ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΓΕΝΙΚΟ ΤΜΗΜΑ ΛΑΡΙΣΑΣ

Μεταπτυχιακό Πρόγραμμα Σπουδών

Τίτλος: "Μηχανικών Η/Υ και Συστημάτων – Ευφυή Συστήματα και IoT"
Επιβλέπων: Δρ. Βέντζας Δημήτριος
Εκπόνηση πτυχιακής: Χατζηδάκης Λάμπρος (Α.Μ. 8117006)

Τίτλος Πτυχιακής:

Σχεδιασμός και ανάπτυξη κόμβου ομιχλώδους υπολογισμού με χρήση πλατφορμών Διαδικτύου των Αντικειμένων

Λέξεις κλειδιά : WSN, Αισθητήρες, Μικροελεγκτές, Xbee, ThingSpeak, Ασύρματη μετάδοση δεδομένων, Πρωτόκολλο API, Βάσεις Δεδομένων, Cloud computing, arduino, raspberry

Dissertation thesis:
Design and Development of Fog Computing Node Based on Internet of Things

Key words: WSN, Sensors, Microcontrollers, Xbee, ThingSpeak, Wireless Data Transmission, Protocol API, Database, Cloud computing, arduino, raspberry

Υπεύθυνη δήλωση μη λογοκλοπής

Δηλώνω υπεύθυνα ότι είμαι ο συγγραφέας αυτής της πτυχιακής εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της, είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, έχω σαφής αναφορές (συντάκτη, χρονολογία, εργασία) τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, προτάσεων, ιδεών ή λέξεων, είτε αυτές αναφέρονται ακριβώς είτε είναι παραφρασμένες. Καταλαβαίνω ότι η αποτυχία να γίνει αυτό, ανέρχεται σε λογοκλοπή και θα θεωρηθεί λόγος αποτυχίας, σε αυτή την Πτυχιακή Εργασία και του συνολικού βαθμού της. Ακόμα δηλώνω ότι αυτή η γραπτή εργασία , προετοιμάστηκε από εμένα προσωπικά και αποκλειστικά και ότι θα αναλάβω πλήρως τις συνέπειες εάν η εργασία αυτή αποδειχθεί ότι δεν μου ανήκει.

Χατζηδάκης Λάμπρος

ΕΠΙΤΡΟΠΗ ΑΞΙΟΛΟΓΗΣΗΣ

1. Δρ. Βέντζας Δημήτριος

2. Δρ. Ξενάκης Απόστολος

3. Δρ. Τσουκάτος Κωνσταντίνος

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή

Τόπος: .....

Ημερομηνία:

Περίληψη

Στην σημερινή εποχή ο τεράστιος όγκος των δεδομένων που παράγεται από την χρήση «έξυπνων» συσκευών έχει δημιουργήσει την ανάγκη διαχείρισης αυτών των δεδομένων, ώστε αφ’ ενός να είναι ασφαλής η αποθήκευσή τους και η ακεραιότητά τους και αφ’ ετέρου, με κατάλληλο τρόπο, να γίνεται η εξαγωγή χρήσιμων συμπερασμάτων.

Όλες οι επιχειρήσεις είναι πλέον συνδεδεμένες στο Διαδίκτυο σε 24ωρη βάση και για την βελτίωση της αποδοτικότητας τους χρησιμοποιούν συστήματα με αισθητήρες ώστε να έχουν την πλήρη παρακολούθηση και έλεγχο του περιβάλλοντος εργασίας. Μέσω του Διαδικτύου η παρακολούθηση αυτή μπορεί να γίνεται απομακρυσμένα με χρήση Cloud από οποιοδήποτε σημείο του πλανήτη και από οποιαδήποτε ηλεκτρονική συσκευή (tablet, προσωπικός υπολογιστής, smart phone κλπ).

Σκοπός της συγκεκριμένης εργασίας είναι η ανάπτυξη ενός πρότυπου συστήματος, το οποίο μετράει την υγρασία και την θερμοκρασία ενός οικιακού/εργασιακού περιβάλλοντος και αποστέλλει τα μετρούμενα δεδομένα σε πλατφόρμα IoT που βρίσκεται εγκατεστημένη στο νέφος. Με κατάλληλο προγραμματισμό της πλατφόρμας αποθηκεύουμε τα δεδομένα και τα αναπαριστούμε με γραφήματα πραγματικού χρόνου για την καλύτερη επιτήρηση του συστήματος από τον χρήστη.

Για το σύστημα το οποίο αναπτύχθηκε, χρησιμοποιήθηκε arduino και raspberry, ενώ η μετάδοση των δεδομένων από την συσκευή των αισθητήρων προς την πύλη IoT (raspberry) έγινε μέσω των ολοκληρωμένων κυκλωμάτων xbee. Στο raspberry πραγματοποιούνται όλες οι απαραίτητες προγραμματιστικές διαδικασίες ομιχλώδους υπολογισμού, για την επεξεργασία των δεδομένων του αισθητήρα και την μετάδοσή τους μέσω επικοινωνίας διεπαφής http RESTful στην πλατφόρμα IoT ThingSpeak (επίπεδο εφαρμογής - application layer). Επιλέξαμε να έναν αισθητήρα υγρασίας/θερμοκρασίας DHT11 λόγω του χαμηλού κόστους. Η παρούσα εργασία αποτελεί πρότυπο οδηγό για την κατασκευή και υλοποίηση αυτόνομων συστημάτων επιτήρησης περιβαλλοντικών συνθηκών.

Λέξεις κλειδιά: WSN, Αισθητήρες, Μικροελεγκτές, Xbee, ThingSpeak, Ασύρματη μετάδοση δεδομένων, Πρωτόκολλο API, Βάσεις Δεδομένων, Cloud computing, arduino, raspberry

Abstract

The use of “smart” devices produces a vast amount of data which have to be securely stored and verified, so that by their manipulation one can come to useful conclusions and identify trends and patters.

Most companies are connected to the Internet 24/7, and in order to ensure optimal operation, the use of sensors is widely adopted. Moreover, the use of Cloud technology in conjunction with Internet applications, monitoring and operating of certain equipment can be done remotely and securely, by just using a “smart” device such as a smart phone, tablet or PC/laptop

In this particular thesis we have developed a novel system which measures the temperature and humidity of the environment (work or home) and sends the data produced to an IoT platform in the cloud. The data are stored with proper coding of the platform. To develop the system, we used both an Arduino and a Raspberry board computer and the transmission of the data was achieved using xbee ic.

Raspberry computer takes over all the programming steps of fog computing for the manipulation of the data and their transmission via an http RESTful interface to the ThingSpeak IoT platform (which lies within the application layer). We chose a DHT11 sensor due to cost restrains. The current thesis can become a prototype template for the construction and realization of environmental parameters.

Key words:

WSN, Sensors, Microcontrollers, Xbee, ThingSpeak, Wireless Data Transmission, Protocol API, Database, Cloud computing, arduino, raspberry

Ευχαριστίες:

Θα ήθελα να ευχαριστήσω τους γονείς μου για την στήριξή τους σε όλη την διάρκεια των σπουδών μου, τον αδερφό μου Νίκο για την πολύτιμη βοήθειά του και την διάθεση να μου μεταδώσει τις γνώσεις του, διαθέτοντάς μου τον πολύτιμο και περιορισμένο χρόνο του, καθώς επίσης και την αρραβωνιαστικιά μου Ευαγγελία, που με στήριξε σε όλες τις δύσκολες στιγμές.
Επίσης να ευχαριστήσω θερμά τον Δρ. Γκιουλέκα για την πολύτιμη καθοδήγησή του και συμβουλές, αλλά και τον επιβλέποντα και διευθυντή του μεταπτυχιακού προγράμματος καθηγητή Βέντζα Δημήτριο για την επίβλεψη, την ηθική συμπαράσταση και την υποστήριξή του.

Πίνακας περιεχομένων

Πίνακας Εικόνων

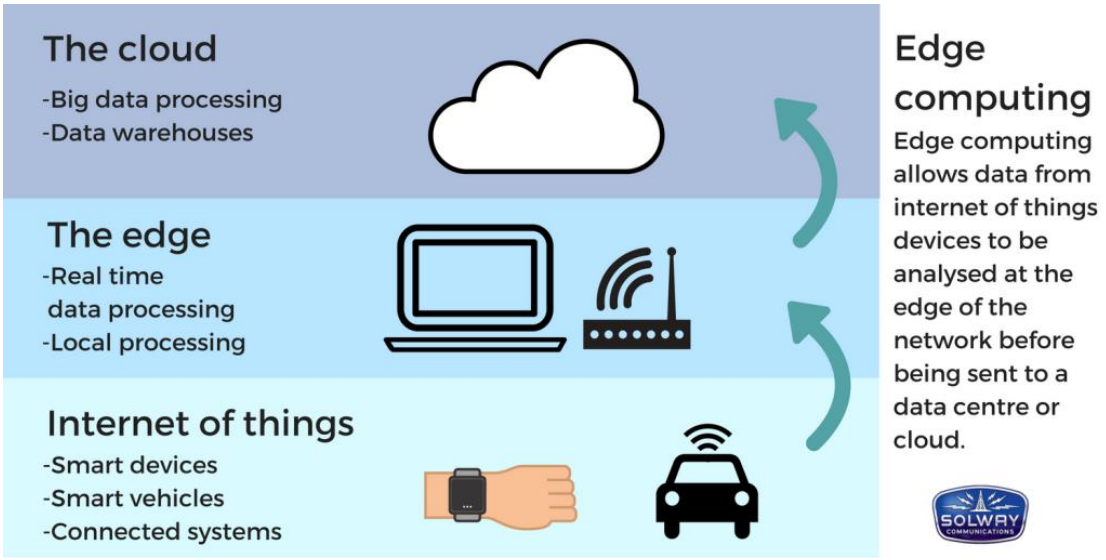
Κεφάλαιο 1ο

1.1 Edge Computing

Στην σημερινή εποχή, πάρα πολλές ηλεκτρονικές συσκευές, αν όχι όλες, που χρησιμοποιούμε καθημερινά είναι συνδεδεμένες με άλλες μέσω ασύρματων δικτύων και ανταλλάσσουν τεράστιο όγκο δεδομένων μέσω cloud. Αυτό που ξεχωρίζει το edge computing (Εικόνα 1) είναι η επεξεργασία των παραγόμενων δεδομένων όσο πιο κοντά στον χρήστη ή αλλιώς στην άκρη (edge) του δικτυού, σε συσκευές που έχουν υπολογιστικές δυνατότητες και διαθέτουν μνήμη όπως για παράδειγμα είναι οι servers, gateways κλπ.

Ο λόγος για τον οποίο χρειαζόμαστε αυτού του είδους την επεξεργασία είναι το γεγονός ότι καθημερινά παράγονται δεδομένα τα οποία αποθηκεύονται σε clouds για περεταίρω χρήση από άλλους χρήστες και συσκευές, τα οποία όμως πρέπει να είναι άξια χρήσης και όχι απλά παραγόμενα δεδομένα. Ένας άλλος λόγος είναι η ελαχιστοποίηση των υπολογιστικών αναγκών από τη μεριά του χρήστη. Με τα δεδομένα προ-επεξεργασμένα σε τέτοιες συσκευές, η δυνατότητα του χρήστη να χρησιμοποιήσει τα δεδομένα άμεσα, μεγιστοποιείται.

Ένα παράδειγμα για την κατανόηση του edge computing είναι ένα σύστημα από κάμερες. Η κάθε κάμερα καταγράφει το περιβάλλον στο οποίο την έχουμε εγκατεστημένη και να παράγει δεδομένα με την μορφή βίντεο. Εάν εμείς θέλουμε να πάρουμε μόνο τα στιγμιότυπα στα οποία ανιχνεύεται κάποια κίνηση θα πρέπει να κάνουμε επεξεργασία σε τοπικό επίπεδο και να ανεβάσουμε στο cloud μόνο αυτά τα στιγμιότυπα και όχι βίντεο το οποίο έχει διάρκεια ώρες ή μέρες. Έτσι λοιπόν περιορίζεται σημαντικά και η χρήση δικτύου από και προς το cloud αλλά και η αποθήκευση μόνο χρήσιμων δεδομένων κάνοντας έτσι το cloud όσο πιο χρηστικό γίνεται χωρίς να γεμίζει με ανεπιθύμητα δεδομένα. Επίσης με αυτόν τον τρόπο μειώνεται σημαντικά το κόστος των συσκευών διότι στο παραπάνω παράδειγμα αντί να έχουμε κάμερες όπου η κάθε μία θα πρέπει να έχει εγκατεστημένο λογισμικό ώστε να ξεχωρίζει τα δεδομένα που θέλουμε, έχουμε πιο απλές κάμερες που απλά καταγράφουν και η διαχείρισή του βίντεο γίνεται σε κάποιον edge server που τρέχει τον αλγόριθμο που απαιτείται και στην συνέχεια τα δεδομένα αποστέλλονται στο cloud.



Εικόνα 1: Edge Computing

1.2 Ομιχλώδης Υπολογισμός

Το Fog Computing (ομιχλώδης υπολογισμός) είναι μια προηγμένη ή, εναλλακτικά, μια εκτεταμένη έκδοση του Cloud computing όπου η επεξεργασία, η αποθήκευση και η εν μέρει ανάλυση των δεδομένων μας πραγματοποιείται στην λεγόμενη «άκρη» του δικτύου των διασυνδεμένων συσκευών μας, δηλαδή, εκεί όπου δημιουργούνται τα δεδομένα και χρησιμοποιούνται πιο συχνά.

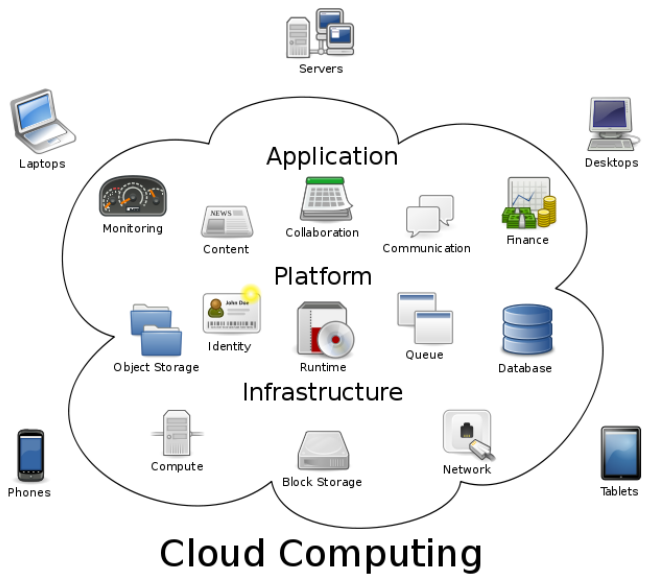
Είναι παρόμοιο με το edge computing, αλλά είναι πολύ πιο πυκνό στη γεωγραφική κατανομή και τη γεωγραφική θέση καθώς και την εγγύτητά του προς τους τελικούς χρήστες η οποία είναι μεγαλύτερη, πράγμα που σημαίνει ότι παρέχει ταχύτερη εμπειρία και καλύτερες επιδόσεις στον τελικό χρήστη από το edge computing.

Η CISCO παρουσίασε πρόσφατα το Vision of fog computing για να επιτρέψει εφαρμογές να τρέχουν απευθείας στην άκρη του δικτύου σε δισεκατομμύρια συνδεδεμένες συσκευές.

1.3 Cloud Computing

Το Cloud computing (Εικόνα 2) είναι το νεότερο μοντέλο υπολογιστικής δομής που καθιστά τους πόρους υπολογιστών διαθέσιμους μέσω του Διαδικτύου. Προσφέρει πολλά πλεονεκτήματα στους χρήστες όσον αφορά το μειωμένο κόστος, τον περιορισμό της διαχείρισης του συστήματος, την αυξημένη ευελιξία, την καλύτερη αξιοπιστία και την ανεξαρτησία της τοποθεσίας. Αν και αυτά είναι καθοριστικά πλεονεκτήματα, το cloud computing όμως "πάσχει" από ορισμένους περιορισμούς. Η φιλοξενία κέντρων δεδομένων cloud στο Διαδίκτυο δημιουργεί μεγάλες και απρόβλεπτες καθυστερήσεις στο δίκτυο και απροσδιόριστα ζητήματα ασφάλειας, καθώς ευαίσθητα δεδομένα ανατίθενται πλέον για αποθήκευση και επεξεργασία σε τρίτους.

Οι μικρές και μεσαίες επιχειρήσεις, ιδίως οι νεοσύστατες, επιλέγουν όλο και περισσότερο την εξωτερική ανάθεση και επεξεργασία δεδομένων τους στο cloud. Αυτό προφανώς είναι η καλύτερη λειτουργική λύση τόσο από άποψη αποτελεσματικότητας όσο και από άποψη κόστους, αλλά έρχεται πλέον σε αντίθεση με μεγάλους κινδύνους, με πιο σοβαρό εκείνον της κλοπής δεδομένων και κωδικών πρόσβασης. Αυτό θεωρείται ως μία από τις κορυφαίες απειλές για το cloud computing.



Εικόνα 2: Cloud computing

Ενώ οι περισσότεροι

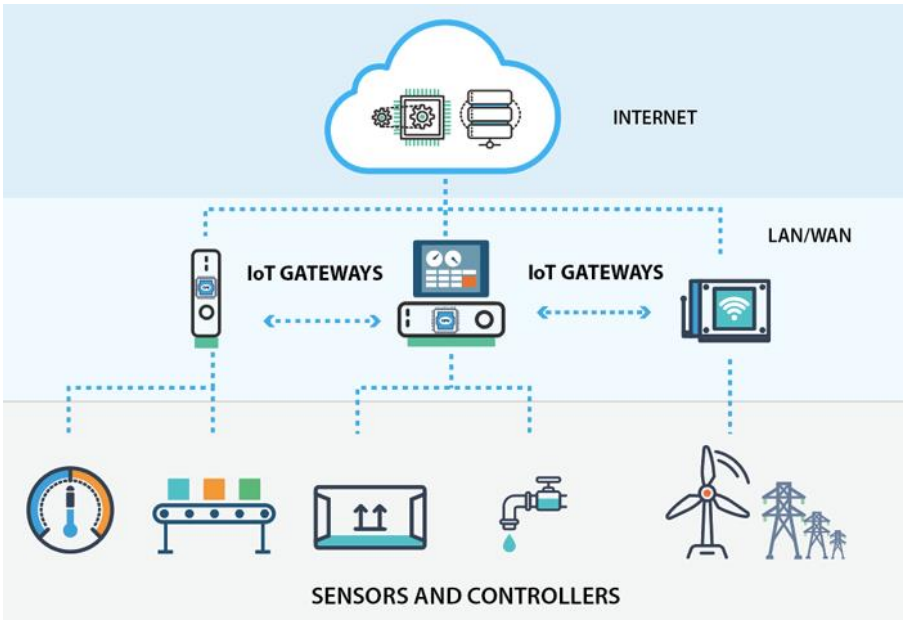
πελάτες Cloud computing γνωρίζουν καλά αυτήν την απειλή, δεν μπορούν να κάνουν τίποτα άλλο παρά να εμπιστευτούν τον πάροχο των υπηρεσιών όταν πρόκειται για την προστασία των δεδομένων τους. Έτσι η ασφάλεια των δεδομένων τους θεωρείται ελλιπής όταν και οι υπηρεσίες ασφάλειας που προσφέρουν οι μεγάλες εταιρείες του χώρου όπως ο έλεγχος, η ταυτοποίηση και εξουσιοδότηση μπορούν να αποτελέσουν κίνητρο για εισβολή (breatch) .

1.4 Gateway IoT

Μία πύλη Διαδικτύου αντικειμένων (Internet of Things gateway) (Εικόνα 3) είναι είτε μία συσκευή, είτε κάποιο λογισμικό που λειτουργεί σαν σημείο σύνδεσης μεταξύ του cloud και των αντίστοιχων ελεγκτών, αισθητήρων και έξυπνων συσκευών. Όλα τα δεδομένα που διακινούνται, είτε από το cloud προς τις συσκευές είτε από τις συσκευές προς το cloud, περνούν από πύλες. Οι πύλες μπορούν συχνά να λειτουργούν σαν «έξυπνες» πύλες σε επίπεδο στρώματος ελέγχου.

Κάποιοι αισθητήρες μπορούν να παράγουν ένα τεράστιο όγκο δεδομένων ανά δευτερόλεπτο λειτουργίας. Η πύλη παρέχει την δυνατότητα της τοπικής προ-επεξεργασίας των δεδομένων πριν αυτά αποσταλούν στο cloud. Τα δεδομένα συναθροίζονται, αναλύονται στην πύλη, ο όγκος τους ελαχιστοποιείται και μετά αποστέλλονται στο cloud. Αυτός ο τρόπος λειτουργίας μπορεί να έχει τεράστια επίπτωση τόσο στο χρόνο απόκρισης του δικτύου συνολικά, όσο και στον όγκο της διακινούμενης πληροφορίας.

Μία πύλη μπορεί επίσης να προσφέρει ένα επιπλέον επίπεδο ασφάλειας στο δίκτυο IoT και τα δεδομένα που διακινούνται. Δεδομένου ότι η πύλη παρέχει αμφίδρομο έλεγχο, μπορεί να προστατέψει την ακεραιότητα των δεδομένων που διακινούνται προς το cloud από απώλειες/διαρροές, όπως επίσης να προστατέψει και τους κόμβους IoT από κακόβουλες επιθέσεις διαφόρων ειδών.



Εικόνα 3: Gateway IoT

Συνοπτικά, μια πύλη IoT εκτελεί τις εξής λειτουργίες[1]:

- Εξασφαλίζει την επικοινωνία συσκευών με το cloud ακόμα και σε περίπτωση που οι συσκευές δεν παρείχαν αυτή τη δυνατότητα.
- Αποθηκεύει προσωρινά τα δεδομένα και ελέγχει τη ροή τους από και προς το cloud
- Προεπεξεργάζεται τα δεδομένα, αφαιρεί διπλότυπες εγγραφές και τα βελτιστοποιεί
- Παρέχει επικοινωνίες μεταξύ των συσκευών (machine-to-machine, M2M)
- Παρέχει διάφορες υπηρεσίες δικτύου και αποθήκευσης δεδομένων.
- Με τη χρήση ειδικού λογισμικού μπορεί να παρέχει οπτικοποίηση των δεδομένων και βασικές υπηρεσίες data analytics.
- Παρέχει βραχυχρόνιο ιστορικό των δεδομένων και των υπηρεσιών που παρείχε.
- Παρέχει υπηρεσίες ασφαλείας και ελέγχου πρόσβασης
- Παρέχει έλεγχο και διαχείριση της συσκευής
- Παρέχει δυνατότητες διάγνωσης της συσκευής

Ειδικά στην περίπτωση edge computing, μία πύλη παρέχει ένα ισχυρό εργαλείο επεξεργασίας, έτσι ώστε να αποφεύγεται η συγκέντρωση των υπολογιστικών αναγκών σε ένα κεντρικό σημείο. Εκεί οφείλεται και ο όρος «έξυπνες» πύλες.

[1] https://whatis.techtarget.com/definition/IoT-gateway

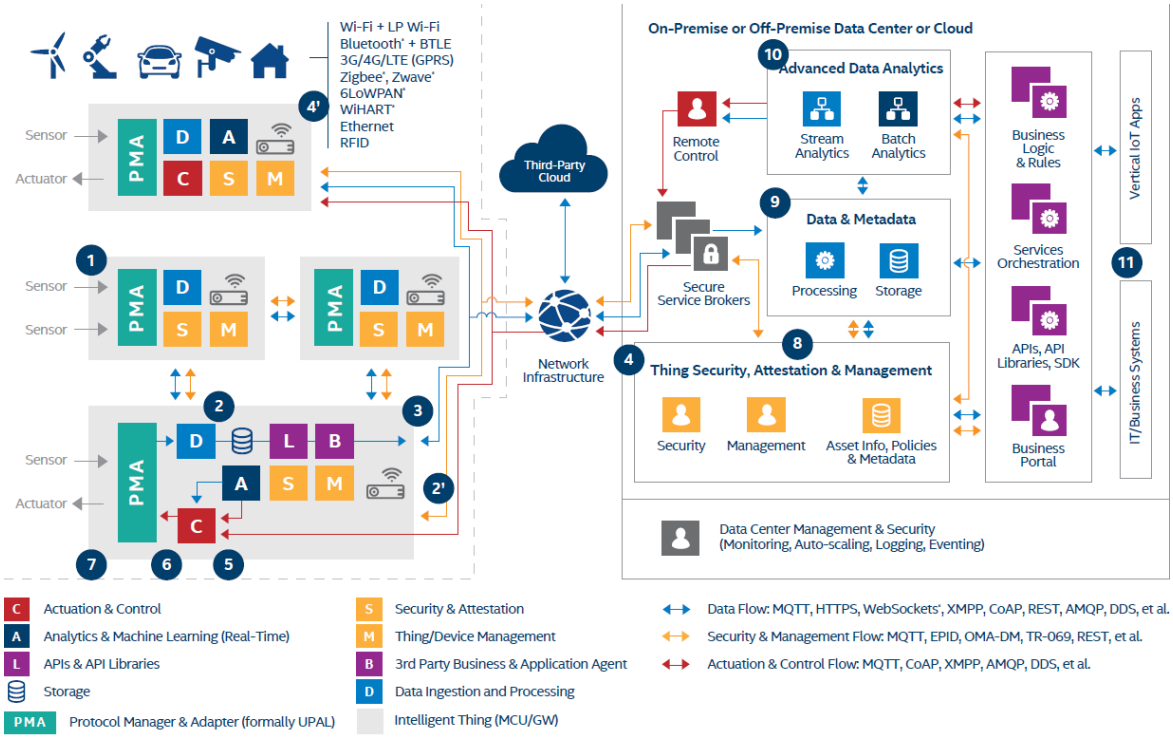
Κεφάλαιο 2ο

2.1 IoT Πλατφόρμες για fog computing

2.1.1 Intel IoT

Η αρχιτεκτονική της Intel (Εικόνα 4), βασίζεται σε middleware (συσκευές και λογισμικό) που μπορούν να λάβουν δεδομένα από αισθητήρες, αλλά και να εκτελέσουν κάποια εργασία με τους actuators, να αποθηκεύσουν ή/και να επεξεργαστούν σε πρώτο στάδιο τα δεδομένα και να τα αποστείλουν με ασφαλή τρόπο σε κάποιο cloud, ανεξάρτητο της Intel, στο διαδίκτυο.

Το Data Center ή Cloud της Intel συνδέεται στο Διαδίκτυο, στο cloud όπου είναι συνδεδεμένες οι middleware συσκευές και λαμβάνει δεδομένα ή στέλνει εντολές ελέγχου. Το Data Center κατανέμει και επεξεργάζεται τα δεδομένα και μέσω των κατάλληλων APIs, αυτά φτάνουν στους χρήστες/χειριστές του συστήματος.



Εικόνα 4: Intel IoT

Χαρακτηριστικά της Intel IoT πλατφόρμας:

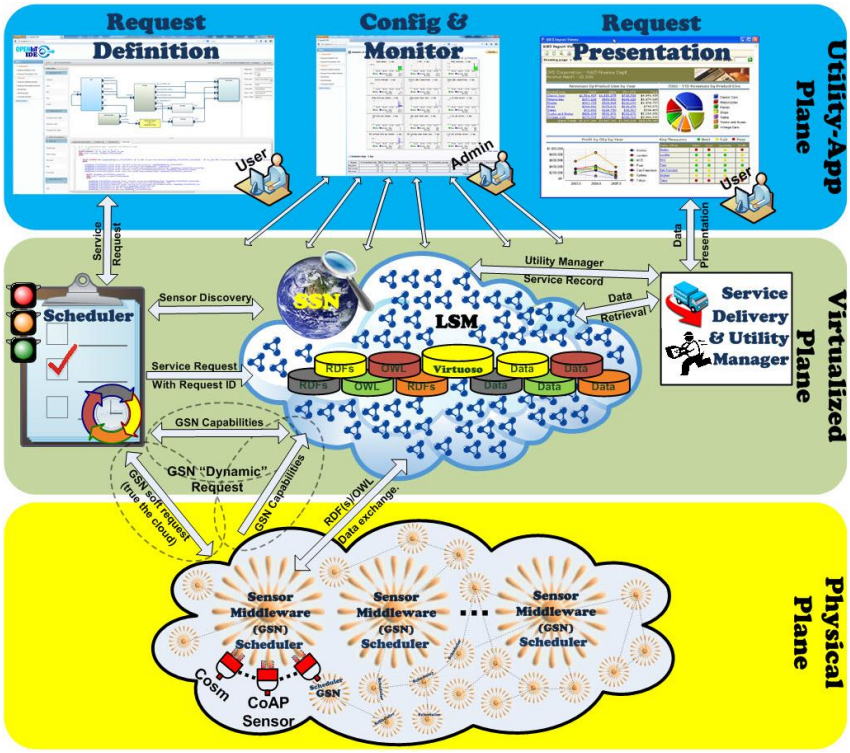
Χρησιμοποιεί σας μοντέλο αναφοράς End-to-End, ξεκινώντας από τον αισθητήρα που θα μετρήσει μία τιμή και καταλήγει στον χειριστή που με βάση τα δεδομένα που βλέπει, θα πάρει μια απόφαση.

Εργαλεία Ανάπτυξης: Galileo, Edison, Intel Open Labs, Freemium Model

2.1.2 Open IoT

Η αρχιτεκτονική της OpenIOT (Εικόνα 5), αποτελείται από 7 κύρια στοιχεία τα οποία ανήκουν σε 3 διαφορετικά λογικά πλάνα. Το Utility/Application το οποίο περιέχει τα στοιχεία Request Definition, Request Presentation και το Configuration and

Monitoring. Το δεύτερο πλάνο είναι το Virtualized το οποίο με την σειρά του εμπεριέχει το Scheduler, το Cloud Data Storage και το Service Delivery & Utility Manager και τέλος είναι το Physical το οποίο αποτελείται από το Sensor



Εικόνα 5: Open IoT

Middleware.

Λειτουργίες στοιχείων

Physical plane

Ξεκινώντας από το Physical plane συναντάμε το Sensor Middleware X-GSN (E**x**tended **G**lobal **S**ensor **N**etwork) το οποίο είναι υπεύθυνο για την συλλογή, την διαλογή και τον συνδυασμό των ροών δεδομένων που προκύπτουν είτε από εικονικούς «αισθητήρες» (όπως αλγορίθμους επεξεργασίας σημάτων, δεδομένα από κοινωνικά δίκτυα κ.λπ.) είτε από αισθητήρες φυσικών συσκευών (θερμόμετρα, μετεωρολογικούς σταθμούς κ.λπ.).

Στην πραγματικότητα το Middleware λειτουργεί ως μεσολαβητής ανάμεσα στην OpenIoT πλατφόρμα και τον φυσικό κόσμο με το να μεταφέρει δεδομένα από τους διάφορα «ερεθίσματα» τα οποία μπορεί να προέρχονται ακόμα και από άλλες πλατφόρμες IoT. Μία βασική ιδιότητα του middleware είναι αυτή της μεταφοράς δεδομένων προς το cloud ανάλογα με την μορφή τους.

Virtualized Plane

Ο Scheduler επεξεργάζεται τα αιτήματα από το Request Definition και εξασφαλίζει την πρόσβαση στους διαφόρους πόρους που απαιτούνται. Οι λειτουργία του είναι η εξής: εντοπίζει τους αισθητήρες και τις συσχετιζόμενες ροές δεδομένων που απαιτούνται ρύθμιση των υπηρεσιών και τις διαχειρίζεται, ενεργοποιώντας τους κατάλληλους πόρους που απαιτούνται.

Το Cloud Data Storage επιτρέπει την αποθήκευση δεδομένων που παρέχονται από τους διάφορους αισθητήρες και λειτουργεί ως cloud database. Επιπλέον, αποθηκεύονται πληροφορίες οι οποίες απαιτούνται για την λειτουργία της OpenIoT. Το λειτουργικό που χρησιμοποιεί είναι το LSM Middleware (Linked Stream Middleware Light) το οποίο είναι σχεδιασμένο να μεταφέρει δεδομένα από και προς το cloud.

Το Service Delivery & Utility Manager εκτελεί 2 λειτουργίες. Η μία είναι να συνδυάζει δεδομένα από το Scheduler και να τα προωθεί στο Request Presentation ή σε μία εφαρμογή. Η δεύτερη είναι να λειτουργεί ως «μετρητής» υπηρεσιών. Αυτές οι μετρήσεις είναι απαραίτητες για την λειτουργία διαφόρων υπηρεσιών όπως λογιστικής, τιμολόγησης κλπ.

Utility/Application Plane

Το Request Definition επιτρέπει την εξυπηρέτηση αιτημάτων σε πραγματικό χρόνο προς την OpenIoT πλατφόρμα μέσω του Web 2.0 interface. Περιλαμβάνει ένα σύνολο υπηρεσιών για τον προσδιορισμό και τη διατύπωση τέτοιων αιτημάτων και ταυτόχρονα τις υποβάλλει στον Scheduler.

Το Request Presentation επιλέγει στοιχεία από κατάλληλες βιβλιοθήκες ώστε να γίνει απεικόνιση μιας υπηρεσίες στο Web 2.0 interface. Για να αποκτήσουν οπτική υπόσταση αυτές οι υπηρεσίες, επικοινωνεί με το Service Delivery & Utility Manager με σκοπό την ανάκτηση συγκεκριμένων δεδομένων.

Το Configuration and Monitoring επιτρέπει τη διαχείριση και διαμόρφωση των λειτουργιών μέσω των αισθητήρων και των υπηρεσιών, που αναπτύσσονται μέσα στην πλατφόρμα OpenIoT.

2.1.3 Sentilo

Η πλατφόρμα sentilo (ονομασία του αισθητήρα στην γλώσσα Esperanto) (Εικόνα 9) έχει σχεδιαστεί για λειτουργία σε περισσότερες πλατφόρμες (cross platform) με σκοπό την ανταλλαγή δεδομένων ανάμεσα σε ετερογενή συστήματα και την εύκολη ενσωμάτωση διαφόρων εφαρμογών.

Είναι ένα σύστημα δημοσίευσης και συνδρομής δεδομένων Java, το οποίο βασίζεται στην Redis, η οποία προσφέρει τα ακόλουθα χαρακτηριστικά:

Restful API

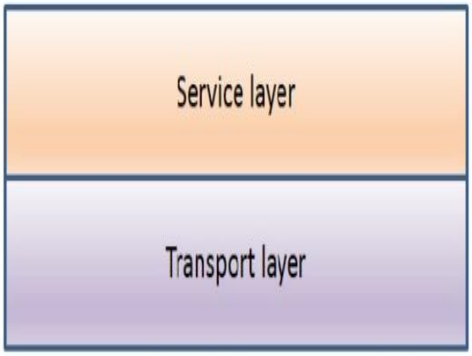
Περιλαμβάνει:

Διαχείριση εφαρμογών web.

Πράκτορες για αποθήκευση δεδομένων σε μία relational βάση δεδομένων.

Πράκτορες για τη σήμανση alarm.

Αποτελείται από τον διακομιστή PubSub (Εικόνα 6). ο οποίος είναι μια αυτόνομη εφαρμογή Java, του οποίου ο σχεδιασμός έχει επικεντρωθεί σε δύο layers υπηρεσίας (service) και μεταφοράς (transport).



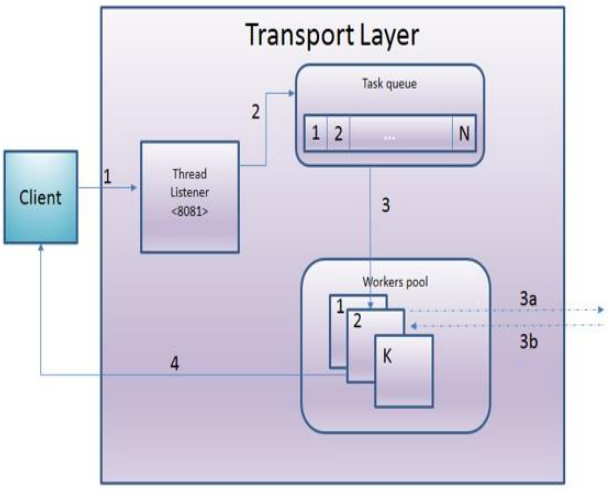
Εικόνα 6: Επίπεδα pubsub

Το επίπεδο μεταφοράς έχει σχεδιαστεί με βάση το μοντέλο Thread Pool.

Το επίπεδο υπηρεσίας έχει βασιστεί στο Spring και Redis, με σκοπό την επίτευξη υψηλών επιπέδων απόδοσης.

Το επίπεδο μεταφοράς (Εικόνα 7)

Ο χρήστης στέλνει αίτημα http στην πλατφόρμα REST. Ο διακομιστής το αποδέχεται και το δρομολογεί στην λίστα των αιτημάτων που εκκρεμούν. Όταν ένας εργάτης (worker) είναι διαθέσιμος, του ανατίθεται η εργασία που εκκρεμεί για επεξεργασία, ενώ ταυτόχρονα αφαιρείται από την λίστα των εκκρεμών αιτημάτων. Ο εργάτης



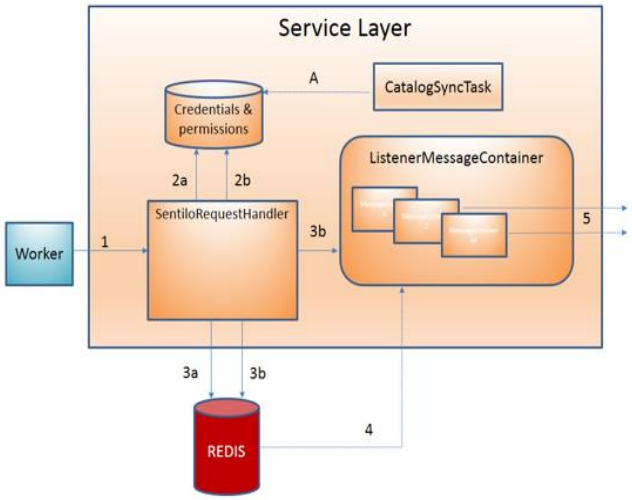
Εικόνα 7: Επίπεδο μεταφοράς

αναθέτει το αίτημα σε ένα στοιχείο του επιπέδου υπηρεσίας και κατασκευάζει την απάντηση http από τα δεδομένα που έχει λάβει. Στέλνει την απάντηση στο αίτημα του χρήστη.

Το επίπεδο υπηρεσίας (Service Layer)

Η σχεδίαση αυτού του επιπέδου έχει σαν κύριο στόχο την ελαχιστοποίηση του χρόνου επεξεργασίας των αιτημάτων, με τέτοιο τρόπο ώστε η κύρια εργασία να κρατείται στη μνήμη (Redis) Το Redis αποθηκεύει δεδομένα σε μια βάση δεδομένων μνήμης, αλλά έχει επίσης τη δυνατότητα αποθήκευσης σε δίσκο, έτσι ώστε να εξασφαλιστεί η διάρκεια των δεδομένων.

Το ακόλουθο διάγραμμα (Εικόνα 8) δείχνει την κύρια ροή για ένα αίτημα μέσα σε αυτό το επίπεδο: Εφαρμογή καταλόγου (Catalog Application)



Εικόνα 8: Ροή αιτήματος

Εφαρμογή καταλόγου

Η πλατφόρμα εφαρμογής καταλόγου είναι μια εφαρμογή ιστού, η οποία έχει υλοποιηθεί με χρήση Spring στην πλευρά του διακομιστή (Spring MVC, Spring Security,..) χρησιμοποιώντας jQuery και bootstrap σαν layer παρουσίασης και την MongoDb σαν βάση αποθήκευσης δεδομένων.

Η εφαρμογή web αποτελείται από:

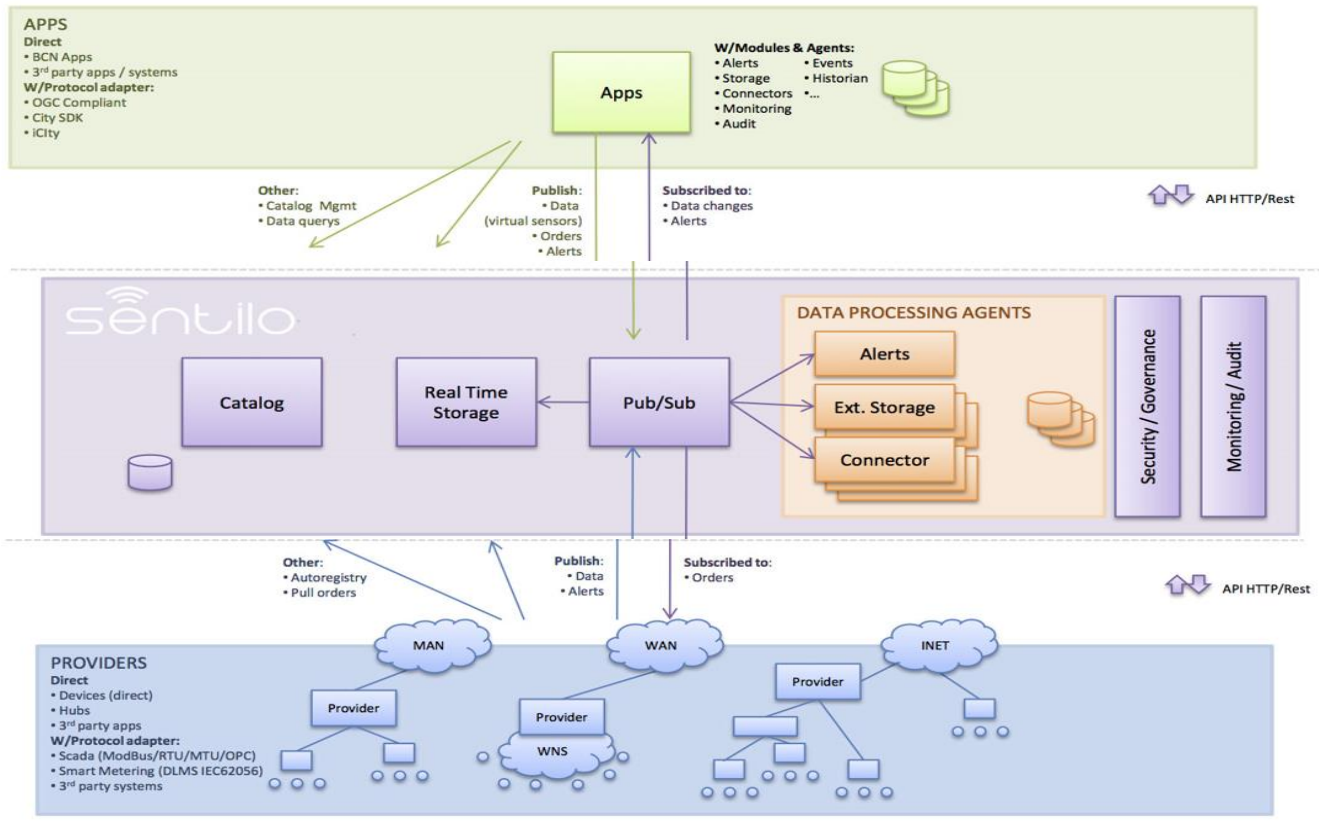
Μία δημόσια κονσόλα για την ένδειξη δημόσιων δεδομένων των τμημάτων της πλατφόρμας και των αισθητήρων.

Ένα ασφαλές τμήμα για την διαχείριση των πόρων: χορηγοί (providers), εφαρμογές χρηστών, αισθητήρες, τμήματα, ειδοποιήσεις, και άδειες.

Έχει πλήρως ενσωματωθεί στην πλατφόρμα δημοσίευσης/συνδρομής για τον συγχρονισμό των δεδομένων:

Δεδομένα αδειών και ταυτοποιήσεων

Καταχώρηση στατιστικών δεδομένων και τα τελευταία δεδομένα που έχουν παραληφθεί για την ένδειξή τους σε διάφορα γραφήματα στην εφαρμογή web.



Εικόνα 9: Πλατφόρμα Sentilo

Κεφάλαιο 3ο

3.1 Βάσεις δεδομένων, τρόποι διαχείρισης δεδομένων

3.1.1 Τι είναι μία Βάση Δεδομένων

Με τον όρο δεδομένα εννοούμε είτε πληροφορίες που παράγονται από τη λειτουργία εξειδικευμένων συσκευών και λογισμικού (όπως για παράδειγμα από αισθητήρες πίεσης, υγρασίας, θερμοκρασίας κλπ.) είτε πληροφορίες που απαιτούνται για τη λειτουργία παρόμοιων συσκευών ή/και λογισμικού. Τα δεδομένα αυτά που παρουσιάζουν μία συσχέτιση μεταξύ τους, είναι χρήσιμο να αποθηκεύονται σε μία βάση δεδομένων (ΒΔ). Είναι, λοιπόν, η ΒΔ μία συλλογή συσχετιζόμενων δεδομένων τα οποία αποθηκεύονται, είτε τοπικά, είτε κατανεμημένα (ενδεχομένως σε κάποιο cloud) και τα οποία είναι διαθέσιμα για μελλοντική ανάκτηση, διαχείριση και χρήση.

Αυτή η ανάκτηση, η διαχείριση και χρήση των δεδομένων, όπως και η εξαγωγή συμπερασμάτων είναι δυνατή χάρη σε λογισμικό που υποστηρίζει τη ΒΔ και συνήθως αναφέρεται με το γενικό όρο Σύστημα Διαχείρισης Βάσης Δεδομένων (ΣΔΒΔ, Data Base Management System - DBMS). Με το ΣΔΒΔ οι χρήστες μπορούν να έχουν πρόσβαση στη ΒΔ, με κατάλληλη εξουσιοδότηση, να επεξεργάζονται τα στοιχεία της (πρόσθεση, αφαίρεση, τροποποίηση), όπως επίσης και να παράγουν αναφορές, σχεδιαγράμματα και γενικά αναπαραστάσεις των δεδομένων, έτσι ώστε να γίνονται αντιληπτές συσχετίσεις των δεδομένων που δεν ήταν εξ αρχής φανερές. Παρά την εξειδίκευση και την πολυπλοκότητα των ΣΔΒΔ, τα πρώτα τέτοια συστήματα χρησιμοποιούνται ήδη από το 1960, με πρώτο ΣΔΒΔ να θεωρείται το Integrated Data Store (IDS) του Charles Bachmen.

Με την πάροδο του χρόνου οι σχετικές τεχνολογίες εξελίχθηκαν πολύ και δεδομένου του τεράστιου όγκου των πληροφοριών που παράγονται από την εφαρμογή του IoT αναμένεται πολύ μεγαλύτερη εξέλιξή τους.

3.2 Είδη Βάσεων Δεδομένων

Η εξέλιξη των ΒΔ αποτυπώνεται στο παρακάτω διάγραμμα (Εικόνα 10)

Εικόνα 10: Είδη Βάσεων Δεδομένων

Υπάρχουν τέσσερεις βασικοί τύποι ΒΔ

Ιεραρχικός:

Βασίζεται για την αποθήκευση των δεδομένων στο σχήμα «γονέα-παιδιού». Πλέον θεωρείται ξεπερασμένο σαν σχήμα και χρησιμοποιείται σπάνια. Η δομή του είναι δενδροειδής, όπου οι εγγραφές παριστάνονται σαν κόμβοι, ενώ τα πεδία παριστάνονται σαν κλάδοι. Κλασσικό παράδειγμα τέτοιας ΒΔ είναι η registry των Windows XP.

Δικτυακές ΒΔ:

Αυτός ο τύπος ΒΔ υποστηρίζει σχέσεις πολλών αντικειμένων με πολλά αντικείμενα (many-to-many) με αποτέλεσμα να οδηγούμαστε σε περίπλοκες δομές ΒΔ. Ένα παράδειγμα τέτοιας ΒΔ είναι ο RDM server.

Σχεσιακές ΒΔ:

Αυτός ο τύπος ΒΔ αναπαριστά τις συσχετίσεις των δεδομένων, με μορφή πινάκων. Σε αντίθεση με τις Δικτυακές Βάσεις Δεδομένων δεν υποστηρίζει σχέσεις πολλών-προς-πολλά. Συνήθως η δομή των δεδομένων είναι προσχεδιασμένη, έτσι ώστε η ΒΔ να υποστηρίζει μόνο τα δεδομένα που έχουν τη συγκεκριμένη προσχεδιασμένη δομή. Προς το παρόν είναι ο δημοφιλέστερος τύπος ΒΔ με παραδείγματα όπως η MySQL, Oracle, και η Microsoft SQL Server.

Αντικειμενοστρεφής Σχεσιακή ΒΔ:

Αυτός ο τύπος ΒΔ υποστηρίζει την αποθήκευση και τον χειρισμό νέων τύπων δεδομένων. Τα δεδομένα αποθηκεύονται με τη μορφή αντικειμένων (object) που έχουν συγκεκριμένες ιδιότητες. Οι ιδιότητες των αντικειμένων μπορούν να προσπελαστούν με «μεθόδους» που ορίζουν συγκεκριμένα πως θα γίνει ο χειρισμός των συγκεκριμένων ιδιοτήτων. Παράδειγμα τέτοιας ΒΔ είναι η PostgreSQL.

Τι είναι η SQL?

Η SQL (Structured Query language) είναι η καθιερωμένο εργαλείο για να χειρίζεται κανείς δεδομένα στις σχεσιακές βάσεις δεδομένων. Με την SQL μπορούμε να εισάγουμε και να διαγράψουμε εγγραφές, να εκτελέσουμε αναζητήσεις και να ανανεώσουμε τις εγγραφές των δεδομένων σε μία ΒΔ. Μπορούμε να εκτελέσουμε εργασίες όπως βελτιστοποίηση των δεδομένων και της ίδιας της ΒΔ, όπως και εργασίες συντήρησης και αναβάθμισης. Σχεσιακές ΒΔ όπως οι: MySQL, Oracle, Ms SQL server, Sybase, κλπ. χρησιμοποιούν την SQL.

Τι είναι η NoSQL

Η NoSQL είναι ένα νέο σύστημα διαχείρισης βάσης δεδομένων. Το κύριο χαρακτηριστικό της είναι ότι δεν είναι αποκλειστικά προσκολλημένη με την σχεσιακή βάση, όπως δείχνει και το όνομά της που σημαίνει: Not only SQL.

Υποστηρίζεται από την Google, το Facebook, την Amazon κλπ. που παράγουν τεράστιες ποσότητες δεδομένων. Τα δεδομένα αυτά, δεδομένου του τεράστιου όγκου τους, είναι πολύ δύσκολο να τα διαχειριστεί κανείς σε εύλογο χρόνο και προκαλούνται καθυστερήσεις. Μία λύση θα ήταν η διαρκής αναβάθμιση του εξοπλισμού (scale up), η οποία όμως δεν αποτελεί πάντα υλοποιήσιμη επιλογή. Εναλλακτικά, μπορεί να κατανεμηθεί το υπολογιστικό κόστος σε πολλαπλούς κόμβους και να μειωθεί έτσι ο χρόνος επεξεργασίας (scale out).

Η NoSQL είναι πολύ καλή στην κατανομή αυτή, και έχει σχεδιαστεί λαμβάνοντας υπόψη εφαρμογές web.

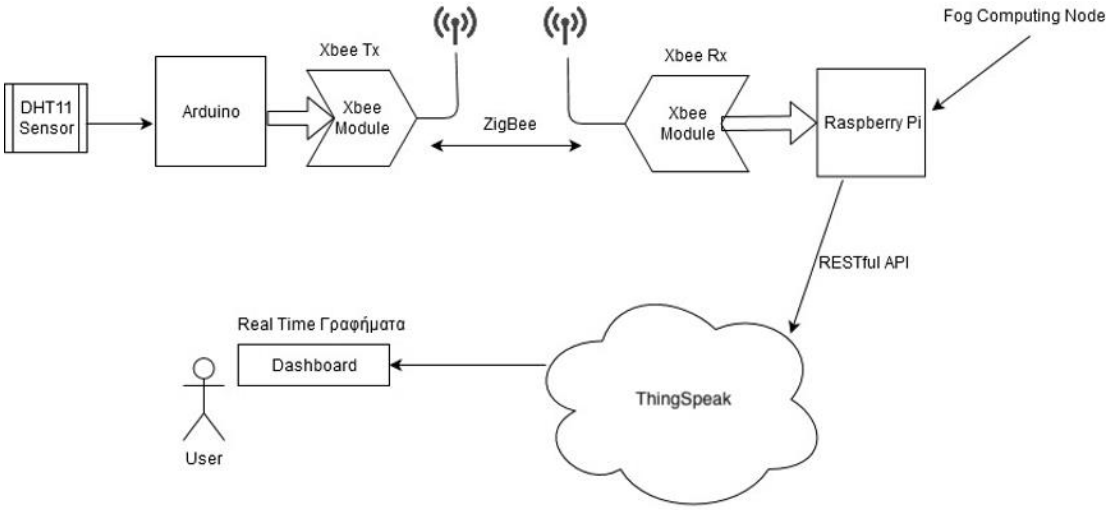
Δεν χρησιμοποιεί τη σύνταξη SQL για τις αναζητήσεις, καθώς δεν ακολουθεί κάποιο αυστηρό σχήμα. Αποτέλεσμα αυτού του γεγονότος είναι να μην μπορεί να εξασφαλιστεί η εξατομίκευση, η συνέπεια, η μοναδικότητα της απάντησης ούτε ακόμα η διάρκειά της.

Κεφάλαιο 4ο

4.1 Αρχιτεκτονική Συστήματος IoT επιτήρησης υγρασίας και θερμοκρασίας

4.1.1 Προδιαγραφές/σχεδιασμός κόμβου fog computing

Στην παρακάτω εικόνα (Εικόνα 11) δίνεται η σχηματική αναπαράσταση του συστήματος που έχουμε υλοποιήσει.



Εικόνα 11: Το σύστημα που υλοποιήθηκε

Τί είναι το Arduino

Το Arduino Board (Εικόνα 13) είναι ένας single-board μικροελεκτής, δηλαδή μια απλή μητρική πλακέτα ανοικτού κώδικα, με ενσωματωμένο μικροελεγκτή και εισόδους/εξόδους. Μπορεί να προγραμματιστεί με τη γλώσσα Wiring. Ουσιαστικά πρόκειται για τη γλώσσα προγραμματισμού C++ και ένα σύνολο από βιβλιοθήκες, υλοποιημένες επίσης στην C++.

Το Arduino μπορεί να χρησιμοποιηθεί για την ανάπτυξη ανεξάρτητων διαδραστικών αντικειμένων αλλά και να συνδεθεί με υπολογιστή μέσω προγραμμάτων. Οι περισσότερες εκδόσεις του Arduino μπορούν να αγοραστούν προ-συναρμολογημένες. Το διάγραμμα και πληροφορίες για το υλικό είναι ελεύθερα διαθέσιμα για αυτούς που θέλουν να συναρμολογήσουν το Arduino μόνοι τους.

Ουσιαστικά πρόκειται για ένα κύκλωμα που χρησιμοποιεί έναν μικροελεγκτή ATmega328 το οποίο μας δίνει ένα αριθμό πυλών οι οποίες μπορεί να λειτουργήσουν είτε ως είσοδοι είτε ως έξοδοι. Αυτές τις εισόδους και εξόδους μπορούμε να τις διαχειριστούμε γράφοντας κώδικα στο περιβάλλον προγραμματισμού Arduino IDE.

Τί είναι το Raspberry Pi

Το Raspberry pi (Εικόνα 18) είναι ένας υπολογιστής σε μέγεθος μίας πιστωτικής κάρτας, η ιδέα του οποίου αναπτύχθηκε και εξελίχθηκε από μία ομάδα εργαζομένων στο πανεπιστήμιο του Cambridge, το 2006, λόγω του μειωμένου ενδιαφέροντος των φοιτητών πληροφορικής και των περιορισμένων γνώσεων τους για αυτούς. Προκειμένου να το καταφέρουν όμως έπρεπε να δημιουργήσουν κάτι το οποίο να είναι οικονομικά προσιτό και εύκολο στην χρήση. Θεωρητικά ένα τέτοιο σύστημα θα βοηθούσε και στην διδασκαλία πληροφορικής στα σχολεία και θα είχε ενδιαφέρον για τους μαθητές.

Η δημιουργία του Raspberry αντιμετώπιζε προβλήματα καθώς υπήρχαν αρκετοί περιορισμοί λόγω του υψηλού κόστους και χαμηλής ισχύος των επεξεργαστών για κινητές/φορητές συσκευές. Δύο χρόνια αργότερα, δημιουργήθηκε το φιλανθρωπικό ίδρυμα Raspberry Pi Foundation και 3 χρόνια αργότερα κυκλοφόρησε το πρώτο Raspberry με τα Model A, Model A+ και Model B.
Τα πρώτα μοντέλα ήταν σχετικά περιορισμένα σε δυνατότητες όμως με την πάροδο του χρόνου έχει φτάσει σήμερα να αποτελείται από έναν επεξεργαστή τύπου ARM Cortex-A53 στα 1200Mhz 1GB Ram και κάρτα γραφικών χρονισμένη στα 250 Mhz.
Γιατί χρησιμοποιούμε το Raspberry;
Το Raspberry όπως προαναφέραμε είναι ένας υπολογιστής ο οποίος όταν τον αγοράσει κάποιος έρχεται ως μία πλακέτα με εξόδους USB, HDMI, ενσωματωμένη κάρτα γραφικών, θύρα ethernet, κάρτα ασύρματου δικτύου και bluetooth. Δεν έχει εγκατεστημένο λειτουργικό σύστημα αλλά δίνεται η δυνατότητα στον χρήστη να επιλέξει από μία λίστα διαθέσιμων δωρεάν λειτουργικών δικτύων το οποία εγκαθίσταται σε μία κάρτα microSD.
Ανάλογα με την χρήση που θέλει να κάνει ο καθένας επιλέγει και το αντίστοιχο λειτουργικό σύστημα που καλύπτει τις ανάγκες του. Μπορεί για παράδειγμα να χρησιμοποιήσει κάποιος το Raspberry ως εξομοιωτή κονσόλας (game emulator), να το χρησιμοποιήσει ως cloud server, torrent downloader, IoT server, web server και πολλά άλλα.

Περιγραφή Xbee

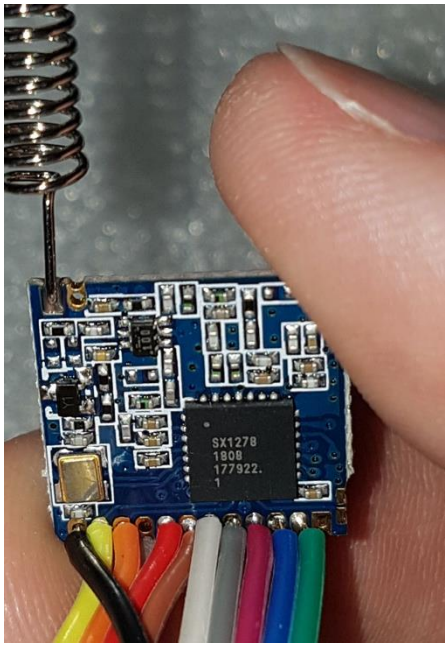
Το Xbee είναι το εμπορικό όνομα μίας κεραίας της εταιρίας Digi International και σχεδιάστηκαν ώστε να πληρούν τα πρωτόκολλο επικοινωνίας Zigbee IEEE 802.15.4-2003 και πραγματοποιούν μετάδοση δεδομένων point-to-point και star communications ασύρματα με ταχύτητες 250kbps.
Αυτή τη στιγμή στην αγορά είναι διαθέσιμοι δύο τύποι Xbee module. Το Xbee το οποίο είναι χαμηλού κόστους και έχει ισχύ 1mW και το Xbee-Pro με ισχύ 100 mW. Μπορούν να χρησιμοποιηθούν κάνοντας χρήση ελαχίστου αριθμού pin/συνδέσεων οι οποίες είναι η τροφοδοσία (3,3 V), ground, data in και data out.

Περιγραφή XCTU

Το XCTU είναι μία δωρεάν εφαρμογή η οποία είναι σχεδιασμένη ώστε να επιτρέπει τον προγραμματισμό και την επικοινωνία μεταξύ των Digi RF modules. Ο λόγος για τον οποίο χρησιμοποιούμε το XCTU στην εργασία μας είναι για τον προγραμματισμό των Xbee modules ώστε να παραμετροποιήσουμε το ένα ως αποστολέα και το άλλο ως αποδέκτη και να παρακολουθήσουμε την μετάδοση των δεδομένων. Επίσης μέσω του XCTU μπορούμε να παρακολουθήσουμε την μετάδοση των πακέτων μεταξύ του coordinator και του receiver καθώς και την ανάγνωσή τους.
Για να γίνουν όλα τα παραπάνω χρειάζεται να προσαρμόσουμε το Xbee module σε έναν «αντάπτορα» και να την συνδέσουμε στον υπολογιστή μέσω μίας θύρας USB.

Για μετάδοση δεδομένων σε μεγαλύτερη απόσταση και εκτός κτηρίου χρησιμοποιείται το Lora η οποία είναι πιο ακριβή λύση και δεν ενδείκνυται για την εφαρμογή μας. Το Lora χρησιμοποιεί συχνότητες μικρότερες του 1GHz, συγκεκριμένα 433MHz, 868MHz (στην Ευρώπη) και 915MHz (Αυστραλία και Β.

Αμερική). Χρησιμοποιώντας αυτό το πρωτόκολλο μπορούν να επιτευχθούν επικοινωνίες και ανταλλαγές δεδομένων και πληροφορίας σε μεγάλες αποστάσεις που σε αγροτικές περιοχές φτάνουν ή υπερβαίνουν τα 10km. Κατά βάση πρόκειται για πρωτόκολλο που αφορά το φυσικό επίπεδο, έχουν όμως εμφανιστεί και



Εικόνα 12: LoRa module

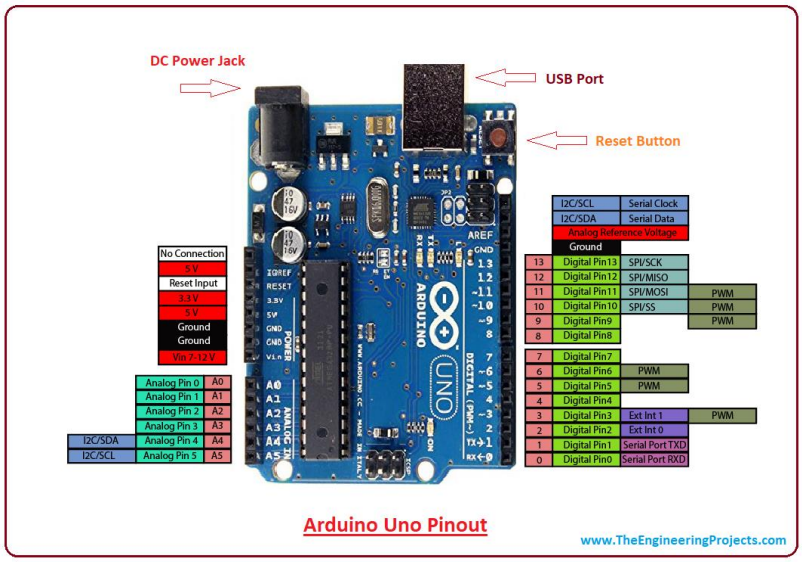
παραλλαγές του όπως το LoRaWAN (Long Range Wide Area Network) (Εικόνα 12) που αφορούν και ανώτερα επίπεδα. Το 2018 ανακοινώθηκαν νέα chipset με μειωμένη κατανάλωση, αυξημένα επίπεδα ισχύος και μειωμένες διαστάσεις. Οι συσκευές LoRa έχουν δυνατότητες γεωεντοπισμού και συχνά χρησιμοποιούνται με χρονοσήμανση από πύλες για τον εντοπισμό των συσκευών με τη μέθοδο του τριγωνισμού.

4.2 Σχεδιασμός εργασίας και υλικά που χρησιμοποιήθηκαν

Η εργασία η οποία υλοποιήθηκε είναι η ασύρματη μετάδοση δεδομένων ενός αισθητήρα υγρασίας και θερμοκρασίας, ο οποίος είναι συνδεδεμένος σε ένα arduino και αποστέλλονται μέσω Xbee module σε ένα Raspberry στο οποίο γίνεται η αποθήκευση των δεδομένων αυτών, σε μία βάση δεδομένων.

Τα υλικά τα οποία χρησιμοποιήθηκαν για την πραγμάτωση της εργασίας είναι τα εξής:

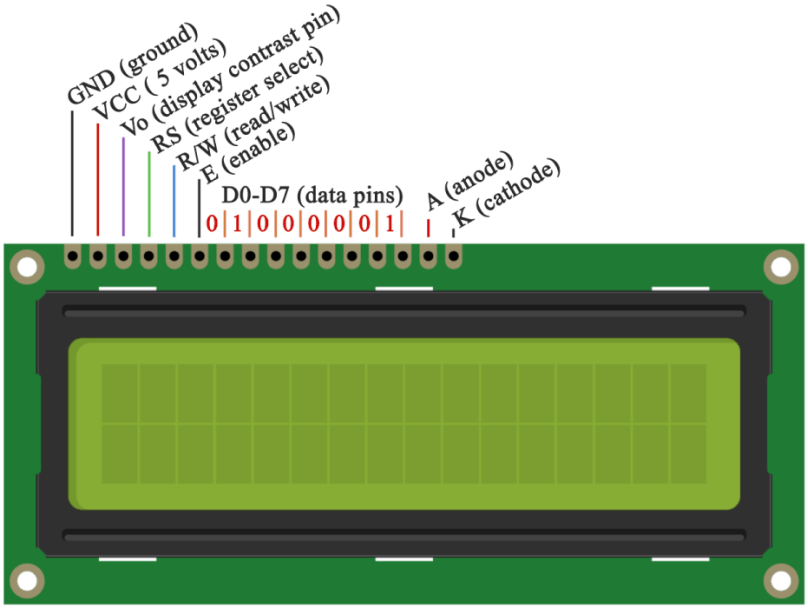
Μία πλακέτα arduino (Εικόνα 13), την οποία συνδέσαμε με ένα breadboard και την προγραμματίσαμε κατάλληλα ώστε να γίνει η επικοινωνία με τον αισθητήρα και η μετάδοση των δεδομένων ασύρματα προς το Raspberry.



Εικόνα 13: Arduino

- Μικροελεγκτής: ATmega328
- Τάση λειτουργίας: 5V
- Τάση εισόδου: 7-12V
- Τάση εισόδου (limits): 6-20V
- Ψηφιακά I/O Pins: 14
- Αναλογικές εισόδους: 6
- PWM εισόδους: 6
- DC Ρεύμα ανά I/O Pin: 20 mA
- DC Ρεύμα για 3.3V Pin: 50mA
- Μνήμη Flash: 32 KB
- Μνήμη SRAM: 2 KB (ATmega328)
- Μνήμη EEPROM: 1 KB (ATmega328)
- Ταχύτητα (Clock Speed): 16 MHz

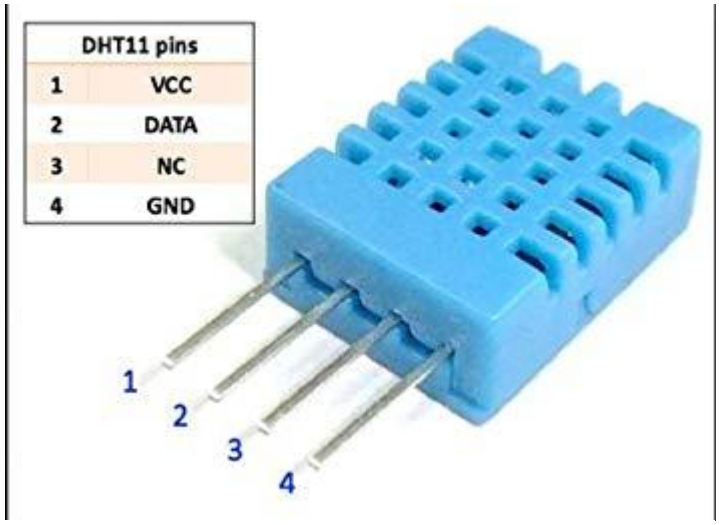
Μία οθόνη LCD 16 χαρακτήρων και 2 γραμμών (Εικόνα 14), η οποία χρησιμοποιήθηκε τοπικά στο arduino ώστε να παρακολουθούμε αν τα δεδομένα



Εικόνα 14: Οθόνη LCD

που μετράει ο αισθητήρας είναι σωστά με αυτά τα οποία αποστέλλονται στο Raspberry

Ένας αισθητήρας θερμοκρασίας και υγρασίας DHT11 (Εικόνα 15). Ο λόγος που έγινε χρήση αυτού του αισθητήρα είναι το χαμηλό κόστος και η δυνατότητα πραγματοποίησης πάρα πολλών μετρήσεων χωρίς να αντιμετωπίζουμε πρόβλημα διάρκειας ζωής όπως έχουμε με χημικούς αισθητήρες στους οποίους ο αριθμός μετρήσεων είναι αρκετά περιορισμένος για την δική μας περίπτωση όπου στα πλαίσια αυτής της διπλωματικής εργασίας χρειάστηκε να γίνουν πάρα πολλές μετρήσεις μέχρι να καταφέρουμε να ολοκληρώσουμε την επικοινωνία μεταξύ του arduino και του raspberry.



Εικόνα 15: Αισθητήρας DHT11

Τύπος Αισθητήρα:

- Θερμοκρασίας
- Υγρασίας

Τυπική Τάση Εισόδου:

- 3VDC
- 3.3VDC
- 5VDC

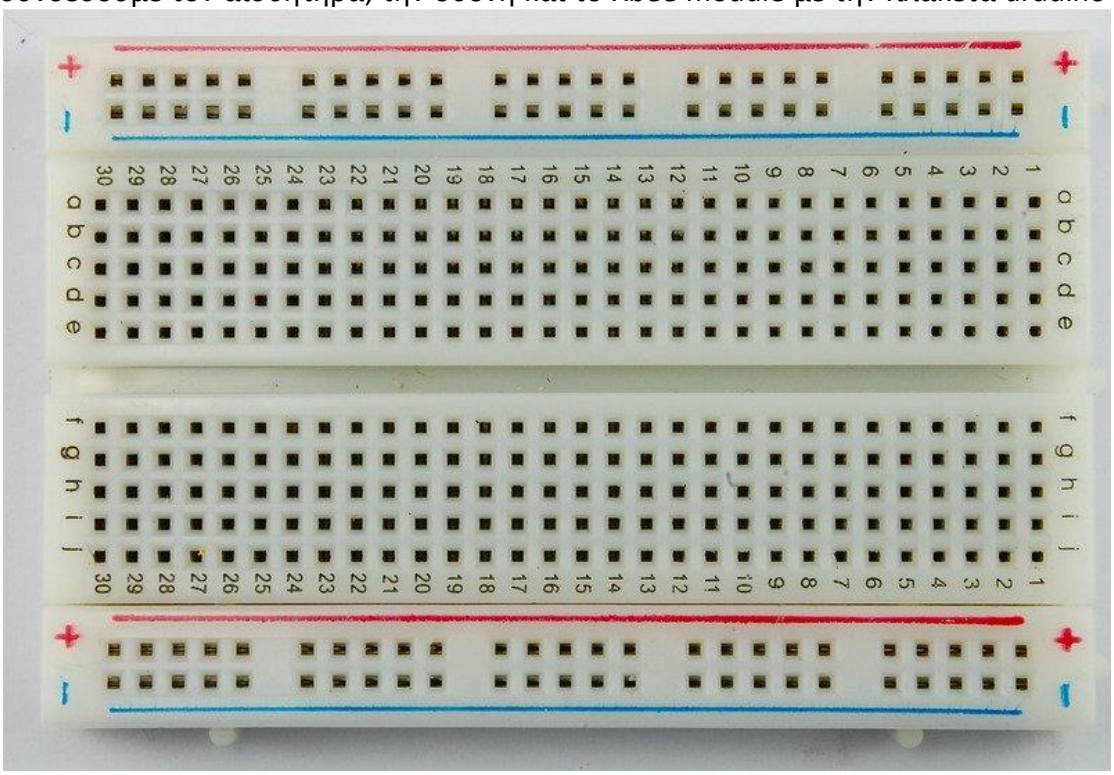
Ρεύμα Λειτουργίας:

- 2.5mA

Διασύνδεση:

- Ψηφιακή

Μία πλακέτα arduino breadboard (Εικόνα 16) η οποία χρειάστηκε για να συνδέσουμε τον αισθητήρα, την οθόνη και το Xbee module με την πλακέτα arduino

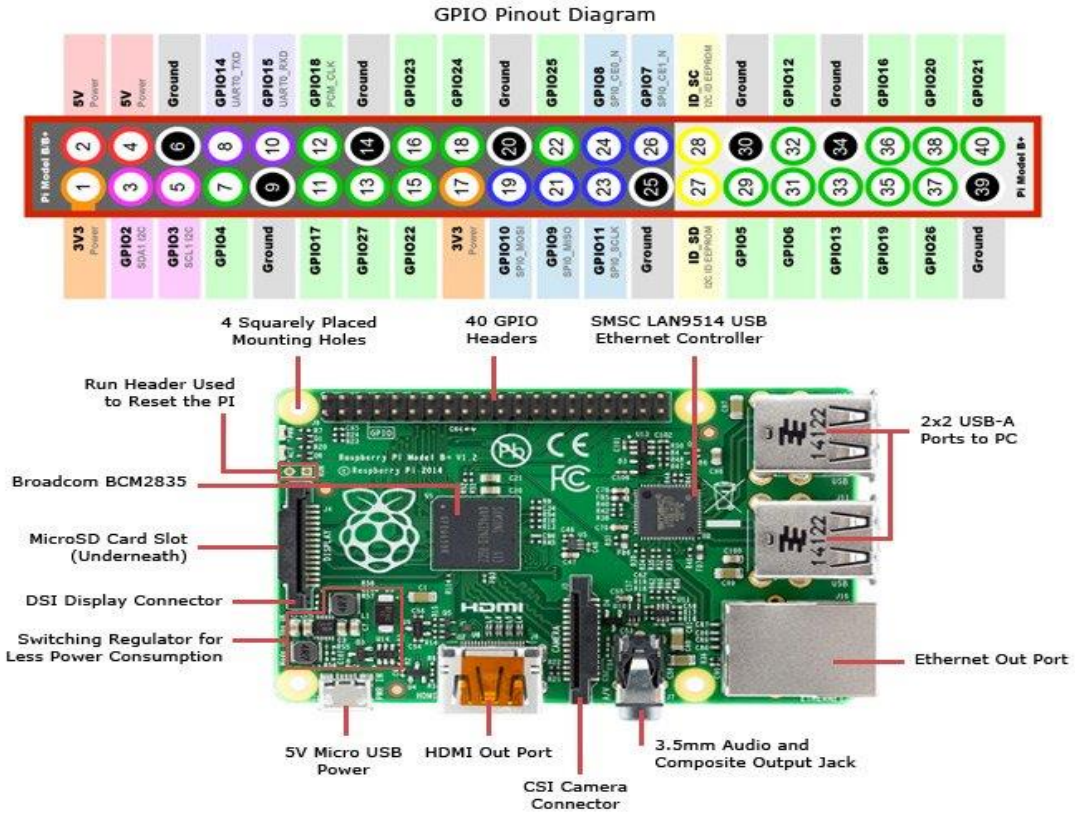


Εικόνα 16: Arduino breadboard

Ένα ποτενσιόμετρο (Εικόνα 17) αντίστασης 10K Ohm

Εικόνα 17: Ποτενσιόμετρο

Μία πλακέτα Raspberry Pi 3 (Εικόνα 18) στην οποία έγινε εγκατάσταση του λογισμικού Raspbian, η δημιουργία της βάσης δεδομένων και η σύνδεση του Xbee module Rx.

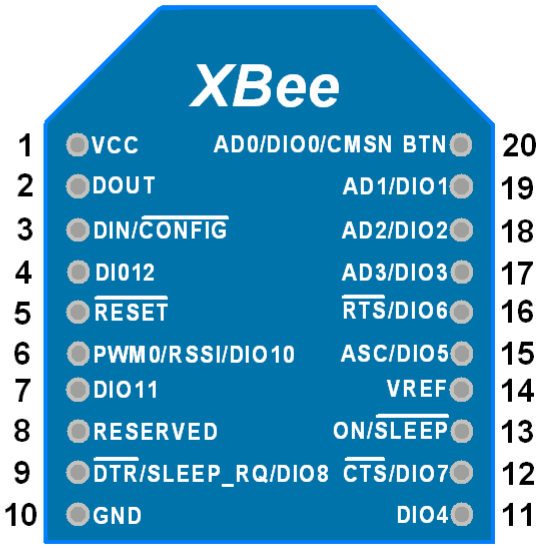


Εικόνα 18: Raspberry Pi3

- Broadcom BCM2837B0, Cortex-A53, 64-bit, quad-core SoC @ 1.4GHz
- 1GB LPDDR2 SDRAM
- 2.4GHz/5GHz IEEE 802.11 b/g/n/ac Wireless LAN (WLAN)
- Bluetooth Low Energy v4.2 (BLE)
- Gigabit Ethernet over USB 2.0 (maximum throughput 300Mbps)
- 4 x USB 2.0 Ports
- Extended 40-pin GPIO Header
- Full Size HDMI, MIPI DSI display port, MIPI CSI camera port
- 4-pole stereo audio/composite video output port
- MicroSD card slot for operating system and data storage
- Power over Ethernet (PoE) enabled (requires separate PoE HAT)
- Power supply requirements - 5V/2.5A DC via micro USB or GPIO

Δύο Xbee module (Εικόνα 19) και 2 Xbee usb programming module (Εικόνα 20). Τα Xbee module χρησιμοποιήθηκαν για να γίνει η αποστολή και η λήψη των

XBee Module



XBee Pin Configuration

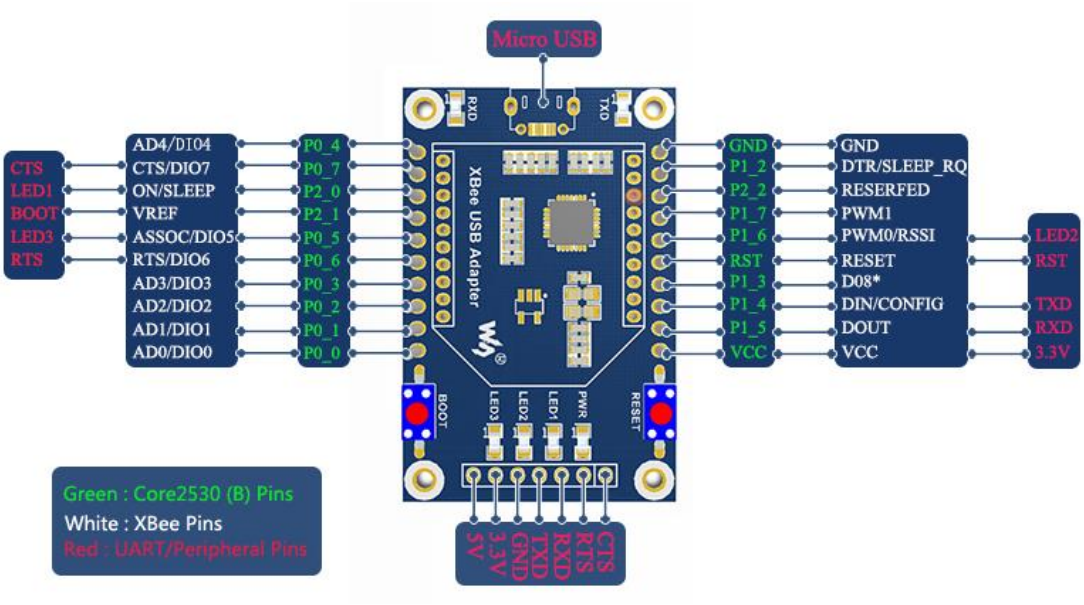
δεδομένων και τα Xbee USB programming module χρειάστηκαν ώστε να συνδεθούν τα modules με τα arduino και raspberry αντίστοιχα καθώς και να γίνει ο προγραμματισμός και ο συγχρονισμός μεταξύ τους μεσω του προγράμματος XCTU

Εικόνα 19: XBee Module

Χαρακτηριστικά[2]:

- 3.3V @ 40mA
- 250kbps Max data rate
- 2mW output (+3dBm)
- 400ft (120m) range
- Built-in antenna
- Fully FCC certified
- 6 10-bit ADC input pins
- 8 digital IO pins
- 128-bit encryption
- Local or over-air configuration
- AT or API command set

[2] https://grobotronics.com/xbee-2mw-wire-antenna-series-2.html



Εικόνα 20: XBee USB programming module

Κεφάλαιο 5ο

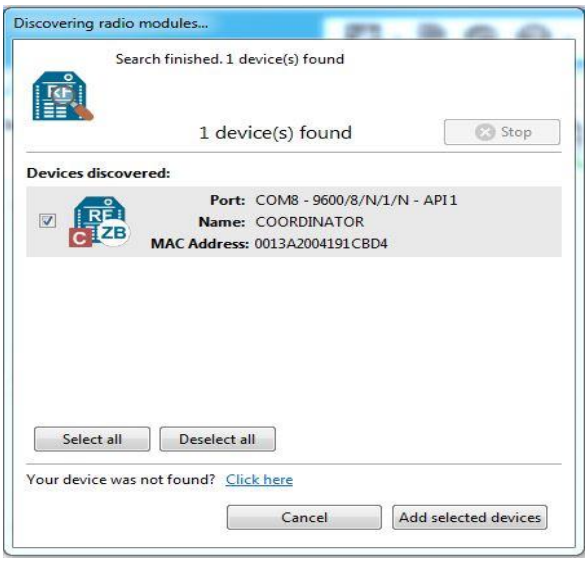
5.1 Προγραμματισμός fog computing με IoT πλατφόρμα

Για να στείλουμε δεδομένα στο cloud απαιτείται να έχουμε ολοκληρώσει τα εξής βήματα:

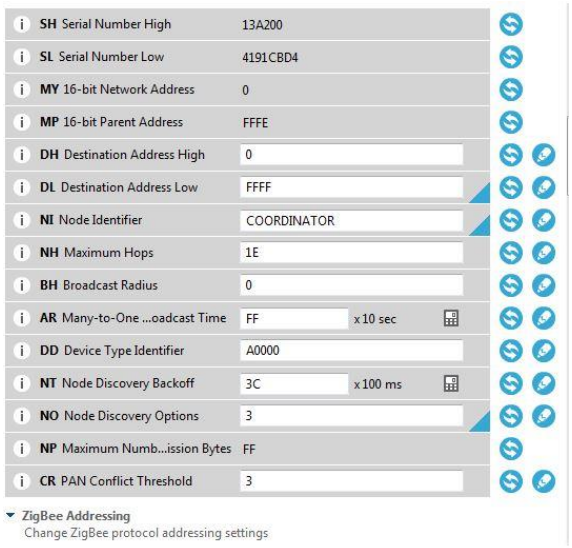
5.1.1 Σχεδιασμός μεταφοράς δεδομένων από Xbee στη βάση δεδομένων

Για να μεταφέρουμε τα δεδομένα από το Xbee στην βάση δεδομένων θα πρέπει πέρα από τον αντίστοιχο κώδικα στα arduino και raspberry, να προγραμματίσουμε τα Xbee μέσω του XCTU και να ορίσουμε το ένα module ως αποστολέα (Tx) και το άλλο ως αποδέκτη (Rx). Στην εργασία μας έχουμε έναν αποστολέα και έναν αποδέκτη αλλά αντίστοιχα μπορούν να προστεθούν και άλλοι κόμβοι αποστολείς και ο αποδέκτης να συλλέγει τα δεδομένα από όλους και να εκτελεί τις περαιτέρω ενέργειες.

Για να γίνει ο προγραμματισμός των Xbee module, αφού έχουμε εγκαταστήσει το XCTU, συνδέουμε μέσω USB και παραμετροποιούμε το κάθε module ξεχωριστά βάζοντας τις κατάλληλες τιμές στα αντίστοιχα πεδία (Εικόνα 22 και Εικόνα 21)

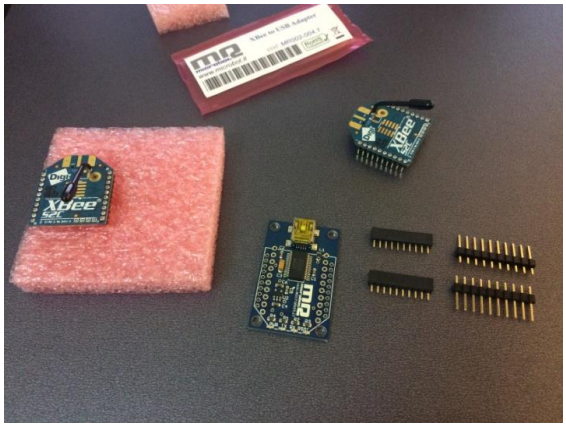


Εικόνα 22: Προγραμματιστικό βήμα 1

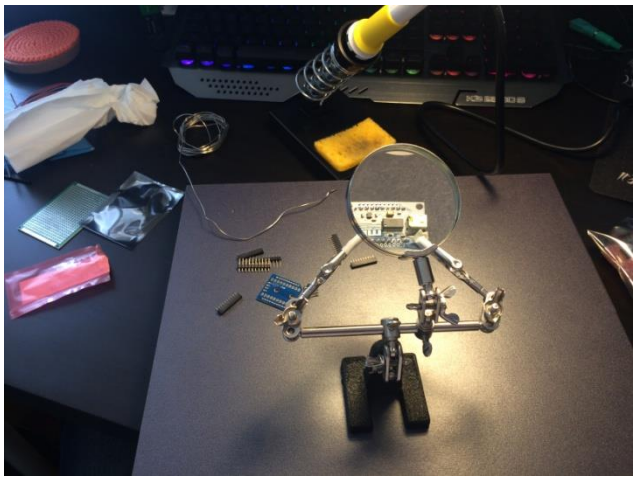


Εικόνα 21: Προγραμματιστικό βήμα 2

Σημείωση: Για να μπορέσουμε να συνδέσουμε τα Xbee module απαιτείται να έχουμε ένα usb module πάνω στο οποίο θα προσαρτήσουμε το Xbee module ώστε να μπορέσει να προγραμματιστεί. Το module θα πρέπει να το "κατασκευάσουμε" (εικόνα 21) μιας και έρχεται σε κομμάτια (Εικόνα 23)



Εικόνα 24: Υλικά Xbee Tx (Rx)

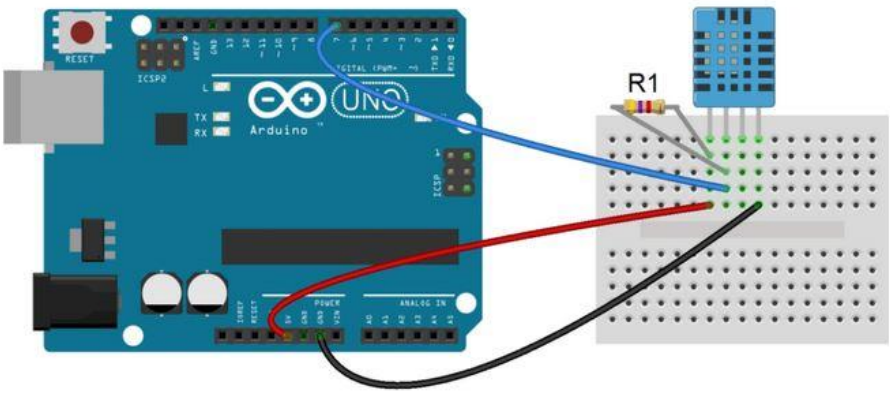


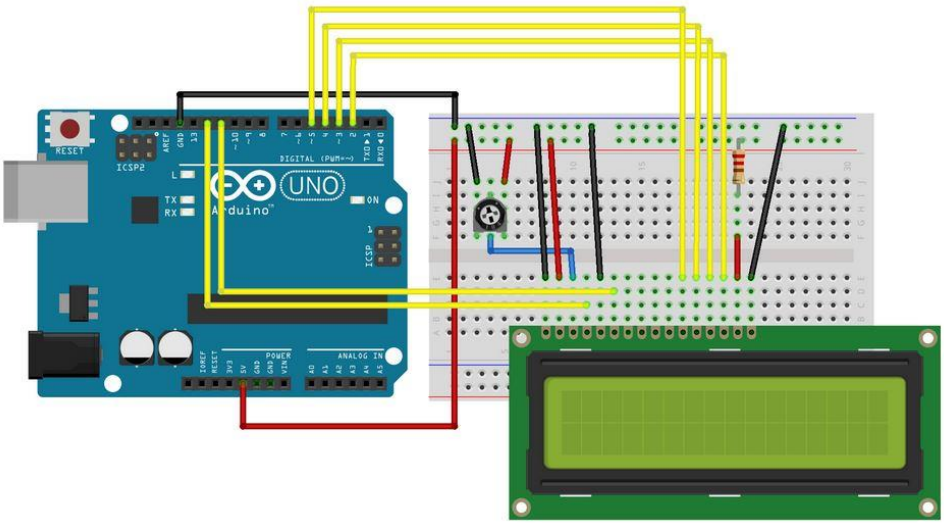
Εικόνα 23: Συναρμολόγηση Xbee Tx (Rx)

5.2 Υλοποίηση κόμβου μετάδοσης μετρήσεων

1. Συνδέουμε το arduino μέσω USB στον υπολογιστή μας ώστε να αποκτήσει τροφοδοσία και να μπορέσουμε να το προγραμματίσουμε.
2. Συνδέουμε στο breadboard την οθόνη και τον αισθητήρα DHT11 και με βάση τα παραπάνω διαγράμματα κάνουμε την κατάλληλη συνδεσμολογία μεταξύ breadboard και arduino ώστε να τροφοδοτήσουμε τα εξαρτήματα και να κάνουμε μετάδοση σήματος. Συγκεκριμένα η συνδεσμολογία είναι η εξής:
   - DHT11 Pin 1 στο 5V Power pin
   - DHT11 Pin 2 στο Digital pin 7
   - DHT11 Pin 4 στο GND Power pin

   Ανάμεσα στα Pin 1 και 2 του αισθητήρα παρεμβάλουμε μία αντίσταση 10K ohm.





Εικόνα 25: Συνδεσμολογία

Η Συνδεσμολογία της οθόνης γίνεται ως εξής (Εικόνα 25):

- LCD VDD και A pin στο 5V Power pin
- LCD VSS, RW και K στο GND pin
- LCD RS pin to digital pin 12
- LCD Enable pin στο digital pin 11
- LCD D4 pin στο digital pin 5
- LCD D5 pin στο digital pin 4
- LCD D6 pin στο digital pin 3
- LCD D7 pin στο digital pin 2
- LCD V0 pin στο μεσαίο pin του ποτενσιόμετρου
- Pin 1 του ποτενσιόμετρου με το 5V Power pin
- Pin 3 του ποτενσιόμετρου με το GND Power pin
- 10K Ohm αντίσταση μεταξύ του LCD Pin A και του 5V Power pin

3. Αφού γίνει η συνδεσμολογία σωστά, ανοίγουμε στον υπολογιστή την εφαρμογή ArduinoIDE και γράφουμε τον κώδικα (παράρτημα Α) για να διαβάσουμε τα δεδομένα από τον αισθητήρα και να τα προβάλουμε στην οθόνη LCD.
4. Αποσυνδέουμε το arduino και κάνουμε εγκατάσταση το πρόγραμμα XCTU.
5. Συνδεσμολογία Xbee Tx trasmitter. Ο transmitter Xbee αποτελείται από την κάθετη σύνδεση ενός Xbee module (Εικόνα 19) με το Xbee USB programming module (Εικόνα 20)
6. Συνδεσμολογία Xbee Rx Receiver. Ο Receiver Xbee αποτελείται από την κάθετη σύνδεση ενός άλλου Xbee module (Εικόνα 19) και ενός άλλου αντίστοιχου Xbee USB programming module (Εικόνα 20)
7. Συνδέουμε μέσω usb εναλλάξ το Xbee Tx και το Xbee Rx στον υπολογιστή και τα προγραματίζουμε έτσι ώστε το Xbee Tx να εκτελεί διαδικασίες αποστολέα και το Xbee Rx διαδικασία παραλήπτη.

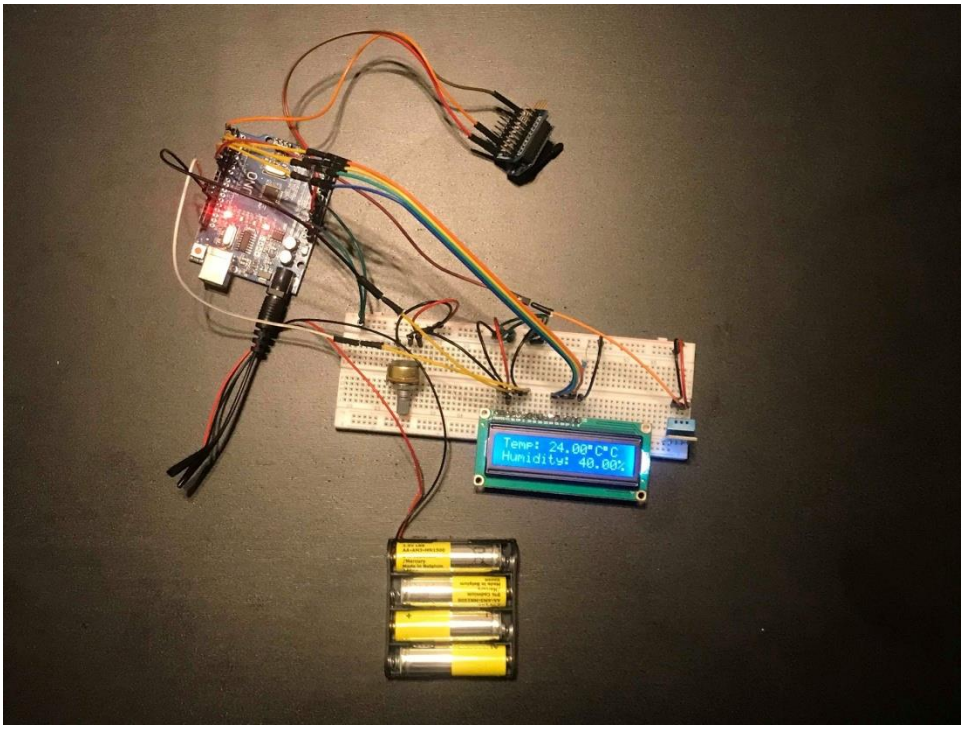
8. Για την σύνδεση του Xbee Tx "αποστολέα" στο arduino, συνδέουμε το pin "DIN" (pin 5 εικόνα 18) στο Tx ➔ 1 pin του arduino και τα pin 1 και 10 με τα 3.3V και GND αντίστοιχα.

Στο σημείο αυτό το αποτέλεσμα θα είναι ένα πλήρως λειτουργικό σύστημα μέτρησης θερμοκρασίας και υγράσιας τα οποία προβάλλονται στην οθόνη και αποστέλλεται μέσω του Xbee Tx.

Στη φωτογραφία (Εικόνα 26) φαίνεται τον εν λόγω σύστημα, η τροφοδοσία του οποίου γίνεται μέσω μπαταριών για την επίτευξη φορητότητας.

Ο κώδικας με τον οποίο προγραμματίσαμε το arduino δίνεται στο παράρτημα Α.

Συγκεκριμένα με τον κώδικα αυτό λαμβάνουμε τα δεδομένα του αισθητήρα DHT11 μέσω του PIN 7 ψηφιακής εισόδου του arduino, το τυπώνουμε στην οθόνη υγρών κρυστάλλων και στην σειριακή έξοδο του arduino. Το Xbee Tx λαμβάνει τα δεδομένα που τυπώνουμε στην σειριακή μέσω της εντολής "serial.print" και τα αποστέλλει.



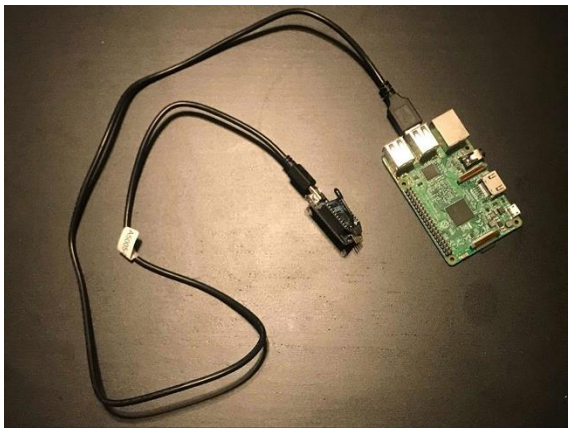
Εικόνα 26: Σύστημα αισθητήρα και υπολογιστή

5.3 Υλοποίηση κόμβου ομιχλώδους Υπολογισμού

1. Κατεβάζουμε στον υπολογιστή μας το λογισμικό Raspbian[3] το οποίο θα εγκαταστήσουμε στο Raspberry και το αποθηκεύουμε σε μία κάρτα microSD.

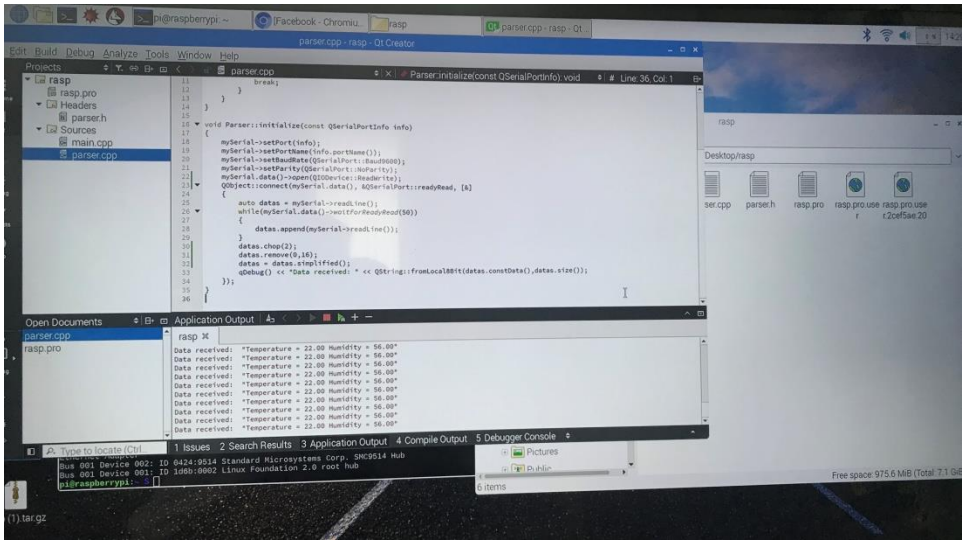
[3]https://thepi.io/how-to-install-raspbian-on-the-raspberry-pi/

2. Τοποθετούμε την κάρτα SD στην πλακέτα Raspberry και της δίνουμε τροφοδοσία και σύνδεση με monitor. Το raspberry τροδοτείται μέσω θύρας usb και ιδανικά το συνδέουμε σε μία θύρα USB μίας smart TV (όπως στην περίπτωσή μας) και με ένα καλώδιο HDMI οπότε είναι έτοιμο να λειτουργήσει.
3. Από το site που κατεβάσαμε το λειτουργικό ακολουθούμε τα βήματα για την εγκατάσταση του
4. Αφού ολοκληρώσουμε την εγκατάσταση του λειτουργικού συστήματος, εγκαθιστούμε την βάση δεδομένων την οποία πρόκειται να χρησιμοποιήσουμε, η οποία στην περίπτωσή μας είναι η MariaDB.
5. Συνδέουμε στο Raspberry τον παραλήπτη Xbee Rx που προγραμματίσαμε στο βήμα 6 και βήμα 7 της προηγούμενης παραγράφου (Εικόνα 27).

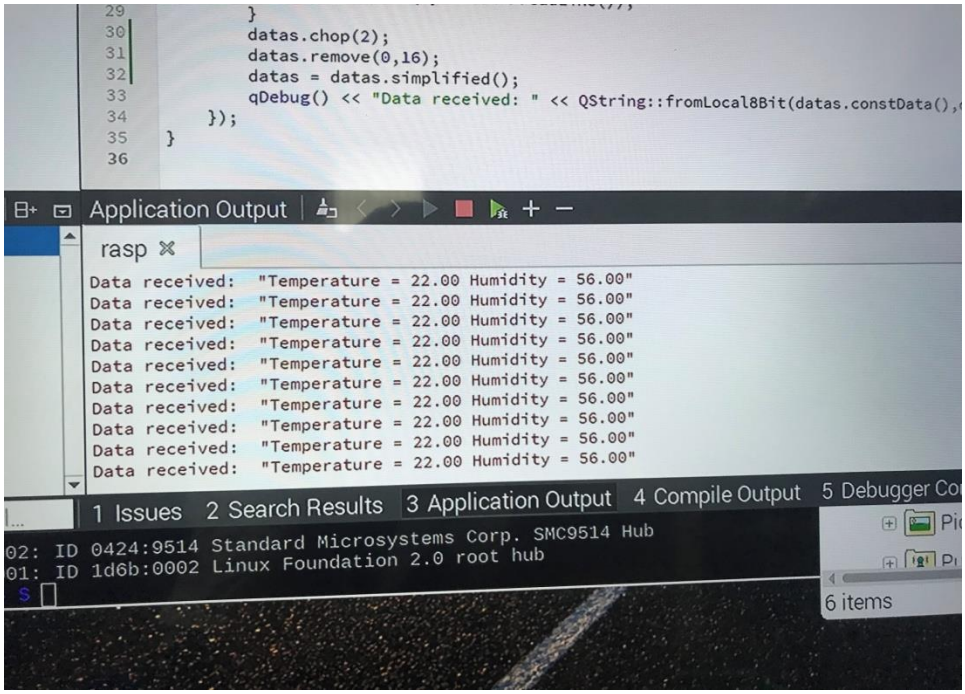


Εικόνα 27: Συνδεσμολογία Raspberry Pi και Xbee Rx μέσω USB

6. Εγκαθιστούμε το QT Creator (προγραμματιστικό περιβάλλον) (Εικόνα 28 και Εικόνα 29) βάσει του οποίου αναπτύξαμε τον κώδικα του παραρτήματος Β. Ο κώδικας C++ που αναπτύξαμε υλοποιεί την επεξεργασία των δεδομένων που λαμβάνουμε από το Xbee Rx μέσω του USB στο raspberry και εκτελείται πάνω στο raspberry Pi. Επισημαίνουμε ότι τα δεδομέναν έχουν σταλλεί από τον κόμβο της προηγούμενης παραγράφου και μέσω του προγράμματος cpp που αναπτύξαμε και εκτελείται πάνω στο raspberry Pi, αποθηκεύονται σε βάση δεδομένων SQL (MariaDB) και μεταδίδονται μέσω της προγραμματιστικής διεπαφής Http RESTful API στην πλατφόρμα IoT ThingSpeak.

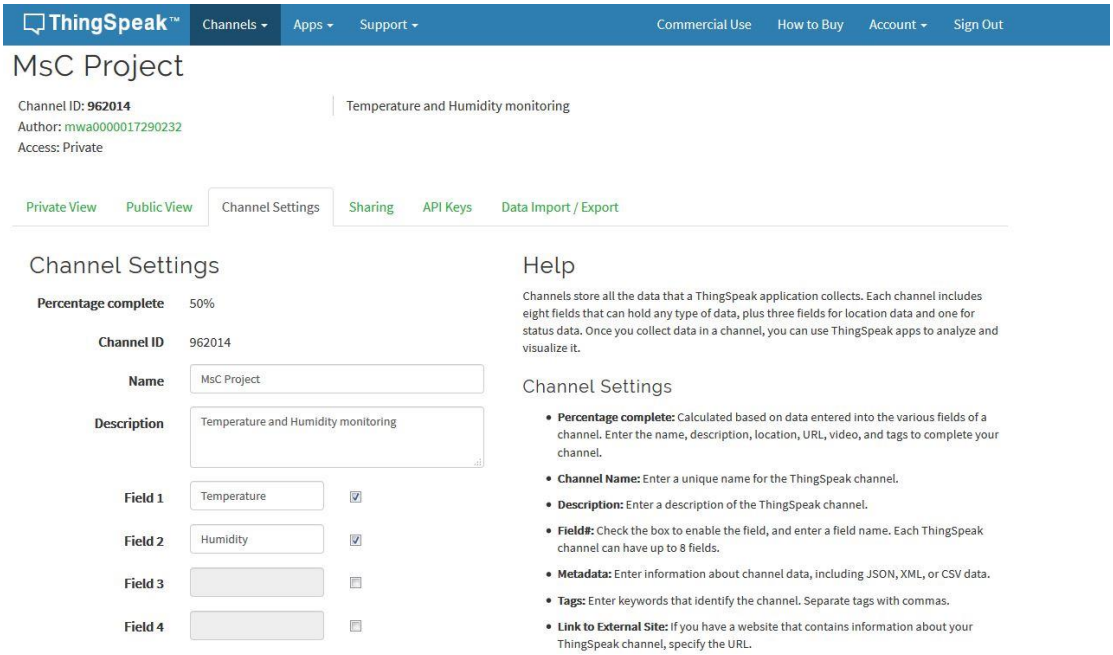


Εικόνα 28: Προγραμματισμός με QT Creator



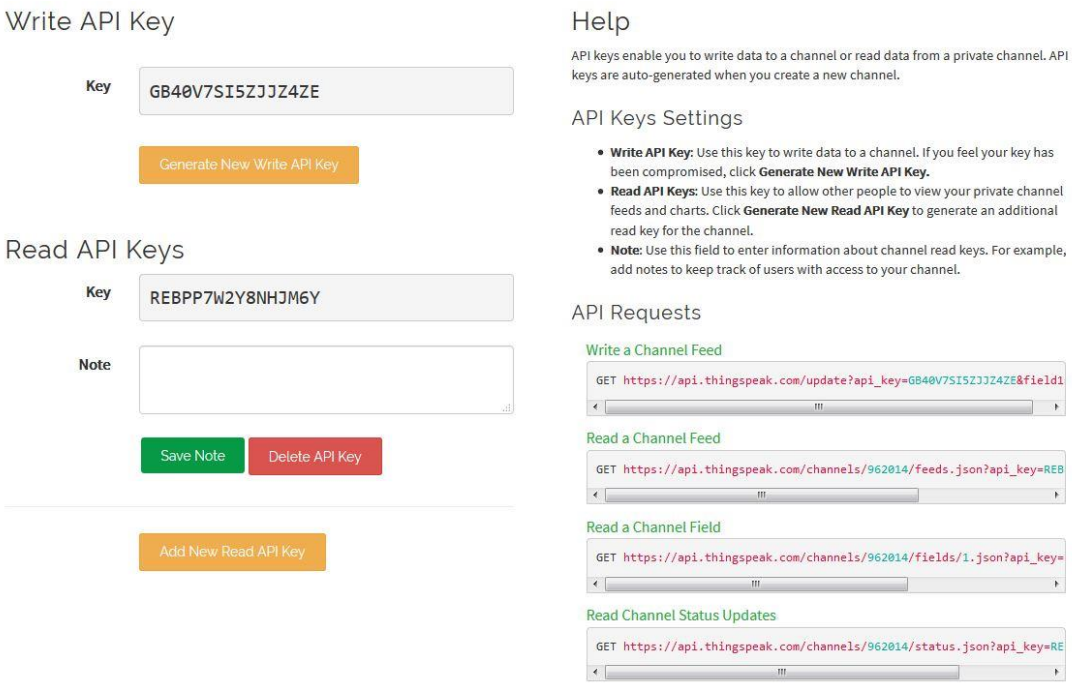
Εικόνα 29: QT Creator

7. Δημιουργούμε λογαριασμό στην πλατφόρμα ThingSpeak (Εικόνα 30) και φτιάχνουμε ένα κανάλι στο οποίο εισάγουμε τα πεδία για τις τιμές που μας ενδιαφέρουν.



Εικόνα 30: Δημιουργία καναλιού ThingSpeak

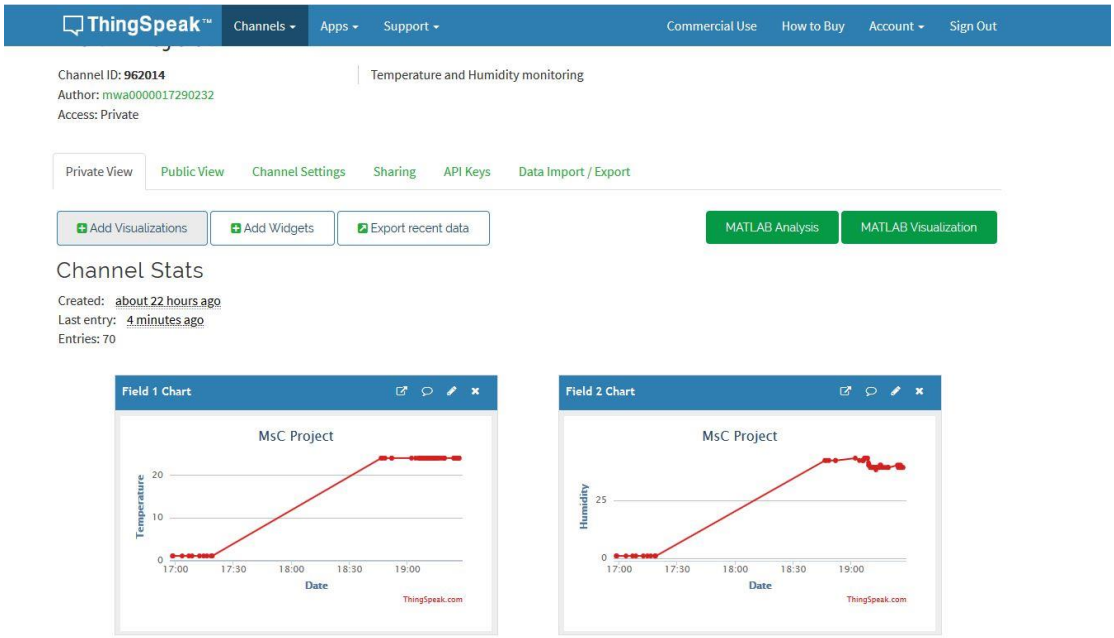
8. Μετά την δημιουργία του καναλιού δημιουργούνται αυτόματα τα API για εγγραφή και ανάγνωση (Εικόνα 31).



Εικόνα 31: API εγγραφής - ανάγνωσης

9. Γράφουμε στον QT Creator τον κώδικα για την λήψη δεδομένων από το arduino, την αποθήκευσή τους στην βάση δεδομένων που έχουμε δημιουργήσει και κάνοντας χρήση του Write API τα στέλνουμε στο cloud.

   Στο σημείο αυτό όταν τρέχουμε τον κώδικα που έχουμε γράψει, έχουμε μετάδοση δεδομένων από το arduino στο raspberry και την αποστολή αυτών στο cloud όπου γίνεται η προβολή τους και τα γραφήματα αυτών (Εικόνα 32).



Εικόνα 32: Αποστολή - Αναπαράσταση δεδομένων/υγρασίας θερμοκρασίας στην πλατφόρμα ThingSpeak

Συμπεράσματα

Με την παρούσα διπλωματική εργασία αναπτύξαμε ένα πρότυπο σύστημα που μετράει την υγρασία και την θερμοκρασία του περιβάλλοντος και αποστέλλει τα μετρούμενα δεδομένα σε μία πλατφόρμα IoT που βρίσκεται στο cloud. Αναφέρθηκαν όλα τα κατασκευαστικά βήματα που φαίνονται και στις αντίστοιχες εικόνες, καθώς και όλα τα προγραμματιστικά βήματα που φαίνονται στα σχετικά παραρτήματα.

Δεδομένου ότι οι συσκευές που συνδέονται στο διαδίκτυο αυξάνονται εκθετικά με το χρόνο οπότε για λόγους βέλτιστης μετάδοσης των δεδομένων και ελαχιστοποίησης κατανάλωσης πόρων και ενέργειας βελτιστοποιούμε στην εφαρμογή την επεξεργασία που λαμβάνει χώρα στα διάφορα στάδια του δικτύου(cloud,fog,edge και device)

Για την υλοποίηση χρησιμοποιήσαμε χαμηλού κόστους αισθητήρες, ανοιχτού λογισμικού κώδικα και ανοιχτού υλισμικού συσκευές. Ενώ για την πλατφόρμα χρησιμοποιήσαμε την εμπορική πλατφόρμα IoT ThingSpeak αξιοποιώντας τις δωρεάν παροχές που προσφέρει. Με τον τρόπο αυτό ο τελικός χρήστης μπορεί σε πραγματικό χρόνο να έχει την εποπτεία των περιβαλλοντολογικών συνθηκών σε ένα κτήριο. Η προτεινόμενη αρχιτεκτονική είναι επεκτάσιμη με την έννοια ότι μπορούμε να συνδέσουμε επιπλέον μετρητικούς κόμβους μέσα σε ένα κτήριο και να λαμβάνουμε τα δεδομένα τους στον κόμβο ομιχλώδους υπολογισμού. Αυτό βέβαια προϋποθέτει την επιπλέον παραμετροποίηση του ασύρματου δικτύου ZigBee.

Η βάση δεδομένων MariaDB χρησιμοποιήθηκε για λόγους εκσφαλμάτωσης του συστήματος και την επαλήθευση ότι λαμβάνουμε σωστά τα δεδομένα μέσω των Xbee. Σε μελλοντική χρήση η βάση δεδομένων δύναται να χρησιμοποιηθεί για τη διατήρηση των δεδομένων και την αποστολή τους στο νέφος σε περίπτωση που διακοπεί η σύνδεση με το διαδίκτυο. Με αυτόν τον τρόπο το fog computing μεταπίπτει σε edge computing δεδομένου ότι εκτελούνται επεξεργαστικές διαδικασίες.

Παράρτημα Α.

Κώδικας Arduino για λήψη δεδομένων από αισθητήρα:

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

LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

dht DHT;

#define DHT11_PIN 7

void setup(){
  lcd.begin(16, 2);
  Serial.begin(9600);
}

void loop()
{
  int chk = DHT.read11(DHT11_PIN);
  lcd.setCursor(0,0);
  lcd.print("Temp: ");
  lcd.print(DHT.temperature);
  lcd.print((char)223);
  lcd.print("C");
  lcd.setCursor(0,1);
  lcd.print("Humidity: ");
  lcd.print(DHT.humidity);
  lcd.print("%");
  Serial.print("Temperature = ");
  Serial.println(DHT.temperature);
  Serial.print("Humidity = ");
  Serial.println(DHT.humidity);
  delay(10000);
}
```

Παράρτημα Β

Προγραμματισμός κόμβου ομιχλώδους υπολογισμού

Main

```
#include <QCoreApplication>
#include "parser.h"

int main(int argc, char *argv[])
{
    QCoreApplication a(argc, argv);

    //ScopedPointer pou dimiourgei ena neo pointer dunamika
    //pros tin klasi Parser() kanontas tin initialize.
    QScopedPointer<Parser> myParser(new Parser());
    return a.exec();
}
```

parser.cpp

```
#include "parser.h"
Parser::Parser(QObject *parent) : QObject(parent),
    mySerial (new QSerialPort),
    myDBConnector (new DBConnector(this)),
    myCloudManager (new CloudManager(this))
{
#ifdef DEBUG
    qDebug() << "Initializing parser...";
#endif

    //Metavliti pou dilwnei an vrethike to device pou psaxnoume.
    //Tha ti xreiastoume parakatw gia na proxwrisoume me ton kwdika.
    //An de vrethei to device, tote to programma termatizetai.
    bool bDeviceFound = false;

    //H klasi QSerialPortInfo mas dinei plirofories gia ta serial
    //devices pou einai sundedemena panw sto mixanima.
    QSerialPortInfo *tempPort = new QSerialPortInfo();

    //Gia kathe serial device, kanoume elegxo na doume an vrethike
    //auto pou psaxnoume.
    foreach(*tempPort, QSerialPortInfo::availablePorts())
    {
        //Periexei to device mas to string "UART" sto description? An
        //nai, tote proxwrame kai to kanoume initialize. Kathe device
        //exei diko tou anagnwristiko, emeis kseroume oti to diko mas
periexei
        //mesa to string "UART".
        if(tempPort-
>description().contains("UART",Qt::CaseInsensitive))
        {
            //An de mporesoume na kanoume initialize to device, to
programma
            //termatizetai.
            if(!initialize(tempPort))
            {
#ifdef DEBUG
```

```
                qDebug() << Q_FUNC_INFO << ": failed to initialize
device.";
#endif
                exit(1);
            }

            //To device vrethike, egine initialized, opote den
psaxnoume gia alla
            //serial devices.
            bDeviceFound = true;
            break;
        }
    }

    //Kanoume delete to tempPort to opoio ftiaksame gia na vroume to
diko mas
    //serial
    delete(tempPort);

    //An de vrethike to device pou psaxname, to programma
termatizetai.
    if(!bDeviceFound)
    {
#ifdef DEBUG
        qDebug() << Q_FUNC_INFO << ": antenna not found, exiting
now.";
#endif
        exit(1);
    }

    //Edw exoume QObject::connect me lamda SLOT. Kathe fora pou to
serial
    //device dinei ena readyRead signal, tote mpainoume mesa sto
lamda.
    QObject::connect(mySerial.data(), &QSerialPort::readyRead,this,
[&]
    {
        //Apothikeuoume ta data apo to device stin metavliti data.
        auto data = mySerial.data()->readLine();
        //Perimenoume 50msec mexri to device na mas dwsei ola ta
        //data, kai ta prosthetoume sti metavliti mas. An ta data
        //htan perissotera, tha eprepe na auksisoume to
waitForReadyRead().
        while(mySerial.data()->waitForReadyRead(50))
        {
            data.append(mySerial.data()->readLine());
        }
        //Kovoume tous prwtous duo xaraktires apo t o string, kai sti
sunexeia
        //afairoume osi pliroforia de xreiazomaste, mexri to simeio
pou ksekina to
        //"Temperature", wste na to kanoume parse.
        data.chop(2);
        data.remove(0,data.indexOf("Temperature",0));
        data = data.simplified();
        //Pername to string pou mas edwse to device stin extractor().

extractor(QString::fromLocal8Bit(data.constData(),data.size()));
    });
}
```

```
Parser::~Parser()
{
    //Kleinoume to serial device ston destructor.
    if(mySerial.data()->isOpen())
    {
        mySerial.data()->close();
    }
}

void Parser::extractor(const QString temp) const
{
    //Ftiaxnoume mia lista opo tha valoume tis duo times,
    //ugrasia kai thermokrasia.
    QList<float> result;

    //Edw exoume ena regexp to opoio tha mas epistrepsei ta duo
    //floats pou periexei to string mas.
    QRegularExpression reg("[+-]?\\d*\\.?\\d+");
    QRegularExpressionMatchIterator i = reg.globalMatch(temp);

    //To kathe "i" krataei kai mia timi apo autes pou vrike sto
string.
    //H prwti timi pou tha dwsei einai h thermokrasia, i deuteri h
ugrasia.
    while (i.hasNext())
    {
        QRegularExpressionMatch match = i.next();
        //Pername tin timi pou vrikame sti lista mas.
        result << match.captured().toFloat();
    }

    //Kanoume elegxo an to string mas edwse 2 times. An oxi, dinoume
katallilo
    //minima se DEBUG mode, alliws to prospername.
    if(result.length()!=2)
    {
#ifdef DEBUG
        qDebug() << Q_FUNC_INFO << ": failed to parse data from
antenna.";
#endif
    }
    else
    {
        //An vrikame swsta 2 times, tote pername ta apotelesmata ston
        //cloudmanager kai ston dbconnector.
        myCloudManager.data()->uploadData(&result[0],&result[1]);
        myDBConnector.data()->addDataToDB(&result[0],&result[1]);
    }
}

bool Parser::initialize(const QSerialPortInfo *info) const
{
    //Kanoume initialize to serial device me tis plirofories
    //pou mas dinei o kataskeuastis.
    mySerial.data()->setPort(*info);
    mySerial.data()->setPortName(info->portName());
    mySerial.data()->setBaudRate(QSerialPort::Baud9600);
    mySerial.data()->setParity(QSerialPort::NoParity);

    return mySerial.data()->open(QIODevice::ReadOnly);
```

```
}
```

parser.h

```
#ifndef PARSER_H
#define PARSER_H

#include <QObject>
#include <QtSerialPort/QtSerialPort>
#include <QtSerialPort/QSerialPortInfo>
#include <QRegularExpression>
#include <QRegularExpressionMatchIterator>
#include <QThread>

#include "dbconnector.h"
#include "cloudmanager.h"

class Parser : public QObject
{
    Q_OBJECT

public:
    explicit Parser(QObject *parent = nullptr);
    ~Parser();

private:
    bool initialize(const QSerialPortInfo *info) const;
    void extractor(const QString temp) const;

    QScopedPointer<QSerialPort> mySerial;
    QScopedPointer<DBConnector> myDBConnector;
    QScopedPointer<CloudManager> myCloudManager;
};

#endif // PARSER_H
```

dbconnector.cpp

```
#include "dbconnector.h"
DBConnector::DBConnector(QObject *parent) : QObject(parent),
    myDB            (new
QSqlDatabase(QSqlDatabase::addDatabase(m_qsDatabase))),
    myQuery         (new QSqlQuery),
    myCurrentDate   (new QDateTime)
{
#ifdef DEBUG
    qDebug() << "Connecting to Database...";
#endif

    //Edw kanoume initialize tin SQL Database me tis times
    //pou exoume orisei sto header.
    myDB.data()->setDatabaseName(m_qsDBName);
    myDB.data()->setHostName    (m_qsHostName);
    myDB.data()->setUserName    (m_qsUserName);
    myDB.data()->setPassword    (m_qsPassword);
    myDB.data()->setPort        (m_iPort);
```

```
    //An den mporesei o kwdikas na sundethei stin DB,
    //tote to software termatizetai.
    if(myDB.data()->open())
    {
#ifdef DEBUG
        qDebug() << "Connection established.";
#endif
    }
    else
    {
#ifdef DEBUG
        qDebug() << "Connection failed, check DB settings again
and/or database installation. Exiting now.";
#endif
        exit(1);
    }
}

DBConnector::~DBConnector()
{
    //Kleinoume ti sundesi me tin DB ston destructor
    if(myDB.data()->isOpen())
    {
        myDB.data()->close();
    }
}

void DBConnector::addDataToDB(float *temp, float *hum) const
{
    //Ekteloume to SQL query gia na kanoume insert ta duo values
    //pou theloume stin SQL database.
    bool res = myQuery.data()->exec("INSERT INTO "+ m_qsDBTableName
+" (date, temp, humidity)"
                                    "VALUES ("+
QString::number(myCurrentDate.data()->currentMSecsSinceEpoch()/1000)
+ ","+ QString::number(*temp) + " ," + QString::number(*hum)+")");
    #ifdef DEBUG
        if(!res)
        {
            qDebug() << Q_FUNC_INFO << ": " << "Failed to write data
to DB: " << myCurrentDate.data()->currentMSecsSinceEpoch()/1000 << ",
" << QString::number(*temp) << ", " << QString::number(*hum);
        }
    #else
        Q_UNUSED(res)
    #endif
}
```

dbconnector.h

```
#ifndef DBCONNECTOR_H
#define DBCONNECTOR_H

#include <QObject>
#include <QtSql/QSqlDatabase>
#include <QtSql/QSqlQuery>
#include <QScopedPointer>
#include <QDateTime>
```

```
#include <QString>

#ifdef DEBUG
#include <QDebug>
#endif

class DBConnector : public QObject
{
    Q_OBJECT

public:
    explicit DBConnector(QObject *parent = nullptr);
    ~DBConnector();
    void addDataToDB(float *temp, float *hum) const;

private:
    /*=============== DB SETTINGS ===============*/
    const QString m_qsDatabase      = "QMYSQL3";
    const QString m_qsDBName        = "msc";
    const QString m_qsDBTableName   = "ant1Data";
    const QString m_qsHostName      = "localhost";
    const QString m_qsUserName      = "root";
    const QString m_qsPassword      = "XXX";
    const int m_iPort = 3306; //Default MySQL port
    /*===========================================*/

    QScopedPointer<QSqlDatabase>    myDB;
    QScopedPointer<QSqlQuery>       myQuery;
    QScopedPointer<QDateTime>       myCurrentDate;
};

#endif // DBCONNECTOR_H
```

cloudmanager.cpp

```
#include "cloudmanager.h"
CloudManager::CloudManager(QObject *parent) : QObject(parent)
{

}

CloudManager::~CloudManager()
{

}

void CloudManager::uploadData(float *field1, float *field2)
{
    //Kathe fora pou kaletai i sunartisi kanoume ena neo
    //QNetworkAccessManager
    QNetworkAccessManager *manager = new QNetworkAccessManager();

    //Orizoume tis parametrous gia to link pou tha kanei
    //update tis times sto cloud (API KEY)
    QString qsURL = "http://api.thingspeak.com/update";
    QUrl url = QUrl(qsURL);
```

```
    QString qsAPIKEY = "GB40V7SI5ZJJZ4ZE";
    QUrlQuery params;

    QNetworkRequest request(url);

    //Sumfwna me to documentation tou cloud, orizoume to http header
gia to request,
    //kathws kai tis duo metavlites pou tha anevoun sto cloud mazi me
ta tags tous.
    request.setHeader(QNetworkRequest::ContentTypeHeader,
"application/x-www-form-urlencoded");

    params.addQueryItem("key",qsAPIKEY);
    params.addQueryItem("field1",QString::number(*field1));
    params.addQueryItem("field2",QString::number(*field2));
#ifdef DEBUG
    qDebug() << "Posting: " << *field1 << ", " << *field2;
#endif
    //Afou kaname construct to request, kanoume POST mesw tou manager
    //kai oi times anevainoun sto cloud. Se periptwsi pou to reply
    //den einai OK, apla prospername tis times.
    manager->post(request, params.query().toUtf8());
}
```

cloudmanager.h

```
#ifndef CLOUDMANAGER_H
#define CLOUDMANAGER_H

#include <QObject>
#include <QUrl>
#include <QString>
#include <QNetworkAccessManager>
#include <QNetworkRequest>
#include <QUrlQuery>
#include <QNetworkReply>
#include <QDebug>

class CloudManager : public QObject
{
    Q_OBJECT
public:
    explicit CloudManager(QObject *parent = nullptr);
    ~CloudManager();

    void uploadData(float *field1, float *field2);
};

#endif // CLOUDMANAGER_H
```

Βιβλιογραφία

[1] (http://www.instructables.com/id/Configuring-XBees-for-API-Mode/, 2020)
[2] (http://www.instructables.com/id/PART-1-Send-Arduino-data-to-the-Web-PHP-MySQL-D3js/#step2, 2020)
[3] (http://www.toptechboy.com/arduino/python-with-arduino-lesson-14-introduction-to-xbee-radios-and-wireless-communication/, 2020)
[4] (http://www.toptechboy.com/arduino/python-with-arduino-lesson-14-introduction-to-xbee-radios-and-wireless-communication/, 2020)
[5] (http://www.toptechboy.com/tutorial/python-with-arduino-lesson-15-configuring-and-using-the-xbee-radios/, 2020)
[6] (https://www.digikey.com/, 2020)
[7] (http://www.circuitbasics.com/how-to-set-up-the-dht11-humidity-sensor-on-an-arduino/, 2020)
[8] (https://thepi.io/how-to-install-raspbian-on-the-raspberry-pi/, 2020)
[9] (https://pimylifeup.com/category/guides/, 2020)
[10](https://iotdesignpro.com/projects/how-to-send-data-to-thingspeak-cloud-using-raspberry-pi, 2020)
[11](https://iotdesignpro.com/projects/iot-based-temperature-monitoring-using-raspberry-pi-and-lm35, 2020)
[12](http://www.sentilo.io/wordpress/, 2020)
[13](https://joinup.ec.europa.eu/document/sentilo-sensor-and-actuator-platform-smart-cities, 2020)
[14](http://iotip.tech/blog-single-sentilo, 2020)
[15](https://howlingpixel.com/wiki/Sentilo_Platform, 2020)
[16](https://www.cisco.com/assets/global/ZA/tomorrow-starts-here/pdf/barcelona_jurisdiction_profile_za.pdf, 2020)
[17](http://searchnetworking.techtarget.com/definition/Cisco-LISP-Cisco-Locator-ID-Separation-Protocol, 2020)
[18](https://www.cloudflare.com/learning/serverless/glossary/what-is-edge-computing/, 2020)
[19](https://www.ge.com/digital/blog/what-edge-computing#to-section-index=section-3, 2020)
[20](https://whatis.techtarget.com/definition/IoT-gateway, 2020)
[21](https://openautomationsoftware.com/blog/iiot-edge-computing-vs-cloud-computing/, 2020)
[22](https://openautomationsoftware.com/blog/iiot-edge-computing-vs-cloud-computing/, 2020)
[23](https://www.cn.ntua.gr/sites/default/files/LoRaWAN.pdf, 2020)
[24](https://www.w3schools.com/sql/sql_intro.asp, 2020)
[25](http://raspberrywebserver.com/sql-databases/using-mysql-on-a-raspberry-pi.html, 2020)
[26](https://www.w3schools.com/sql/sql_syntax.asp, 2020)
[27](https://blogs.mulesoft.com/biz/tech-ramblings-biz/what-are-apis-how-do-apis-work/, 2020)

[28]Tushar B. Dhore, Shailesh S. Chandankhede, Anup Todsam, International Journal of Research In Science & Engineering, Special Issue: Techno-Xtreme 16, e-ISSN: 2394-8299, p-ISSN: 2394-8280 , FOG Computing

[29]Bonomi, Flavio & Milito, Rodolfo. (2012). Fog Computing and its Role in the Internet of Things. Proceedings of the MCC workshop on Mobile Cloud Computing. 10.1145/2342509.2342513.

[30]Firdhous, Mohamed & Ghazali, Osman & Hassan, Suhaidi. (2014). Fog Computing: Will it be the Future of Cloud Computing?

[31]Maksimovic, Mirjana & Vujovic, Vladimir & Davidović, Nikola & Milosevic, Vladimir & Perisic, Branko. (2014). Raspberry Pi as Internet of Things hardware: Performances and Constraints.

[32]Louis, Leo. (2018). WORKING PRINCIPLE OF ARDUINO AND USING IT AS A TOOL FOR STUDY AND RESEARCH. 10.5121/ijcacs.2016.1203.

[33]Barik, Lal. (2019). IoT based Temperature and Humidity Controlling using Arduino and Raspberry Pi. International Journal of Advanced Computer Science and Applications. 9. 494-502. 10.14569/IJACSA.2019.0100966.

[34]Botta, Alessio & Donato, Walter & Persico, Valerio & Pescapè, Antonio. (2015). Integration of Cloud computing and Internet of Things: A survey. Future Generation Computer Systems. 56. 10.1016/j.future.2015.09.021.

[35]Atzori, Luigi & Iera, Antonio & Morabito, Giacomo. (2010). The Internet of Things: A Survey. Computer Networks. 2787-2805. 10.1016/j.comnet.2010.05.010.