained = 0;
238.        MQTTClient_publishMessage(client, TOPIC, &pubmsg, &token);
239.           printf("Waiting for up to %d sec to publish %i on topic %s | client: %s\n",
    (int)(TIMEOUT/1000), batt, TOPIC, CLIENTID);
240.        rc = MQTTClient_waitForCompletion(client, token, TIMEOUT);
241.        printf("Message with delivery token %d delivered\n", token);
242.
243.        */
244.
245.        //Send luminosity
246.         sprintf(topic_buffer,"diploma/%s/GW_%s/node_%i/dev_%i/lumi/",MUNID,GWID,nodeID,deviceID
    );
247.        //printf("%s\n",topic_buffer); // e.g. "diploma/Ilioup/GW_1/node_2/dev_1/lumi/"
248.        TOPIC=topic_buffer; //printf("%s\n",TOPIC);
249.        sprintf(measure_buffer,"%d", lumi);  //int to string
250.        //printf("topic_buffer lumi = %s\n", measure_buffer);
251.        pubmsg.payload = measure_buffer;
252.                    pubmsg.payloadlen = strlen(measure_buffer); //printf("pubmsg.payloadlen =
    %i\n",pubmsg.payloadlen);
253.        pubmsg.qos = QOS;
254.        pubmsg.retained = 0;
255.        MQTTClient_publishMessage(client, TOPIC, &pubmsg, &token);
256.           printf("Waiting for up to %d sec to publish %d on topic %s | client:
    %s\n", (int)(TIMEOUT/1000), lumi, TOPIC, CLIENTID);
257.        rc = MQTTClient_waitForCompletion(client, token, TIMEOUT);
258.        printf("Message with delivery token %d delivered\n", token);
259.
260.
261.        if (deviceID == 1 || deviceID == 3){
262.
263.                //Send temperature
264.                 sprintf(topic_buffer,"diploma/%s/GW_%s/node_%i/dev_%i/temp/",MUNID,GWID,nodeID,
    deviceID);
265.                //printf("%s\n",topic_buffer); // e.g. "diploma/Ilioup/GW_1/node_2/dev_1/temp/"
266.                TOPIC=topic_buffer; //printf("%s\n",TOPIC);
267.                sprintf(measure_buffer,"%.2f", temp);  //float to string
268.                //printf("topic_buffer temp = %s\n", measure_buffer);
269.                pubmsg.payload = measure_buffer;
270.                     pubmsg.payloadlen = strlen(measure_buffer); //printf("pubmsg.payloadlen =
    %i\n",pubmsg.payloadlen);
271.                pubmsg.qos = QOS;
272.                pubmsg.retained = 0;
273.                MQTTClient_publishMessage(client, TOPIC, &pubmsg, &token);
274.                   printf("Waiting for up to %d sec to publish %.2f on topic %s | client:
    %s\n", (int)(TIMEOUT/1000), temp, TOPIC, CLIENTID);
275.                rc = MQTTClient_waitForCompletion(client, token, TIMEOUT);
```

```
276.                printf("Message with delivery token %d delivered\n", token);
277.
278.                //Send humidity
279.                 sprintf(topic_buffer,"diploma/%s/GW_%s/node_%i/dev_%i/humi/",MUNID,GWID,nodeID,
    deviceID);
280.                //printf("%s\n",topic_buffer); // e.g. "diploma/Ilioup/GW_1/node_2/dev_1/humi/"
281.                TOPIC=topic_buffer; //printf("%s\n",TOPIC);
282.                sprintf(measure_buffer,"%.2f", humi);  //float to string
283.                //printf("topic_buffer humi = %s\n", measure_buffer);
284.                pubmsg.payload = measure_buffer;
285.                      pubmsg.payloadlen = strlen(measure_buffer); //printf("pubmsg.payloadlen  =
    %i\n",pubmsg.payloadlen);
286.                pubmsg.qos = QOS;
287.                pubmsg.retained = 0;
288.                MQTTClient_publishMessage(client, TOPIC, &pubmsg, &token);
289.                   printf("Waiting for up to %d sec to publish %.2f on topic %s | client:
    %s\n", (int)(TIMEOUT/1000), humi, TOPIC, CLIENTID);
290.                rc = MQTTClient_waitForCompletion(client, token, TIMEOUT);
291.                printf("Message with delivery token %d delivered\n", token);
292.
293.                /* not yet supported
294.                //Send rain
295.                 sprintf(topic_buffer,"diploma/%s/GW_%s/node_%i/dev_%i/rain/",MUNID,GWID,nodeID,
    deviceID);
296.                //printf("%s\n",topic_buffer); // e.g. "diploma/Ilioup/GW_1/node_2/dev_1/rain/"
297.                TOPIC=topic_buffer; //printf("%s\n",TOPIC);
298.                sprintf(measure_buffer,"%d", rain);  //int to string
299.                //printf("topic_buffer rain = %s\n", measure_buffer);
300.                pubmsg.payload = measure_buffer;
301.                    pubmsg.payloadlen = strlen(measure_buffer); //printf("pubmsg.payloadlen =
    %i\n",pubmsg.payloadlen);
302.                pubmsg.qos = QOS;
303.                pubmsg.retained = 0;
304.                MQTTClient_publishMessage(client, TOPIC, &pubmsg, &token);
305.                  printf("Waiting for up to %d sec to publish %d on topic %s | client: %s\n",
    (int)(TIMEOUT/1000), rain, TOPIC, CLIENTID);
306.                rc = MQTTClient_waitForCompletion(client, token, TIMEOUT);
307.                printf("Message with delivery token %d delivered\n", token)
308.
309.                */
310.        }
311.
312.        if (deviceID == 2 || deviceID == 3){
313.
314.                /* not yet supported
315.                //Send UV
316.                 sprintf(topic_buffer,"diploma/%s/GW_%s/node_%i/dev_%i/UV/",MUNID,GWID,nodeID,de
    viceID);
317.                //printf("%s\n",topic_buffer); // e.g. "diploma/Ilioup/GW_1/node_2/dev_1/UV/"
```

```
318.                TOPIC=topic_buffer; //printf("%s\n",TOPIC);
319.                sprintf(measure_buffer,"%d", UV);  //int to string
320.                //printf("topic_buffer UV = %s\n", measure_buffer);
321.                pubmsg.payload = measure_buffer;
322.                    pubmsg.payloadlen = strlen(measure_buffer); //printf("pubmsg.payloadlen =
    %i\n",pubmsg.payloadlen);
323.                pubmsg.qos = QOS;
324.                pubmsg.retained = 0;
325.                MQTTClient_publishMessage(client, TOPIC, &pubmsg, &token);
326.                 printf("Waiting for up to %d sec to publish %d on topic %s | client: %s\n",
    (int)(TIMEOUT/1000), UV, TOPIC, CLIENTID);
327.                rc = MQTTClient_waitForCompletion(client, token, TIMEOUT);
328.                printf("Message with delivery token %d delivered\n", token)
329.
330.                //Send IR
331.                 sprintf(topic_buffer,"diploma/%s/GW_%s/node_%i/dev_%i/IR/",MUNID,GWID,nodeID,de
    viceID);
332.                //printf("%s\n",topic_buffer); // e.g. "diploma/Ilioup/GW_1/node_2/dev_1/IR/"
333.                TOPIC=topic_buffer; //printf("%s\n",TOPIC);
334.                sprintf(measure_buffer,"%d", IR);  //int to string
335.                //printf("topic_buffer IR = %s\n", measure_buffer);
336.                pubmsg.payload = measure_buffer;
337.                    pubmsg.payloadlen = strlen(measure_buffer); //printf("pubmsg.payloadlen =
    %i\n",pubmsg.payloadlen);
338.                pubmsg.qos = QOS;
339.                pubmsg.retained = 0;
340.                MQTTClient_publishMessage(client, TOPIC, &pubmsg, &token);
341.                 printf("Waiting for up to %d sec to publish %d on topic %s | client: %s\n",
    (int)(TIMEOUT/1000), IR, TOPIC, CLIENTID);
342.                rc = MQTTClient_waitForCompletion(client, token, TIMEOUT);
343.                printf("Message with delivery token %d delivered\n", token)
344.
345.                */
346.
347.        }
348.
349.        MQTTClient_disconnect(client, 10000);
350.        MQTTClient_destroy(&client);
351.}
352.
353.void send2mySQL(){
354.
355.        //initialize MYSQL object for connections
356.        MYSQL *mysql1;
357.        mysql1 = mysql_init(NULL);
358.
359.        if(mysql1 == NULL){
360.                fprintf(stderr, "%s\n", mysql_error(mysql1));
361.                //exit(-1);
```

```
362.                //return;
363.        }
364.
365.        //Connect to the database
366.        if(mysql_real_connect(mysql1, SERVER, USER, PASSWORD, DATABASE, 0, NULL, 0) == NULL)
367.        {
368.                fprintf(stderr, "%s\n", mysql_error(mysql1));
369.                //exit(-1);
370.        }
371.        else
372.        {
373.                printf("Database connection successful.\n");
374.        }
375.
376.        char statement[512];
377.
378.        //Send data
379.        if(mysql1 != NULL)
380.        {
381.                snprintf(statement, sizeof(statement), \
382.                        "INSERT INTO `diplomadb`.`measurements` (\
383.                        `id` ,`datetime` ,`country` ,`region` ,`municipality` ,`gwid` ,`nodeid`
    ,`deviceid` ,`lat` ,`lng` ,\
384.                        `cons` ,`prod` ,`batt` ,`lumi` ,`temp` ,`humi`)VALUES (\
385.                          NULL , CURRENT_TIMESTAMP , '%s', '%s', '%s', '%s', '%i', '%i', '%f',
    '%f', \
386.                        '%.2f', NULL , NULL , '%i', NULL , NULL\
387.                        )"\
388.                           ,COUNTRY, REGION, MUNID, GWID, nodeID, deviceID, coordinates[nodeID-
    1][LAT], coordinates[nodeID-1][LNG], \
389.                        cons, lumi);
390.                //printf("SQL query %s\n",statement);
391.
392.                if (mysql_query(mysql1, statement)){
393.                        fprintf(stderr, "%s\n", mysql_error(mysql1));
394.                }
395.        }
396.
397.        //Disconnect from the database
398.        mysql_close(mysql1);
399.        printf("Disconnected from database.\n");
400.}
```

Makefile

```
1.  CCFLAGS= -lpaho-mqtt3c -lwiringPi  `mysql_config --cflags` `mysql_config --libs`
2.
```

```
# define all programs
PROGRAMS =  rfm69_gateway rfm69_rangetest
SOURCES = ${PROGRAMS:=.c}

all: ${PROGRAMS}

${PROGRAMS}: ${SOURCES}
        gcc ${CCFLAGS} rfm69.c $@.c -o $@
#       gcc ${CCFLAGS} -Wall rfm69.c $@.c -o $@

clean:
        rm -rf $(PROGRAMS)
```

Raspberry Pi 2 + nRF24L01+

Στο directory nrf24_gateway τοποθετούμε τα παρακάτω αρχεία:

nrf24_gateway.c

```
#include <cstdlib>
#include <iostream>
#include <sstream>
#include <string>
#include <RF24/RF24.h>
#include <mysql/mysql.h>
  extern "C"
{
        #include "MQTTClient.h"
}

#define COUNTRY         "Greece"
#define REGION          "Attica"
#define GWID        "1"  // Gateway ID
#define MUNID       "Ilioupoli" // municipality ID
#define ADDRESS     "tcp://83.212.112.98:1883"
#define PAYLOAD     ""
#define QOS         1
#define TIMEOUT     10000L
#define CLIENTID    "GWID1-nRF24"

#define SERVER          "83.212.112.98"
#define USER            "evgenia"
#define PASSWORD        "1111"
#define DATABASE        "measurements"

#define NUMBEROFNODES 2 //how many nodes in this network?
#define LAT 0
```

```
29. #define LNG 1
30. float coordinates[NUMBEROFNODES][2] = { {37.924926,23.7454326},\
31.                                                                 {37.924927,23.7
    454325}};
32.
33. // coordinates[NODEID-1][0] == LAT
34. // coordinates[NODEID-1][1] == LNG
35.
36. using namespace std;
37.
38. // Hardware configuration
39. // Configure the appropriate pins for your connections
40. RF24 radio(25,1);
41.
42. typedef struct {
43.         uint8_t nodeID;
44.         uint8_t deviceID;
45.         uint8_t batt;
46.         uint8_t pad[1];
47.         uint16_t prod_x100;
48.         uint16_t cons_x100;
49.         uint16_t lumi;
50.         int16_t temp_x100;
51.         uint16_t humi_x100;
52.         uint16_t rain;
53.         uint16_t UV;
54.         uint16_t IR;
55. }Payload;
56. Payload gotData;
57.
58. // Radio pipe addresses for the 2 nodes to communicate.
59. const uint8_t addresses[][6] = {"1Node","2Node"};
60.
61. uint8_t counter = 1;    // A single byte to keep track of the data being sent back and forth
62.
63. uint8_t nodeID, deviceID;
64. //uint8_t batt;
65. //float prod, cons;
66. uint16_t lumi;
67. float temp, humi;
68. //uint16_t rain, UV, IR;
69.
70.
71. //int nodeID, deviceID;
72. //float cons, prod;
73. //
74. /* int16_t lumi;
75. float temp, humi;
76. uint8_t batt;  */
```

```
77.
78. char *TOPIC;
79.
80. void send2MQTT(void);
81. void send2MySQL(void);
82.
83. int main(int argc, char** argv){
84.
85.         printf("RPi/RF24/examples/gettingstarted_call_response\n");
86.         radio.begin();
87.         radio.enableAckPayload();               // Allow optional ack payloads
88.         radio.enableDynamicAck();
89.          radio.printDetails();                    // Dump the configuration of the rf unit for
    debugging
90.
91.         radio.openWritingPipe(addresses[1]);
92.         radio.openReadingPipe(1,addresses[0]);
93.
94.         radio.startListening();
95.         radio.writeAckPayload(1,&counter,sizeof(counter));
96.
97.         radio.stopListening();
98.         radio.startListening();
99.
100.        // forever loop
101.        while (1){
102.
103.                delay(10);
104.                 uint8_t pipeNo;                                 // Declare variables for the
    pipe
105.
106.                if( radio.available(&pipeNo))
107.                {
108.                        // Read all available payloads
109.                        //printf("radio available\n");
110.                        radio.read( &gotData, sizeof(gotData) );
111.                        printf("\nI received %d bytes!\n", sizeof(gotData) );
112.
113.                        /*** Data processing*****************************/
114.
115.                        nodeID = gotData.nodeID;  printf("nodeID:%d\n",nodeID);
116.                        deviceID = gotData.deviceID; printf("deviceID:%d\n",deviceID);
117.                        /* consumption, production, battery not yet supported
118.                        batt = gotData.batt;
119.                        prod = (float)gotData.prod_x100/100;
120.                        cons = (float)gotData.cons_x100/100;
121.                        */
122.                        lumi = gotData.lumi; printf("lumi:%d\n",lumi);
123.                        if (deviceID == 1 || deviceID == 3){
```

```
124.                                temp = (float)gotData.temp_x100/100; printf("temp:%.2f\n",temp)
    ;
125.                                humi = (float)gotData.humi_x100/100; printf("humi:%.2f\n",humi)
    ;
126.                                //rain not yet supported
127.                                //rain = gotData.rain;
128.                        }
129.                        /* UV, IR not yet supported
130.                        if (deviceID == 2 || deviceID == 3){
131.                                UV = gotData.UV;
132.                                IR = gotData.IR;
133.                        }
134.                        */
135.                        /**********************************/
136.
137.                        send2MQTT();
138.                        send2MySQL();
139.                }
140.} //while 1
141.} //main
142.
143.void send2MQTT(){
144.
145.        MQTTClient client;
146.        MQTTClient_connectOptions conn_opts = MQTTClient_connectOptions_initializer;
147.        MQTTClient_message pubmsg = MQTTClient_message_initializer;
148.        MQTTClient_deliveryToken token;
149.        int rc;
150.
151.        MQTTClient_create(&client, ADDRESS, CLIENTID, MQTTCLIENT_PERSISTENCE_NONE, NULL);
152.        conn_opts.keepAliveInterval = 20;
153.        //  conn_opts.cleansession = 1;
154.
155.        if ((rc = MQTTClient_connect(client, &conn_opts)) != MQTTCLIENT_SUCCESS){
156.                printf("Failed to connect, return code %d\n", rc);
157.                //exit(-1);
158.        }
159.
160.        char topic_buffer[80];
161.        char measure_buffer[7];
162.
163.        /* Consumption, production, battery not yet supported
164.        //Send consumption
165.         sprintf(topic_buffer,"diploma/%s/GW_%s/node_%i/dev_%i/cons/",MUNID,GWID,nodeID,deviceID
    );
166.        //printf("%s\n",topic_buffer); // e.g. "diploma/Ilioup/GW_1/node_2/dev_1/cons/"
167.        TOPIC=topic_buffer; //printf("%s\n",TOPIC);
168.        sprintf(measure_buffer,"%.2f", cons);  //float to string
169.        //printf("topic_buffer cons = %s\n", measure_buffer);
```

```
170.        pubmsg.payload = measure_buffer;
171.              pubmsg.payloadlen   =   strlen(measure_buffer);   //printf("pubmsg.payloadlen   =
    %i\n",pubmsg.payloadlen);
172.        pubmsg.qos = QOS;
173.        pubmsg.retained = 0;
174.        MQTTClient_publishMessage(client, TOPIC, &pubmsg, &token);
175.          printf("Waiting for up to %d sec to publish %.2f on topic %s | client: %s\n",
    (int)(TIMEOUT/1000), cons, TOPIC, CLIENTID);
176.        rc = MQTTClient_waitForCompletion(client, token, TIMEOUT);
177.        printf("Message with delivery token %d delivered\n", token);
178.
179.
180.        //Send production
181.         sprintf(topic_buffer,"diploma/%s/GW_%s/node_%i/dev_%i/prod/",MUNID,GWID,nodeID,deviceID
    );
182.        //printf("%s\n",topic_buffer); // e.g. "diploma/Ilioup/GW_1/node_2/dev_1/prod/"
183.        TOPIC=topic_buffer; //printf("%s\n",TOPIC);
184.        sprintf(measure_buffer,"%.2f", prod);  //float to string
185.        //printf("topic_buffer prod = %s\n", topic_buffer);
186.        pubmsg.payload = measure_buffer;
187.              pubmsg.payloadlen   =   strlen(measure_buffer);   //printf("pubmsg.payloadlen   =
    %i\n",pubmsg.payloadlen);
188.        pubmsg.qos = QOS;
189.        pubmsg.retained = 0;
190.        MQTTClient_publishMessage(client, TOPIC, &pubmsg, &token);
191.          printf("Waiting for up to %d sec to publish %.2f on topic %s | client: %s\n",
    (int)(TIMEOUT/1000), prod, TOPIC, CLIENTID);
192.        rc = MQTTClient_waitForCompletion(client, token, TIMEOUT);
193.        printf("Message with delivery token %d delivered\n", token);
194.
195.        //Send battery voltage
196.         sprintf(topic_buffer,"diploma/%s/GW_%s/node_%i/dev_%i/batt/",MUNID,GWID,nodeID,deviceID
    );
197.        //printf("%s\n",topic_buffer); // e.g. "diploma/Ilioup/GW_1/node_2/dev_1/batt/"
198.        TOPIC=topic_buffer; //printf("%s\n",TOPIC);
199.        sprintf(measure_buffer,"%if", batt);  //float to string
200.        //printf("topic_buffer batt = %s\n", topic_buffer);
201.        pubmsg.payload = measure_buffer;
202.              pubmsg.payloadlen   =   strlen(measure_buffer);   //printf("pubmsg.payloadlen   =
    %i\n",pubmsg.payloadlen);
203.        pubmsg.qos = QOS;
204.        pubmsg.retained = 0;
205.        MQTTClient_publishMessage(client, TOPIC, &pubmsg, &token);
206.          printf("Waiting for up to %d sec to publish %i on topic %s | client: %s\n",
    (int)(TIMEOUT/1000), batt, TOPIC, CLIENTID);
207.        rc = MQTTClient_waitForCompletion(client, token, TIMEOUT);
208.        printf("Message with delivery token %d delivered\n", token);
209.
210.        */
```

```
211.
212.         //Send luminosity
213.          sprintf(topic_buffer,"diploma/%s/GW_%s/node_%i/dev_%i/lumi/",MUNID,GWID,nodeID,deviceID
    );
214.         //printf("%s\n",topic_buffer); // e.g. "diploma/Ilioup/GW_1/node_2/dev_1/lumi/"
215.         TOPIC=topic_buffer; //printf("%s\n",TOPIC);
216.         sprintf(measure_buffer,"%d", lumi);  //int to string
217.         //printf("topic_buffer lumi = %s\n", measure_buffer);
218.         pubmsg.payload = measure_buffer;
219.                     pubmsg.payloadlen = strlen(measure_buffer); //printf("pubmsg.payloadlen    =
    %i\n",pubmsg.payloadlen);
220.         pubmsg.qos = QOS;
221.         pubmsg.retained = 0;
222.         MQTTClient_publishMessage(client, TOPIC, &pubmsg, &token);
223.             printf("Waiting  for  up  to  %d  sec  to  publish  %d  on  topic  %s  |  client:
    %s\n", (int)(TIMEOUT/1000), lumi, TOPIC, CLIENTID);
224.         rc = MQTTClient_waitForCompletion(client, token, TIMEOUT);
225.         printf("Message with delivery token %d delivered\n", token);
226.
227.
228.         if (deviceID == 1 || deviceID == 3){
229.
230.                 //Send temperature
231.                  sprintf(topic_buffer,"diploma/%s/GW_%s/node_%i/dev_%i/temp/",MUNID,GWID,nodeID,
    deviceID);
232.                 //printf("%s\n",topic_buffer); // e.g. "diploma/Ilioup/GW_1/node_2/dev_1/temp/"
233.                 TOPIC=topic_buffer; //printf("%s\n",TOPIC);
234.                 sprintf(measure_buffer,"%.2f", temp);  //float to string
235.                 //printf("topic_buffer temp = %s\n", measure_buffer);
236.                 pubmsg.payload = measure_buffer;
237.                       pubmsg.payloadlen = strlen(measure_buffer); //printf("pubmsg.payloadlen  =
    %i\n",pubmsg.payloadlen);
238.                 pubmsg.qos = QOS;
239.                 pubmsg.retained = 0;
240.                 MQTTClient_publishMessage(client, TOPIC, &pubmsg, &token);
241.                    printf("Waiting  for  up  to  %d  sec  to  publish  %.2f  on  topic  %s  |  client:
    %s\n", (int)(TIMEOUT/1000), temp, TOPIC, CLIENTID);
242.                 rc = MQTTClient_waitForCompletion(client, token, TIMEOUT);
243.                 printf("Message with delivery token %d delivered\n", token);
244.
245.                 //Send humidity
246.                  sprintf(topic_buffer,"diploma/%s/GW_%s/node_%i/dev_%i/humi/",MUNID,GWID,nodeID,
    deviceID);
247.                 //printf("%s\n",topic_buffer); // e.g. "diploma/Ilioup/GW_1/node_2/dev_1/humi/"
248.                 TOPIC=topic_buffer; //printf("%s\n",TOPIC);
249.                 sprintf(measure_buffer,"%.2f", humi);  //float to string
250.                 //printf("topic_buffer humi = %s\n", measure_buffer);
251.                 pubmsg.payload = measure_buffer;
```

```
252.                     pubmsg.payloadlen = strlen(measure_buffer); //printf("pubmsg.payloadlen =
    %i\n",pubmsg.payloadlen);
253.                 pubmsg.qos = QOS;
254.                 pubmsg.retained = 0;
255.                 MQTTClient_publishMessage(client, TOPIC, &pubmsg, &token);
256.                    printf("Waiting for up to %d sec to publish %.2f on topic %s | client:
    %s\n", (int)(TIMEOUT/1000), humi, TOPIC, CLIENTID);
257.                 rc = MQTTClient_waitForCompletion(client, token, TIMEOUT);
258.                 printf("Message with delivery token %d delivered\n", token);
259.
260.                 /* not yet supported
261.                 //Send rain
262.                  sprintf(topic_buffer,"diploma/%s/GW_%s/node_%i/dev_%i/rain/",MUNID,GWID,nodeID,
    deviceID);
263.                 //printf("%s\n",topic_buffer); // e.g. "diploma/Ilioup/GW_1/node_2/dev_1/rain/"
264.                 TOPIC=topic_buffer; //printf("%s\n",TOPIC);
265.                 sprintf(measure_buffer,"%d", rain);  //int to string
266.                 //printf("topic_buffer rain = %s\n", measure_buffer);
267.                 pubmsg.payload = measure_buffer;
268.                     pubmsg.payloadlen = strlen(measure_buffer); //printf("pubmsg.payloadlen =
    %i\n",pubmsg.payloadlen);
269.                 pubmsg.qos = QOS;
270.                 pubmsg.retained = 0;
271.                 MQTTClient_publishMessage(client, TOPIC, &pubmsg, &token);
272.                   printf("Waiting for up to %d sec to publish %d on topic %s | client: %s\n",
    (int)(TIMEOUT/1000), rain, TOPIC, CLIENTID);
273.                 rc = MQTTClient_waitForCompletion(client, token, TIMEOUT);
274.                 printf("Message with delivery token %d delivered\n", token)
275.
276.                 */
277.         }
278.
279.         if (deviceID == 2 || deviceID == 3){
280.
281.                 /* not yet supported
282.                 //Send UV
283.                  sprintf(topic_buffer,"diploma/%s/GW_%s/node_%i/dev_%i/UV/",MUNID,GWID,nodeID,de
    viceID);
284.                 //printf("%s\n",topic_buffer); // e.g. "diploma/Ilioup/GW_1/node_2/dev_1/UV/"
285.                 TOPIC=topic_buffer; //printf("%s\n",TOPIC);
286.                 sprintf(measure_buffer,"%d", UV);  //int to string
287.                 //printf("topic_buffer UV = %s\n", measure_buffer);
288.                 pubmsg.payload = measure_buffer;
289.                     pubmsg.payloadlen = strlen(measure_buffer); //printf("pubmsg.payloadlen =
    %i\n",pubmsg.payloadlen);
290.                 pubmsg.qos = QOS;
291.                 pubmsg.retained = 0;
292.                 MQTTClient_publishMessage(client, TOPIC, &pubmsg, &token);
```

```
293.                 printf("Waiting for up to %d sec to publish %d on topic %s | client: %s\n",
    (int)(TIMEOUT/1000), UV, TOPIC, CLIENTID);
294.                 rc = MQTTClient_waitForCompletion(client, token, TIMEOUT);
295.                 printf("Message with delivery token %d delivered\n", token)
296.
297.                 //Send IR
298.                 sprintf(topic_buffer,"diploma/%s/GW_%s/node_%i/dev_%i/IR/",MUNID,GWID,nodeID,de
    viceID);
299.                 //printf("%s\n",topic_buffer); // e.g. "diploma/Ilioup/GW_1/node_2/dev_1/IR/"
300.                 TOPIC=topic_buffer; //printf("%s\n",TOPIC);
301.                 sprintf(measure_buffer,"%d", IR);  //int to string
302.                 //printf("topic_buffer IR = %s\n", measure_buffer);
303.                 pubmsg.payload = measure_buffer;
304.                     pubmsg.payloadlen = strlen(measure_buffer); //printf("pubmsg.payloadlen =
    %i\n",pubmsg.payloadlen);
305.                 pubmsg.qos = QOS;
306.                 pubmsg.retained = 0;
307.                 MQTTClient_publishMessage(client, TOPIC, &pubmsg, &token);
308.                 printf("Waiting for up to %d sec to publish %d on topic %s | client: %s\n",
    (int)(TIMEOUT/1000), IR, TOPIC, CLIENTID);
309.                 rc = MQTTClient_waitForCompletion(client, token, TIMEOUT);
310.                 printf("Message with delivery token %d delivered\n", token)
311.
312.                 */
313.
314.         }
315.
316.         MQTTClient_disconnect(client, 10000);
317.         MQTTClient_destroy(&client);
318.}
319.
320.void send2MySQL(){
321.
322.         //initialize MYSQL object for connections
323.         MYSQL *mysql1;
324.         mysql1 = mysql_init(NULL);
325.
326.         if(mysql1 == NULL){
327.                 fprintf(stderr, "%s\n", mysql_error(mysql1));
328.                 //exit(-1);
329.                 //return;
330.         }
331.
332.         //Connect to the database
333.                         if(mysql_real_connect(mysql1,       SERVER,       USER,       PASSWORD,
    DATABASE, 0, NULL, 0) == NULL)    {
334.                 fprintf(stderr, "%s\n", mysql_error(mysql1));
335.                 //exit(-1);
336.         }else{
```

```
337.                printf("Database connection successful.\n");
338.        }
339.
340.        char statement[512];
341.
342.        //Send data
343.        if(mysql1 != NULL){
344.                snprintf(statement, sizeof(statement), \
345.                        "INSERT INTO `diplomadb`.`measurements` (\
346.                        `id` ,`datetime` ,`country` ,`region` ,`municipality` ,`gwid` ,`nodeid`
    ,`deviceid` ,`lat` ,`lng` ,\
347.                        `cons` ,`prod` ,`batt` ,`lumi` ,`temp` ,`humi`)VALUES (\
348.                          NULL , CURRENT_TIMESTAMP , '%s', '%s', '%s', '%s', '%i', '%i', '%f',
    '%f', \
349.                        NULL, NULL , NULL , '%i', '%.2f' , '%.2f'\
350.                        )"\
351.                          ,COUNTRY, REGION, MUNID, GWID, nodeID, deviceID, coordinates[nodeID-
    1][LAT], coordinates[nodeID-1][LNG], \
352.                        lumi, temp, humi);
353.                //printf("SQL query %s\n",statement);
354.
355.                if (mysql_query(mysql1, statement)){
356.                        fprintf(stderr, "%s\n", mysql_error(mysql1));
357.                }
358.        }
359.
360.        //Disconnect from the database
361.        mysql_close(mysql1);
362.        printf("Disconnected from database.\n");
363.}
```

Makefile

```
1.  prefix := /usr/local
2.  # Detect the Raspberry Pi by the existence of the bcm_host.h file
3.  BCMLOC=/opt/vc/include/bcm_host.h
4.
5.  ifneq ("$(wildcard $(BCMLOC))","")
6.  # The recommended compiler flags for the Raspberry Pi
7.  CCFLAGS=-Ofast -mfpu=vfp -mfloat-abi=hard -march=armv6zk -mtune=arm1176jzf-s -lpaho-mqtt3c
    `mysql_config --cflags` `mysql_config --libs`
8.  endif
9.
10. # define all programs
11. PROGRAMS =  nrf24_gateway
12. #test test2
13. SOURCES = ${PROGRAMS:=.cpp}
```

```
14.
15. all: ${PROGRAMS}
16.
17. ${PROGRAMS}: ${SOURCES}
18.         g++ ${CCFLAGS} -Wall -I../ -lrf24-bcm $@.cpp -o $@
19.
20. clean:
21.         rm -rf $(PROGRAMS)
22.
23. install: all
24.         test -d $(prefix) || mkdir $(prefix)
25.         test -d $(prefix)/bin || mkdir $(prefix)/bin
26.         for prog in $(PROGRAMS); do \
27.           install -m 0755 $$prog $(prefix)/bin; \
28.         done
29.
30. .PHONY: install
```

Server

Mosquitto configuration file

Mosquito.conf.diploma

```
1.  # Config file for mosquitto
2.  #
3.  # Use the # character to indicate a comment, but only if it is the
4.  # very first character on the line.
5.
6.  # =================================================================
7.  # General configuration
8.  # =================================================================
9.
10. # Write process id to a file. Default is a blank string which means
11. # a pid file shouldn't be written.
12. # This should be set to /var/run/mosquitto.pid if mosquitto is
13. # being run automatically on boot with an init script and
14. # start-stop-daemon or similar.
15. pid_file /var/run/mosquitto.pid
16.
17. # =================================================================
18. # Default listener
19. # =================================================================
20.
21. # Port to use for the default listener.
22. port 1883
23.
24. # Choose the protocol to use when listening.
```

```
25. # This can be either mqtt or websockets.
26. protocol mqtt
27.
28. # ================================================================
29. # Extra listeners
30. # ================================================================
31.
32. # Listen on a port/ip address combination. By using this variable
33. # multiple times, mosquitto can listen on more than one port. If
34. # this variable is used and neither bind_address nor port given,
35. # then the default listener will not be started.
36. # The port number to listen on must be given. Optionally, an ip
37. # address or host name may be supplied as a second argument. In
38. # this case, mosquitto will attempt to bind the listener to that
39. # address and so restrict access to the associated network and
40. # interface. By default, mosquitto will listen on all interfaces.
41. # listener port-number [ip address/host name]
42. listener 8000 127.0.0.1
43.
44. # Choose the protocol to use when listening.
45. # This can be either mqtt or websockets.
46. # Certificate based TLS may be used with websockets, except that only the
47. # cafile, certfile, keyfile and ciphers options are supported.
48. protocol websockets
49.
50. # When a listener is using the websockets protocol, it is possible to serve
51. # http data as well. Set http_dir to a directory which contains the files you
52. # wish to serve. If this option is not specified, then no normal http
53. # connections will be possible.
54. http_dir /var/www/html/diploma/
55.
56. # ================================================================
57. # Persistence
58. # ================================================================
59.
60. # If persistence is enabled, save the in-memory database to disk
61. # every autosave_interval seconds. If set to 0, the persistence
62. # database will only be written when mosquitto exits. See also
63. # autosave_on_changes.
64. # Note that writing of the persistence database can be forced by
65. # sending mosquitto a SIGUSR1 signal.
66. autosave_interval 1800
67.
68. # Save persistent message data to disk (true/false).
69. # This saves information about all messages, including
70. # subscriptions, currently in-flight messages and retained
71. # messages.
72. # retained_persistence is a synonym for this option.
73. persistence true
```

```
74.
75. # The filename to use for the persistent database, not including
76. # the path.
77. persistence_file mosquitto.db
78.
79. # Location for persistent database. Must include trailing /
80. # Default is an empty string (current directory).
81. # Set to e.g. /var/lib/mosquitto/ if running as a proper service on Linux or
82. # similar.
83. persistence_location /var/lib/mosquitto/
84.
85. # =================================================================
86. # Logging
87. # =================================================================
88.
89. # Places to log to. Use multiple log_dest lines for multiple
90. # logging destinations.
91. # The file will be closed and reopened when the broker receives a HUP signal.
92. # Only a single file destination may be configured.
93. log_dest file /var/log/mosquitto/mosquitto.log
94.
95. # If set to true, client connection and disconnection messages will be included
96. # in the log.
97. connection_messages true
98.
99. # If set to true, add a timestamp value to each log message.
100.log_timestamp true
```

index.html

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
    <title>Remote monitoring - Home page</title>
</head>
<body>

<h1>Remote monitoring of power consumption, luminosity, temperature, humidity</h1>
<h2>Graphical represantation of collected data</h2>
<ul>
        <li><a href="cons-live.html">Consumption: Live data</a></li>
        <li><a href="cons-lasthour.php">Consumption: Last hour</a></li>
        <li><a href="cons-lastday.php">Consumption: Last day</a></li>
        <li><a href="cons-lastweek.php">Consumption: Last week</a></li>
        <li><a href="cons-lastmonth.php">Consumption: Last month</a></li>
        <br/>
        <li><a href="lumi-live.html">Luminosity: Live data</a></li>
        <li><a href="lumi-lastday.php">Luminosity: Last day</a></li>
        <li><a href="lumi-lastweek.php">Luminosity: Last week</a></li>
        <br/>
        <li><a href="temp-live.html">Temperature: Live data</a></li>
        <li><a href="humi-live.html">Humidity: Live data</a></li>
        <li><a href="temp-humi-lasthour.php">Temperature-Humidity: Last hour</a></li>
        <li><a href="temp-humi-lastday.php">Temperature-Humidity: Last day</a></li>
        <li><a href="temp-humi-lastweek.php">Temperature-Humidity: Last week</a></li>
        <li><a href="temp-humi-lastmonth.php">Temperature-Humidity: Last month</a></li>

</ul>
</body>
</html>
```

cons-live.html

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Consumption - Live data</title>
<script                                                    type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/1.8.2/jquery.min.js"></script>
<script src="js/mqttws31_new.js" type="text/javascript"></script>
<script type="text/javascript">

        var MQTTbroker = '83.212.112.98';
        var MQTTport = 8000;
        var MQTTsubTopic1 = 'diploma/Ilioup/GW_1/node_2/dev_0/cons/';
        var chart; // global variable for chart
        var dataTopics = new Array();

//mqtt broker
        var client = new Messaging.Client(MQTTbroker, MQTTport,
                                "myclientid_" + parseInt(Math.random() * 100, 10));
        client.onMessageArrived = onMessageArrived;
        client.onConnectionLost = onConnectionLost;
        //connect to broker is at the bottom of the init() function !!!!


//mqtt connecton options including the mqtt broker subscriptions
        var options = {
                timeout: 3,
                onSuccess: function () {
                        console.log("mqtt connected");
                        // Connection succeeded; subscribe to our topics
                        client.subscribe(MQTTsubTopic1, {qos: 1});
                        //client.subscribe(MQTTsubTopic2, {qos: 1});
                        //client.subscribe(MQTTsubTopic3, {qos: 1});
                        },
                onFailure: function (message) {
                        console.log("Connection      failed,      ERROR:      "      +
message.errorMessage);
                        //window.setTimeout(location.reload(),20000);  //wait  20seconds
before trying to connect again.
                }
        };

//can be used to reconnect on connection lost
        function onConnectionLost(responseObject) {
                console.log("connection lost: " + responseObject.errorMessage);
                //window.setTimeout(location.reload(),20000);  //wait  20seconds  before
trying to connect again.
        };

//what is done when a message arrives from the broker
        function onMessageArrived(message) {
                console.log(message.destinationName, '',message.payloadString);

                //check if it is a new topic, if not add it to the array
                if (dataTopics.indexOf(message.destinationName) < 0){

                    dataTopics.push(message.destinationName); //add new topic to array
                    var  y  =  dataTopics.indexOf(message.destinationName);  //get  the
index no

                    //create new data series for the chart
                        var newseries = {
                             id: y,
                             name: message.destinationName,
                             data: []
                             };

                        chart.addSeries(newseries); //add the series

                    };

                var y = dataTopics.indexOf(message.destinationName); //get the index no
of the topic from the array
                var myEpoch = new Date().getTime(); //get current epoch time
                var thenum = message.payloadString.replace( /^\D+/g, ''); //remove any
text spaces from the message
```

```
                var plotMqtt = [myEpoch, Number(thenum)]; //create the array
                if (isNumber(thenum)) { //check if it is a real number and not text
                        console.log('is a propper number, will send to chart.')
                        plot(plotMqtt, y);     //send it to the plot function
                };
        };

//check if a real number
        function isNumber(n) {
          return !isNaN(parseFloat(n)) && isFinite(n);
        };

//function that is called once the document has loaded
        function init() {

                //i find i have to set this to false if i have trouble with timezones.
                Highcharts.setOptions({
                        global: {
                                useUTC: false
                        }
                });

                // Connect to MQTT broker
                client.connect(options);

        };


//this adds the plots to the chart
    function plot(point, chartno) {
        console.log(point);

                var series = chart.series[0],
                    shift = series.data.length > 20; // shift if the series is
                                                     // longer than 20
                // add the point
                chart.series[chartno].addPoint(point, true, shift);

        };

//settings for the chart
        $(document).ready(function() {
            chart = new Highcharts.Chart({
                chart: {
                    renderTo: 'container',
                    defaultSeriesType: 'spline'
                },
                title: {
                    text: 'Consumption - live data'
                },
                subtitle: {
                                text: 'myMQTTbroker: ' + MQTTbroker + ' | port: ' +
MQTTport
                        },
                xAxis: {
                    type: 'datetime',
                    tickPixelInterval: 150,
                    maxZoom: 20 * 1000
                },
                yAxis: {
                    minPadding: 0.2,
                    maxPadding: 0.2,
                    title: {
                        text: 'Watt',
                        margin: 80
                    }
                },
                series: []
            });
        });

</script>

<script src="http://code.highcharts.com/stock/highstock.js"></script>
<script src="http://code.highcharts.com/stock/modules/exporting.js"></script>

</head>
```

```
<body>
<body onload="init();"><!--Start the javascript ball rolling and connect to the mqtt
broker-->
        <h1>Remote monitoring of power consumption, luminosity, temperature,
humidity</h1>
        <h2>Consumption diagram</h2>
        <h3>Node #0, Gateway#1, Ilioupoli</h3>
        <h4>Live data</h4>
        <div id="container" style="height: 500px; min-width: 500px"></div>
</body>
</html>
```

cons-lasthour.php

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
    <title>Consumption - Last hour</title>
    <script src="canvasjs/canvasjs.min.js"></script>

        <?php
                $SERVER         = "localhost";
                $USER           = "diploma_user";
                $PASSWORD       = "1111";
                $DATABASE       = "diplomadb";

                $con = mysqli_connect($SERVER,$USER,$PASSWORD,$DATABASE);

                // Check connection
                if (mysqli_connect_errno($con))
                {
                        echo "Failed to connect to DataBase: " . mysqli_connect_error();
                }else
                {
                        $energy=0;
                        $cons_points = array();

                        $result1 = mysqli_query($con, "SELECT `datetime` , `cons` FROM
`measurements` WHERE `datetime` >= NOW( ) - INTERVAL 1 HOUR ");

                        while($row = mysqli_fetch_array($result1))
                        {
                                $point = array(label => $row[0] , y => $row[1]);
                                $energy = $energy+$row[1]*10;
                                array_push($cons_points, $point);
                                array_push($result1, $cons_points);
                        }

                        $energy = $energy/(3600*1000); //W*s -> kW*h
                        $energy = ROUND($energy, 3); //round to 3 decimals
                        }
                mysqli_close($con);

        ?>

      <script type="text/javascript">
         var cons_dps = <?php echo json_encode($cons_points, JSON_NUMERIC_CHECK); ?>;

         window.onload = (function() {


                        var chart_temp = new CanvasJS.Chart("chartContainer1",{
                                title :{
                                        text: "Consumption"
                                },
                                axisX: {
                                        title: "Date and Time",
                                        //intervalType: "hour",
                                        //interval: 300,
                                        //valueFormatString: "hh:mm",
                                        labelAngle: -50
                                },
                                axisY: {
                                        title: "Watt"
                                },
                                data: [{
                                        type: "line",
```

```
                                        xValueType: "dateTime",
                                        dataPoints : cons_dps
                                }]
                        });
                        chart_temp.render();

                });
    </script>
</head>
<body>
        <h1>Remote monitoring of power consumption, luminosity, temperature, humidity</h1>
        <h2>Consumption diagram</h2>
        <h3>Node #0, Gateway#1, Ilioupoli</h3>
        <h4>Last hour</h4>
        <div id="chartContainer1" style="width: 80%; height: 50%; margin-bottom: 25%;"></div>
        <p>Estimated energy consumed: <?php echo $energy; ?> kWh</p>
</body>
</html>
```

cons-lastday.php

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
    <title>Consumption - Last day</title>
    <script src="canvasjs/canvasjs.min.js"></script>

        <?php
                $SERVER         = "localhost";
                $USER           = "diploma_user";
                $PASSWORD       = "1111";
                $DATABASE       = "diplomadb";

                $con = mysqli_connect($SERVER,$USER,$PASSWORD,$DATABASE);

                // Check connection
                if (mysqli_connect_errno($con))
                {
                        echo "Failed to connect to DataBase: " . mysqli_connect_error();
                }else
                {
                        $energy=0;
                        $cons_points = array();

                        $result1 = mysqli_query($con, "SELECT `datetime` , `cons` FROM `measurements` WHERE `datetime` >= NOW( ) - INTERVAL 1 DAY ");

                        while($row = mysqli_fetch_array($result1))
                        {
                                $point = array(label => $row[0] , y => $row[1]);
                                $energy = $energy+$row[1]*10;
                                array_push($cons_points, $point);
                                array_push($result1, $cons_points);
                        }

                        $energy = $energy/(3600*1000); //W*s -> kW*h
                        $energy = ROUND($energy, 3); //round to 3 decimals
                                }
                mysqli_close($con);
        ?>

      <script type="text/javascript">
         var cons_dps = <?php echo json_encode($cons_points, JSON_NUMERIC_CHECK); ?>;

         window.onload = (function() {
                        var chart_temp = new CanvasJS.Chart("chartContainer1",{
                                title :{
                                        text: "Consumption"
                                },
                                axisX: {
                                        title: "Date and Time",
                                        intervalType: "hour",
                                        //interval: 300,
                                        valueFormatString: "hh:mm",
                                        labelAngle: -50
```

```
                                },
                                axisY: {
                                        title: "Watt"
                                },
                                data: [{
                                        type: "line",
                                        xValueType: "dateTime",
                                        dataPoints : cons_dps
                                }]
                        });
                        chart_temp.render();
                });
    </script>
</head>
<body>
        <h1>Remote monitoring of power consumption, luminosity, temperature, humidity</h1>
        <h2>Consumption diagram</h2>
        <h3>Node #0, Gateway#1, Ilioupoli</h3>
        <h4>Last δαυ</h4>
        <div id="chartContainer1" style="width: 80%; height: 50%; margin-bottom: 25%;"></div>
        <p>Estimated energy consumed: <?php echo $energy; ?> kWh</p>
</body>
</html>
```

cons-lastweek.php

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
    <title>Consumption - Last week</title>
    <script src="canvasjs/canvasjs.min.js"></script>

        <?php
                $SERVER         = "localhost";
                $USER           = "diploma_user";
                $PASSWORD       = "1111";
                $DATABASE       = "diplomadb";

                $con = mysqli_connect($SERVER,$USER,$PASSWORD,$DATABASE);

                // Check connection
                if (mysqli_connect_errno($con))
                {
                        echo "Failed to connect to DataBase: " . mysqli_connect_error();
                }else
                {
                        $energy=0;
                        $cons_points = array();

                        $result1 = mysqli_query($con, "SELECT `datetime` , `cons` FROM `measurements` WHERE `datetime` >= NOW( ) - INTERVAL 7 DAY ");

                        while($row = mysqli_fetch_array($result1))
                        {
                                $point = array(label => $row[0] , y => $row[1]);
                                $energy = $energy+$row[1]*10;
                                array_push($cons_points, $point);
                                array_push($result1, $cons_points);
                        }

                        $energy = $energy/(3600*1000); //W*s -> kW*h
                        $energy = ROUND($energy, 3); //round to 3 decimals
                }
                mysqli_close($con);
        ?>

      <script type="text/javascript">
         var cons_dps = <?php echo json_encode($cons_points, JSON_NUMERIC_CHECK); ?>;

         window.onload = (function() {
                        var chart_temp = new CanvasJS.Chart("chartContainer1",{
                                title :{
                                        text: "Consumption"
                                },
                                axisX: {
```

```
                                        title: "Date and Time",
                                        intervalType: "hour",
                                        //interval: 300,
                                        valueFormatString: "hh:mm",
                                        labelAngle: -50
                                },
                                axisY: {
                                        title: "Watt"
                                },
                                data: [{
                                        type: "line",
                                        xValueType: "dateTime",
                                        dataPoints : cons_dps
                                }]
                        });
                        chart_temp.render();
                });
    </script>
</head>
<body>
        <h1>Remote monitoring of power consumption, luminosity, temperature, humidity</h1>
        <h2>Consumption diagram</h2>
        <h3>Node #0, Gateway#1, Ilioupoli</h3>
        <h4>Last week</h4>
        <div id="chartContainer1" style="width: 80%; height: 50%; margin-bottom: 25%;"></div>
        <p>Estimated energy consumed: <?php echo $energy; ?> kWh</p>
</body>
</html>
```

cons-lastmonth.php

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
    <title>Consumption - Last month</title>
    <script src="canvasjs/canvasjs.min.js"></script>
        <?php
                $SERVER         = "localhost";
                $USER           = "diploma_user";
                $PASSWORD       = "1111";
                $DATABASE       = "diplomadb";

                $con = mysqli_connect($SERVER,$USER,$PASSWORD,$DATABASE);

                // Check connection
                if (mysqli_connect_errno($con))
                {
                        echo "Failed to connect to DataBase: " . mysqli_connect_error();
                }else
                {
                        $energy=0;
                        $cons_points = array();

                        $result1 = mysqli_query($con, "SELECT `datetime` , `cons` FROM `measurements` WHERE `datetime` >= NOW( ) - INTERVAL 1 MONTH ");

                        while($row = mysqli_fetch_array($result1))
                        {
                                $point = array(label => $row[0] , y => $row[1]);
                                $energy = $energy+$row[1]*10;
                                array_push($cons_points, $point);
                                array_push($result1, $cons_points);
                        }

                        $energy = $energy/(3600*1000); //W*s -> kW*h
                        $energy = ROUND($energy, 3); //round to 3 decimals
                }
                mysqli_close($con);
        ?>

      <script type="text/javascript">
         var cons_dps = <?php echo json_encode($cons_points, JSON_NUMERIC_CHECK); ?>;

         window.onload = (function() {
                        var chart_temp = new CanvasJS.Chart("chartContainer1",{
```

```
                                title :{
                                        text: "Consumption"
                                },
                                axisX: {
                                        title: "Date and Time",
                                        intervalType: "hour",
                                        //interval: 300,
                                        valueFormatString: "hh:mm",
                                        labelAngle: -50
                                },
                                axisY: {
                                        title: "Watt"
                                },
                                data: [{
                                        type: "line",
                                        xValueType: "dateTime",
                                        dataPoints : cons_dps
                                }]
                        });
                        chart_temp.render();
                });
    </script>
</head>
<body>
        <h1>Remote monitoring of power consumption, luminosity, temperature, humidity</h1>
        <h2>Consumption diagram</h2>
        <h3>Node #0, Gateway#1, Ilioupoli</h3>
        <h4>Last month</h4>
        <div id="chartContainer1" style="width: 80%; height: 50%; margin-bottom: 25%;"></div>
        <p>Estimated energy consumed: <?php echo $energy; ?> kWh</p>
</body>
</html>
```

lumi-live.html

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Luminosity - Live data</title>
<script type="text/javascript" src="http://ajax.googleapis.com/ajax/libs/jquery/1.8.2/jquery.min.js"></script>
<script src="js/mqttws31_new.js" type="text/javascript"></script>
<script type="text/javascript">

        var MQTTbroker = '83.212.112.98';
        var MQTTport = 8000;
        var MQTTsubTopic1 = 'diploma/Ilioupoli/GW_1/node_1/dev_1/lumi/';
        var MQTTsubTopic2 = 'diploma/Ilioup/GW_1/node_2/dev_0/lumi/';
        var chart; // global variable for chart
        var dataTopics = new Array();

//mqtt broker
        var client = new Messaging.Client(MQTTbroker, MQTTport,
                                "myclientid_" + parseInt(Math.random() * 100, 10));
        client.onMessageArrived = onMessageArrived;
        client.onConnectionLost = onConnectionLost;
        //connect to broker is at the bottom of the init() function !!!!


//mqtt connecton options including the mqtt broker subscriptions
        var options = {
                timeout: 3,
                onSuccess: function () {
                        console.log("mqtt connected");
                        // Connection succeeded; subscribe to our topics
                        client.subscribe(MQTTsubTopic1, {qos: 1});
                        client.subscribe(MQTTsubTopic2, {qos: 1});
                        },
                onFailure: function (message) {
                        console.log("Connection failed, ERROR: " + message.errorMessage);
                        //window.setTimeout(location.reload(),20000);  //wait  20seconds before trying to connect again.
                }
        };
```

```
//can be used to reconnect on connection lost
	function onConnectionLost(responseObject) {
		console.log("connection lost: " + responseObject.errorMessage);
		//window.setTimeout(location.reload(),20000);  //wait  20seconds  before
trying to connect again.
	};

//what is done when a message arrives from the broker
	function onMessageArrived(message) {
		console.log(message.destinationName, '',message.payloadString);

		//check if it is a new topic, if not add it to the array
		if (dataTopics.indexOf(message.destinationName) < 0){

		    dataTopics.push(message.destinationName); //add new topic to array
		    var  y  =  dataTopics.indexOf(message.destinationName);  //get  the
index no

		    //create new data series for the chart
		        var newseries = {
		            id: y,
		            name: message.destinationName,
		            data: []
		            };

		        chart.addSeries(newseries); //add the series

		    };

		var y = dataTopics.indexOf(message.destinationName); //get the index no
of the topic from the array
		var myEpoch = new Date().getTime(); //get current epoch time
		var thenum = message.payloadString.replace( /^\D+/g, ''); //remove any
text spaces from the message
		var plotMqtt = [myEpoch, Number(thenum)]; //create the array
		if (isNumber(thenum)) { //check if it is a real number and not text
			console.log('is a propper number, will send to chart.')
			plot(plotMqtt, y);     //send it to the plot function
		};
	};

//check if a real number
	function isNumber(n) {
	  return !isNaN(parseFloat(n)) && isFinite(n);
	};

//function that is called once the document has loaded
	function init() {

		//i find i have to set this to false if i have trouble with timezones.
		Highcharts.setOptions({
			global: {
				useUTC: false
			}
		});

		// Connect to MQTT broker
		client.connect(options);

	};


//this adds the plots to the chart
    function plot(point, chartno) {
	console.log(point);

		var series = chart.series[0],
		    shift = series.data.length > 20; // shift if the series is
		                                     // longer than 20
		// add the point
		chart.series[chartno].addPoint(point, true, shift);

	};

//settings for the chart
	$(document).ready(function() {
```

```
            chart = new Highcharts.Chart({
                chart: {
                    renderTo: 'container',
                    defaultSeriesType: 'spline'
                },
                title: {
                    text: 'Luminosity - live data'
                },
                subtitle: {
                                text: 'myMQTTbroker: ' + MQTTbroker + ' | port: ' +
MQTTport
                        },
                xAxis: {
                    type: 'datetime',
                    tickPixelInterval: 150,
                    maxZoom: 20 * 1000
                },
                yAxis: {
                    minPadding: 0.2,
                    maxPadding: 0.2,
                    title: {
                        text: '',
                        margin: 80
                    }
                },
                series: []
            });
        });

</script>

<script src="http://code.highcharts.com/stock/highstock.js"></script>
<script src="http://code.highcharts.com/stock/modules/exporting.js"></script>

</head>
<body>
<body onload="init();"><!--Start the javascript ball rolling and connect to the mqtt
broker-->
        <h1>Remote monitoring of power consumption, luminosity, temperature,
humidity</h1>
        <h2>Luminosity diagram</h2>
        <h3>Node #1 and #2, Gateway #1, Ilioupoli</h3>
        <h4>Live data</h4>
        <div id="container" style="height: 500px; min-width: 500px"></div>
</body>
</html>
```

lumi-lastday.php

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
    <title>Luminosity - Last day</title>
    <script src="canvasjs/canvasjs.min.js"></script>

        <?php
                $SERVER         = "localhost";
                $USER           = "diploma_user";
                $PASSWORD       = "1111";
                $DATABASE       = "diplomadb";

                $con = mysqli_connect($SERVER,$USER,$PASSWORD,$DATABASE);

                // Check connection
                if (mysqli_connect_errno($con))
                {
                        echo "Failed to connect to DataBase: " . mysqli_connect_error();
                }else
                {
                        $lumi_points = array();

                        $result1 = mysqli_query($con, "SELECT `datetime` , `lumi` FROM
`measurements` WHERE `datetime` >= NOW( ) - INTERVAL 1 DAY AND `lumi`<1023 AND
`lumi`>0");

                        while($row = mysqli_fetch_array($result1))
                        {
```

```
                                $point = array(label => $row[0] , y => $row[1]);
                                $energy = $energy+$row[1]*10;
                                array_push($lumi_points, $point);
                                array_push($result1, $lumi_points);
                        }
                }
                mysqli_close($con);
        ?>

    <script type="text/javascript">
                var     lumi_dps     =     <?php     echo     json_encode($lumi_points,
JSON_NUMERIC_CHECK); ?>;

                window.onload = (function() {
                        var chart_lumi = new CanvasJS.Chart("chartContainer1",{
                                title :{
                                        text: "Luminosity"
                                },
                                axisX: {
                                        title: "Date and Time",
                                        intervalType: "hour",
                                        //interval: 300,
                                        valueFormatString: "hh:mm",
                                        labelAngle: -50
                                },
                                axisY: {
                                        title: ""
                                },
                                data: [{
                                        type: "line",
                                        xValueType: "dateTime",
                                        dataPoints : lumi_dps
                                }]
                        });
                        chart_lumi.render();

                });
    </script>
</head>
<body>
        <h1>Remote   monitoring   of   power   consumption,   luminosity,   temperature,
humidity</h1>
        <h2>Luminosity diagram</h2>
        <h3>Node #2, Gateway #1, Ilioupoli</h3>
        <h4>Last day</h4>
        <div   id="chartContainer1"   style="width:   80%;   height:   50%;   margin-bottom:
25%;"></div>
</body>
</html>
```

lumi-lastweek.php

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
    <title>Luminosity - Last week</title>
    <script src="canvasjs/canvasjs.min.js"></script>

        <?php
                $SERVER         = "localhost";
                $USER           = "diploma_user";
                $PASSWORD       = "1111";
                $DATABASE       = "diplomadb";

                $con = mysqli_connect($SERVER,$USER,$PASSWORD,$DATABASE);

                // Check connection
                if (mysqli_connect_errno($con))
                {
                        echo "Failed to connect to DataBase: " . mysqli_connect_error();
                }else
                {
                        $lumi_points = array();

                        $result1 = mysqli_query($con, "SELECT `datetime` , `lumi` FROM
`measurements` WHERE `datetime` >= NOW( ) - INTERVAL 7 DAY AND `lumi`<1023 AND
`lumi`>0");
```

```
                        while($row = mysqli_fetch_array($result1))
                        {
                                $point = array(label => $row[0] , y => $row[1]);
                                $energy = $energy+$row[1]*10;
                                array_push($lumi_points, $point);
                                array_push($result1, $lumi_points);
                        }
                }
                mysqli_close($con);
        ?>

    <script type="text/javascript">
                var lumi_dps = <?php echo json_encode($lumi_points,
JSON_NUMERIC_CHECK); ?>;

                window.onload = (function() {
                        var chart_lumi = new CanvasJS.Chart("chartContainer1",{
                                title :{
                                        text: "Luminosity"
                                },
                                axisX: {
                                        title: "Date and Time",
                                        intervalType: "hour",
                                        //interval: 300,
                                        valueFormatString: "hh:mm",
                                        labelAngle: -50
                                },
                                axisY: {
                                        title: ""
                                },
                                data: [{
                                        type: "line",
                                        xValueType: "dateTime",
                                        dataPoints : lumi_dps
                                }]
                        });
                        chart_lumi.render();
                });
    </script>
</head>
<body>
        <h1>Remote monitoring of power consumption, luminosity, temperature,
humidity</h1>
        <h2>Luminosity diagram</h2>
        <h3>Node #2, Gateway #1, Ilioupoli</h3>
        <h4>Last week</h4>
        <div id="chartContainer1" style="width: 80%; height: 50%; margin-bottom:
25%;"></div>
</body>
</html>
```

temp-live.html

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Temperature - Live data</title>
<script                                                    type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/1.8.2/jquery.min.js"></script>
<script src="js/mqttws31_new.js" type="text/javascript"></script>
<script type="text/javascript">


//settings BEGIN
        var MQTTbroker = '83.212.112.98';
        var MQTTport = 8000;
        var MQTTsubTopic1 = 'diploma/Ilioupoli/GW_1/node_1/dev_1/temp/';

        //settings END

        var chart; // global variable for chart
        var dataTopics = new Array();

//mqtt broker
        var client = new Messaging.Client(MQTTbroker, MQTTport,
                                "myclientid_" + parseInt(Math.random() * 100, 10));
```

```
        client.onMessageArrived = onMessageArrived;
        client.onConnectionLost = onConnectionLost;
        //connect to broker is at the bottom of the init() function !!!!


//mqtt connecton options including the mqtt broker subscriptions
        var options = {
                timeout: 3,
                onSuccess: function () {
                        console.log("mqtt connected");
                        // Connection succeeded; subscribe to our topics
                        client.subscribe(MQTTsubTopic1, {qos: 1});
                        },
                onFailure: function (message) {
                        console.log("Connection failed, ERROR: " +
message.errorMessage);
                        //window.setTimeout(location.reload(),20000);  //wait  20seconds
before trying to connect again.
                }
        };

//can be used to reconnect on connection lost
        function onConnectionLost(responseObject) {
                console.log("connection lost: " + responseObject.errorMessage);
                //window.setTimeout(location.reload(),20000);  //wait  20seconds  before
trying to connect again.
        };

//what is done when a message arrives from the broker
        function onMessageArrived(message) {
                console.log(message.destinationName, '',message.payloadString);

                //check if it is a new topic, if not add it to the array
                if (dataTopics.indexOf(message.destinationName) < 0){

                    dataTopics.push(message.destinationName); //add new topic to array
                    var  y  =  dataTopics.indexOf(message.destinationName);  //get  the
index no

                    //create new data series for the chart
                        var newseries = {
                             id: y,
                             name: message.destinationName,
                             data: []
                             };

                        chart.addSeries(newseries); //add the series

                    };

                var y = dataTopics.indexOf(message.destinationName); //get the index no
of the topic from the array
                var myEpoch = new Date().getTime(); //get current epoch time
                var  thenum = message.payloadString.replace( /^\D+/g, ''); //remove any
text spaces from the message
                var plotMqtt = [myEpoch, Number(thenum)]; //create the array
                if (isNumber(thenum)) { //check if it is a real number and not text
                        console.log('is a propper number, will send to chart.')
                        plot(plotMqtt, y);     //send it to the plot function
                };
        };

//check if a real number
        function isNumber(n) {
          return !isNaN(parseFloat(n)) && isFinite(n);
        };

//function that is called once the document has loaded
        function init() {

                //i find i have to set this to false if i have trouble with timezones.
                Highcharts.setOptions({
                        global: {
                                useUTC: false
                        }
                });
```

```
                // Connect to MQTT broker
                client.connect(options);

        };


//this adds the plots to the chart
    function plot(point, chartno) {
        console.log(point);

                var series = chart.series[0],
                    shift = series.data.length > 20; // shift if the series is
                                                     // longer than 20
                // add the point
                chart.series[chartno].addPoint(point, true, shift);

        };

//settings for the chart
        $(document).ready(function() {
            chart = new Highcharts.Chart({
                chart: {
                    renderTo: 'container',
                    defaultSeriesType: 'spline'
                },
                title: {
                    text: 'Temperature - live data'
                },
                subtitle: {
                                text: 'myMQTTbroker: ' + MQTTbroker + ' | port: ' +
MQTTport
                        },
                xAxis: {
                    type: 'datetime',
                    tickPixelInterval: 150,
                    maxZoom: 20 * 1000
                },
                yAxis: {
                    minPadding: 0.2,
                    maxPadding: 0.2,
                    title: {
                        text: 'Watt',
                        margin: 80
                    }
                },
                series: []
            });
        });

</script>

<script src="http://code.highcharts.com/stock/highstock.js"></script>
<script src="http://code.highcharts.com/stock/modules/exporting.js"></script>

</head>
<body>
<body onload="init();"><!--Start the javascript ball rolling and connect to the mqtt
broker-->
        <h1>Remote monitoring of power consumption, luminosity, temperature,
humidity</h1>
        <h2>Temperature diagram</h2>
        <h3>Node #1, Gateway#1, Ilioupoli</h3>
        <h4>Live data</h4>
        <div id="container" style="height: 500px; min-width: 500px"></div>
</body>
</html>
```

humi-live.html

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Humidity - Live data</title>
<script                                                    type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/1.8.2/jquery.min.js"></script>
<script src="js/mqttws31_new.js" type="text/javascript"></script>
<script type="text/javascript">
```

```
//settings BEGIN
	var MQTTbroker = '83.212.112.98';
	var MQTTport = 8000;
	var MQTTsubTopic1 = 'diploma/Ilioupoli/GW_1/node_1/dev_1/humi/';

	//settings END

	var chart; // global variable for chart
	var dataTopics = new Array();

//mqtt broker
	var client = new Messaging.Client(MQTTbroker, MQTTport,
				"myclientid_" + parseInt(Math.random() * 100, 10));
	client.onMessageArrived = onMessageArrived;
	client.onConnectionLost = onConnectionLost;
	//connect to broker is at the bottom of the init() function !!!!


//mqtt connecton options including the mqtt broker subscriptions
	var options = {
		timeout: 3,
		onSuccess: function () {
			console.log("mqtt connected");
			// Connection succeeded; subscribe to our topics
			client.subscribe(MQTTsubTopic1, {qos: 1});
			},
		onFailure: function (message) {
			console.log("Connection failed, ERROR: " +
message.errorMessage);
			//window.setTimeout(location.reload(),20000);  //wait  20seconds
before trying to connect again.
		}
	};

//can be used to reconnect on connection lost
	function onConnectionLost(responseObject) {
		console.log("connection lost: " + responseObject.errorMessage);
		//window.setTimeout(location.reload(),20000);  //wait  20seconds  before
trying to connect again.
	};

//what is done when a message arrives from the broker
	function onMessageArrived(message) {
		console.log(message.destinationName, '',message.payloadString);

		//check if it is a new topic, if not add it to the array
		if (dataTopics.indexOf(message.destinationName) < 0){

		    dataTopics.push(message.destinationName); //add new topic to array
		    var  y  =  dataTopics.indexOf(message.destinationName);  //get  the
index no

		    //create new data series for the chart
			var newseries = {
			    id: y,
			    name: message.destinationName,
			    data: []
			    };

			chart.addSeries(newseries); //add the series

		    };

		var y = dataTopics.indexOf(message.destinationName); //get the index no
of the topic from the array
		var myEpoch = new Date().getTime(); //get current epoch time
		var thenum = message.payloadString.replace( /^\D+/g, ''); //remove any
text spaces from the message
		var plotMqtt = [myEpoch, Number(thenum)]; //create the array
		if (isNumber(thenum)) { //check if it is a real number and not text
			console.log('is a propper number, will send to chart.')
			plot(plotMqtt, y);     //send it to the plot function
		};
	};
```

```
//check if a real number
        function isNumber(n) {
          return !isNaN(parseFloat(n)) && isFinite(n);
        };

//function that is called once the document has loaded
        function init() {

                //i find i have to set this to false if i have trouble with timezones.
                Highcharts.setOptions({
                        global: {
                                useUTC: false
                        }
                });

                // Connect to MQTT broker
                client.connect(options);

        };


//this adds the plots to the chart
    function plot(point, chartno) {
        console.log(point);

                var series = chart.series[0],
                    shift = series.data.length > 20; // shift if the series is
                                                     // longer than 20
                // add the point
                chart.series[chartno].addPoint(point, true, shift);

        };

//settings for the chart
        $(document).ready(function() {
            chart = new Highcharts.Chart({
                chart: {
                    renderTo: 'container',
                    defaultSeriesType: 'spline'
                },
                title: {
                    text: 'Humidity - live data'
                },
                subtitle: {
                                text: 'myMQTTbroker: ' + MQTTbroker + ' | port: ' +
MQTTport
                        },
                xAxis: {
                    type: 'datetime',
                    tickPixelInterval: 150,
                    maxZoom: 20 * 1000
                },
                yAxis: {
                    minPadding: 0.2,
                    maxPadding: 0.2,
                    title: {
                        text: 'Watt',
                        margin: 80
                    }
                },
                series: []
            });
        });

</script>

<script src="http://code.highcharts.com/stock/highstock.js"></script>
<script src="http://code.highcharts.com/stock/modules/exporting.js"></script>

</head>
<body>
<body onload="init();"><!--Start the javascript ball rolling and connect to the mqtt
broker-->
        <h1>Remote monitoring of power consumption, luminosity, temperature,
humidity</h1>
        <h2>Humidity diagram</h2>
        <h3>Node #1, Gateway#1, Ilioupoli</h3>
```

```
        <h4>Live data</h4>
        <div id="container" style="height: 500px; min-width: 500px"></div>
</body>
</html>
```

temp-humi-lasthour.php

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
    <title>Temperature and Humidity - Last hour</title>
    <script src="canvasjs/canvasjs.min.js"></script>

        <?php
                $SERVER         = "localhost";
                $USER           = "diploma_user";
                $PASSWORD       = "1111";
                $DATABASE       = "diplomadb";

                $con = mysqli_connect($SERVER,$USER,$PASSWORD,$DATABASE);

                // Check connection
                if (mysqli_connect_errno($con))
                {
                        echo "Failed to connect to DataBase: " . mysqli_connect_error();
                }else
                {
                        $temp_points = array();
                        $humi_points = array();

                        $result1 = mysqli_query($con, "SELECT `datetime` , `temp` FROM
`measurements` WHERE `datetime` >= NOW( ) - INTERVAL 1 HOUR ");
                        $result2 = mysqli_query($con, "SELECT `datetime` , `humi` FROM
`measurements`          WHERE `datetime` >= NOW( ) - INTERVAL 1 HOUR ");

                        while($row = mysqli_fetch_array($result1))
                        {
                                $point = array(label => $row[0] , y => $row[1]);

                                array_push($temp_points, $point);
                                array_push($result1, $temp_points);
                        }

                        while($row = mysqli_fetch_array($result2))
                        {
                                $point = array(label => $row[0] , y => $row[1]);

                                array_push($humi_points, $point);
                                array_push($result2, $humi_points);
                        }

                }
                mysqli_close($con);

?>

      <script type="text/javascript">
         var temp_dps = <?php echo json_encode($temp_points, JSON_NUMERIC_CHECK); ?>;
         var humi_dps = <?php echo json_encode($humi_points, JSON_NUMERIC_CHECK); ?>;
         window.onload = (function() {


                        var chart_temp = new CanvasJS.Chart("chartContainer1",{
                                title :{
                                        text: "Temperature"
                                },
                                axisX: {
                                        title: "Date and Time",
                                        intervalType: "hour",
                                        //interval: 300,
                                        valueFormatString: "hh:mm",
                                        labelAngle: -50
                                },
                                axisY: {
                                        title: "oC"
                                },
                                data: [{
```

```
                                type: "line",
                                xValueType: "dateTime",
                                dataPoints : temp_dps
                        }]
                });
                chart_temp.render();

                var chart_humi = new CanvasJS.Chart("chartContainer2",{
                        title :{
                                text: "Humidity"
                        },
                        axisX: {
                                title: "Date and Time",
                                intervalType: "hour",
                                //interval: 300,
                                valueFormatString: "hh:mm",
                                labelAngle: -50
                        },
                        axisY: {
                                title: "%"
                        },
                        data: [{
                                type: "line",
                                xValueType: "dateTime",
                                dataPoints : humi_dps
                        }]
                });
                chart_humi.render();
        });


    </script>
</head>
<body>
        <h1>Remote monitoring of power consumption, luminosity, temperature, humidity</h1>
        <h2>Temperature and humidity diagrams</h2>
        <h3>Node #1, Gateway#1, Ilioupoli</h3>
        <h4>Last hour</h4>
        <div id="chartContainer1" style="width: 80%; height: 50%; margin-bottom: 25%;"></div>
        <div id="chartContainer2" style="width: 80%; height: 50%; margin-top: 25%;"></div>
</body>
</html>
```

temp-humi-lastday.php

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
    <title>Temperature and Humidity - Last day</title>
    <script src="canvasjs/canvasjs.min.js"></script>

        <?php
                $SERVER         = "localhost";
                $USER           = "diploma_user";
                $PASSWORD       = "1111";
                $DATABASE       = "diplomadb";

                $con = mysqli_connect($SERVER,$USER,$PASSWORD,$DATABASE);

                // Check connection
                if (mysqli_connect_errno($con))
                {
                        echo "Failed to connect to DataBase: " . mysqli_connect_error();
                }else
                {
                        $temp_points = array();
                        $humi_points = array();

                        $result1 = mysqli_query($con, "SELECT `datetime` , `temp` FROM `measurements` WHERE `datetime` >= NOW( ) - INTERVAL 1 DAY ");
                        $result2 = mysqli_query($con, "SELECT `datetime` , `humi` FROM `measurements`         WHERE `datetime` >= NOW( ) - INTERVAL 1 DAY ");

                        while($row = mysqli_fetch_array($result1))
```

```
                        {
                                $point = array(label => $row[0] , y => $row[1]);

                                array_push($temp_points, $point);
                                array_push($result1, $temp_points);
                        }

                        while($row = mysqli_fetch_array($result2))
                        {
                                $point = array(label => $row[0] , y => $row[1]);

                                array_push($humi_points, $point);
                                array_push($result2, $humi_points);
                        }

                }
                mysqli_close($con);

        ?>

        <script type="text/javascript">
                var temp_dps = <?php echo json_encode($temp_points, JSON_NUMERIC_CHECK); ?>;
                var humi_dps = <?php echo json_encode($humi_points, JSON_NUMERIC_CHECK); ?>;
                window.onload = (function() {


                        var chart_temp = new CanvasJS.Chart("chartContainer1",{
                                title :{
                                        text: "Temperature"
                                },
                                axisX: {
                                        title: "Date and Time",
                                        intervalType: "hour",
                                        //interval: 300,
                                        valueFormatString: "hh:mm",
                                        labelAngle: -50
                                },
                                axisY: {
                                        title: "oC"
                                },
                                data: [{
                                        type: "line",
                                        xValueType: "dateTime",
                                        dataPoints : temp_dps
                                }]
                        });
                        chart_temp.render();

                        var chart_humi = new CanvasJS.Chart("chartContainer2",{
                                title :{
                                        text: "Humidity"
                                },
                                axisX: {
                                        title: "Date and Time",
                                        intervalType: "hour",
                                        //interval: 300,
                                        valueFormatString: "hh:mm",
                                        labelAngle: -50
                                },
                                axisY: {
                                        title: "%"
                                },
                                data: [{
                                        type: "line",
                                        xValueType: "dateTime",
                                        dataPoints : humi_dps
                                }]
                        });
                        chart_humi.render();
                });

    </script>
</head>
<body>
        <h1>Remote monitoring of power consumption, luminosity, temperature, humidity</h1>
```

```
        <h2>Temperature and humidity diagrams</h2>
        <h3>Node #1, Gateway#1, Ilioupoli</h3>
        <h4>Last day</h4>
        <div  id="chartContainer1"  style="width:  80%;  height:  50%;  margin-bottom:
25%;"></div>
        <div   id="chartContainer2"   style="width:   80%;   height:   50%;   margin-top:
25%;"></div>
</body>
</html>
```

temp-humi-lastweek.php

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
    <title>Temperature and Humidity - Last week</title>
    <script src="canvasjs/canvasjs.min.js"></script>

        <?php
                $SERVER         = "localhost";
                $USER           = "diploma_user";
                $PASSWORD       = "1111";
                $DATABASE       = "diplomadb";

                $con = mysqli_connect($SERVER,$USER,$PASSWORD,$DATABASE);
                //$con = mysql_connect("localhost","diploma_user","1111","diplomadb");

                // Check connection
                if (mysqli_connect_errno($con))
                {
                        echo "Failed to connect to DataBase: " . mysqli_connect_error();
                }else
                {
                        $temp_points = array();
                        $humi_points = array();

                        $result1 = mysqli_query($con, "SELECT `datetime` , `temp` FROM
`measurements` WHERE `datetime` >= NOW( ) - INTERVAL 7 DAY ");
                        $result2 = mysqli_query($con, "SELECT `datetime` , `humi` FROM
`measurements`          WHERE `datetime` >= NOW( ) - INTERVAL 7 DAY ");

                        while($row = mysqli_fetch_array($result1))
                        {
                                $point = array(label => $row[0] , y => $row[1]);

                                array_push($temp_points, $point);
                                array_push($result1, $temp_points);
                        }

                        while($row = mysqli_fetch_array($result2))
                        {
                                $point = array(label => $row[0] , y => $row[1]);

                                array_push($humi_points, $point);
                                array_push($result2, $humi_points);
                        }

                        //echo json_encode($temp_points, JSON_NUMERIC_CHECK);
                        //echo json_encode($humi_points, JSON_NUMERIC_CHECK);
                }
                mysqli_close($con);

        ?>

      <script type="text/javascript">
          var temp_dps = <?php echo json_encode($temp_points, JSON_NUMERIC_CHECK); ?>;
          var humi_dps = <?php echo json_encode($humi_points, JSON_NUMERIC_CHECK); ?>;
          window.onload = (function() {


                        var chart_temp = new CanvasJS.Chart("chartContainer1",{
                                title :{
                                        text: "Temperature"
                                },
                                axisX: {
                                        title: "Date and Time",
                                        intervalType: "hour",
```

```
                                        //interval: 300,
                                        valueFormatString: "hh:mm",
                                        labelAngle: -50
                                },
                                axisY: {
                                        title: "oC"
                                },
                                data: [{
                                        type: "line",
                                        xValueType: "dateTime",
                                        dataPoints : temp_dps
                                }]
                        });
                        chart_temp.render();

                        var chart_humi = new CanvasJS.Chart("chartContainer2",{
                                title :{
                                        text: "Humidity"
                                },
                                axisX: {
                                        title: "Date and Time",
                                        intervalType: "hour",
                                        //interval: 300,
                                        valueFormatString: "hh:mm",
                                        labelAngle: -50
                                },
                                axisY: {
                                        title: "%"
                                },
                                data: [{
                                        type: "line",
                                        xValueType: "dateTime",
                                        dataPoints : humi_dps
                                }]
                        });
                        chart_humi.render();
                });


    </script>
</head>
<body>
        <h1>Remote monitoring of power consumption, luminosity, temperature, humidity</h1>
        <h2>Temperature and humidity diagrams</h1>
        <h3>Node #1, Gateway#1, Ilioupoli</h3>
        <h4>Last week</h4>
        <div id="chartContainer1" style="width: 80%; height: 50%; margin-bottom: 25%;"></div>
        <div id="chartContainer2" style="width: 80%; height: 50%; margin-top: 25%;"></div>
</body>
</html>
```

temp-humi-lastmonth.php

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
    <title>Temperature and Humidity - Last month</title>
    <script src="canvasjs/canvasjs.min.js"></script>

        <?php
                $SERVER         = "localhost";
                $USER           = "diploma_user";
                $PASSWORD       = "1111";
                $DATABASE       = "diplomadb";

                $con = mysqli_connect($SERVER,$USER,$PASSWORD,$DATABASE);
                //$con = mysql_connect("localhost","diploma_user","1111","diplomadb");

                // Check connection
                if (mysqli_connect_errno($con))
                {
                        echo "Failed to connect to DataBase: " . mysqli_connect_error();
                }else
                {
```

```
                        $temp_points = array();
                        $humi_points = array();

                        $result1 = mysqli_query($con, "SELECT `datetime` , `temp` FROM
`measurements` WHERE `datetime` >= NOW( ) - INTERVAL 1 MONTH ");
                        $result2 = mysqli_query($con, "SELECT `datetime` , `humi` FROM
`measurements`          WHERE `datetime` >= NOW( ) - INTERVAL 1 MONTH ");

                        while($row = mysqli_fetch_array($result1))
                        {
                                $point = array(label => $row[0] , y => $row[1]);

                                array_push($temp_points, $point);
                                array_push($result1, $temp_points);
                        }

                        while($row = mysqli_fetch_array($result2))
                        {
                                $point = array(label => $row[0] , y => $row[1]);

                                array_push($humi_points, $point);
                                array_push($result2, $humi_points);
                        }

                }
                mysqli_close($con);

        ?>

        <script type="text/javascript">
                var temp_dps = <?php echo json_encode($temp_points, JSON_NUMERIC_CHECK); ?>;
                var humi_dps = <?php echo json_encode($humi_points, JSON_NUMERIC_CHECK); ?>;
                window.onload = (function() {


                        var chart_temp = new CanvasJS.Chart("chartContainer1",{
                                title :{
                                        text: "Temperature"
                                },
                                axisX: {
                                        title: "Date and Time",
                                        intervalType: "hour",
                                        //interval: 300,
                                        valueFormatString: "hh:mm",
                                        labelAngle: -50
                                },
                                axisY: {
                                        title: "oC"
                                },
                                data: [{
                                        type: "line",
                                        xValueType: "dateTime",
                                        dataPoints : temp_dps
                                }]
                        });
                        chart_temp.render();

                        var chart_humi = new CanvasJS.Chart("chartContainer2",{
                                title :{
                                        text: "Humidity"
                                },
                                axisX: {
                                        title: "Date and Time",
                                        intervalType: "hour",
                                        //interval: 300,
                                        valueFormatString: "hh:mm",
                                        labelAngle: -50
                                },
                                axisY: {
                                        title: "%"
                                },
                                data: [{
                                        type: "line",
                                        xValueType: "dateTime",
                                        dataPoints : humi_dps
                                }]
                        });
```

```
                chart_humi.render();
        });


    </script>
</head>
<body>
        <h1>Remote monitoring of power consumption, luminosity, temperature, humidity</h1>
        <h2>Temperature and humidity diagrams</h2>
        <h3>Node #1, Gateway#1, Ilioupoli</h3>
        <h4>Last month</h4>
        <div id="chartContainer1" style="width: 80%; height: 50%; margin-bottom: 25%;"></div>
        <div id="chartContainer2" style="width: 80%; height: 50%; margin-top: 25%;"></div>
</body>
</html>
```

map.html

```
<!DOCTYPE html>
<html>
        <head>
                <TITLE>Map view - Demo</title>

                <meta name=viewport content="user-scalable=no,width=device-width" />
                <meta name="viewport" content="width=device-width, initial-scale=1.0, user-scalable=no">
                <meta charset="utf-8">
                <style>
                      html, body, #map {
                        margin: 0;
                        padding: 0;
                        height: 99%;
                      }
                      #legend {
                        font-family: Arial, sans-serif;
                        background: #fff;
                        padding: 5px;
                        margin: 5px;
                        border: 1px solid #000;
                      }
                      #legend h3 {
                        margin-top: 0;
                      }
                      #legend img {
                        vertical-align: middle;
                      }
                </style>


                <script src="http://maps.googleapis.com/maps/api/js?sensor=false"></script>
                <script src="http://ajax.googleapis.com/ajax/libs/jquery/1.6.4/jquery.min.js" type="text/javascript"></script>
                <script src="mqttws31.js" type="text/javascript"></script>

                <script>
                // To be used with websockets, mqtt and http://owntracks.org/ project for ios/android.
                //   You   will   also   need   to   setup   a   google   maps   api key.https://code.google.com/apis/console
                // and need the mqtt javascript library
                // http://git.eclipse.org/c/paho/org.eclipse.paho.mqtt.javascript.git/plain/src/mqttws31.js
                // by Matthew Bordignon @bordignon September 2013 updated June 2014

                // Don't forget to add the googlemaps api key above as well.

                        var MQTTbroker = '83.212.112.98';
                        var MQTTport = 8000;
                        var MQTTsubTopic ='diploma/Ilioup/GW_1/#';

                        var personArray = new Array();
```

```
                        var       pinColourArray       =       new       Array('blue',
'blue','blue','blue','blue','blue','blue','blue','blue','blue','blue');

                        var client = new Messaging.Client(MQTTbroker, MQTTport,
                                                          "myclientid_"                    +
parseInt(Math.random() * 100, 10));

                        client.onConnectionLost = function (responseObject) {
                                alert("connection          lost:          "          +
responseObject.errorMessage);
                                console.log("connection lost");
                        };

                        client.onMessageArrived = function (message) {
                                console.log(message.destinationName   +   '   --   '   +
message.payloadString);
                                var recievedmsg = message.payloadString;
                                // sample message expected
                                //{"_type": "dev2", "lat": "38.1935", "lon": "23.6379",
"production": "80W", "consumption": "0W", "temp": "22.5", "humi": "55.5", "CO2": "-",
"rain":"-", "w_speed":"-", "batt":"0.6KW", "tst": "1431957928" }
                                //check to see if payload is JSON
                                try {
                                        var   myObj   =   jQuery.parseJSON(recievedmsg);
//parse payload
                        } catch (e) {
                                console.log(e);
                                return false;
                        };

                                //check to see if latittude (lat) is in the JSON message
                                if ("lat" in myObj){
                                        //property exists
                                        var myDate = new Date(myObj.tst *1000); //convert
epoch time to readible local datetime
                                        var       markerinfo       =       myDate+'<br
/><b>'+message.destinationName+'</b><br />'+message.payloadString;
                                        addMarker(myObj.lat,myObj.lon,markerinfo,
message.destinationName);
                                                //  add  marker  based  on  lattitude  and
longittude, using timestamp for description for now
                                        center = bounds.getCenter(); //center on marker,
zooms in to far atm, needs to be fixed!
                                        map.fitBounds(bounds);
                                };
                        };

                        var options = {
                                timeout: 60,
                                onSuccess: function () {
                                        // alert("Connected");
                                        console.log("mqtt connected");
                                        // Connection succeeded; subscribe to our topic
                                        client.subscribe(MQTTsubTopic, {qos: 0});
                                },
                                onFailure: function (message) {
                                        alert("Connection       failed:       "       +
message.errorMessage);
                                        console.log("connection failed");
                                }
                        };


                var center = null;
                var map = null;
                var currentPopup;
                var bounds = new google.maps.LatLngBounds();
                var x = 0;
                var markers = {};
                var latestlocations = []; // creating dictionary of latest locations.


                function addMarker(lat, lng, info, person) {
//                      console.log('addmaker -- ' + lat + ' ' + lng + ' ' + info + ' '
+ person);

                        var pinColour = 'blue' //default colour of pin
```

```
                        //check the array if the person is already added to the array,
if not add it.
                        if (personArray.indexOf(person) < 0) {
                                personArray.push(person); //add the new person to the
array
                                pinColour = pinColourArray[personArray.indexOf(person)];
                                // add the new person to the legend on the map
                                var div = document.createElement('div');
                                div.innerHTML     =     '<img     onclick="selectmarker('+
personArray.indexOf(person)  +')"  src="http://maps.google.com/mapfiles/ms/micons/'  +
pinColour + '.png"> ' + person;
                                legend.appendChild(div);
                                //create a new item for person
                                latestlocations.push({
                                        key:    personArray.indexOf(person),
                                        value:  0,
                                });

                        }

                        //check the pincolour for the person
                        pinColour = pinColourArray[personArray.indexOf(person)];

                        //assign the marker colour
                        var                  icon                  =                  new
google.maps.MarkerImage("http://maps.google.com/mapfiles/ms/micons/"    +    pinColour
+".png",
                                new        google.maps.Size(32,        32),        new
google.maps.Point(0, 0),
                                new google.maps.Point(16, 32));
                        var pt = new google.maps.LatLng(lat, lng);
                        bounds.extend(pt);
                        x = x + 1;
                        var marker = new google.maps.Marker({
                                position: pt,
                                icon: icon,
                                map: map,
                                id: x,
                         animation: google.maps.Animation.DROP

                        });

                        markers[x] = marker; //add to the markers array so we can call
them later
                        latestlocations[personArray.indexOf(person)] =      markers[x];
//update array with their latest location

                        //create the popup for marker
                        var popup = new google.maps.InfoWindow({
                                content: info,
                                maxWidth: 280
                        });

                        //what happens when you click on a marker
                        google.maps.event.addListener(marker, "click", function() {
                                if (currentPopup != null) {
                                        currentPopup.close();
                                        currentPopup = null;
                                }

                                //zoom in on the marker
                                map.setZoom(18);
                                map.setCenter(marker.getPosition());
                                //open the info box popup
                                popup.open(map, marker);
                                currentPopup = popup;
                        });

                        // what happens when you close a marker
        //              google.maps.event.addListener(popup, "closeclick", function() {
                                // zoom back out to see all markers
                //              center = bounds.getCenter();
                //      map.fitBounds(bounds);
                //               map.panTo(center);
                //              currentPopup = null;
                //      });
```

```
                        //zoom in on this latest marker
        //              selectmarker(personArray.indexOf(person));
                };

                //function for selecting the latest marker for a persion
                function selectmarker(person) {
                google.maps.event.trigger(latestlocations[person], 'click');
                };

                function initMap() {
                        map = new google.maps.Map(document.getElementById("map"), {
                                center: new google.maps.LatLng(38.0650173,23.7954726),
                                zoom: 12,
                                mapTypeId: google.maps.MapTypeId.ROADMAP,
                                mapTypeControl: true,
                                mapTypeControlOptions: {
                                        style:
google.maps.MapTypeControlStyle.HORIZONTAL_BAR
                                },
                                navigationControl: true,
                                navigationControlOptions: {
                                        style:
google.maps.NavigationControlStyle.ZOOM_PAN
                                }
                        });

                        var legend = document.getElementById('legend');

        map.controls[google.maps.ControlPosition.RIGHT_BOTTOM].push(legend);
                        center = bounds.getCenter();
                        map.fitBounds(bounds);

                        /* Connect to MQTT broker */
                        client.connect(options);
                };
                </script>

        </head>

        <body>
                <body onload="initMap()" >
                <div id="map"></div>
                <div id="legend"><h3>LamppostID</H3></div>
        </body>
</html>
```

Range tests

Πομπός rfm69 (Arduino) – rfm69_rangetest.ino

```
1.  #include <RFM69.h>
2.  #include <RFM69registers.h>
3.  #include <SPI.h>
4.  #include <Wire.h>
5.  #include <stdlib.h>
6.  #include "printf.h"
7.
8.  #define NODEID        2    //unique for each node on same network
9.  #define DEVICEID      1
10. #define NETWORKID     100  //the same on all nodes that talk to each other
11. #define GATEWAYID     1
12.
13. //Match frequency to the hardware version of the radio on your Arduino
```

```
14. #define FREQUENCY   RF69_868MHZ // other options: RF69_433MHZ  RF69_868MHZ RF69_915MHZ
15. #define ENCRYPTKEY    "0123456789012345" //exactly the same 16 characters/bytes on all nodes!
16. 
17. //uncomment accordingly
18. #define RF_BITRATEMSB_4800            0x1A  // Default
19. //#define RF_BITRATEMSB_9600            0x0D
20. //define RF_BITRATEMSB_19200           0x06
21. //#define RF_BITRATEMSB_38400           0x03
22. 
23. #define SERIAL_BAUD   115200  //must be 9600 for GPS, use whatever if no GPS
24. 
25. boolean requestACK = true;
26. unsigned long loop_start_time;
27. RFM69 radio;
28. byte number;
29. 
30. void setup()
31. {
32.   Serial.begin(SERIAL_BAUD);          //  setup serial
33.   Wire.begin();
34.   delay(1000);
35.   Serial.println("Range testing");
36.   Serial.print("Initializing Radio...");
37.   radio.initialize(FREQUENCY, NODEID, NETWORKID);
38.   Serial.println("OK!!");
39. 
40.   radio.encrypt(ENCRYPTKEY);
41.   char buff[50];
42.   sprintf(buff, "\nTransmitting           0-100           at           %d           Mhz...",
    FREQUENCY == RF69_433MHZ ? 433 : FREQUENCY == RF69_868MHZ ? 868 : 915);
43.   Serial.println(buff);
44.   number=0;
45. }
46. 
47. void loop() {
48. 
49.     loop_start_time = millis();
50. 
51.     Serial.print("Message (");
52.     Serial.print(sizeof(number));
53.     Serial.print(" bytes) | ");
54.     Serial.print("number: ");
55.     Serial.println(number);
56. 
57.     radio.send(GATEWAYID, (const void*)(&number), sizeof(number), requestACK);
58.     number++;
59.     delay(1000);
60. 
61.     if (number==100){
```

```
62.       number=0;
63.     }
64. }//end loop
```

Δέκτης rfm69 (Raspberry Pi) – rfm69_rangetest.c

```
1.  #include "rfm69.h"
2.  #include <wiringPi.h>
3.  #include <stdio.h>
4.  #include <stdlib.h>
5.  #include <stdint.h>
6.  #include <time.h>
7.  #include <unistd.h>
8.  #include <fcntl.h>
9.  #include <string.h>
10. #include <pthread.h>
11. #include <errno.h>
12.
13. #define FREQUENCY RF69_868MHZ
14. #define NODEID 1
15. #define NETWORKID 100
16. #define TXPOWER 31
17. #define CRYPTPASS "0123456789012345"    // A 16 bit password
18.
19. unsigned char received[63];
20. int rssi;
21. char datalen;
22. char senderId;
23. int rssi_sum=0;
24. int i=0;
25.
26. int main(int argc, char* argv[]) {
27.
28.         printf("Strarting rangetest.c ...\n\n");
29.         rfm69_initialize(FREQUENCY, NODEID, NETWORKID);
30.
31.         rfm69_encrypt(CRYPTPASS);
32.         rfm69_setPowerLevel(TXPOWER); // Max Power
33.
34.         while(1){
35.                 rfm69_receive();
36.                 datalen = rfm69_getDataLen();
37.                 if(datalen > 0){
38.
39.                         rssi = rfm69_getRssi();
40.                         rfm69_getData(received);
41.                         senderId = rfm69_getSenderId();
42.
43.                         printf("RSSI: %i\t", rssi);
44.                         printf("Received:");
45.                         printf(" = %i\n\n",(int)received[0]);
46.
47.                         rssi_sum=rssi_sum+rssi;
```

```
                        i++;
                        if (i==10){
                                printf("\n============\n");
                                                        printf("RSSI  average  of  10  packets:
%.1f \n", (float)rssi_sum/10);
                                printf("============\n\n");
                                i=0;
                                rssi_sum=0;
                                exit(0);
                        }
                }
        }
}
```

Πομπός nRF24L01+ (Arduino Nano) – nrf24_send.ino

```
#include <SPI.h>
#include "RF24.h"

/* Hardware configuration: Set up nRF24L01 radio on SPI bus plus pins 9 & 10 */
RF24 radio(9,10);

//        Radio        pipe        addresses        for        the        2        nodes        to
communicate.
byte addresses[][6] = {"1Node","2Node
byte counter = 0;       // A single byte to keep track of the data being sent back and forth
int success=0;
int fails=0;

void setup(){

  Serial.begin(115200);
  Serial.println(F("RF24 rangetest: nrf24_send"));

  radio.begin();

  //RF24_250KBPS for 250kbs, RF24_1MBPS for 1Mbps, or RF24_2MBPS for 2Mbps
  if (radio.setDataRate(RF24_250KBPS)) {
  //if (radio.setDataRate(RF24_1MBPS)) {
  //if (radio.setDataRate(RF24_2MBPS)) {
    Serial.println(F("DataRate set ok!"));
    }

  radio.enableAckPayload();                    // Allow optional ack payloads
  radio.enableDynamicPayloads();               // Ack payloads are dynamic payloads

  radio.openWritingPipe(addresses[0]);
  radio.openReadingPipe(1,addresses[1]);

```

```
33.   radio.startListening();                       // Start listening
34.
35.   radio.writeAckPayload(1,&counter,1);// Pre-load an ack-paylod into the FIFO buffer for pipe 1
36.   radio.printDetails();
37. }
38.
39. void loop(void) {
40.   byte gotByte;                    // Initialize a variable for the incoming response
41.
42.   radio.stopListening();           // First, stop listening so we can talk.
43.   Serial.print(F("Now sending ")); // Use a simple byte counter as payload
44.   Serial.println(counter);
45.
46.   unsigned long time = micros();   // Record the current microsecond count
47.
48.   if ( radio.write(&counter,1) ){    // Send the counter variable to the other radio
49.     if(!radio.available()){         // If nothing in the buffer, we got an ack but it is blank
50.       unsigned long timer = micros();
51.       Serial.print(F("Got blank response. "));
52.       Serial.print(F(" round-trip delay: "));
53.       Serial.print(timer-time);
54.       Serial.println(F(" microseconds"));
55.       success++;
56.     }else{
57.       while(radio.available() ){                      // If an ack with payload was received
58.         radio.read( &gotByte, 1 );                  // Read it, and display the response time
59.         unsigned long timer = micros();
60.
61.         Serial.print(F("Got response "));
62.         Serial.print(gotByte);
63.         Serial.print(F(" round-trip delay: "));
64.         Serial.print(timer-time);
65.         Serial.println(F(" microseconds"));
66.         success++;
67.       }
68.     }
69.   }else{
70.     Serial.println(F("Sending failed."));           // If no ack response, sending failed
71.     fails++;
72.   }
73.   counter++;
74.   delay(250);  // Try again later
75.
76.   if (counter==100){
77.     counter=0;
78.     Serial.print(F("success: "));
79.     Serial.println(success);
80.     Serial.print(F("fails: "));
81.     Serial.println(fails);
```

```
82.     success=0;
83.     fails=0;
84.     //delay(300000);
85.   }
86. }
```

Δέκτης nRF24L01+ (Arduino Uno) – nrf24_receive.ino

```
1.  #include <SPI.h>
2.  #include "RF24.h"
3.
4.  /* Hardware configuration: Set up nRF24L01 radio on SPI bus plus pins 9 & 10 */
5.  RF24 radio(9,10);
6.
7.  // Radio pipe addresses for the 2 nodes to communicate.
8.  byte addresses[][6] = {"1Node","2Node"};
9.  byte counter=0;
10. int success=0;
11. int fails=0;
12.
13. void setup(){
14.
15.   Serial.begin(115200);
16.   Serial.println(F("RF24 rangetest: nrf24_receive"));
17.
18.   radio.begin();
19.
20.   //RF24_250KBPS for 250kbs, RF24_1MBPS for 1Mbps, or RF24_2MBPS for 2Mbps
21.   if (radio.setDataRate(RF24_250KBPS)) {
22.   //if (radio.setDataRate(RF24_1MBPS)) {
23.   //if (radio.setDataRate(RF24_2MBPS)) {
24.     Serial.println(F("DataRate set ok!"));
25.   }
26.
27.   radio.enableAckPayload();                    // Allow optional ack payloads
28.   radio.enableDynamicPayloads();               // Ack payloads are dynamic payloads
29.
30.    radio.openWritingPipe(addresses[1]);         // Both radios listen on the same pipes by
    default, but opposite addresses
31.   radio.openReadingPipe(1,addresses[0]);      // Open a reading pipe on address 0, pipe 1
32.
33.   radio.startListening();                      // Start listening
34.
35.   radio.writeAckPayload(1,&counter,1);         // Pre-load an ack-paylod into the FIFO buffer
    for pipe 1
36.   radio.printDetails();
37. }
```

```
38.
39. void loop(void) {
40.   byte pipeNo, gotByte;                          // Declare variables for the pipe and the byte
    received
41.   while( radio.available(&pipeNo)){              // Read all available payloads
42.     radio.read( &gotByte, 1 );
43.     Serial.print(F("Received "));
44.     Serial.println(gotByte);
45.   }
46. }
```