ΑΝΩΤΑΤΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΠΕΙΡΑΙΑ Τ.Τ.
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ
Τμήμα Ηλεκτρονικών Μηχανικών Τ.Ε.

PIRAEUS UNIVERSITY of APPLIED SCIENCES
FACULTY OF ENGINEERING
Department of Electronics Engineering

Πρόγραμμα Μεταπτυχιακών Σπουδών

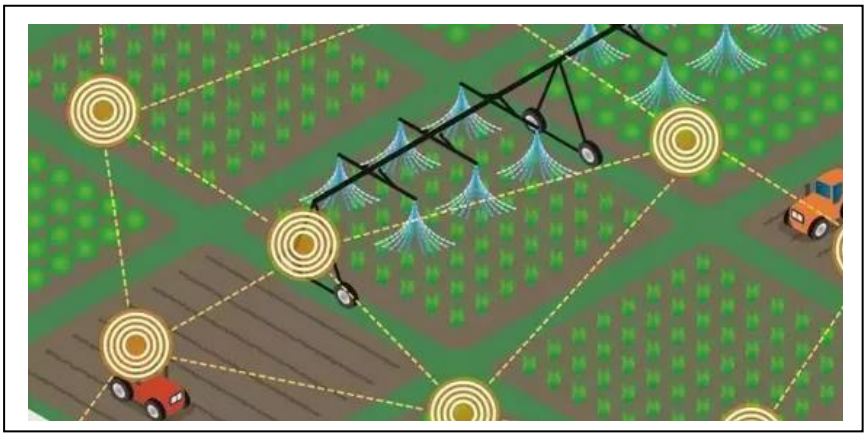
Διαδικτυωμένα Ηλεκτρονικά Συστήματα

Master of Science in

Internetworked Electronic Systems

ΜΕΤΑΠΤΥΧΙΑΚΗ ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Υλοποίηση ασύρματου κόμβου μικρού εύρους ζώνης μεγάλης εμβέλειας για εφαρμογές ακριβείας στη γεωργία.



Μεταπτυχιακός Φοιτητής: Δόγας Ιωάννης, Α.Μ. 0008

Επιβλέπων: Παπαγέωργας Παναγιώτης, Καθηγητής

ΑΙΓΑΛΕΩ, ΟΚΤΩΒΡΙΟΣ 2017

ΑΝΩΤΑΤΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΠΕΙΡΑΙΑ Τ.Τ.
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ
Τμήμα Ηλεκτρονικών Μηχανικών Τ.Ε.

PIRAEUS UNIVERSITY of APPLIED SCIENCES
FACULTY OF ENGINEERING
Department of Electronics Engineering

Πρόγραμμα Μεταπτυχιακών Σπουδών

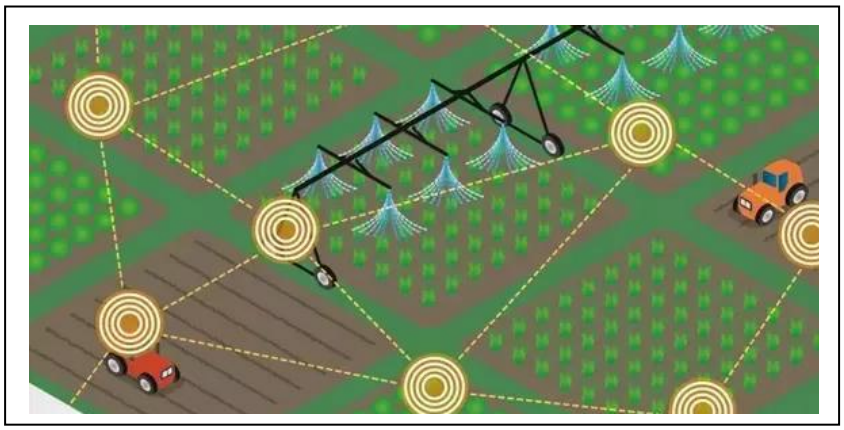
Διαδικτυωμένα Ηλεκτρονικά Συστήματα

Master of Science in

Internetworked Electronic Systems

MSc Thesis

Development of a long range – narrow band wireless sensor node for Precision Agriculture applications.



Student: Dogas Ioannis, Reg. Nr. 0008

MSc Thesis Supervisor: Papageorgas Panagiotis, Professor

ATHENS-EGALEO, OCTOBER 2017

ΠΕΡΙΛΗΨΗ

Αντικείμενο της παρούσας εργασίας αποτελεί η εξέταση και αξιολόγηση των τεχνολογιών που δύναται να χρησιμοποιηθούν σε ασύρματους κόμβους μεγάλης εμβέλειας στην γεωργία ακριβείας. Αρχικά μελετώνται σε θεωρητικό επίπεδο διάφορες παράμετροι όπως η εμβέλεια, η κατανάλωση ενέργειας, η πολυπλοκότητα που εισάγουν σε ένα σύστημα, το κόστος, η χρήση συχνοτήτων και ισχύος εκπομπής σε δημοφιλείς τεχνολογίες ασύρματης δικτύωσης, με σκοπό την εύρεση της βέλτιστης για το πεδίο εφαρμογής της γεωργίας. Εξετάζονται επίσης τόσο τα απαιτούμενα βέλτιστα υλικά, –μικροελεγκτές, πλακέτες ανάπτυξης, επεκτάσεις πλακετών, αισθητήρες- όσο και οι απαιτούμενες βέλτιστες τεχνολογίες προγραμματισμού και πρωτοκόλλων για την υλοποίηση ενός τέτοιου συστήματος.

Στην συνέχεια υλοποιήθηκε ένα σύστημα βάσει των προδιαγραφών που τέθηκαν στο θεωρητικό μέρος. Στην συγκεκριμένη υλοποίηση διάφοροι κόμβοι λαμβάνουν μετρήσεις από αισθητήρες είτε αναλογικούς, είτε ψηφιακούς, επεξεργάζονται τα δεδομένα και τα αποστέλλουν ασύρματα σε μεγάλες αποστάσεις σε έναν κεντρικό κόμβο ο οποίος τα συλλέγει. Στην συνέχεια από τον κεντρικό κόμβο δύναται τα δεδομένα αυτά να είναι προσβάσιμα μέσω μιας πύλης διαδικτύου από οποιοδήποτε μέρος του κόσμου. Αυτό είναι εφικτό είτε με την χρήση κάποιας εφαρμογής cloud τρίτου παρόχου, είτε κατευθείαν σε υλοποιημένη εφαρμογή διαδικτύου.

ΛΕΞΕΙΣ – ΚΛΕΙΔΙΑ: Διαδίκτυο των Πραγμάτων, Συλλογή Δεδομένων, Διαδίκτυο των Πραγμάτων στην Γεωργία, Δίκτυα Μεγάλων Αποστάσεων, Γεωργία ακριβείας, Ασύρματα δίκτυα αισθητήρων

ABSTRACT

This thesis evaluates the technologies that can be used in wireless long range nodes used in precision agriculture. Firstly, parameters and specifications such as range, energy consumption, introduced complexity, cost, restrictions about frequency bands of operation and the transceivers sensitivities , are examined, in order to select the most suitable wireless communication technologies for use in Agriculture. The most appropriate hardware components like microcontrollers, development boards, expansion shields, sensors- and programming techniques are also examined, in order to realize an experimental platform for proof of concept.

An experimental setup was built, according to the specifications and the requirements resulted from the theoretical framework of the thesis. The implemented sensor nodes collect data from either analog or digital sensors, process data and transmit the signals to a master node. The data from the sensors can then be accessible worldwide via a network gateway. This is feasible via, either a cloud app from a third party provider, or a custom built web application.

KEYWORDS: IoT, Data Collection, Internet of Things, IoT in Agriculture, Long Range Network, LoRa, Precision Agriculture, Wireless Sensor Networks

ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα να εκφράσω τις θερμότατες ευχαριστίες μου στον επιβλέποντα καθηγητή κ. Παναγιώτη Παπαγέωργα, για τη στήριξη και καθοδήγησή του σε επιστημονικά και διαδικαστικά ζητήματα καθ' όλη την διάρκεια της εκπόνησης της Μεταπτυχιακής Διπλωματικής Εργασίας μου.

Επίσης, θα ήθελα να ευχαριστήσω το σύνολο των καθηγητών και του προσωπικού του Μεταπτυχιακού Προγράμματος "Διαδικτυωμένα Ηλεκτρονικά Συστήματα" του Τμήματος Ηλεκτρονικών Μηχανικών του Α.Ε.Ι. Πειραιά Τ.Τ. για τις γνώσεις που μετέδωσαν κατά την διάρκεια της φοίτησής μου αλλά και για το σύνολο του έργου τους.

Ιδιαίτερες ευχαριστίες οφείλω στους φίλους και συμφοιτητές του Προγράμματος Μεταπτυχιακών Σπουδών, αλλά και στους συναδέλφους μηχανικούς της Ελληνικής Αεροπορικής Βιομηχανίας.

Τέλος, ευχαριστώ την οικογένειά μου για τη στήριξη και την δύναμη που μου δίνουν να συνεχίζω και να εκπληρώνω τους στόχους που θέτω στην ζωή μου.

ΠΙΝΑΚΑΣ ΣΥΜΒΟΛΩΝ-ΑΚΡΩΝΥΜΙΩΝ-ΣΥΝΤΟΜΟΓΡΑΦΙΩΝ

3GPP	3rd Generation Partnership Project
6LoWPAN	IPv6 over Low-Power Wireless Personal Area Networks
ADC	Analog to Digital Converter
ADR	Adaptive Data Rate
BNC	Bayonet Neill–Concelman
DDR2	DDR2 (Double Data Rate 2)
FDMA	Frequency division multiple access
GND	GrouND
GPS	Global Positioning System
GSM	Global System for Mobile Communications
HAL	Hardware Abstruction Layer
HTTP	Hypertext Transfer Protocol
IDE	Integrated Development Environment
IoT	Internet of Things
ISM	Industrial Scientific Medical
LED	Light Emitting Diode
LoRa	Long Range
LPWAN	Low Power Wide Area Network
LTE	Long-Term Evolution
MAC	Media Access Control
MCU	Microcontroller Unit
MQTT	Message Queue Telemetry Transport
NB-IoT	NarrowBand Internet of Things
OFDM	Orthogonal frequency Division Multiplexing
PWM	Pulse Width Modulation
QoS	Quality of Service
QPSK	Quadrature Phase Shift Keying
RAM	Random Access Memory)
REST	Representational State Transfer
RPMA	Random Phase Multiple Access
SPI	Serial Peripheral Interface
SRAM	Static Random Access Memory
TCP	Transmission Control Protocol
TCP/IP	Transmission Control Protocol/Internet Protocol
TDMA	Time-division multiple access
UART	Universal Asynchronous Receiver Transmitter
USB	Universal Serial Bus
WiFi	Wireless Fidelity
WSN	Wireless Sensor Network
ΑΔΑ	Ασύρματο Δίκτυο Αισθητήρων
ΑΠΕ	Ανανεώσιμες Πηγές Ενέργειας
ΔΤΠ	Διαδίκτυο των πραγμάτων

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

ΠΙΝΑΚΑΣ ΕΙΚΟΝΩΝ

ΕΙΣΑΓΩΓΗ:

Αντικείμενο, ερευνητικά ερωτήματα και διάρθρωση της εργασίας

Η διάδοση του διαδικτύου των πραγμάτων κατέστησε σαφείς τις ευκολίες που παρέχει τόσο σε οικιακό, όσο και σε βιομηχανικό περιβάλλον. Βασικά πλεονεκτήματα αποτελούν η δυνατότητα απομακρυσμένης συνεχούς επιτήρησης παραμέτρων από αισθητήρες (sensors), αλλά και απομακρυσμένου ελέγχου ενεργοποιητών (actuators), εάν αυτό απαιτείται. Ο έλεγχος των ενεργοποιητών μπορεί να γίνεται επίσης μέσω αυτοματισμών, οι παράμετροι των οποίων μπορούν να αλλάζουν δυναμικά βάσει των συνθηκών, ή των επιλογών του χρήστη.

Κατά συνέπεια και η γεωργία αποτελεί έναν κλάδο ο οποίος δύναται να επωφεληθεί στο μέγιστο από ένα ασύρματο δίκτυο αισθητήρων και ενεργοποιητών, το οποίο να μπορεί να ελεγχθεί απομακρυσμένα μέσω του διαδικτύου. Οι συνθήκες υγρασίας, θερμοκρασίας, οξύτητας χώματος, φωτός μπορούν να επηρεάσουν ανά πάσα στιγμή την ποιότητα των γεωργικών προϊόντων, συνεπώς η συνεχής επιτήρησή τους –και η λήψη ενεργειών όπου αυτό απαιτείται- κρίνεται απαραίτητη. Για παράδειγμα οι αντιπαγετικοί ανεμιστήρες ή και οι λάμπες υπερύθρων δύναται να ενεργοποιούνται αυτόματα, για αποφυγή συνθηκών παγετού σε οπωρώνες, διαδικασία που πραγματοποιείται ακόμα χειροκίνητα. Έτσι δύναται να επιτευχθεί μεγιστοποίηση της παραγωγής ποιοτικότερων προϊόντων, αλλά και ελαχιστοποίηση της πιθανότητας καταστροφής της παραγωγής.

Παράλληλα, και οι περισσότερες βιομηχανικές λύσεις του διαδικτύου των πραγμάτων προσανατολισμένες στην γεωργία παρουσιάζουν ορισμένα προβλήματα. Αρχικά το κόστος τους παραμένει πολύ υψηλό το οποίο τις καθιστά δυσπρόσιτες για τον μέσο

παραγωγό, για τον οποίο η απόσβεση θα είναι μακροπρόθεσμη. Επίσης υστερούν στην ευκολία χρήσης τους, στην συνδεσιμότητα και στην επεκτασιμότητα τους, το οποίο περιορίζει το όφελος από την χρήση τους.

Οι γνωστότερες διαθέσιμες εμπορικές λύσεις είναι το Parrot Flower Power, το Edyn Smart Garden System, και το Pycno. Το Parrot Flower αποτελεί ένα προϊόν τόσο για εσωτερική όσο και για εξωτερική χρήση. Δύναται να μετρήσει φωτεινότητα, θερμοκρασία, υγρασία και γονιμότητα εδάφους. Βασικό μειονέκτημα του αποτελεί ότι κάθε Module πρέπει να συνδέεται με φορητή συσκευή μέσω Bluetooth. Αυτό συνεπάγεται μεγάλη κατανάλωση ενέργειας (άρα μειωμένη αυτονομία), μειωμένη εμβέλεια, δυσκολία στην χρήση, αλλά και ανύπαρκτη επεκτασιμότητα. Το εν λόγω προϊόν είναι κατάλληλο μόνο για μικρούς οικιακούς κήπους. Αντίστοιχα το Edyn Smart Garden είναι ένα παρόμοιο προϊόν, το οποίο χρησιμοποιεί φωτοβολταϊκά για βελτίωση της αυτονομίας. Χρησιμοποιεί το WiFi (Wireless Fidelity) το οποίο προσφέρει ελαφρώς μεγαλύτερη εμβέλεια, το οποίο όμως και αυτό αποτελεί ενεργοβόρο τεχνολογία. Δύναται να μετρήσει φωτεινότητα, υγρασία αέρος και εδάφους, θερμοκρασία, και θερπτικά στοιχεία στο χώμα. Τέλος η υλοποίηση της Pycno αποτελεί το πιο πολύπλοκο προϊόν, μιας και χρησιμοποιεί πολλά slave nodes τα οποία επικοινωνούν με master node/gateway, που καθιστά προσβάσιμα τα δεδομένα μέσω GSM (Global System for Mobile Communications). Μέγιστη απόσταση λειτουργίας των nodes του συστήματος είναι τα 2 χιλιόμετρα. Οι λύσεις αυτού του είδους είναι συνήθως απομονωμένες χωρίς την δυνατότητα γραμμικής κλιμάκωσης για μεγάλο αριθμό αισθητήρων ενώ δεν ακολουθούν ανοικτές σε επίπεδο λογισμικού και υλικού υλοποιήσεις με αποτέλεσμα να μην είναι δυνατή η διαλειτουργικότητά τους. [1]

Εικόνα 1 – Το πρόβλημα των διαφορετικών πυλών διαδικτύου για εφαρμογές του ΔΤΠ (Διαδικτύου Των Πραγμάτων) [2]

Κίνητρο για την εκπόνηση της παρούσας εργασίας αποτέλεσε η αντιμετώπιση των προαναφερθέντων προβλημάτων στις υλοποιήσεις ασύρματων δικτύων αισθητήρων με συνδεσιμότητα προς το Internet προσανατολισμένες στην γεωργία ακριβείας.

Η προτεινόμενη υλοποίηση πρέπει να είναι εξαιρετικά χαμηλού κόστους, εύκολη στην χρήση, με ανοιχτού κώδικα λογισμικό και υλικό, καθώς και με δυνατότητες εύκολης επέκτασης της χρηστικότητάς της. Επιπρόσθετα η χρήση κατάλληλου ενδιάμεσου λογισμικού (middleware) με την χρήση του πρωτοκόλλου MQTT (Message Queue Telemetry Transport) για την ανταλλαγή μηνυμάτων μεταξύ των κόμβων και των εφαρμογών καταγραφής-ελέγχου είναι η βέλτιστη επιλογή για την γραμμική κλιμάκωση της προτεινόμενης λύσης σε συνδυασμό με τεχνολογίες «νέφους» cloud και την υλοποίησή της με κατανεμημένο υλικό-λογισμικό.

Στο πρώτο κεφάλαιο της παρούσας εργασίας εξετάζεται θεωρητικά το πλαίσιο του θέματος, γίνεται αναφορά και επεξήγηση των βασικών εννοιών και τεχνολογιών, όπως του διαδικτύου των πραγμάτων, των ασυρμάτων δικτύων αισθητήρων, και της γεωργίας ακριβείας

Στο δεύτερο κεφάλαιο εξετάζεται η χρήση τεχνολογιών υλικών και λογισμικού τα οποία πληρούν τις προϋποθέσεις για την υλοποίηση ενός συστήματος ασυρμάτου δικτύου αισθητήρων για την γεωργίας ακριβείας.

Στο τρίτο κεφάλαιο πραγματοποιείται η υλοποίηση του συστήματος με την χρήση των υλικών, εργαλείων και λογισμικού που έχουν επιλεχθεί.

Τέλος γίνεται αναφορά στο κατά πόσον η παρούσα υλοποίηση καλύπτει τις απαιτήσεις που τέθηκαν, αλλά και κατά πόσον υφίσταται δυνατότητα μελλοντικής εξέλιξής της.

1. Διαδίκτυο των Πραγμάτων

1.1 Εισαγωγή στο Διαδίκτυο των Πραγμάτων

Ως Διαδίκτυο των Πραγμάτων (IoT, Internet of Things) ορίζουμε το σύνολο των φυσικών αντικειμένων και συσκευών, τα οποία περιέχουν ενσωματωμένα ηλεκτρονικά συστήματα και ανταλλάσσουν πληροφορίες μεταξύ τους αλλά και με το διαδίκτυο. Μέσω της πρόσβασης στο διαδίκτυο δίνεται η δυνατότητα απομακρυσμένου ελέγχου τους και επίβλεψής τους από οποιοδήποτε σημείου του κόσμου. Συνεπώς, προσφέρουν σύνδεση του φυσικού κόσμου με υπολογιστικά συστήματα, το οποίο αυξάνει την αποτελεσματικότητα, την ακρίβεια και την αποδοτικότητα τέτοιων συσκευών σε σύγκριση με τις κοινές.

Πολλές τεχνολογίες συνδυάζονται για την υλοποίηση συσκευών που ανήκουν στο διαδίκτυο των πραγμάτων. Για την επικοινωνία μεταξύ των συσκευών, είτε των συσκευών με το διαδίκτυο χρησιμοποιούνται ασύρματες τεχνολογίες όπως Zigbee, LoRa (Long Range), Sigfox, RPMA (Random Phase Multiple Access), NB-IoT (NarrowBand Internet of Things), Weightless-N, 6LoWPAN (IPv6 over Low-Power Wireless Personal Area Networks), WiFi, Bluetooth, GSM και λοιπές, με μετρικές παραμέτρους διαφοροποίησης όπως η εμβέλεια, η κατανάλωση ενέργειας, ο ρυθμός μετάδοσης δεδομένων, να καθιστούν κάθε μία από αυτές περισσότερο ή λιγότερο ελκυστικές για κάθε υλοποίηση. Ανάλυση και σύγκριση των δημοφιλέστερων εκ των τεχνολογιών αυτών πραγματοποιείται σε επόμενο κεφάλαιο της παρούσας εργασίας. [3]

Τα αντικείμενα του διαδικτύου των πραγμάτων σχεδόν πάντα διαθέτουν διάφορους αισθητήρες, με σκοπό την συλλογή δεδομένων και πληροφοριών από το περιβάλλον τους. Στην συνέχεια τις πληροφορίες αυτές μπορεί να τις χρησιμοποιήσουν είτε για να εκτελέσουν κάποια ενέργεια, είτε για να τις διαβιβάσουν σε λοιπούς κόμβους, και προς το διαδίκτυο. Ταυτόχρονα, ο ενσωματωμένος επεξεργαστής ή μικροελεγκτής που διαθέτουν, αλλά και η δυνατότητα αποθήκευσης δεδομένων τους παρέχει την

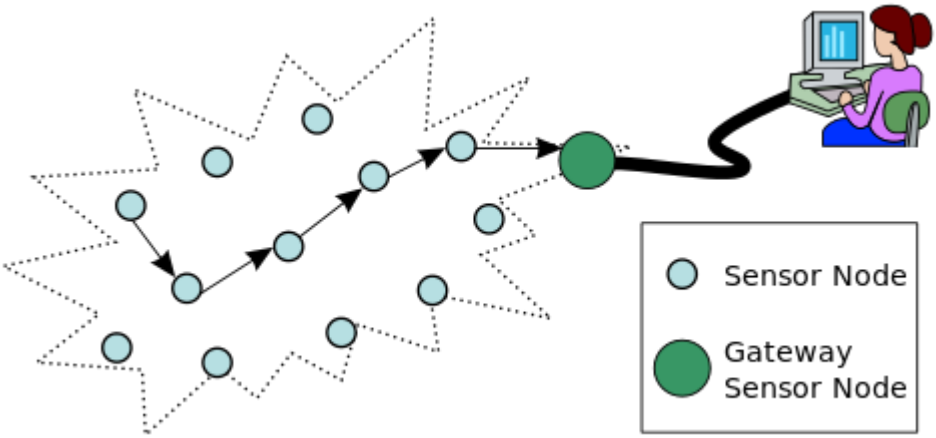
δυνατότητα αύξησης της ακρίβειας των δεδομένων από τους αισθητήρες μέσω δειγματοληψίας και στατιστικών τεχνικών.

Εικόνα 2 – Κόμβος διαδικτύου των πραγμάτων για την υλοποίηση «έξυπνης πόλης» [4]

Διάφοροι ενεργοποιητές δίνουν επίσης την δυνατότητα εκτέλεσης ενεργειών από τις συσκευές, και όχι μόνο λήψη δεδομένων από τους αισθητήρες τους. Αυτό γίνεται με την μετατροπή ηλεκτρικών σημάτων σε μηχανικές κινήσεις μέσω electromechanical/electronic Relays, ή κινητήρες. Οι ενεργοποιητές σε ένα σύστημα του διαδικτύου των πραγμάτων μπορούν είτε να ελεγχθούν από τον χρήστη από απόσταση, είτε να λειτουργήσουν αντιδρώντας σε πληροφορίες από τους αισθητήρες, μέσω αυτοματοποιήσεων.

1.2 Ασύρματα δίκτυα αισθητήρων και τεχνολογίες τους

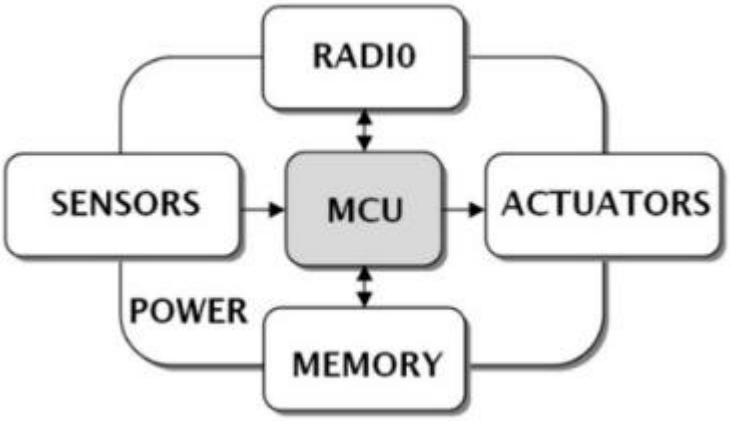
Ένα ασύρματο δίκτυο αισθητήρων (ΑΔΑ / Wireless Sensor Network - WSN) αποτελείται από διασκορπισμένους αυτόνομους αισθητήρες για την παρακολούθηση φυσικών ή περιβαλλοντολογικών συνθηκών, όπως η θερμοκρασία, η υγρασία, η ατμοσφαιρική πίεση κτλ. να μεταφέρει τα δεδομένα μέσω του δικτύου σε μια συγκεκριμένη τοποθεσία. Τα πιο μοντέρνα δίκτυα είναι ικανά και να δίνουν αλλά και να δέχονται πληροφορίες πράγμα που τους επιτρέπει να ελέγχουν την δραστηριότητα των αισθητήρων. Το κίνητρο για την ανάπτυξη των ασύρματων δικτύων με αισθητήρες ήταν οι στρατιωτικές εφαρμογές όπως η παρακολούθηση των πεδίων μάχης. Σήμερα τέτοια δίκτυα χρησιμοποιούνται σε πολλές καταναλωτικές και βιομηχανικές εφαρμογές, η παρακολούθηση και ο έλεγχος της βιομηχανικής παραγωγής. [5]



Εικόνα 3 - Ασύρματο δίκτυο αισθητήρων με συνδεσιμότητα προς το Internet (εικόνα υπό Public Domain)

Το ασύρματο δίκτυο αισθητήρων αποτελείται από κόμβους, όπου κάθε κόμβος συνδέετε σε έναν (η κάποιες φορές σε αρκετούς) αισθητήρες. Κάθε τέτοιος κόμβος του δικτύου αισθητήρων έχει χαρακτηριστικά υποσυστήματα: ένα ραδιοπομποδέκτη με μια εσωτερική κεραία ή μια σύνδεση με μια εξωτερική κεραία, ένα μικροελεγκτή, ένα ηλεκτρονικό κύκλωμα για τη διασύνδεση με τους αισθητήρες και μια πηγή ενέργειας, συνήθως μια μπαταρία η μια ενσωματωμένη μορφή συγκομιδής ενέργειας. Ένας αισθητήριος κόμβος

μπορεί να ποικίλει σε μέγεθος. Το κόστος των αισθητήριων κόμβων επίσης ποικίλει, ξεκινώντας από μερικές δεκάδες και φτάνοντας σε εκατοντάδες ευρώ, αναλόγως την πολυπλοκότητα των μεμονωμένων αισθητήριων κόμβων. Οι περιορισμοί σε μέγεθος και κόστος έχουν ως αποτέλεσμα αντίστοιχους περιορισμούς σε πόρους όπως κατανάλωση ενέργειας, μνήμη, επεξεργαστική ισχύς και τεχνολογία επικοινωνιών. Η τοπολογία των αισθητήρων μπορεί να διαφέρει, παρ' όλα αυτά οι περισσότερες υλοποιήσεις χρησιμοποιούν την τοπολογία αστέρα.



Εικόνα 4 – Τυπική αρχιτεκτονική ενός κόμβου ασύρματου δικτύου αισθητήρων [6]

1.2.1 Ασύρματα Δίκτυα Αισθητήρων Χαμηλής Κατανάλωσης Ενέργειας

Σε πολλές περιπτώσεις η τροφοδοσία των κόμβων ενός δικτύου αισθητήρων δεν είναι δυνατόν να πραγματοποιηθεί από την παροχή του δικτύου. Αυτό συνεπάγεται την απαίτηση τροφοδοσίας τους από μπαταρίες, οι οποίες με την σειρά τους μπορεί να μην είναι δυνατόν να αλλαχθούν για όλη την διάρκεια ζωής του συστήματος. Έτσι δημιουργείται η απαίτηση για ασύρματα δίκτυα αισθητήρων χαμηλής κατανάλωσης ενέργειας (LPWAN - Low Power Wireless Access Networks).

Αν και κάθε εφαρμογή είναι διαφορετική, ένα ασύρματο δίκτυο αισθητήρων χαμηλής κατανάλωσης ενέργειας προϋποθέτει κάποιες κοινές απαιτήσεις για τον σχεδιασμό του. Αρχικά πρέπει να είναι δυνατή η λειτουργία των κόμβων του δικτύου από πολύ περιορισμένες πηγές ενέργειας, όπως για παράδειγμα από μικρές μπαταρίες τύπου

νομίσματος. Η επίτευξη της απαιτούμενης εμβέλειες, συνεπώς, πρέπει να επιτυγχάνεται χωρίς την χρήση μεγάλης ισχύος εκπομπής, το οποίο θα συνεπαγόταν μεγάλη κατανάλωση ενέργειας. Επίσης, συνηθισμένη απαίτηση σε τέτοια συστήματα αποτελεί το χαμηλό κόστος υλοποίησης, εγκατάστασης, αλλά και συντήρησής του. Αυτό συνεπάγεται μικρή πολυπλοκότητα τόσο σε επίπεδο υλικού όσο και σε επίπεδο λογισμικού. Το επίπεδο δραστηριότητας μπορεί να αλλάζει αναλόγως των απαιτήσεων της κάθε εφαρμογής, αλλά για την μείωση της κατανάλωσης ενέργειας ένας κόμβους (ή το εκάστοτε υποσύστημα ενός κόμβου) δεν πρέπει να βγαίνει από το sleep mode εκτός και αν απαιτείται η αποστολή ή λήψη δεδομένων. Αυτό συνεπάγεται την απόρριψη συγχρονισμένων ή δικτύων πλέγματος (mesh networks), και χρήση πρωτοκόλλων όπως το ALOHA, σε τοπολογία αστέρα. Επιπλέον η υποδομή του δικτύου, αλλά και η επέκτασή του (για παράδειγμα η προσθήκη επιπλέον κόμβων) πρέπει να δύναται να πραγματοποιηθεί με εύκολο τρόπο, άρα απαιτείται η χρήση κάποιου πρωτοκόλλου που να το επιτρέπει, ταυτόχρονα με λειτουργίες βελτιστοποίησης της αποδοτικότητας του δικτύου, όπως ο δυναμικός ορισμός ρυθμού μετάδοσης δεδομένων (ADR – Adaptive Data Rate) και ο έλεγχος ισχύος εκπομπής (TPC – Transmit Power Control). Η κρυπτογράφηση των δεδομένων που αποστέλλονται, καθώς και η προστασία από παρεμβολές αποτελούν με την σειρά τους κρίσιμες απαιτήσεις για τις περισσότερες εφαρμογές. Αντίστοιχα, σε αρκετές εφαρμογές είναι χρήσιμο κάθε κόμβος ο οποίος είναι τοποθετημένος στο πεδίο, να μπορεί να εντοπισθεί χωρίς την χρήση ενεργοβόρων τεχνολογιών όπως το GPS (Global Positioning System). Τέλος, από πλευράς εφαρμογών, τα δεδομένα που συλλέχτηκαν θα πρέπει να μπορούν να χρησιμοποιηθούν για την βελτίωση πολλών λοιπών προϊόντων και υπηρεσιών. [7]

1.2.2 Τεχνολογίες ασύρματης δικτύωσης LPWAN

Ενώ υπάρχουν πολλές τεχνολογίες και πρότυπα Low Power Wide Area Network (LPWAN), θα επικεντρωθούμε στα SigFox, LoRa, RPMA, Symphony Link και Weightless, επειδή αυτές οι τεχνολογίες βρίσκονται υπό ενεργό ανάπτυξη και εξάπλωση. Αν και υπάρχουν άλλα ιδιόκτητα πρωτόκολλα και στοίβες όπως το πρωτόκολλο Dash7, δεν έχουν ευρεία απήχηση/δεν έχουν προσελκύσει το άμεσο ενδιαφέρον και δεν θα αναλυθούν στο παρόν άρθρο. [8]

SigFox

Εικόνα 5 – Sigfox Logo

Με έτος ίδρυσης το 2009, η SigFox αποτελεί μια γαλλική εταιρεία με έδρα την πόλη Labège της Γαλλίας. Η SigFox έχει αναμφισβήτητα την μεγαλύτερη απήχηση στον χώρο των τεχνολογιών LPWAN (ή τουλάχιστον είναι η πιο αναγνωρίσιμη, λόγω των επιτυχημένων εκστρατειών μάρκετινγκ στην Ευρώπη. Διαθέτει επίσης ένα μεγάλο δίκτυο πωλητών, συμπεριλαμβανομένων των Texas Instruments, Silicon Labs και Axom.

Το (πρότυπο) SigFox χρησιμοποιεί ιδιόκτητη τεχνολογία, με διαμόρφωσης συχνότητας και χαμηλό ρυθμό μετάδοσης δεδομένων για την επίτευξη μεγαλύτερης εμβέλειας. Λόγω αυτής της επιλογής σχεδιασμού, το SigFox είναι μια εξαιρετική επιλογή για εφαρμογές όπου το σύστημα χρειάζεται μόνο να στείλει μικρό αριθμό bytes ανά μεγάλα χρονικά διαστήματα (για παράδειγμα 10 bytes /μέρα).

Πιθανές εφαρμογές περιλαμβάνουν αισθητήρες στάθμευσης, μετρητές νερού ή έξυπνους κάδους απορριμμάτων. Ωστόσο, έχει επίσης κάποια μειονεκτήματα. Η αποστολή δεδομένων πίσω στους αισθητήρες / συσκευές (δυνατότητα καθοδικής/κατερχόμενης ζεύξης-downlink) είναι πολύ περιορισμένη και η παρεμβολές σημάτων μπορεί να αποτελέσουν πρόβλημα.

LoRa

Εικόνα 6 – LoRa Logo

Η LoRa Alliance είναι ένας ανοικτός μη κερδοσκοπικός οργανισμός που δημιουργήθηκε για να προωθήσει ένα οικοσύστημα για ορισμένες τεχνολογίες LPWAN. Συνεργάζεται με περίπου 400 εταιρείες σε ολόκληρη τη Βόρεια Αμερική, την Ευρώπη, την Αφρική και την Ασία και τα ιδρυτικά της μέλη περιλαμβάνουν τις IBM, MicroChip, Cisco, Semtech, Bouygues Telecom, Singtel, KPN, Swisscom, Fastnet και Belgacom.

Το LoRaWAN αποτελεί ένα επίπεδο δικτύωσης ανοιχτού προτύπου, το οποίο διαχειρίζεται η LoRa Alliance. Ωστόσο, δεν είναι πραγματικά ανοικτού υλικού δεδομένου ότι το υπάρχον κύκλωμα για την υλοποίηση μιας πλήρους στοίβας LoRaWAN είναι διαθέσιμο μόνο μέσω της Semtech.

Η λειτουργικότητα είναι παρόμοια με αυτήν του SigFox, καθώς αφορά κυρίως εφαρμογές μόνο ανερχόμενης ζεύξης (δεδομένα από αισθητήρες / συσκευές σε πύλη) με πολλά τελικά σημεία (endpoints). Δεν χρησιμοποιεί στενή ζώνη μετάδοσης, αλλά διαδίδει πληροφορίες σε διαφορετικά κανάλια συχνότητας και ρυθμούς δεδομένων μέσω της χρήσης κωδικοποιημένων μηνυμάτων. Αυτά τα μηνύματα είναι λιγότερο πιθανό να συγκρουστούν και να αλληλεπιδράσουν μεταξύ τους/επηρεάσουν το ένα το άλλο.

RPMA

Εικόνα 7 – INGENU Logo

Η διαμόρφωση φάσης πολλαπλής πρόσβασης (RPMA) είναι μια ιδιόκτητη στοίβα τεχνολογίας LPWAN που αναπτύχθηκε από την Ingenu. Η εταιρεία ιδρύθηκε το 2008 στο Σαν Ντιέγκο της Καλιφόρνια, από πρώην μηχανικούς της Qualcomm και αρχικά ονομάστηκε On-Ramp Wireless.

Ως ιδρυτικό μέλος της ομάδας εργασίας IEEE 802.15.4k (η οποία ήταν αφιερωμένη στην παρακολούθηση υποδομών χαμηλής ενέργειας), η Ingenu κατέβαλε τεράστια προσπάθεια στην ανάπτυξη του RPMA, ενώ οι ομάδες SigFox και LoRaWAN επικεντρώθηκαν στην πιο γρήγορη διάθεση στην αγορά.

Δεδομένης της αρχιτεκτονικής της, η διαμόρφωση RPMA διαθέτει μεγαλύτερη χωρητικότητα καναλιού τόσο σε downlink όσο και σε uplink σε σχέση με άλλα μοντέλα. Λειτουργεί σε φάσμα 2,4 GHz, το οποίο είναι παγκοσμίως διαθέσιμο (χρησιμοποιείται για

WiFi και Bluetooth). Αυτό σημαίνει ότι δεν υπάρχουν αλλαγές αρχιτεκτονικής ανά περιοχή όπως στο SigFox και στο LoRa.

Symphony Link

Εικόνα 8 – LinkLabs Logo

Η Link Labs είναι μέλος της LoRa Alliance και επομένως χρησιμοποιεί το LoRa chip. Ωστόσο, αντί να χρησιμοποιήσει το LoRaWAN, η Link Labs δημιούργησε ένα ιδιόκτητο επίπεδο δικτύωσης MAC, βασισμένη στα τσιπάκια της Semtech και το οποίο ονομάζεται Symphony Link. Η Link Labs ιδρύθηκε το 2013 από πρώην μέλη του Εργαστηρίου Εφαρμοσμένης Φυσικής του Πανεπιστημίου Johns Hopkins και έχει την έδρα του στην Αννάπολις της πολιτείας Μέριλαντ.

Το Symphony Link προσθέτει ορισμένα σημαντικά χαρακτηριστικά συνδεσιμότητας σε σχέση με το LoRaWAN, όπως: εγγυημένη λήψη μηνύματος, ασύρματη αναβάθμιση λογισμικού στο υλικό, ικανότητα αναμετάδωσης και δυναμικό εύρος.

Weightless

Εικόνα 9 - Weightless Logo

Η Weightless SIG ιδρύθηκε το 2008 με αποστολή της την τυποποίηση των τεχνολογιών LPWAN. Υπάρχουν πέντε μέλη του ομίλου Promotor, συμπεριλαμβανομένων των Accenture, η ARM, η M2COMM, Sony-Europe και Telensa.

Το Weightless είναι το μόνο πραγματικά ανοιχτό πρότυπο που λειτουργεί σε υπό το 1 GHz μη αδειοδοτημένο (ISM - Industrial Scientific Medical) φάσμα. Υπάρχουν τρεις εκδόσεις του Weightless που εξυπηρετούν διαφορετικούς σκοπούς:

Weightless-W: αξιοποιεί/εκμεταλλεύεται τα κενά (αχρησιμοποίητο τοπικό φάσμα σε αδειοδοτούμενη ζώνη τηλεόρασης)

Weightless-N: πρωτόκολλο στενής ζώνης μη αδειοδοτούμενης χρήσης που προήλθε από την τεχνολογία NWave

Weightless-P: αμφίδρομο πρωτόκολλο που προήλθε από την τεχνολογία Platanus M2COMM.

Τα Weightless N και P είναι οι πιο δημοφιλείς επιλογές, καθώς το Weightless-W έχει μεγαλύτερη κατανάλωση ενέργειας

Weightless-N/NWave

Το Nwave μοιάζει πολύ με το SigFox από την άποψη της λειτουργικότητας, αλλά διαθέτει μια καλύτερη υλοποίηση του επιπέδου δικτύωσης MAC (Media Access Control). Υποστηρίζεται ότι χρησιμοποιεί «προηγμένες τεχνικές διαμόρφωσης» για να επιτρέπει στο δίκτυό του να συνυπάρχει με άλλες ραδιοτεχνολογίες χωρίς επιπλέον θόρυβο. Όπως και το SigFox, είναι καλύτερο για δίκτυα με αισθητήρες, μετρήσεις θερμοκρασίας, παρακολούθηση στάθμης δεξαμενών (ή υγρών δεξαμενής), έξυπνες μετρήσεις και άλλες παρόμοιες εφαρμογές.

Weightless-P

Αυτό το πρότυπο χρησιμοποιεί διαμόρφωση FDMA (Frequency Division Multiple Access) + TDMA (Time Division Multiple Access) στενής ζώνης 12,5 kHz (μεγαλύτερη από το SigFox αλλά μικρότερη από το LoRa). Έχει επίσης προσαρμόσιμο ρυθμό δεδομένων, παρόμοιο με το Symphony Link (200 bps έως 100 kbps). Η ευαισθησία είναι αρκετά υψηλή, -134 dBm στα 625 bps και υποστηρίζει τόσο τη διαμόρφωση PSK όσο και τη διαμόρφωση GMSK.

NB-IoT

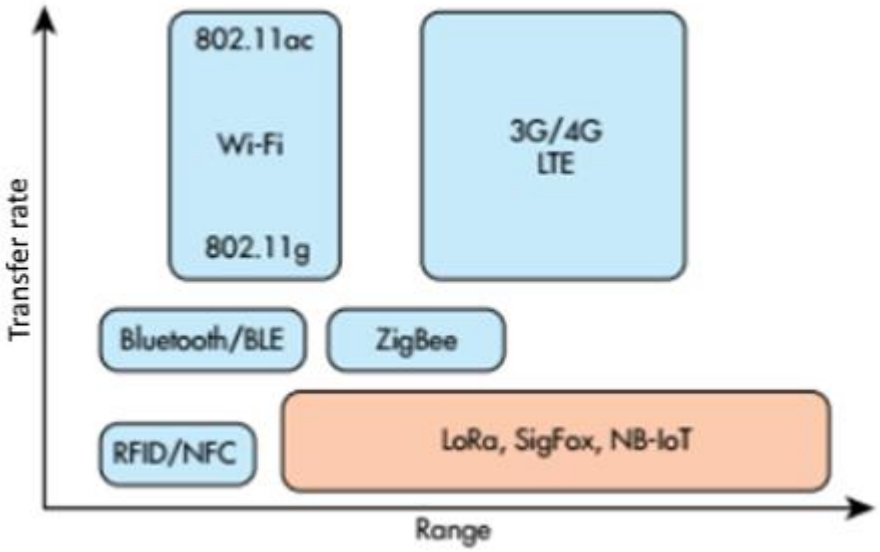
Εικόνα 10 – NB-IoT Logo

Το NB-IoT αποτελεί μία τεχνολογία ασύρματης δικτύωσης, για δίκτυα μεγάλων αποστάσεων χαμηλής κατανάλωσης ενέργειας, η οποία σχεδιάστηκε από την 3GPP (3rd

Generation Partnership Project). Χρησιμοποιεί τις ίδιες συχνότητες με το LTE (Long Term Evolution), το οποίο καθιστά απαραίτητη την άδεια για την χρήση του. Η διαμόρφωσή του είναι η QPSK (Quadrature Phase Shift Keying), ενώ παρέχει ταχύτητες μετάδοσης δεδομένων της τάξης των 200Kbps. Η υλοποίηση του NB-IoT μπορεί να γίνει χρησιμοποιώντας το υπάρχον δίκτυο κινητής τηλεφωνίας.

1.2.2 Σύγκριση ασύρματων τεχνολογιών

Οι βασικότεροι τομείς που καθιστούν μία ασύρματη τεχνολογία κατάλληλη για το ΔΤΠ (Διαδίκτυο των πραγμάτων) είναι το κόστος, η κατανάλωση ενέργειας, η κάλυψη/η εμβέλεια, η μέθοδος διάθεσης/υλοποίησης, ο ρυθμός μετάδοσης, και τυχόν latency στην μεταφορά δεδομένων.



Εικόνα 11 - Εμβέλεια τεχνολογιών ασύρματης δικτύωσης σε σύγκριση με ρυθμό μετάδοσης δεδομένων [9]

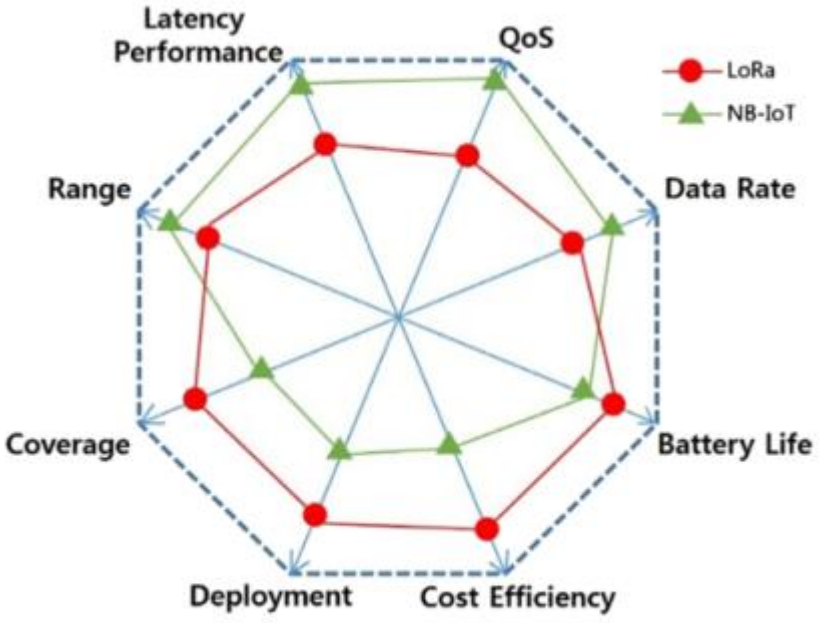
Οι εφαρμογές γεωργίας ακριβείας δεν απαιτούν μεγάλη ταχύτητα μετάδοσης δεδομένων από κόμβο σε κόμβο, αλλά αντίθετα απαιτούν μεγάλη εμβέλεια, και χαμηλή κατανάλωση ενέργειας.

Στον παρακάτω πίνακα, παρουσιάζεται το εύρος ζώνης, ο ρυθμός μετάδοσης δεδομένων, η εμβέλεια, η υποστήριξη multihop, καθώς και λοιπές παρατηρήσεις για επικρατούσες τεχνολογίες ασύρματων δικτύων αισθητήρων του διαδικτύου των πραγμάτων.

Technology	Bandwidth	Data rate	Range	Multihop	Remarks
LoRaWAN	125 kHz/ 250 kHz	250 bps–5.5 kbps/ 11 kbps/50 kbps	2–15 km	No	Open system specification
SigFox	100 Hz	100 bps	3–50 km	No	Single operator, 12 bytes messages
Weightless-N	200 Hz	100 bps	3 km	No	No downlink
Weightless-P	12.5 kHz	200 bps–100 kbps	2 km	No	Under development

Table 1 – Σύγκριση LoRaWan, Sigfox, Weightless-N και Weightless-P [10]

Οι δύο επικρατέστερες τεχνολογίες ασύρματης δικτύωσης χαμηλής κατανάλωσης ενέργειας το 2017, για χρήση σε εφαρμογές του διαδικτύου των πραγμάτων, είναι το LoRa και το NB-IoT.



Εικόνα 12 – Σύγκριση τεχνολογιών LoRa και NB-IoT για χρήση στο ΔΤΠ [11]

Όσον αφορά την κατανάλωση ενέργειας, οι κόμβοι με LoRa μπορούν να βρίσκονται σε sleep mode για όσο χρονικό διάστημα απαιτείται από την εφαρμογή, μιας και πρόκειται για ασύγχρονο πρωτόκολλο. Αντίθετα το NB-IoT απαιτεί τακτικό συγχρονισμό το οποίο καταναλώνει περισσότερη ενέργεια. Εκτός από τον χρόνο σε sleep mode, οι συσκευές με δικτύωση NB-IoT, λόγω της διαμόρφωσης OFDM (Orthogonal Frequency Division Multiplexing) απαιτούν μέγιστη ενέργεια 120-130mA, ενώ αυτές με LoRa 32mA. Πλεονέκτημα του NB-IoT αποτελεί ο πολύ μεγαλύτερος ρυθμός μετάδοσης δεδομένων, άνω των 200kbps, σε αντίθεση με του Lora, που είναι κάτω από 300bps.

Όσον αφορά την μέθοδο υλοποίησης/διάθεσης, το NB-IoT θα χρησιμοποιεί τις κυψέλες δικτύου και την μπάντα του LTE, το οποίο συνεπάγεται ότι είναι διαθέσιμο σε οποιοδήποτε μέρος υπάρχει ήδη κυψελωτή κάλυψη. Για την υλοποίηση ενός δικτύου LoRa απαιτείται ένα Gateway χαμηλού κόστους, το οποίο είναι ικανό να καλύψει σε απόσταση μία ολόκληρη πόλη. Η τεχνολογία NB-IoT θα είναι διαθέσιμη στις περισσότερες χώρες στα μέσα του 2019 επομένως οι τεχνολογίες LPWA όπως η LoRA θα συνεχίσουν να χρησιμοποιούνται σε ευρεία κλίμακα για IoT εφαρμογές. Ένα βασικό πλεονέκτημα της τεχνολογίας LoRA είναι το χαμηλό κόστος των πυλών δικτύου (gateways) που ξεκινούν από μερικές εκατοντάδες Ευρώ καθιστώντας την βέλτιστη για εφαρμογές τύπου Γεωργίας Ακριβείας όπου οι αγρότες δεν μπορούν να υποστηρίξουν μεγάλου κόστους ανάπτυξης και συντήρησης ασύρματων δικτύων αισθητήρων με σύνθετες πύλες δικτύου.

Parameters	LoRa	NB-IoT
Spectrum	Unlicensed	Licensed LTE bandwidth
Modulation	CSS	QPSK
Bandwidth	500–125 kHz	180 kHz
Peak data rate	290 bps–50 kbps (DL/UL)	DL:234.7 kbps; UL:204.8 kbps
Link budget	154 dB	150 dB
Max. # message/day	Unlimited	Unlimited
Duplex operation	–	Half duplex
Power efficiency	Very high	Medium high
Mobility	Better than NB-IoT	No connected mobility (only idle mode reselection)
Connection density	Utilized with NB-IoT	1500 km^2
Energy efficiency	>10 years battery life of devices	>10 years battery life of devices
Spectrum efficiency	Chirp SS CDMA better than FSK	Improved by standalone, in-band, guard band operation
Area traffic capacity	Depends on gateway type	40 devices per household, ~55k devices per cell
Interference immunity	Very high	Low
Peak current	32 mA	120–300 mA
Sleep current	1 μA	5 μA
Standardization	De-facto Standard	3GPP Rel.13 (planned)

Table 2 – Πίνακας σύγκρισης φυσικών παραμέτρων LoRa και NB-IoT [11]

1.3 Διαδίκτυο των πραγμάτων στην γεωργία

Η χρήση ασύρματων δικτύων αισθητήρων στο πλαίσιο του γεωργικού γίνεται όλο και πιο διαδεδομένη. Η χρήση ενός ασύρματου δικτύου απελευθερώνει τον αγρότη από την απαίτηση ύπαρξης καλωδίωσης σε ένα δύσκολο περιβάλλον. Συστήματα νερού, τροφοδοσίας, βαρύτητας, μπορούν να παρακολουθούνται χρησιμοποιώντας πομπούς πίεσης για να παρακολουθούν τα επίπεδα δεξαμενή νερού, αντλίες μπορούν να ελέγχονται με τη χρήση ασύρματων I / O συσκευών και η χρήση του νερού μπορεί να μετρηθεί και να μεταδίδεται ασύρματα σε ένα κεντρικό σημείο ελέγχου για τιμολόγηση. Ο Αυτοματισμός άρδευσης επιτρέπει την πιο αποτελεσματική χρήση του νερού και μειώνει τα απόβλητα.

Εικόνα 13 – Η αλυσίδα συλλογής δεοδμένων από αισθητήρες και εμφάνισης αυτών στον χρήστη [12]

1.3.1 Γεωργία ακριβείας

Τα ασύρματα δίκτυα αισθητήρων επιτρέπουν στους χρήστες να κάνουν ακριβή παρακολούθηση της καλλιέργειας κατά το χρόνο της ανάπτυξής της. Ως εκ τούτου, οι αγρότες μπορούν να γνωρίζουν άμεσα την κατάσταση του αντικειμένου σε όλα τα στάδια του, κάτι το οποίο θα διευκολύνει τη διαδικασία λήψης απόφασης σχετικά με το χρόνο της συγκομιδής. Συνολικά τα πλεονεκτήματα στην γεωργία ακριβείας από έξυπνες εφαρμογές γεωργίας, είναι τα εξής: Η κατανόηση των παραγόντων που επηρεάζουν την γεωργία και μειώνουν τις απώλειες σπαρτών από αρρώστιες, ή τις απότομες αλλαγές στον καιρό και σε συνθήκες περιβάλλοντος, έχουν ως συνέπεια την μεγιστοποίηση της παραγωγής.

Μειώνονται επίσης τα κόστη που απαιτούνται για χρήση λιπασμάτων και παρασιτοκτόνων, μιας και αυτά θα χρησιμοποιούνται μόνο όταν είναι όντως απαραίτητα, και όχι προληπτικά. Επίσης μειώνεται και το κόστος ποτίσματος μιας και βελτιστοποιείται η διαδικασία αυτή. Η δύσκολη δουλειά χειροκίνητου ελέγχου παραμέτρων στο χωράφι, και οι εργατοώρες που απαιτούνται για αυτό, επίσης μειώνονται. Τέλος, η διαδικασία συγκομιδής θα πραγματοποιείται στον βέλτιστο χρόνο, βελτιώνοντας την ποιότητα της παραγωγής. [13]

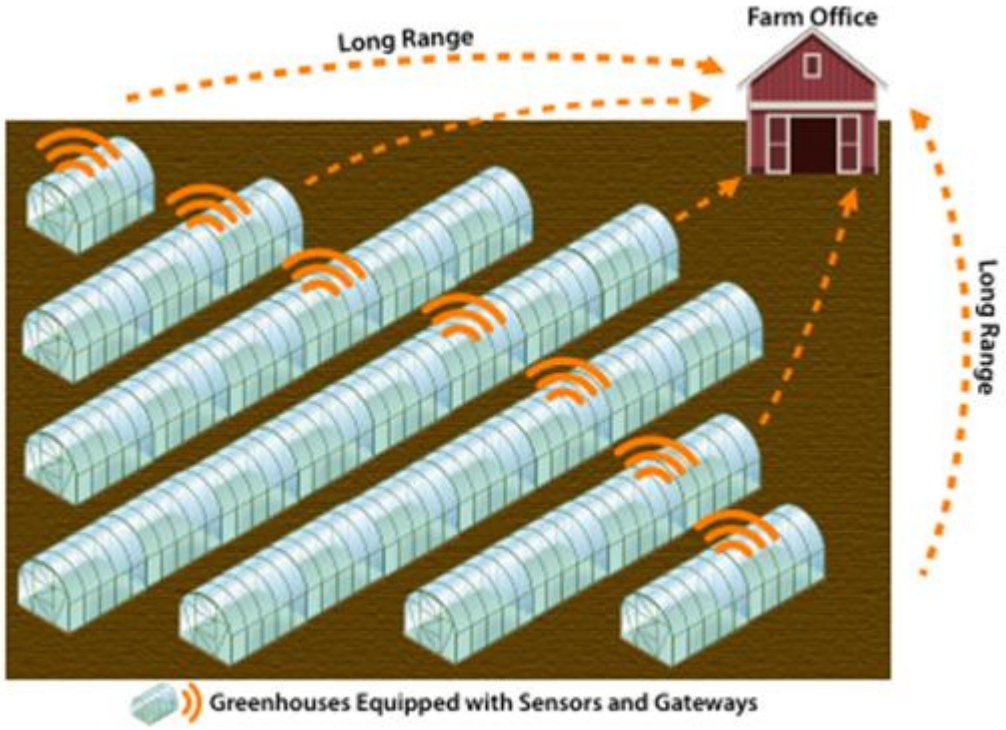
Εικόνα 14 – Το LoRaDrone για χρήση στην γεωργία που παρουσιάστηκε στην CES2017 [14]

1.3.2 Διαχείριση της άρδευσης

Όταν παραδίδονται τα δεδομένα σε πραγματικό χρόνο, οι αγρότες είναι σε θέση να επιτύχουν έξυπνη άρδευση. Τα στοιχεία που αφορούν τα πεδία, όπως η θερμοκρασία και το επίπεδο υγρασίας του εδάφους παραδίδονται στους αγρότες μέσω των ασύρματων δικτύων αισθητήρων. Όταν κάθε φυτό ενώνεται με ένα προσωπικό σύστημα άρδευσης, οι αγρότες μπορούν να παρέχουν το ακριβές ποσό του νερού που χρειάζεται κάθε φυτό και ως εκ τούτου, να επιτύχουν τη μείωση του κόστους και την βελτίωση της ποιότητας του τελικού προϊόντος. Τα δίκτυα μπορούν να χρησιμοποιηθούν για τη διαχείριση των διαφόρων ενεργοποιητών στα συστήματα χρησιμοποιώντας όχι ενσύρματη υποδομή.

1.3.3 Θερμοκήπια

Τα ασύρματα δίκτυα αισθητήρων, επίσης, να χρησιμοποιηθούν για να ελέγχουν τα επίπεδα θερμοκρασίας και υγρασίας στο εσωτερικό εμπορικών θερμοκηπίων. Όταν η θερμοκρασία και η υγρασία πέφτει κάτω από συγκεκριμένα επίπεδα, ο διαχειριστής του θερμοκηπίου πρέπει να ειδοποιείτε μέσω e-mail ή στο κινητό τηλέφωνο με μήνυμα κειμένου, ή τα συστήματα υποδοχής μπορεί να πυροδοτήσουν τα συστήματα υδρονέφωσης, να ανοίξουν τους αεραγωγούς, ενεργοποίηση των περσίδων, ή να ελέγξουν μια ευρεία ποικιλία αντιδράσεων του συστήματος. Πρόσφατες έρευνες σε ασύρματα δίκτυα αισθητήρων στη βιομηχανία γεωργίας δίνουν έμφαση στη χρήση της σε θερμοκήπια, ιδιαίτερα για τις μεγάλες εκμεταλλεύσεις με συγκεκριμένες καλλιέργειες. Τέτοια μικροκλίματα έχουν ανάγκη την διατήρηση ακριβούς κατάστασης καιρικών συνθηκών ανά πάσα στιγμή. Επιπλέον, με τη χρήση πολλαπλών κατανεμημένων αισθητήρων θα ελέγχεται καλύτερα η παραπάνω διαδικασία, σε ανοικτή επιφάνεια, καθώς και στο έδαφος.



Εικόνα 15 – Ασύρματοι κόμβοι μεγάλης εμβέλειας σε θερμοκήπια, αποστέλλουν δεδομένα σε σταθμό βάσης [12]

2. Θεωρητικό υπόβαθρο και προδιαγραφές συστήματος

Για την υλοποίηση ενός ασύρματου δικτύου αισθητήρων είναι απαραίτητη η χρήση πλακετών ανάπτυξης (development boards), τόσο για την δημιουργία κόμβων με αισθητήρες, όσο και για τον κεντρικό κόμβο/gateway που θα συλλέγει και θα μεταβιβάζει τα δεδομένα. Οι πλακέτες ανάπτυξης που θα επιλεγούν πρέπει να δύναται να δεχθούν επεκτάσεις έτσι ώστε να προστεθεί σε αυτές συνδεσιμότητα LoRa, να έχουν όσο το δυνατόν χαμηλότερο κόστος, επαρκή βιβλιογραφία, να είναι ανοιχτού υλικού ώστε να είναι ευκολότερη η παραμετροποίηση τους, χαμηλής κατανάλωσης ενέργειας, με επαρκή μνήμη, επεξεργαστική ισχύ, επαρκή διαθέσιμα I/O Pins, αλλά και ενσωματωμένο ADC (Analog to Digital Converter).

Για την επιλογή των αισθητήρων μελετήθηκαν παράμετροι όπως η επιλογή αναλογικών ή ψηφιακών, το κόστος, η γραμμικότητά τους, το εύρος λειτουργίας τους, η ακρίβειά τους, η διακριτική ικανότητά τους, η ευαισθησία τους και η απόκρισή τους.

2.1 Το Arduino Uno και ο μικροελεγκτής ATMega328P

Για τα nodes των αισθητήρων επιλέχθηκε το Arduino Uno. Αποτελεί μία πλακέτα ανάπτυξης βασισμένη στον μικροελεγκτή ATMega328P χαμηλό κόστους, και ανοιχτού υλικού (open hardware). Το γεγονός ότι αποτελεί ανοιχτό υλικό καθιστά οποιονδήποτε κατασκευαστή ικανό να το αναπαράγει και να το διαθέτει στην αγορά, ως «Arduino Compatible board», το οποίο μειώνει αρκετά το κόστος αγοράς υλοποιήσεών του από τρίτους κατασκευαστές. Διαθέτει 14 ψηφιακές εξόδους/εισόδους, 6 εκ των οποίων μπορούν να χρησιμοποιηθούν και για έξοδο κυματομορφών PWM (Pulse Width Modulation), 6 αναλογικές εισόδους με ADC ανάλυσης 10bit, και κρύσταλλο 16 MHz. [15] Ο μικροελεγκτής 328P διαθέτει 32kb μνήμης Flash, και 2kb SRAM (Static Random Access Memory). [16]

Εικόνα 16 – Το Arduino Uno με τον μικροελεγκτή ATMega328P [15]

Για την τροφοδοσία του μπορούν να χρησιμοποιηθούν είτε η θύρα USB (Universal Serial Bus((+5V), είτε το ενσωματωμένο jack, είτε το Vin pin το οποίο και αυτό επικοινωνεί με τον περιοριστή τάσης των 5V (voltage regulator), με τάση από 6 έως 20V. Η κατανάλωση

ενέργειας του μικροελεγτή όταν αυτός λειτουργεί στα 16 MHz στα 5V, είναι περίπου 17mA αλλά φυσικά οι πλατφόρμες αυτού του τύπου δεν είναι βελτιστοποιημένες για την κατανάλωση ενέργειας χωρίς δυνατότητα ελέγχου των ολοκληρωμένων που τις απαρτίζουν. Τέλος, η δυνατότητα που παρέχει το Arduino Uno να δεχθεί πλήθος από επεκτάσεις-Shields τα οποία ακολουθούν συγκεκριμένο πρότυπο (form factor), το καθιστά συμβατό με πλήθος επεκτάσεων και κατά συνέπεια και εφαρμογών. Συνεπώς κρίνεται ότι το Arduino Uno αποτελεί κατάλληλη επιλογή για τους κόμβους των αισθητήρων της παρούσας υλοποίησης, μιας και καλύπτει τις απαιτήσεις σε επεξεργαστική ισχύ, δυνατότητα επέκτασης και προσθήκης συνδεσιμότητας σε αυτό με άλλους κόμβους, ανάλυση στον ADC, πλήθος I/O Pins, μνήμης, και σε μικρότερο βαθμό κατανάλωσης ενέργειας και κόστους.

2.1.1 Ο προγραμματισμός του μικροελεγκτή ATMega328P

Ο προγραμματισμός του εν λόγω μικροελεγκτή στο Arduino Uno πραγματοποιείται μέσω USB, χρησιμοποιώντας το atmega8u2 chip (αντί για το FTDI FT232RL που χρησιμοποιούσαν οι παλιότερες εκδόσεις), το οποίο μετατρέπει την USB σε ασύγχρονη σειριακή θύρα τύπου UART (Universal Asynchronous Receiver Transmitter).

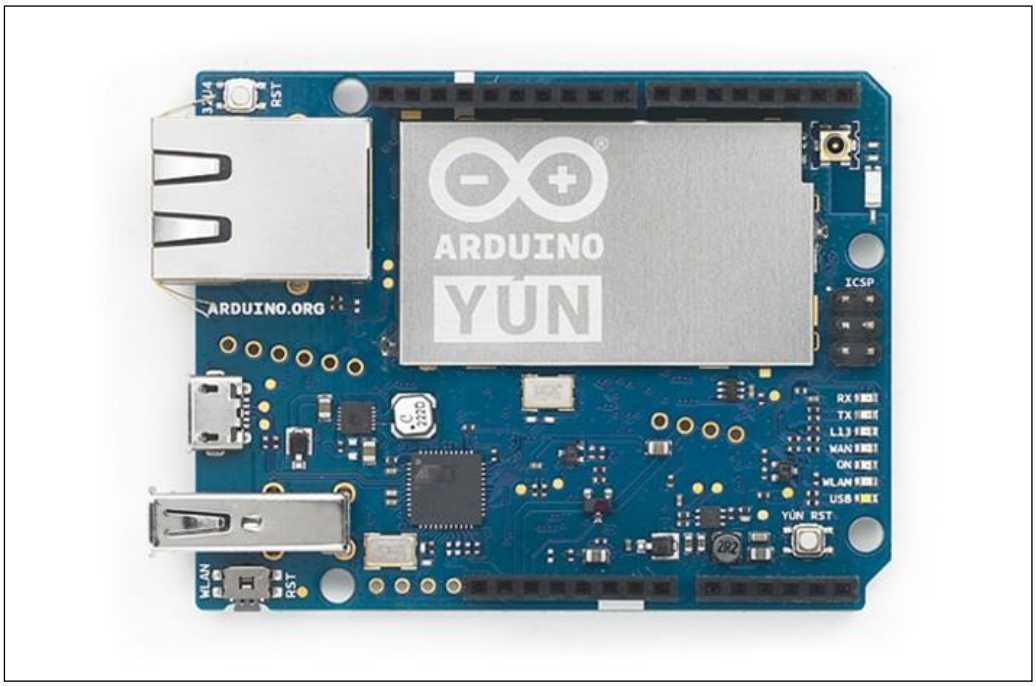
Ο προγραμματισμός του μπορεί να πραγματοποιηθεί είτε χρησιμοποιώντας C, είτε χρησιμοποιώντας C++ με τις έτοιμες συναρτήσεις που προσφέρει το επίπεδο αφαίρεσης υλικού του HAL (Hardware Abstraction Layer) (pinMode(), digitalWrite(), digitalRead(), analogReference(), analogRead(), analogWrite(), millis(), delay(), setup(), loop() , tone(), pulseIn(), micros(), shiftIn κ.α.). Το ενοποιημένο περιβάλλον ανάπτυξης IDE (Integrated Development Environment) του Arduino χρησιμοποιεί τον compiler avr-g++ ανοικτού κώδικα για το compilation του προγράμματος.

Η void setup() η οποία περιέχει κάθε sketch του Arduino που χρησιμοποιεί το abstruction layer του, εκτελείται μία φορά κατά την εκκίνηση, ενώ η void loop() εκτελείται καθ' επανάληψη και αντιστοιχεί στην ατέρμονη for(;;) loop που χρησιμοποιείται στα ενσωματωμένα συστήματα. Ενδεικτικά η δομή ενός βασικού Arduino sketch δίνεται παρακάτω.

```
//#include <stuff.h>
//#define stuff

void setup()
{
    //ekteleitai mia fora kata to bootup
}

void loop()
{
    //ekteleitai kat epanalipsi
    //antistoixi tis while(), h tis for(;;)
}
```

2.2 Το Arduino YUN, ο μικροελεγκτής ATmega32U4 και ο επεξεργαστής AR9331

Για τον κεντρικό κόμβο του ασύρματου δικτύου αισθητήρων, ο οποίος θα λειτουργεί και ως gateway προς το Internet, επιλέχθηκε το Arduino YUN. Αποτελεί μία πλακέτα ανάπτυξης βασισμένη στον μικροελεγκτή ATMega32U4, η οποία διαθέτει και τον μικροεπεξεργαστή της Atheros AR9331 ο οποίος εκτελεί μία διανομή Linux βασισμένη στο OpenWRT. Ταυτόχρονα, διαθέτει συνδεσιμότητα μέσω WiFi και Ethernet, καθώς και υποδοχές για κάρτα SD και USB-A. Αυτό την καθιστά κατάλληλη επιλογή για εφαρμογές που απαιτούν δικτύωση, όπως αυτές του διαδικτύου των πραγμάτων. [17]

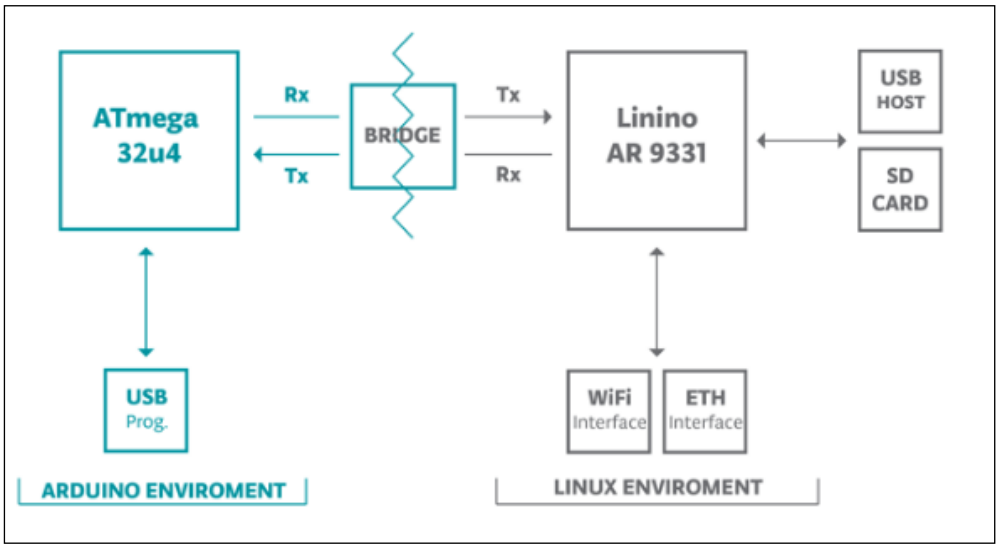


Εικόνα 17 - Το Arduino YUN [15]

Το Arduino YUN διαθέτει 20 ψηφιακές εισόδους/εξόδους, 16Mhz κρύσταλλο και 32kb μνήμης στον μικροελεγκτή. Αντίστοιχα η διαθέσιμη μνήμη για τον μικροεπεξεργαστή που εκτελεί την διανομή Linux είναι 16MB τύπου Flash και 64MB RAM (Random Access Memory) τύπου DDR2 (Double Data Rate 2). Προεγκατεστημένη στην διανομή Linux

υπάρχει και η 2.7 έκδοση της Python, η οποία επιτρέπει να γράψουμε σχετικά scripts εάν αυτό απαιτείται. Αντίστοιχα πακέτα μπορούν να εγκατασταθούν χρησιμοποιώντας τον Linux package manager, opkg. Ο προγραμματισμός του είναι συμβατός με το Abstraction Layer που αναφέρθηκε παραπάνω.

Υπάρχει σειριακή επικοινωνία μεταξύ του μικροελεγκτή ATMega32U4 και του μικροεπεξεργαστή AR9331 όπως παρουσιάζεται στην εικόνα παρακάτω. Η βιβλιοθήκη Bridge απλοποιεί την επικοινωνία αυτή, και εκτελεί προγράμματα στο Linux κομμάτι, όταν ζητηθεί από το κομμάτι του κώδικα του μικροελεγκτή. Ταυτόχρονα παρέχει αποθηκευτικό χώρο για μετρήσεις/δεδομένα, αλλά και συνδεσιμότητα προς το Internet.

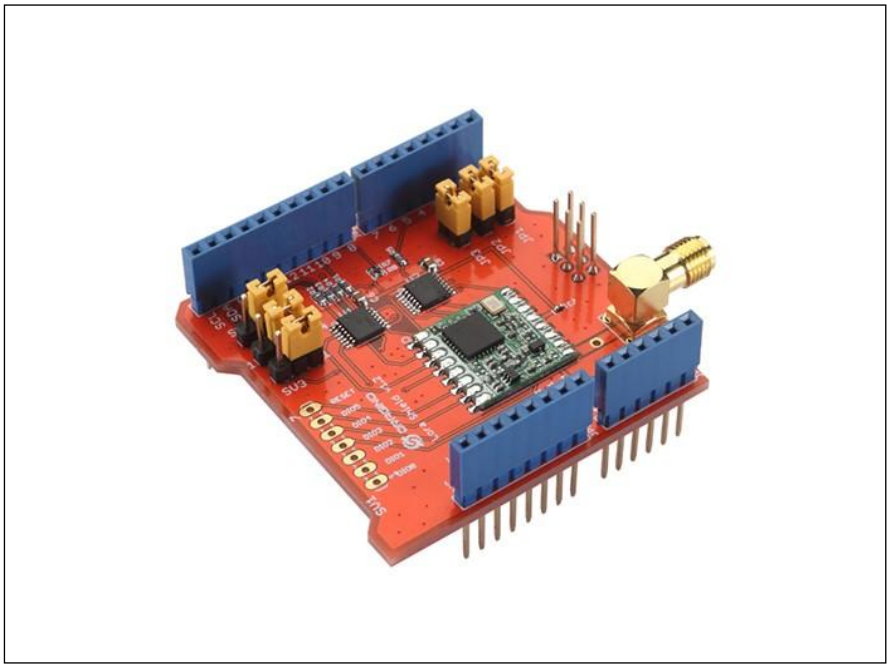


Εικόνα 18 - Η επικοινωνία του μικροελεγκτή με τον επεξεργαστή του Arduino YUN [17]

Τέλος, το Arduino YUN είναι και αυτό συμβατό με τα περισσότερα Shields της αγοράς, και με αυτό που επιλέχθηκε για συνδεσιμότητα LoRa στο υλοποιηθέν σύστημα. Συνεπώς κρίθηκε κατάλληλο για να λειτουργήσει ως κεντρικός κόμβος και gateway προς το διαδίκτυο για το σύστημα που υλοποιήθηκε, μια και καλύπτει τις απαιτήσεις ταχύτητας, μνήμης, συνδεσιμότητας και συμβατότητας.

2.3 Η Dragino LoRa Shield

Η προσθήκη συνδεσιμότητας LoRa στους κόμβους του ασύρματου δικτύου αισθητήρων έγινε με την χρήση της Dragino LoRa Shield. Αποτελεί έναν πομποδέκτη σχετικά χαμηλού κόστους ο οποίος επιτρέπει αποστολή και λήψη δεδομένων με χαμηλό ρυθμό σε πάρα πολύ μεγάλες αποστάσεις. Βασίζεται στο Module της Semtech SX1276 και χρησιμοποιεί την διαμόρφωση της HOPE RF, LoRATM.



Εικόνα 19 - Η Dragino LoRa Shield [18]

Η υψηλή ευαισθησία του δέκτη είναι -148dBm σε συνδυασμό με τον ενσωματωμένο ενισχυτή ισχύος καθιστούν εφικτή την συνδεσιμότητα σε πολύ μεγάλες αποστάσεις. Η ισχύς εκπομπής μπορεί να φτάσει τα 100mW (20dBm). Διαθέτει επίσης sma connector για κεραία για περαιτέρω επέκταση της εμβέλειας, εάν αυτό απαιτείται. Ο ρυθμός αποστολής δεδομένων μπορεί να φτάσει τα 300kbps το οποίο είναι επαρκέστατο για την εφαρμογή που θέλουμε να τα χρησιμοποιήσουμε. Ταυτόχρονα, είναι επιθυμητό στους κόμβους να έχουμε όσο το δυνατόν χαμηλότερη κατανάλωση ενέργειας, το οποίο επιτυγχάνεται στην εν λόγω Shield σε σχέση με ανταγωνιστικά προϊόντα. Η κατανάλωση ενέργειας κατά το

Sleep Mode είναι από 0,2-1μA, ενώ κατά την λήψη δεδομένων η κατανάλωση είναι περίπου 12mA. Σε μία τυπική ισχύ εκπομπής, 13dBm, η κατανάλωση ενέργειας είναι 29mA. Πλήρης πίνακας με όλα τις καταστάσεις λειτουργίας (sleep, idle, standby, synth, rx, tx) και κατανάλωση ενέργειας σε κάθε ένα, δίνεται παρακάτω:

Symbol	Description	Conditions	Min	Typ	Max	Unit
IDDSL	Supply current in Sleep mode		-	0.2	1	uA
IDDIDLE	Supply current in Idle mode	RC oscillator enabled	-	1.5	-	uA
IDDST	Supply current in Standby mode	Crystal oscillator enabled	-	1.6	1.8	mA
IDDFS	Supply current in Synthesizer mode	FSRx	-	5.8	-	mA
IDDR	Supply current in Receive mode	*LnaBoost* Off, band 1 *LnaBoost* On, band 1 Bands 2&3	- - -	10.8 11.5 12.0	- - -	mA
IDDT	Supply current in Transmit mode with impedance matching	RFOP = +20 dBm, on PA_BOOST RFOP = +17 dBm, on PA_BOOST RFOP = +13 dBm, on RFO_LF/HF pin RFOP = + 7 dBm, on RFO_LF/HF pin	- - - -	120 87 29 20	- - - -	mA mA mA mA

Table 3- Καταστάσεις λειτουργίας Dragino Shield και κατανάλωση ενέργειας [18]

Η συχνότητα λειτουργίας της εν λόγω Shield που επιλέχθηκε είναι τα 868 MHz, ενώ είναι διαθέσιμη και σε εκδόσεις στα 915 MHz και στα 433 MHz. Τέλος, η εν λόγω Shield είναι συμβατή με τα περισσότερα Arduino Boards της αγοράς (Uno, Mega, DUE, Leonardo, Yun). [18] Στην παρούσα υλοποίηση χρησιμοποιήθηκε με Arduino Uno στα client nodes και Arduino YUN στο Server/Gateway Node.

Συνεπώς η Dragino Shield καλύπτει τις απαιτήσεις που απαιτούνται στην παρούσα υλοποίηση, δηλαδή όσον αφορά το κόστος, την εμβέλεια, την κατανάλωση ενέργειας, το ρυθμό μετάδοσης αλλά και την συμβατότητα.

2.4 Η βιβλιοθήκη Radiohead Packet Radio

Για την αποστολή και λήψη δεδομένων με την χρήση του Dragino Shield το οποίο προσφέρει συνδεσιμότητα LoRa, έγινε χρήση της βιβλιοθήκης RadioHead Packet Radio. Αποτελεί μία βιβλιοθήκη για μικροελεγκτές σε ενσωματωμένα συστήματα που υποστηρίζει πλήθος πομποδεκτών. Αποτελείται από δύο βασικά σετ κλάσεων, των drivers και των managers. Οι drivers παρέχουν πρόσβαση χαμηλού επιπέδου στους πομποδέκτες, ενώ οι managers παρέχουν υψηλότερου επιπέδου δυνατότητες αποστολής και λήψης μηνυμάτων. Κάθε πρόγραμμα που χρησιμοποιεί την εν λόγω βιβλιοθήκη θα εκτελεί και driver για πρόσβαση στο υλικό του πομποδέκτη, και manager ο οποίος χρησιμοποιεί τον driver για αποστολή και λήψη μηνυμάτων για την εφαρμογή. Για την Dragino Shield έγινε χρήση του οδηγού (driver) RH_RF95, μιας και η εν λόγω Shield χρησιμοποιεί το Module της Semtech SX1276, με τεχνική διαμόρφωσης spread spectrum modulation and forward error correction techniques to increase the range and robustness of radio communication links compared to traditional FSK or OOK based modulation. Υποστηρίζει επίσης δυναμική αλλαγή συχνοτήτων (frequency hopping) και δυνατότητα αποστολής και λήψης μεγάλων payloads εάν αυτό απαιτείται από την εφαρμογή. [19]

Η επεξεργασία του αρχείου RH_RF95.cpp της βιβλιοθήκης μας δίνει την δυνατότητα να ορίσουμε την συχνότητα που επιθυμούμε να χρησιμοποιήσουμε –τα 868Mhz-, αλλά και παραμέτρους όπως η ισχύς εκπομπής:

```
...
// Orismos sixnotitas
setFrequency(868.0);

// Orismos isxios ekpompis, to 13 einai sxetika xamili timi
// Mporei na parei times apo 5 ews 23dBm
setTxPower(13);
...
```

Η ενσωμάτωση της βιβλιοθήκης στο Project, παράλληλα με την επίσης απαιτούμενη βιβλιοθήκη για επικοινωνία μέσω SPI (Serial Peripheral Interface) της Shield με το Arduino, αλλά και η εκκίνηση του radio driver γίνεται ως εξής:

```
#include <SPI.h>
#include <RH_RF95.h>

// Singleton instance of the radio driver
RH_RF95 rf95;

...

void setup()
{
...
if (!rf95.init())
    //Serial.println("init failed"); //gia debugging
...
}
```

Για την αποστολή δεδομένων από ένα node σε ένα άλλο, εκτελείται η παρακάτω ρουτίνα:

```
if (rf95.available())
  {
    ...
    // Apostoli dedomenwn
    uint8_t data[] = "Dedomena pros apostoli";
    rf95.send(data, sizeof(data));
    rf95.waitPacketSent();
    ...

   }
```

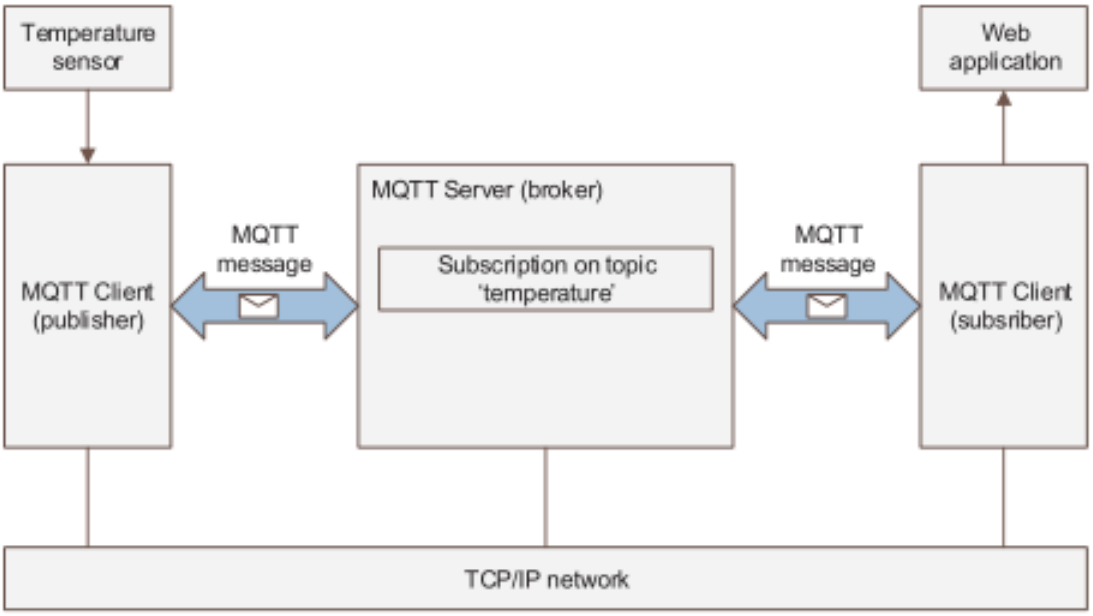
Αντίστοιχα ένα node λαμβάνει δεδομένα όταν εκτελείται η παρακάτω ρουτίνα:

```
if (rf95.available())
  {
    // Gia lipsi minimatos
    uint8_t buf[RH_RF95_MAX_MESSAGE_LEN];
    uint8_t len = sizeof(buf);

    if (rf95.recv(buf, &len))
    {

    // RH_RF95::printBuffer("request: ", buf, len);
      Serial.print("Elava to minima: ");
      Serial.println((char*)buf);
    }

    else
    {
    //  Serial.println("lipsi apetixe"); //gia debugging
    }

  }
}
```

2.5 Το πρωτόκολλο MQTT

Το MQTT αποτελεί ένα πρωτόκολλο συνδεσιμότητας συσκευής προς συσκευή. Σχεδιάστηκε και κατασκευάστηκε με σκοπό να είναι εξαιρετικά ελαφρύ και γρήγορο, καθιστώντας το ιδανικό για εφαρμογές του διαδικτύου των πραγμάτων. Για την χρήση του MQTT απαιτείται ένας Broker, δηλαδή ο Server ο οποίος θα διαχειρίζεται τα δεδομένα και την επικοινωνία μεταξύ των MQTT Clients. Ο σχεδιασμός του επιτρέπει να έχει ελάχιστο αντίκτυπο σε μνήμη και απαιτήσεις επεξεργαστικής ισχύος, οπότε και μικροελεγκτές των 8bit μπορούν να το υποστηρίξουν, ενώ ταυτόχρονα μπορεί να λειτουργήσει αποτελεσματικά με ασταθείς συνδέσεις χαμηλής ταχύτητας.

Κάθε αντικείμενο του ΔΤΠ μπορεί να είναι MQTT Client που στέλνει ή λαμβάνει δεδομένα. Ένας Client μπορεί να είναι κάθε συσκευή από μικροελεγκτή μέχρι ηλεκτρονικό υπολογιστή ή κινητό τηλέφωνο. Ο τύπος του κάθε MQTT Client (subscriber ή publisher) εξαρτάται από τον ρόλο του στο σύστημα, δηλαδή είτε παράγει, είτε λαμβάνει δεδομένα τηλεμετρίας. Και στις δύο περιπτώσεις, κάθε MQTT Client απαιτείται να συνδεθεί σε έναν Server (τον Broker). Αφού επιτευχθεί η σύνδεση, ο Client πρέπει να δηλώσει αν είναι subscriber ή publisher. Για τον διαχωρισμό των δεδομένων που στέλνει ένας publisher MQTT κόμβος, χρησιμοποιείται μία λέξη –ένα string- με το θέμα που αφορά. Για παράδειγμα αν ένας Client αποστέλει δεδομένα θερμοκρασίας και υγρασίας, μπορεί να χρησιμοποιήσει τα strings «temp» και «hum» για να διαχωρίσει τις τιμές ανά κατηγορία. Αντίστοιχα, αν ένας Client λαμβάνει δεδομένα, πρέπει να κάνει subscribe σε ένα (τουλάχιστον) θέμα. Για την δημιουργία ενός MQTT Client, απαιτείται η χρήση αντίστοιχης βιβλιοθήκης αναλόγως συστήματος, αλλά και σύνδεση με έναν MQTT Broker, η λειτουργία του οποίου περιγράφεται παρακάτω.



Εικόνα 20 – Χρήση του MQTT για αποστολή και λήψη δεδομένων στο θέμα temperature μέσω του απαιτούμενου Broker [20]

Ένας MQTT Broker είναι μία κεντρική συσκευή η οποία αναλαμβάνει την επικοινωνία και την διαχείριση των συνδέσεων και της ανταλλαγής μηνυμάτων μεταξύ των MQTT Clients. Κάθε Broker μπορεί να υποστηρίξει χιλιάδες συνδέσεις από MQTT Clients ταυτόχρονα. Όταν ένα μήνυμα ληφθεί, ο Broker πρέπει να βρει όλες τις συσκευές οι οποίες έχουν κάνει subscribe στο συγκεκριμένο θέμα (topic), και να τους το αποστείλει. Λοιπές διεργασίες που πραγματοποιεί ένας Broker περιλαμβάνουν την επικύρωση των στοιχείων εισόδου ενός χρήστη, το οποίο μπορεί και να περιλαμβάνει το εάν αυτός ο χρήστης έχει την δυνατότητα μόνο να λάβει, ή και να λάβει και να αποστείλει δεδομένα. [20]

Το MQTT πατάει πάνω στην στοίβα του πρωτοκόλλου TCP/IP και υποστηρίζει τις παρακάτω μεθόδους:

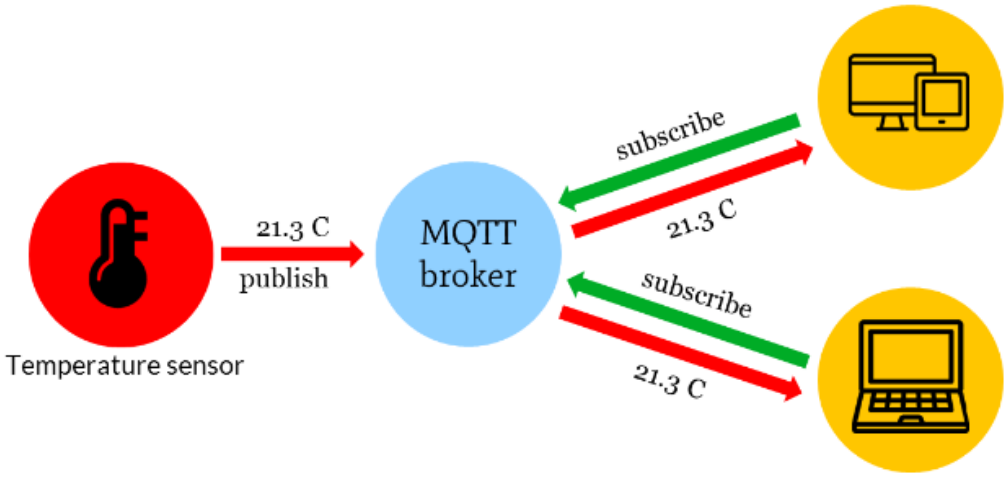
Connect: Εκτελεί σύνδεση με τον server

Disconnect, Αναμένει ο MQTT Client να τελειώσει τις ενέργειες που εκτελεί, και στην συνέχεια αποσυνδέει το TCP/IP (Transmission Control Protocol/Internet Protocol) session

Subscribe: Χρησιμοποιείται για εγγραφή σε topics, από τα οποία οι clients λαμβάνουν δεδομένα

Unsubscribe: Χρησιμοποιείται έτσι ώστε ο ο Client να διαγραφεί από topics από τα οποία ελάμβανε δεδομένα

Publish: Χρησιμοποιείται ώστε οι Clients να αποστέλλουν δεδομένα στον server/broker, τα οποία στην συνέχεια είναι προσβάσιμα από τον χρήστη, ή και άλλους Clients.



Εικόνα 21 – Χρήση MQTT Broker, Publish και Subscribe [21]

Το MQTT υπερτερεί έναντι των υλοποιήσεων HTTP (Hypertext Transfer Protocol) και REST (Representational State Transfer), μιας και δεν απαιτείται ο Client να ζητά δεδομένα –ή το αν υπάρχει κάποια αλλαγή- από τον Server, αλλά μπορεί ο Server να στείλει όποτε εκείνος θέλει δεδομένα στον Client μέσω συνεχώς ανοιχτής TCP (Transmission Control Protocol) σύνδεσης. [22]

Το MQTT διασφαλίζει την αξιόπιστη μετάδοση μηνυμάτων διαθέτοντας τρία επίπεδα διασφάλισης ποιότητας υπηρεσιών (QoS - Quality of Service). Στο QoS=0 ο publisher στέλνει ένα μήνυμα μία φορά, και δεν ελέγχει εάν έφτασε στον προορισμό του. Δεν

αποστέλλεται απάντηση από τον δέκτη, και καμία επαναποστολή δεν πραγματοποιείται. Αυτό το κατώτερο επίπεδο, περιγράφεται ως «αποστέλλω και ξεχνώ» και το μήνυμα μπορεί όντως να χαθεί αναλόγως της κατάστασης του δικτύου. Στο QoS=1 διασφαλίζεται ότι το μήνυμα θα φτάσει τουλάχιστον μία φορά στον παραλήπτη. Ο αποστολέας στέλνει το μήνυμα και ελέγχει για την λήψη του μέσω του PUBACK μηνύματος. Εάν το PUBACK μήνυμα χαθεί, ο Broker επαναλαμβάνει την αποστολή του μηνύματος, μέχρι να ληφθεί το PUBACK. Στο επίπεδο QoS=2, κάθε μήνυμα λαμβάνεται μία ακριβώς φορά χάρη σε χειραψία τεσσάρων κατευθύνσεων (4 way handshake). Σε αυτό το επίπεδο μπορεί η αποστολή να καθυστερεί λίγο παραπάνω, αλλά δεν χάνεται το μήνυμα.

2.5.1 Η βιβλιοθήκη Adafruit_MQTT_Library

Για την χρήση του πρωτοκόλλου MQTT για αποστολή και λήψη πληροφοριών από και προς κάποια υπηρεσία cloud, έγινε χρήση της βιβλιοθήκης Adafruit_MQTT_Library. Είναι συμβατή με οποιονδήποτε broker που ακολουθεί τις προδιαγραφές των εκδόσεων του MQTT 3, ή 3.1.1. Η εν λόγω βιβλιοθήκη υποστηρίζει το Arduino YUN, το Adafruit FONA, το ESP8266, καθώς όλες τις Shields συμβατές με το Client Interface του Arduino, όπως π.χ. η Ethernet Shield. [23]

Για την ενσωμάτωση της βιβλιοθήκης απαιτείται να κάνουμε include τα αρχεία Adafruit_MQTT.h και Adafruit_MQTT_Client.h στο Arduino sketch, ως εξής:

```
#include "Adafruit_MQTT.h"
#include "Adafruit_MQTT_Client.h"
...
```

Στην συνέχεια απαιτείται η δήλωση των παραμέτρων του MQTT Broker που θα χρησιμοποιήσουμε, καθώς και τα credentials για τον server αυτόν. Για παράδειγμα αν γίνει χρήση της υπηρεσίας Cloud της Adafruit τα definitions θα είναι τα εξής:

```
#define AIO_SERVER      "io.adafruit.com"
#define AIO_SERVERPORT  1883
#define AIO_USERNAME    "...το όνομα χρήστη..."
#define AIO_KEY         "...ο κωδικός χρήστη..."
```

Έπειτα περνάμε τις παραπάνω παραμέτρους στο Adafruit_MQTT_Client

```
Adafruit_MQTT_Client mqtt
(&client, AIO_SERVER, AIO_SERVERPORT, AIO_USERNAME, AIO_KEY);
```

Δημιουργούμε τα Adafruit_MQTT_Publish objects που επιθυμούμε, για τα topics στα οποία θα θελήσουμε να κάνουμε Publish δεδομένα στον broker. Αντίστοιχα,

δημιουργούμε και τα Adafruit_MQTT_Subscribe objects, για τα feeds στα οποία θέλουμε να κάνουμε subscribe για λήψη δεδομένων από τον broker. Ενδεικτικά ο κώδικας για την δημιουργία του αντικειμένου sensor για αποστολή δεδομένων, και του αντικειμένου button για την λήψη δεδομένων, δίνεται παρακάτω:

```
Adafruit_MQTT_Publish sensor = Adafruit_MQTT_Publish
(&mqtt, AIO_USERNAME "/feeds/sensor");

Adafruit_MQTT_Subscribe button = Adafruit_MQTT_Subscribe
(&mqtt, AIO_USERNAME "/feeds/button");
```

Συνεπώς το directory του feed στον server στον οποίο αποθηκεύονται τα δεδομένα από τον αισθητήρα είναι το <username>/feeds/sensor, ενώ το directory το οποίο έχει τα δεδομένα της θέσης του διαδικτυακού διακόπτη είναι το <username>/feeds/button.

Εντός της void setup(), της συνάρτησης δηλαδή του Arduino η οποία εκτελείται μία φορά κατά την εκκίνηση κάνουμε subscribe στα subscribe feeds που δημιουργήσαμε:

```
void setup() {
...

mqtt.subscribe(&button);
...

}
```

Εντός της void loop(), της συνάρτησης δηλαδή του Arduino που εκτελείται καθ' επανάληψη, απαιτείται να εκτελεστεί η MQTT_connect(), η οποία επιβεβαιώνει την σύνδεση με τον MQTT Server. Σε περίπτωση που ο Client έχει αποσυνδεθεί από τον Server, η εν λόγω συνάρτηση προχωρά σε επανασύνδεση.

```
void MQTT_connect() {

int8_t ret;


// Έλεγχος αν υπάρχει σύνδεση, είμαστε OK αν υπάρχει
  if (mqtt.connected()) {
    return;
  }

while ((ret = mqtt.connect()) != 0) {
       // Serial.println(mqtt.connectErrorString(ret));
       mqtt.disconnect();
       delay(5000);  // αναμονή 5 δευτ. και νέα προσπάθεια
  }
}
```

Για τον έλεγχο των εισερχόμενων subscriptions, αρχικά δημιουργούμε έναν pointer στο Adafruit_MQTT_Subsribe object.

```
Adafruit_MQTT_Subscribe *subscription;
```

Με τον τρόπο αυτό γίνεται εφικτό να καθορίσουμε για ποιο subscription λάβαμε δεδομένα. Στην συνέχεια εκτελείται η mqtt.readSubscription(<χρόνος>) η οποία αναμένει την λήψη δεδομένων για τα milliseconds που έχει ορίσει ο προγραμματιστής στην παράμετρο <χρόνος>. Εάν γίνει λήψη μηνύματος πριν τον χρόνο αυτό, θα επιστραφεί pointer στον subscription. Εάν δεν ληφθεί μήνυμα στον χρόνο αυτό, θα επιστρέψει 0. Στην περίπτωση που γίνει λήψη δεδομένων, ελέγχουμε για ποιο από τα subscriptions ήταν. Τα δεδομένα που ελήφθησαν βρίσκονται στο feedobject.lastread, τα οποία στην συνέχεια μπορούμε να τα χρησιμοποιήσουμε στο sketch. Ενδεικτικά:

```
while ((subscription = mqtt.readSubscription(3000)))
{
    //anamoni 3 defteroleptwn gia lipsi
```

```
      if (subscription == &onoffbutton) {
      //siggrisi me gnwsta subscriptions

      Serial.print(F("Got: "));
      Serial.println((char *)onoffbutton.lastread);
      //emfanisi dedomenwn se serial output
    }
 }
```

Στην περίπτωση που δεν κάνουμε publish δεδομένα στον broker, αλλά μόνο subscribe σε αυτόν, απαιτείται να στέλνουμε κάθε 300 δευτερόλεπτα ping στον server για να διατηρηθεί ενεργή η σύνδεση. Το ίδιο απαιτείται και στην περίπτωση που στέλνουμε ή λαμβάνουμε δεδομένα με πολύ αργό ρυθμό από τον server, δηλαδή μεταξύ κάθε αποστολής ή λήψης να παρεμβάλλεται χρονικό διάστημα άνω των 300 δευτερολέπτων.

```
  // ping ston server
  if(! mqtt.ping()) {

    //Serial.println(F("Ping apetixe"));
  }
```

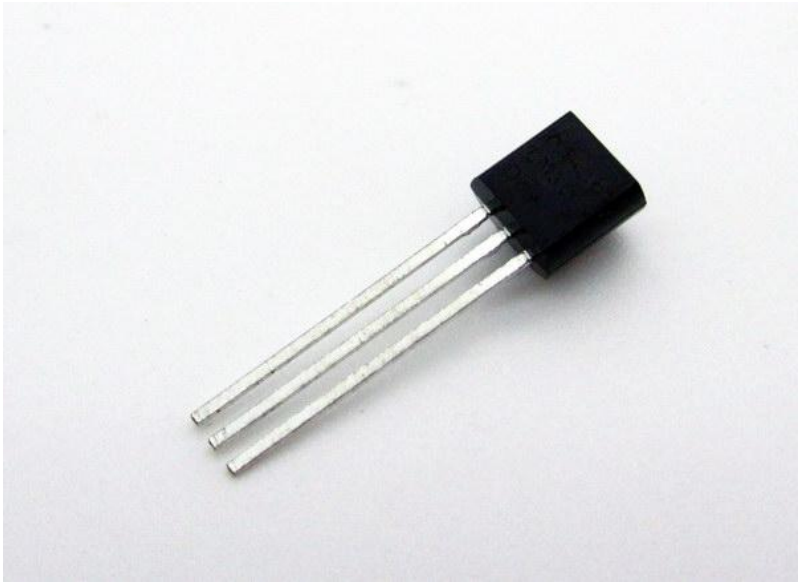
Η επεξεργασία της βιβλιοθήκης Adafruit_MQTT.h μας δίνει την δυνατότητα να αλλάξουμε ορισμένες παραμέτρους, όπως το μέγιστο μέγεθος πακέτων προς αποστολή, το χρονικό timeout των συνδέσεων, τον μέγιστο αριθμό συνδρομών, τις προδιαγραφές που ακολουθεί ο broker με τον οποίο συνδεόμαστε.

```
...
// Use 3 (MQTT 3.0) or 4 (MQTT 3.1.1)
#define MQTT_PROTOCOL_LEVEL 4
...
// Adjust as necessary, in seconds.  Default to 5 minutes.
#define MQTT_CONN_KEEPALIVE 300
...
// how many subscriptions we want to be able to track
```

```
#define MAXSUBSCRIPTIONS 5
...
// Largest full packet we're able to send.
// Need to be able to store at least ~90 chars for a connect packet with full
// 23 char client ID.
#define MAXBUFFERSIZE (150)
...
```

2.6 Ο αισθητήρας θερμοκρασίας LM35D

Για την μέτρηση θερμοκρασίας περιβάλλοντος από τους διάφορους κόμβους του συστήματος, επιλέχθηκε ο LM35D της Texas Instruments. Αποτελεί έναν αναλογικό αισθητήρα εξαιρετικά χαμηλού με γραμμική έξοδο τάσης. Δεν χρειάζεται κάποια περαιτέρω βαθμονόμηση μετά την αγορά του.



Εικόνα 22 - Ο αισθητήρας θερμοκρασίας LM35D

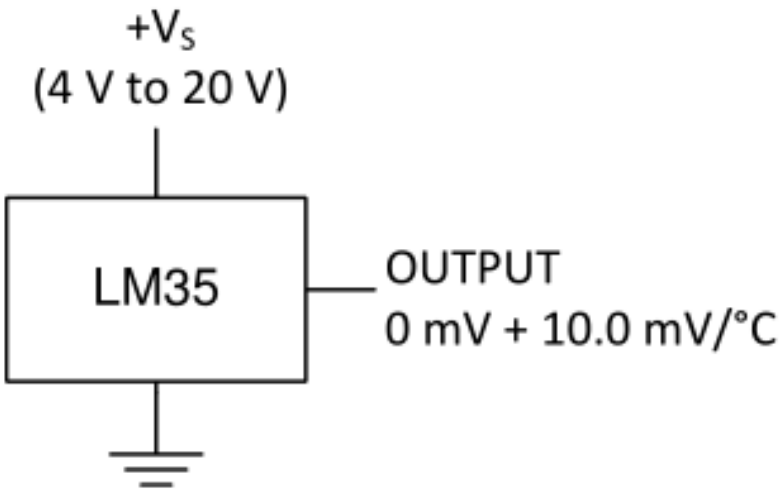
Η τυπική ακρίβειά του LM35 είναι ±0.25 °C σε θερμοκρασίες δωματίου, και ±¾°C σε όλο το εύρος τιμών που μπορεί να μετρήσει, δηλαδή από -55°C έως 150°C (0 έως 100 για τον LM35D). Ταυτόχρονα η κατανάλωση ενέργειάς του είναι μόλις 60μA, το οποίο συνεπάγεται και πολύ μικρή ιδιο-θέρμανση, της τάξης του 0.1°C. Δύναται να τροφοδοτηθεί από +4V έως +30V, οπότε και το +5V Pin του Arduino κρίνεται επαρκές.

		MIN	MAX	UNIT
Specified operating temperature: T_{MIN} to T_{MAX}	LM35, LM35A	–55	150	°C
	LM35C, LM35CA	–40	110	
	LM35D	0	100	
Supply Voltage ($+V_S$)		4	30	V

Table 4 – Παροχή τάσης και εύρος αισθητήρα LM35D

Όσον αφορά την έξοδο του αισθητήρα, για θερμοκρασία 0°C έχουμε 0V τάσης εξόδου, ενώ για θερμοκρασία 100°C έχουμε 1V τάσης εξόδου. Όπως αναφέρθηκε, η έξοδος του είναι γραμμική, το οποίο συνεπάγεται 10mV/°C. [24]

Το σχηματικό διάγραμμα της βασικής συνδεσμολογίας του LM35 δίνεται παρακάτω:

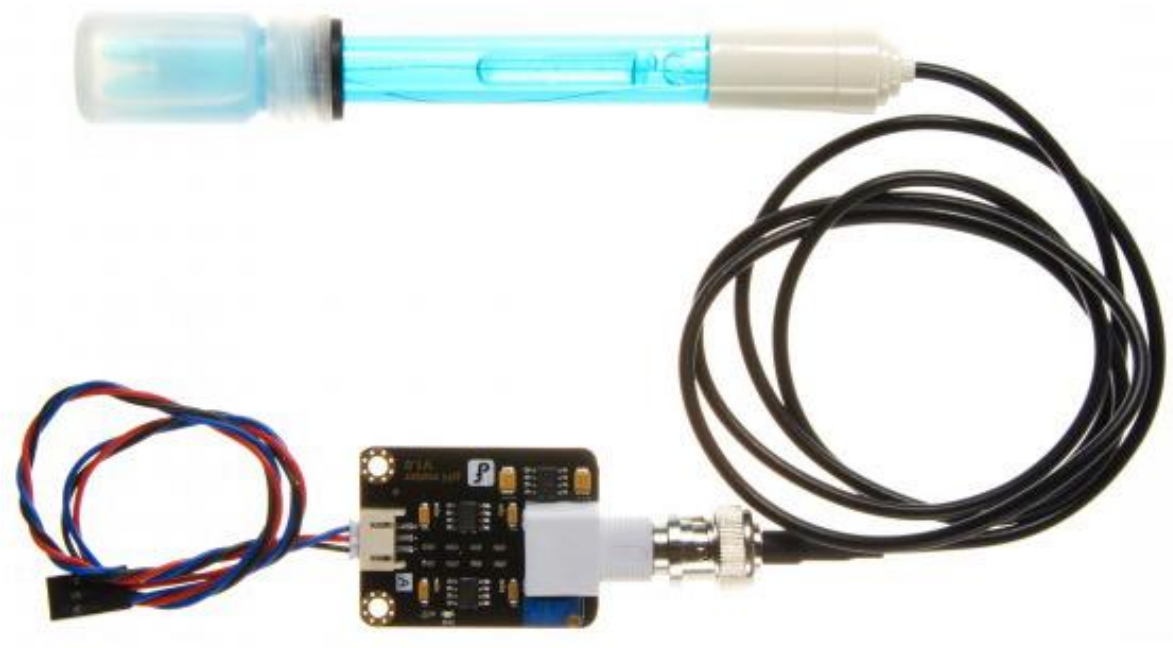


Εικόνα 23- Σχηματικό διάγραμμα σύνδεσης αισθητήρα LM35 [24]

Συνεπώς ο LM35D καλύπτει τις απαιτήσεις του συστήματός μας όσον αφορά το εύρος, το κόστος, την απλότητα χρήσης που προσφέρει η γραμμική απόκριση αλλά και την χαμηλή κατανάλωση ενέργειάς του.

2.7 Ο αισθητήρας οξύτητας Gravity PH Sensor της DFRobot

Για την μέτρηση της οξύτητας επιλέχθηκε ο αναλογικός αισθητήρας χαμηλού κόστους της DFRobot, Gravity PH Sensor. Μπορεί να μετρήσει όλο το εύρος οξύτητας 0 έως 14PH, και σε θερμοκρασίες από 0°C έως 60°C. Η ακρίβειά του είναι 0.1PH στους 25°C. Απαιτείται τροφοδοσία 5V, οπότε και δύναται να χρησιμοποιηθεί η +5V έξοδος του Arduino.



Εικόνα 24 - Ο αισθητήρας οξύτητας DFRobot PH Sensor [25]

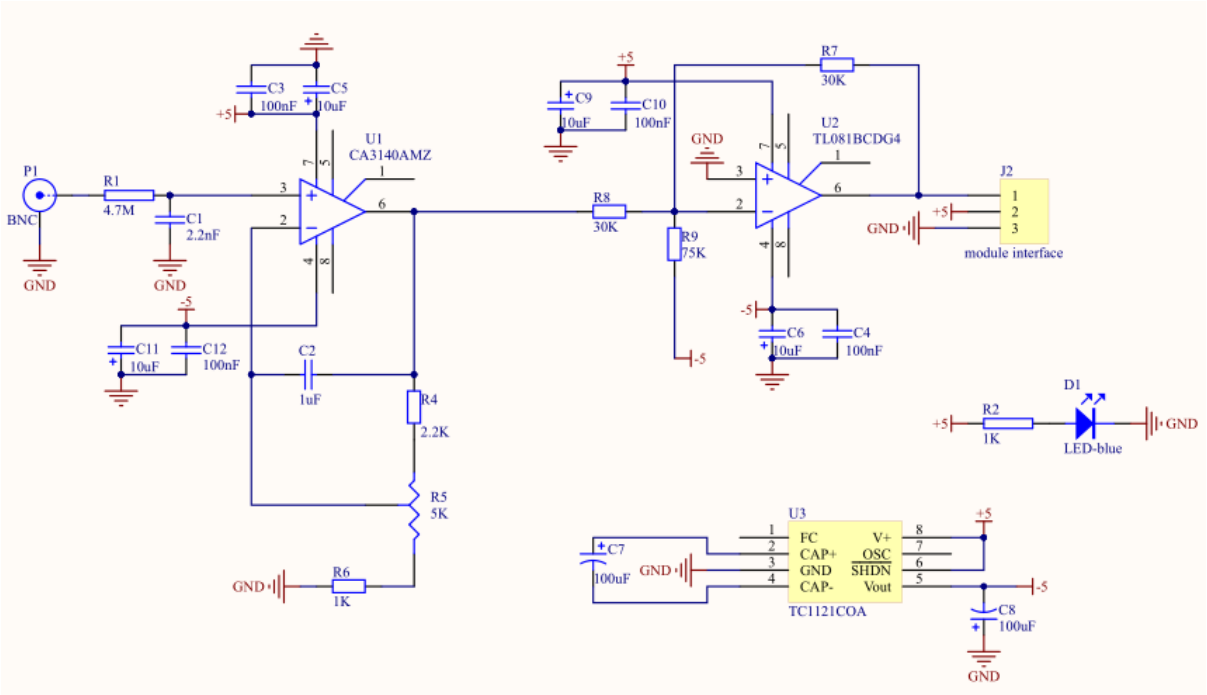
Η απόκριση του αισθητήρα είναι περίπου 60 δευτερόλεπτα, το οποίο αποτελεί τυπική απόκριση ακόμα και για ακριβότερους αισθητήρες της κατηγορίας. Συνδέεται μέσω BNC (Bayonet Neill–Concelman) σε Module με LED (Light Emitting Diode) λειτουργίας και κύκλωμα προσαρμογής, και από εκεί η αναλογική έξοδος μπορεί να οδηγηθεί στο Arduino. Για την βελτιστοποίηση της ακρίβειας απαιτείται διαδικασία βαθμονόμησης του αισθητήρα. [25]

Ο πίνακας αντιστοιχίας τάσης εξόδου του αισθητήρα, και μετρήσεων οξύτητας σε PH, το οποίο θα απαιτηθεί κατά τον προγραμματισμό του Arduino για μετατροπή της μέτρησης στον ADC σε PH, δίνεται παρακάτω:

VOLTAGE (mV)	pH value	VOLTAGE (mV)	pH value
414.12	0.00	-414.12	14.00
354.96	1.00	-354.96	13.00
295.80	2.00	-295.80	12.00
236.64	3.00	-236.64	11.00
177.48	4.00	-177.48	10.00
118.32	5.00	-118.32	9.00
59.16	6.00	-59.16	8.00
0.00	7.00	0.00	7.00

Table 5 - Αντιστοιχία τάσης εξόδου και PH στον αισθητήρα οξύτητας [25]

Τέλος, ο εν λόγω αισθητήρας αποτελεί υλικό ανοιχτού κώδικα, το σχηματικό του οποίου δίνεται παρακάτω.



Εικόνα 25 - Σχηματικό διάγραμμα του αισθητήρα οξύτητας χώματος [25]

Συνεπώς κρίνεται ότι ο αισθητήρας της DFRobot, Gravity PH Sensor, καλύπτει τις απαιτήσεις ακρίβειας, κόστους και απόκρισης που απαιτείται από τις προδιαγραφές που έχουν τεθεί για το σύστημα. Η διαδικασία βαθμονόμησης που απαιτείται κάθε 6 μήνες για μεγιστοποίηση της ακρίβειας, αποτελεί τυπική τακτική σε όλους τους αισθητήρες της κατηγορίας ακόμα και πολλαπλάσιου κόστους και δεν ήταν εφικτό να αποφευχθεί.

2.8 Ο αισθητήρας υγρασίας χώματος FC-28D

Για την μέτρηση του ποσοστού της υγρασίας στο χώμα, επιλέχθηκε ο αισθητήρας υγρασίας χώματος FC-28D. Ο εν λόγω αισθητήρας μετράει τον ογκομετρικό περιεχόμενο νερού στον χώμα, και επιστρέφει το επίπεδο υγρασίας. Ο αισθητήρας διαθέτει και αναλογική και ψηφιακή έξοδο. Τα δύο probes του αισθητήρα επιτρέπουν ηλεκτρικό ρεύμα να περάσει από το χώμα και μετρώντας την αντίσταση δύναται να βρεθεί η υγρασία στο χώμα. Όταν υπάρχει περισσότερο νερό, το χώμα άγει και υπάρχει μικρότερη αντίσταση, συνεπώς και μεγαλύτερη υγρασία. Αντίστοιχα, όταν το χώμα άγει λιγότερο, υπάρχει μικρότερο ποσοστό υγρασίας. [26]

Εικόνα 26 - Ο αισθητήρας υγρασίας χώματος FC28

Η τάση λειτουργίας του συγκεκριμένου αισθητήρα είναι από 3,3 έως 5V, συνεπώς δύναται να συνδεθεί είτε στην έξοδο +3,3V, είτε στην +5V του Arduino. Η κατανάλωση ενέργειας του αισθητήρα είναι περίπου 35mA. Το module με το κύκλωμα προσαρμογής διαθέτει και ποτενσιόμετρο με το οποίο μπορεί να οριστεί μια οριακή τιμή, η οποία θα συγκρίνεται από

τον LM393 συγκριτή. Όταν η τιμή αυτή ξεπεραστεί, το ενσωματωμένο LED στο module αυτό θα ανάψει. Οι τιμές από 0 έως 1023 στην έξοδο του αισθητήρα (στην περίπτωση που έχουμε ADC 10bit ακρίβειας όπως στο Arduino), αντιστοιχούν σε τιμές 0 έως 100% υγρασίας χώματος.

Συνεπώς ο αισθητήρας υγρασίας χώματος FC28 καλύπτει τις απαιτήσεις κόστους, κατανάλωσης ενέργειας, ευκολίας χρήσης και ακρίβειας που απαιτούνται από το σύστημα.

2.9 Ο αισθητήρας φωτεινότητας GL5539

Ο αισθητήρας GL5539 αποτελεί μία φωτοαντίσταση φτιαγμένη από ημιαγώγιμο υλικό, του οποίου η αγωγιμότητα αλλάζει με την αλλαγή της φωτεινότητας. Είναι ανθεκτικός, εξαιρετικά χαμηλού κόστους και δύναται να λειτουργήσει σε μεγάλο εύρος θερμοκρασιών, από -30°C έως +70°C. Η ταχύτητα απόκρισής του είναι της τάξης των 20-30ms, ενώ αλλάζοντας την τιμή της αντίστασης του διαιρέτη τάσης στο κύκλωμα προσαρμογής που απαιτείται, δύναται να μετρηθεί φωτεινότητα μεγαλύτερη και των 500lux. [27]

Εικόνα 27 - Η φωτοαντίσταση GL5539

Συνεπώς κρίνεται ότι ο αισθητήρας φωτεινότητας GL5539 καλύπτει τις απαιτήσεις κόστους, εύρους λειτουργίας, ανθεκτικότητας και ταχύτητας απόκρισης που απαιτούνται από την παρούσα υλοποίηση.

2.10 Λοιπά υλικά (relays, αντλία νερού, φωτισμός)

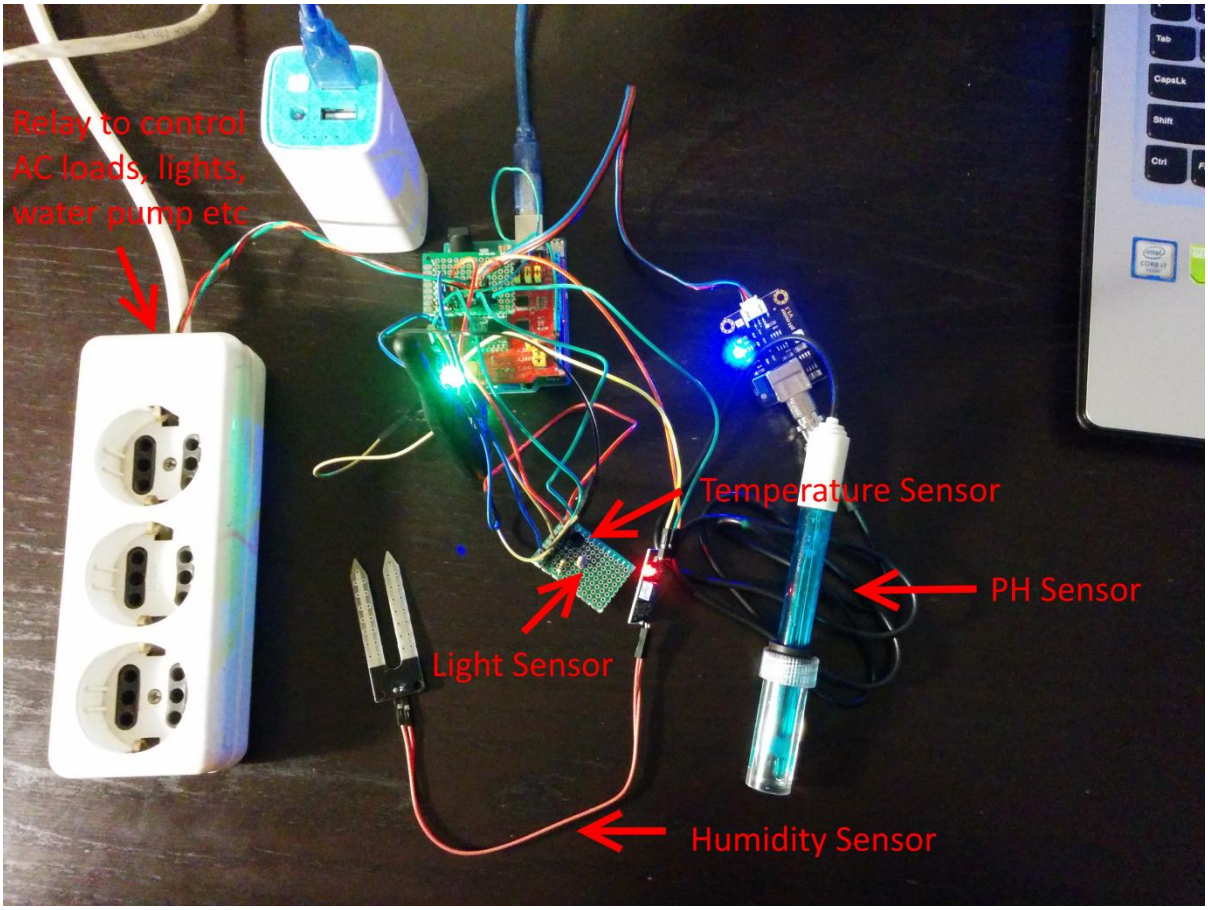
Λοιπά υλικά για τα οποία έγινε έρευνα αγοράς αποτελούν οι αντλίες νερού και οι λάμπες φωτισμού, τα οποία δύναται να ελεγχτούν μέσω του Arduino με Relays. Τα υλικά που επιλέχθηκαν δεν είναι σε θέση να καλύψουν κάποια full-scale υλοποίηση καθώς είναι χαμηλού κόστους και απόδοσης, και δεν θα μπορούσαν να ανταπεξέλθουν σε καθημερινή χρήση σε αγροτικό περιβάλλον. Ενσωματώθηκαν στο project μιας και η διαδικασία ελέγχου τους μέσω relays είναι ακριβώς ίδια με αυτήν πολύ ακριβότερων βιομηχανικών προϊόντων. Συνεπώς ο έλεγχος μίας ακριβότερης αντλίας νερού, πραγματοποιείται με τον ίδιο τρόπο με την ενδεικτική παρούσα υλοποίηση.

Εικόνα 28 - Αντλία νερού και relay για έλεγχό της μέσω του Arduino

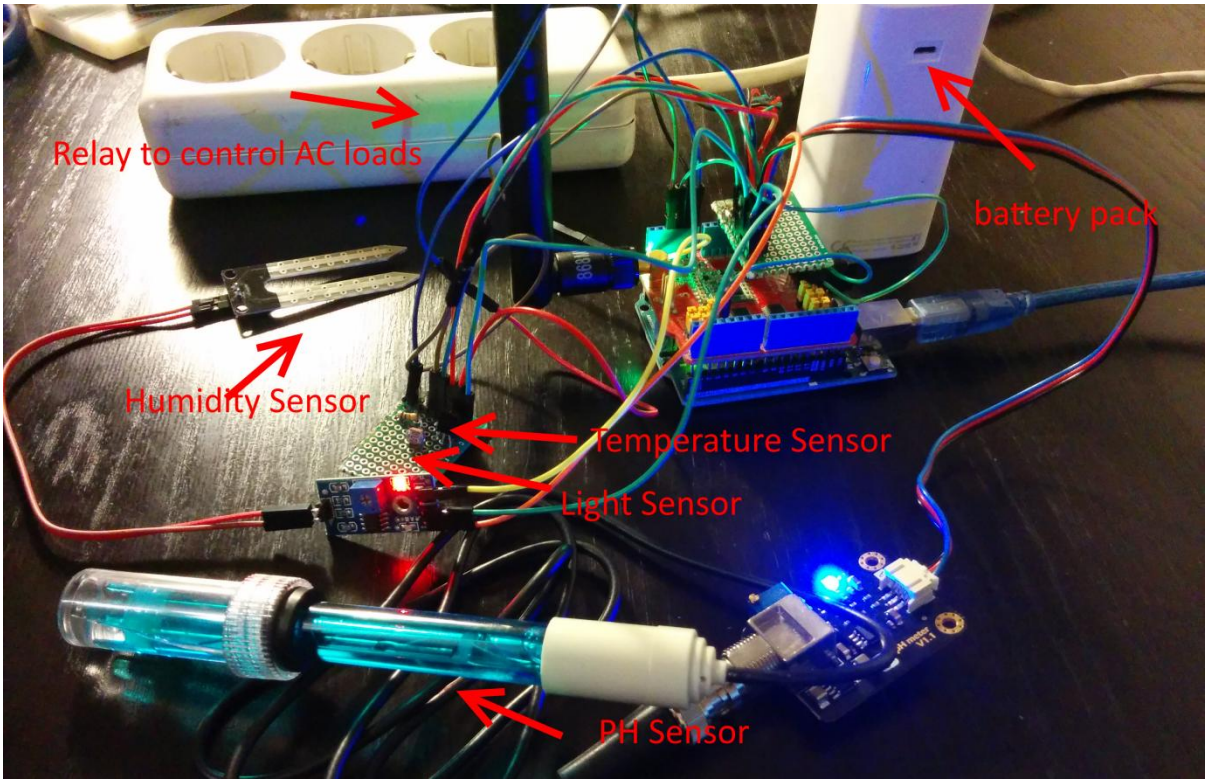
3. Υλοποίηση του προτεινόμενου συστήματος

3.1 Υλοποίηση των client nodes του ασύρματου δικτύου αισθητήρων του συστήματος

Για την υλοποίηση κάθε client node του ασύρματου δικτύου αισθητήρων απαιτείται η χρήση του Arduino Uno με την Dragino Shield για την συνδεσιμότητα LoRa, καθώς και η σύνδεση των αισθητήρων θερμοκρασίας, υγρασίας, οξύτητας και φωτεινότητας στις αναλογικές του εισόδους. Στην συνέχεια, απαιτείται η επεξεργασία των τιμών που διαβάσαμε από τον ADC, για μετατροπή των τιμών από 0 έως 1023 σε κάποια κατανοητή από τον χρήστη, δηλαδή της θερμοκρασίας σε βαθμούς κελσίου, της φωτεινότητας σε lux, της οξύτητας σε PH (0-14), και της υγρασίας σε τοις εκατό υγρασία. Επίσης, απαιτείται η σύνδεση relay για έλεγχο λοιπών ενεργοποιητών, όπως φωτισμού/αυτόματου ποτίσματος.



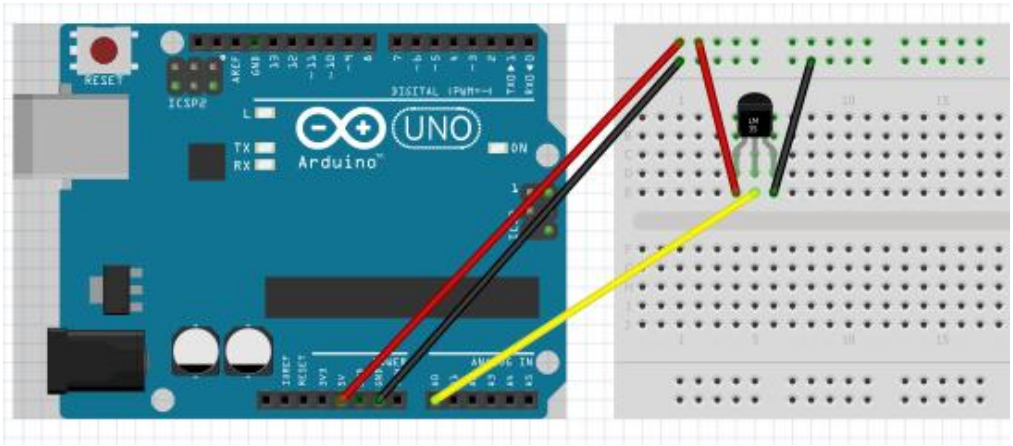
Εικόνα 29 – Το υλοποιηθέν Client Node, overview



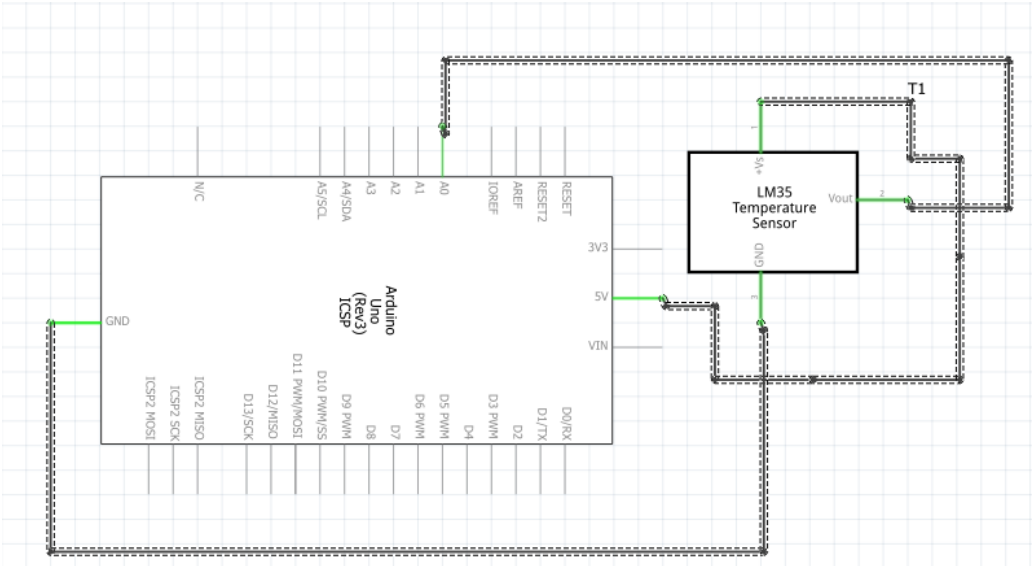
Εικόνα 30 - Το υλοποιηθέν Client Node

3.1.1 Σύνδεση του αισθητήρα θερμοκρασίας σε client node, ανάγνωση και επεξεργασία δεδομένων του

Για την σύνδεση του αισθητήρα θερμοκρασίας LM35 με το Arduino Uno, απαιτείται η τροφοδοσία του αισθητήρα από το +5V του Arduino, η σύνδεση της γείωσης του αισθητήρα με το GND του Arduino, και η αναλογική έξοδος του αισθητήρα στην αναλογική είσοδο A0. Σε περίπτωση που έχουμε θορυβώδες περιβάλλον, προσθέτουμε πυκνωτή 0,1mF όπως προτείνει η Texas Instruments στο Datasheet. Στην παρούσα υλοποίηση δεν κρίθηκε απαραίτητο.



Εικόνα 31 - Σύνδεση LM35 με Arduino



Εικόνα 32 - Σχηματικό διάγραμμα σύνδεσης LM35 με Arduino

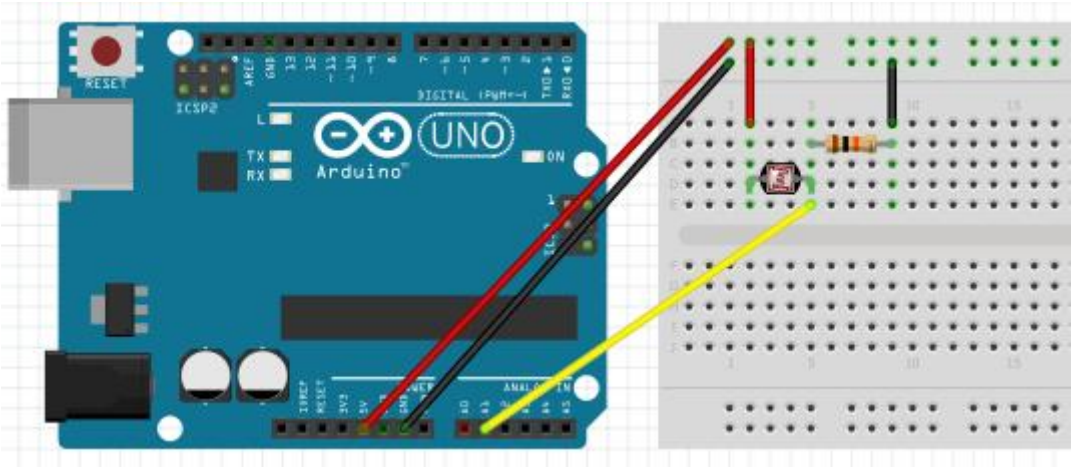
Για την ανάγνωση των δεδομένων του αισθητήρα αναπτύξαμε την συνάρτηση int readTempData(). Αρχικά, μέσω της συνάρτησης analogRead(A0) διαβάζουμε τα δεδομένα από το αναλογικό Pin A0 και τα αποθηκεύουμε στην μεταβλητή tempADCreading. Στην συνέχεια υπολογίζουμε την τάση κανονικοποιώντας με την διακριτική ικανότητα-resolution του ADC (δηλ. διαιρώντας την στάθμη που διαβάσαμε με το σύνολο της διακριτικής ικανότητας του ADC (1024 στάθμες) και πολλαπλασιάζουμε με τα 5000mv του εύρους σήματος). Η μεταβλητή cel περιέχει την μέτρηση σε βαθμούς κελσίου, διαιρώντας την τιμή της μεταβλητής mv με το 10. Η διαδικασία αυτή επαναλαμβάνεται 20 φορές εντός μίας for-loop για την υλοποίηση ενός φίλτρου διέλευσης χαμηλών συχνοτήτων και την εξομάλυνση των μετρήσεων από πιθανό θόρυβο, και το άθροισμα των μετρήσεων αποθηκεύεται στην μεταβλητή sum. Έπειτα, εκτός της επανάληψης, διαιρούμε το sum με το 20 για να πάρουμε μία ακριβή τιμή θερμοκρασίας, την οποία μπορούμε να επιστρέψουμε στο κυρίως πρόγραμμα, όταν καλέσει την readTempData().

```
int readTempData(){
  int sum=0;
  int cel;

  for (int i=0;i<=19;i++){
    int tempADCreading=analogRead(A0);
    int mv = (tempADCreading/1024.0)*5000;
    cel = mv/10;
    //Serial.println(tempADCreading); //gia debugging
    //Serial.println(mv); //gia debugging
    //Serial.println(cel); //gia debugging
    sum+=cel;
  }
  //Serial.println(sum); //gia debugging
  cel=sum/20;

  //Serial.println(cel); //gia debugging
```

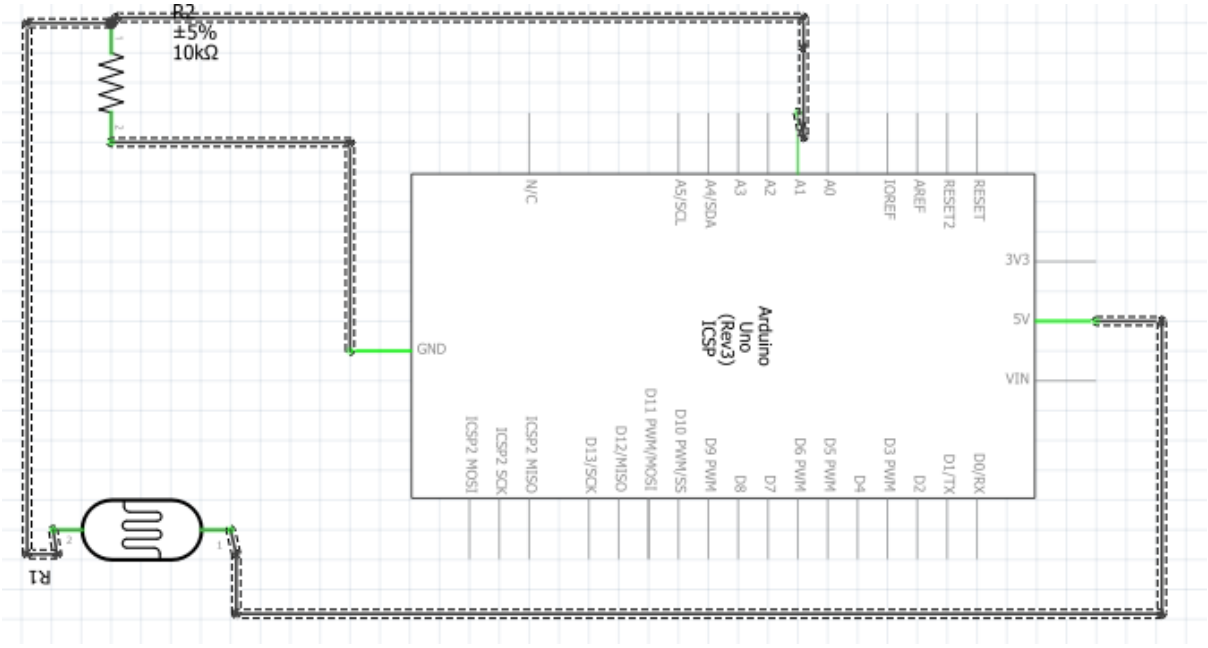
```
  return cel;
}
```

3.1.2 Σύνδεση του αισθητήρα φωτεινότητας σε client node, ανάγνωση και επεξεργασία δεδομένων του

Για την τροφοδοσία του αισθητήρα φωτεινότητας GL5539 χρησιμοποιήθηκε το +5V pin του Arduino. Σε σειρά με αυτόν τοποθετήθηκε αντίσταση 10K, ενώ την μέτρηση της φωτεινότητας την παίρνουμε από την έξοδο του διαιρέτη τάσης που δημιουργήθηκε, και την οδηγούμε στην αναλογική είσοδο A1 του Arduino.



Εικόνα 33 - Σύνδεση αισθητήρα φωτεινότητας GL5539 με Arduino



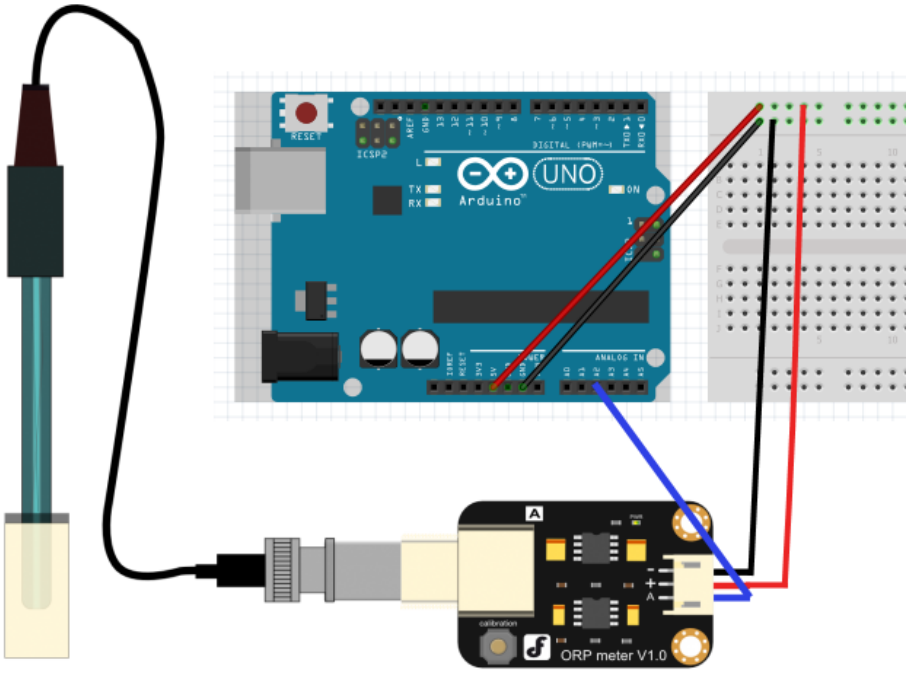
Εικόνα 34 - Σχηματικό σύνδεσης αισθητήρα φωτεινότητας με Arduino

Για την ανάγνωση των δεδομένων του αισθητήρα αναπτύξαμε την συνάρτηση int readLightData(). Μέσω της συνάρτησης analogRead(A1) που προσφέρει το Abstuction Layer, διαβάζουμε τα δεδομένα από τον αναλογικό ακροδέκτη Pin A1 και τα αποθηκεύουμε στην μεταβλητή fwsreading. Τέλος, βεβαιωνόμαστε ότι ακόμα και σε περίπτωση αστοχίας του αισθητήρα δεν θα σταλεί ένδειξη μεγαλύτερη από αυτήν που επιτρέπεται στο κυρίως πρόγραμμα.

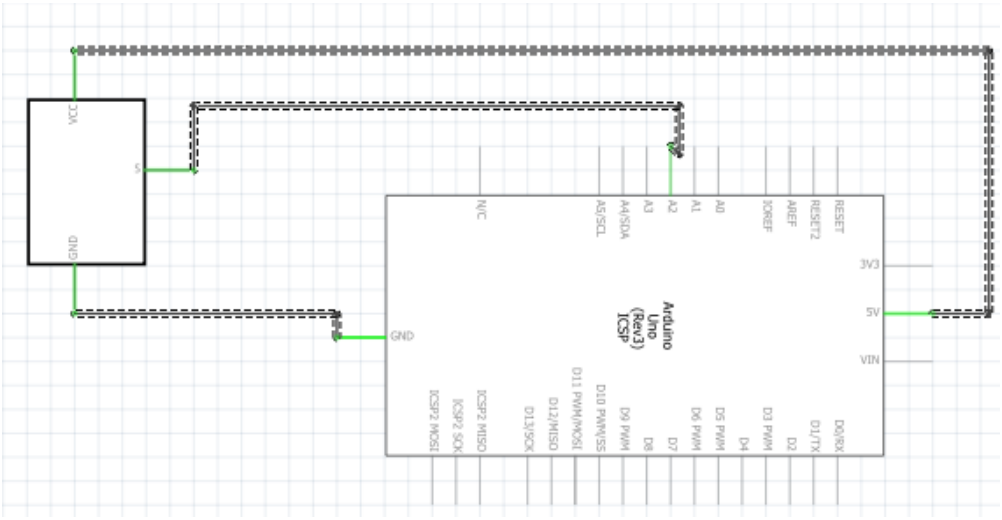
```
int readLightData(){

  int fwsreading=analogRead(A1)/10; //se lux (!)
  //Serial.println(fwsreading);  //gia debugging

  if (fwsreading>99){
    fwsreading=99; //just to make sure
  }

  return fwsreading;
}
```

3.1.3 Σύνδεση του αισθητήρα οξύτητας σε client node, ανάγνωση και επεξεργασία δεδομένων του

Η τροφοδοσία του Module του αισθητήρα οξύτητας πραγματοποιείται από το 5V rail του Arduino Uno, ενώ η αναλογική του έξοδος οδηγείται στην είσοδο A2. Το κύκλωμα προσαρμογής του αισθητήρα περιέχεται στο Module.



Εικόνα 35 - Σύνδεση αισθητήρα οξύτητας με Arduino



Εικόνα 36 – Σχηματικό σύνδεσης αισθητήρα οξύτητας με Arduino

Για την ανάγνωση των δεδομένων του αισθητήρα αναπτύξαμε την συνάρτηση int readPhData(), η οποία χρησιμοποιεί την double averagearray() για βελτιστοποίηση της ακρίβειας μέσω δειγματοληψίας. Στα definitions δηλώνουμε το Pin του Arduino που θα οδηγηθεί η έξοδος του αισθητήρα, δηλαδή το A2, ο ρυθμός δειγματοληψίας, ο αριθμός των δειγμάτων και λοιπές παράμετροι για calibration και debugging. Στο τέλος η averagearray επιστρέφει την μεταβλητή avg στην readPhData, η οποία το διαμορφώνει βάσει το offset που έχει δοθεί στο calibration και στην συνέχεια το επιστρέφει στο κυρίως πρόγραμμα μέσω της μεταβλητής pHValue.

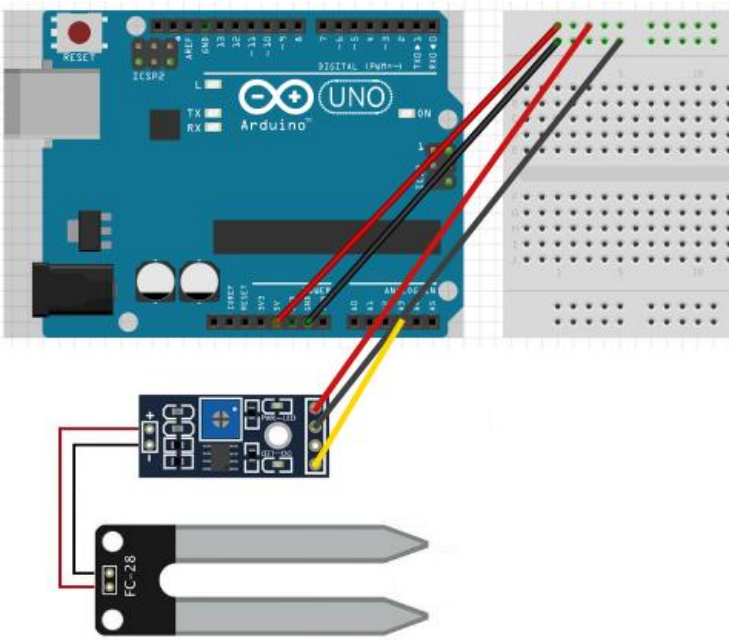
```
//definitions for the PH Meter Sensor
#define SensorPin A2            //pH meter Analog output to Arduino Analog
Input 2
#define Offset 0.00            //calibration
#define LED 5 //gia testing, for now
#define samplingInterval 20 //rithmos digmatolipsias
#define printInterval 800
#define ArrayLenth  40    //deigmata

int pHArray[ArrayLenth];   //Store the average value of the sensor feedback
int pHArrayIndex=0;

int readPhData(){
  static unsigned long samplingTime = millis();
  static unsigned long printTime = millis();
  static float pHValue,voltage;
  //thelw opwsdipote digmatolipsia, gia megaliteri akriveia
  if(millis()-samplingTime > samplingInterval)

{
      pHArray[pHArrayIndex++]=analogRead(SensorPin);
      if(pHArrayIndex==ArrayLenth)pHArrayIndex=0;
      voltage = averagearray(pHArray, ArrayLenth)*5.0/1024;
      pHValue = 3.5*voltage+Offset;
      samplingTime=millis();
  }
```

```
  if(millis() - printTime > printInterval)
//Every 800 milliseconds, print a numerical, convert the state of the LED
indicator


{
  //Serial.print("Voltage:");
  //Serial.print(voltage,2);
  //Serial.print("    pH value: ");
  //Serial.println(pHValue,2);
  //digitalWrite(LED,digitalRead(LED)^1);
  printTime=millis();
  }
  return pHValue;
}

//deigmatolipsia gia megaliteri akriveia ston PH sensotr
double averagearray(int* arr, int number){
  int i;
  int max,min;
  double avg;
  long amount=0;

  if(number<=0){
    // Serial.println("Error number for the array to avraging!/n");
    return 0;
  }

  if(number<5){   //less than 5, calculated directly statistics
    for(i=0;i<number;i++){
      amount+=arr[i];
    }
    avg = amount/number;
    return avg;
  }else{
    if(arr[0]<arr[1]){
      min = arr[0];max=arr[1];
    }
    else
    {
```

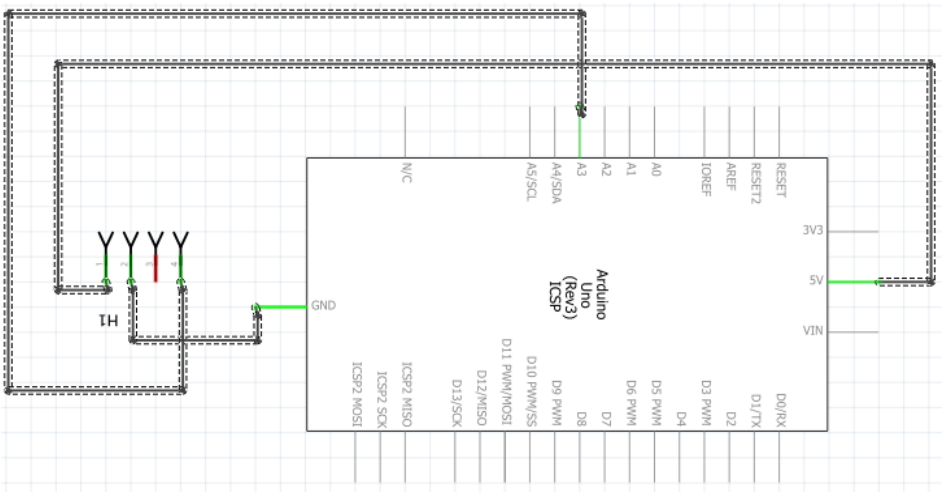
```
      min=arr[1];max=arr[0];
    }
    for(i=2;i<number;i++){
      if(arr[i]<min){
        amount+=min;         //arr<min
        min=arr[i];
      }else {
        if(arr[i]>max){
          amount+=max;     //arr>max
          max=arr[i];
        }else{
          amount+=arr[i]; //min<=arr<=max
        }
      }//if
    }//for
    avg = (double)amount/(number-2);
  }//if
  return avg;
}
```

3.1.4 Σύνδεση του αισθητήρα υγρασίας χώματος σε client node, ανάγνωση και επεξεργασία δεδομένων του

Η τροφοδοσία του Module του αισθητήρα υγρασίας πραγματοποιείται από το 5V Pin του Arduino, ενώ το GND (Ground) του Module του συστήματος προσαρμογής συνδέεται με το GND Rail του Arduino. Η ψηφιακή έξοδος του αισθητήρα δεν χρησιμοποιήθηκε. Η αναλογική έξοδος του αισθητήρα συνδέεται με την αναλογική είσοδο Pin A3.



Εικόνα 37 – Σύνδεση αισθητήρα υγρασίας με το Arduino



Εικόνα 38 - Σχηματικό σύνδεσης αισθητήρα υγρασίας με το Arduino

Για την ανάγνωση των δεδομένων του αισθητήρα υγρασία αναπτύχθηκε η συνάρτηση int readHumData(). Μετρώντας με τελείως ξηρό χώμα η μέτρηση που πήραμε ήταν 550, ενώ με τελείως υγρό ήταν περίπου 10. Συνεπώς το διάστημα τιμών από 550 έως 10, αντιστοιχίζεται στην περιοχή τιμών σχετικής υγρασίας από 100% έως 0%. Τέλος η μεταβλητή humid επιστρέφει την τιμή της στο κυρίως πρόγραμμα.

```
int readHumData(){

int humid=analogRead(A3);
humid=map(humid,550,0,0,100);
return humid;

}
```

3.1.5 Αποστολή δεδομένων από τα client nodes στο server node

Εντός της void loop(), η οποία εκτελείται καθ' επανάληψη στο Arduino, αναπτύχθηκε η ρουτίνα αποστολής των δεδομένων των αισθητήρων από τα client nodes προς το κεντρικό server node μέσω LoRa. Μετά τις δηλώσεις των μεταβλητών καλούνται οι συναρτήσεις readTempData(), readPhData(), readLightData() και readHumData τις οποίες δείξαμε σε προηγούμενα κεφάλαια, και τις τιμές που επιστρέφουν τις αποθηκεύουμε στις αντίστοιχες μεταβλητές, temp, ph, light και hum. Στην συνέχεια υλοποιούμε την λέξη προς αποστολή, επιλέγοντας τον αριθμό των ψηφίων που θα αφιερώσουμε σε κάθε μέτρηση, στην μεταβλητή sendstring που ορίσαμε. Πολλαπλασιάζοντας για παράδειγμα την τιμή του ph με το 10000, στη λέξη που θα δημιουργηθεί θα χρησιμοποιήσει το 5ο και το 6ο ψηφίο της λέξη. Ταυτόχρονα, μιας και η τιμή του ph, δεν μπορεί να λάβει τιμές παραπάνω από 14, δεν χρειάζεται να της αφιερώσουμε τρίτο ψηφίο. Συνεπώς πολλαπλασιάζοντας με δυνάμεις του 10 τις μεταβλητές των μετρήσεων, δημιουργείται η λέξη προς αποστολή, η οποία περιέχει ανά δυο (ή περισσότερα αν απαιτηθεί) ψηφία το σύνολο των μετρήσεων. Έπειτα η λέξη προς αποστολή μετατρέπεται από long σε string και αποστέλλεται μέσω της συνάρτησης rf95.send() που αναπτύχθηκε για αυτό τον σκοπό.

```
void loop()
{
  ...

  int temp=0;
  int ph=0;
  int light=0;
  int hum=0;
  String datastring="";
  char databuf[10];
  uint8_t dataoutgoing[10];


  temp=readTempData();
```

```
  ph=readPhData();
  light=readLightData();
  hum=readHumData();

  //Serial.println(temp);
  //Serial.println(ph);
  //Serial.println(light);
  //Serial.println(hum);
  long sendstring;
  sendstring=((temp*1000000L)+(ph*10000L)+(light*100L)+hum);
  //praxeis me long, took some time to debug :(
  //Serial.println(sendstring);
  //Serial.println("Sending to rf95_server");
  datastring =dtostrf(sendstring, 8, 0, databuf);
  Serial.println(datastring);
  strcpy((char *)dataoutgoing,databuf);
  rf95.send(dataoutgoing, sizeof(dataoutgoing));

  rf95.waitPacketSent();

  ...

  delay(1000);
}
```

3.1.6 Λήψη δεδομένων από τα client nodes που αποστέλλει το server node

Το server node/gateway, το οποίο έχει κάνει subscribe σε topics στον MQTT Broker, μεταβιβάζει τις πληροφορίες από αυτόν, στα client nodes. Για παράδειγμα όταν ο χρήστης επιλέξει από την Cloud εφαρμογή να ενεργοποιήσει τον φωτισμό, ή την αντλία νερού για πότισμα, η πληροφορία της αλλαγής της θέσης του εικονικού διακόπτη μεταβιβάζεται από το Internet, μέσω του κεντρικού node το οποίο δρα και ως gateway, στα client nodes.

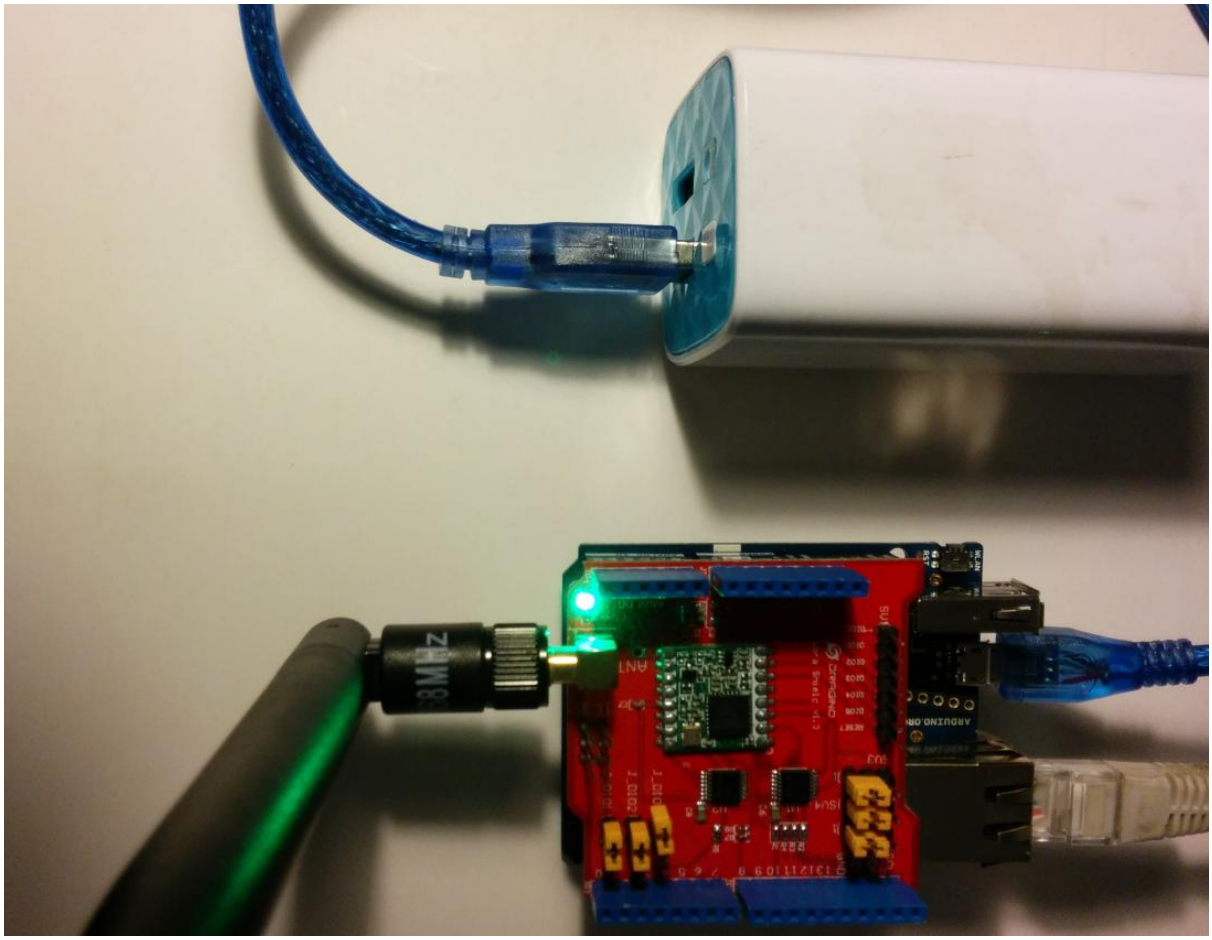
Στην συνέχεια, πραγματοποιείται σύγκριση της πληροφορίας που έλαβαν τα client nodes, με τις γνωστές κωδικές λέξεις στις οποίες έχει αντιστοιχηθεί κάποια ενέργεια. Για παράδειγμα η λέξη ON αντιστοιχεί στην ενεργοποίηση του φωτισμού, η λέξη ONP αντιστοιχεί στην ενεργοποίηση της αντλίας του νερού για πότισμα.

```
void loop(){

...

uint8_t buf[RH_RF95_MAX_MESSAGE_LEN];
uint8_t len = sizeof(buf);

if (rf95.waitAvailableTimeout(3000))
  {
    // An iparxei lifthen minima
    if (rf95.recv(buf, &len))
   {
      Serial.print("got reply: ");
      Serial.println((char*)buf);
      //Serial.print("RSSI: "); //debugging
      //Serial.println(rf95.lastRssi(), DEC);

      String wordd=(char*)buf;
      Serial.println(wordd);

      if (wordd=="ON"){ //energopoihsh fwtismou
            digitalWrite(3,HIGH);
```

```
      }
      else if (wordd=="OFF"){ //apenergopoihsh fwtismou
            digitalWrite(3,LOW);
      }
      else if (wordd=="ONP"){ //energopoihsh antlias nerou
            digitalWrite(2,HIGH);
      }
      else if (wordd=="OFFP"){ //apenergopoihsh antlias nerou
            digitalWrite(2,LOW);
      }

    }
    else
    {
      Serial.println("recv failed");
    }
  }
else
{
//Serial.println("No reply...");
//debugging
}
...
}
```

3.2 Υλοποίηση του κεντρικού node και gateway του συστήματος

Για την υλοποίηση του κεντρικού server node του συστήματος, το οποίο συλλέγει όλα τα δεδομένα από τα client nodes, χρησιμοποιήθηκε το Arduino YUN με την Dragino Shield για την συνδεσιμότητα LoRa. Για την τροφοδοσία του κεντρικού node μπορεί να χρησιμοποιηθεί οποιοδήποτε 5V rail είτε στο Vin Pin, είτε στην θύρα MicroUSB, είτε PoE (Power over Ethernet). Η σύνδεση του κόμβου με το Router, πραγματοποιείται είτε μέσω WiFi, είτε μέσω Ethernet. Για Redundancy στις δοκιμές χρησιμοποιήθηκαν ταυτόχρονα και οι δυο τρόποι. Τα δεδομένα που έχουν συλλεχθεί, χρησιμοποιώντας το πρωτόκολλο MQTT ανεβαίνουν σε οποιονδήποτε MQTT Broker επιθυμούμε και είναι προσβάσιμα από το Cloud.



Εικόνα 39 – Το κεντρικό Node και Gateway του συστήματος

3.2.1 Αποστολή δεδομένων από το server node στα client nodes

Το Server Node αναλαμβάνει να στείλει δεδομένα στα client nodes, τα οποία έλαβε από το Cloud, έχοντας κάνει subscribe σε topics στον MQTT Broker. Αρχικά ορίζουμε τις παραμέτρους του cloud που επιλέξαμε, δηλαδή το IoT Dashboard της Adafruit, καθώς και τα topics στα οποία θέλουμε να κάνουμε Subscribe, στην περίπτωσή μας στα onoff και onoffp τα οποία αντιστοιχούν στην αλλαγή της θέσης του εικονικού διακόπτη στο cloud, και ελέγχουν τον φωτισμό και την αντλία νερού για το πότισμα αντίστοιχα.

```
...
#define AIO_SERVER      "io.adafruit.com"
#define AIO_SERVERPORT  1883
#define AIO_USERNAME    "giannisdogas"
#define AIO_KEY         "***cb75********9d9**dcdbc******"


/****** Global State , Bridge Client kai MQTT Client ******/
// Create a BridgeClient instance to communicate using the Yun's bridge &
Linux OS.
BridgeClient client;

// Setup the MQTT client class by passing in the client and MQTT server and
login details.
Adafruit_MQTT_Client mqtt
(&client, AIO_SERVER, AIO_SERVERPORT, AIO_USERNAME, AIO_KEY);


// Setup feeds for subscribing to changes.
Adafruit_MQTT_Subscribe onoffbutton = Adafruit_MQTT_Subscribe(&mqtt,
AIO_USERNAME "/feeds/onoff");

Adafruit_MQTT_Subscribe onoffbuttonp = Adafruit_MQTT_Subscribe(&mqtt,
AIO_USERNAME "/feeds/onoffp");
...
```

Εντός της void setup(), η οποία εκτελείται μία φορά κατά την εκκίνηση, αρχικοποιούμε τα subscriptions τα οποία δημιουργήσαμε και δηλώσαμε παραπάνω.

```
void setup()
{
  ...

  // Setup MQTT subscription for onoff and onoffp feeds.
  mqtt.subscribe(&onoffbutton);
  mqtt.subscribe(&onoffbuttonp);

  ...

}
```

Εντός της void loop(), καλείται η mqttloop(), η ποία συγκρίνει το subscription που ελήφθησε με τα δηλωθέντα (onoffbutton και onoffbuttonp, και αποστέλει μέσω της rf95.send() το περιεχόμενων των δεδομένων (ON, OFF, ONP ή OFFP όπως έχουν οριστεί στο cloud.

```
void mqttloop(){
...

  // this is our 'wait for incoming subscription packets' busy subloop
Adafruit_MQTT_Subscribe *subscription;
while ((subscription = mqtt.readSubscription(1000))) {
if (subscription == &onoffbutton) {
      Serial.print(F("Got: "));
      Serial.println((char *)onoffbutton.lastread);
      // Send a reply
      //uint8_t data[] = "test data";
      rf95.send((char *)onoffbutton.lastread, 8);
      rf95.waitPacketSent();
      //Serial.println("Sent a reply");
    }
```

```
else if (subscription == &onoffbuttonp) {
      Serial.print(F("Got: "));
      Serial.println((char *)onoffbuttonp.lastread);
      // Send a reply
      //uint8_t data[] = "test data";
      rf95.send((char *)onoffbuttonp.lastread, 8);
      rf95.waitPacketSent();
      //Serial.println("Sent a reply");
    }
  }
...
}
```

3.2.2 Λήψη δεδομένων από το server node και ανέβασμα αυτών στο cloud

Τα δεδομένα των αισθητήρων που αποστέλλουν τα client nodes, συλλέγονται από το κεντρικό server node. Στην συνέχεια ανεβαίνουν από αυτό προς το cloud/τον broker της επιλογής μας. Αρχικά στον κώδικα ορίζουμε τα directories των Feeds που θα χρησιμοποιήσουμε, αλλά και αρχικοποιούμε τις τιμές των μεταβλητών που θα λάβουμε. Τα τέσσερα feeds για Publish δεδομένων που χρησιμοποιούμε είναι τα /giannisdogas/feeds/temp, /giannisdogas/feeds/ph, giannisdogas/feeds/light και giannisdogas/feeds/hum για την θερμοκρασία, την οξύτητα, την φωτεινότητα και την υγρασία αντίστοιχα.

```
...

/************* Feeds *************/
// Feed Paths
Adafruit_MQTT_Publish thermokrasia = Adafruit_MQTT_Publish
(&mqtt, AIO_USERNAME "/feeds/temp");
Adafruit_MQTT_Publish oxitita = Adafruit_MQTT_Publish
(&mqtt, AIO_USERNAME "/feeds/ph");
Adafruit_MQTT_Publish fwteinotita = Adafruit_MQTT_Publish
(&mqtt, AIO_USERNAME "/feeds/light");
Adafruit_MQTT_Publish igrasia = Adafruit_MQTT_Publish
(&mqtt, AIO_USERNAME "/feeds/hum");


uint32_t temp=0;
uint32_t ph=0;
uint32_t light=0;
uint32_t hum=0;

...

String incoming="";
```

Στην συνέχεια εντός της void loop() η οποία εκτελείται καθ' επανάληψη, καλείται η ρουτίνα λήψης δεδομένων μέσω LoRa από τα client nodes. Η ληφθείσα λέξη μετατρέπεται στην συνέχεια σε integer μέσω της .toInt() για να είναι εφικτή η ευκολότερη επεξεργασία της.

Έπειτα γίνεται διαχωρισμός αναλόγως των ψηφίων που έχουν αφιερωθεί σε κάθε μέτρηση, με ακέραια διαίρεση με δυνάμεις του 10. Για παράδειγμα, τα δύο πρώτα ψηφία της 8ψήφιας λέξης αφορούν την θερμοκρασία, συνεπώς δύναται να τα χωρίσουμε διαιρώντας ακέραια την μεταβλητή intincoming με το 1000000. Αντίστοιχα, αφαιρώντας τα δυο πρώτα ψηφία, και διαιρώντας με το 10000 προκύπτει η μέτρηση του ph από την ληφθείσα λέξη.

Οι μετρήσεις που προκύπτουν αποθηκεύονται στις μεταβλητές temp, ph, light και hum για την θερμοκρασία, την οξύτητα, την φωτεινότητα και την υγρασία αντίστοιχα.

```
void loop()
{
  ...

  if (rf95.available())
  {
    // An iparxei lifthen minima
    uint8_t buf[RH_RF95_MAX_MESSAGE_LEN];
    uint8_t len = sizeof(buf);
    if (rf95.recv(buf, &len))
    {
      //digitalWrite(led, HIGH);
      //RH_RF95::printBuffer("request: ", buf, len);
      //Serial.print("got request: ");
      //Serial.println((char*)buf);
      incoming=((char*)buf);
      Serial.println(incoming);
      long int intincoming = incoming.toInt();
      Serial.println(intincoming);
```

```
      //spliting the digits, gia na xexwrisw tis metriseis
      temp=intincoming/1000000;
      //Serial.println(temp);
      ph=((intincoming-temp*1000000)/10000);
      //Serial.println(ph);
      light=((intincoming-temp*1000000-ph*10000)/100);
      //Serial.println(light);
      hum=(intincoming-temp*1000000-ph*10000-light*100);
      //Serial.println(hum);
    }
    else
    {
      //Serial.println("recv failed");
    }
  }
}
```

Εντός της void loop(), επίσης, καλείται η mqttloop(), η οποία έχει γραφεί για να ανεβάζει τα δεδομένα στον server, η οποία με την σειρά της καλεί την mqttconnect(), η οποία είναι υπεύθυνη για την σύνδεση με τον Broker.

```
void MQTT_connect() {
  int8_t ret;

  // Stop if already connected.
  if (mqtt.connected()) {
    return;
  }
  //Serial.print("Connecting to MQTT... ");

  while ((ret = mqtt.connect()) != 0) {
       //Serial.println(mqtt.connectErrorString(ret));
       mqtt.disconnect();
       delay(3000);  // wait 3 seconds
  }
  //Serial.println("MQTT Connected!");

}
```

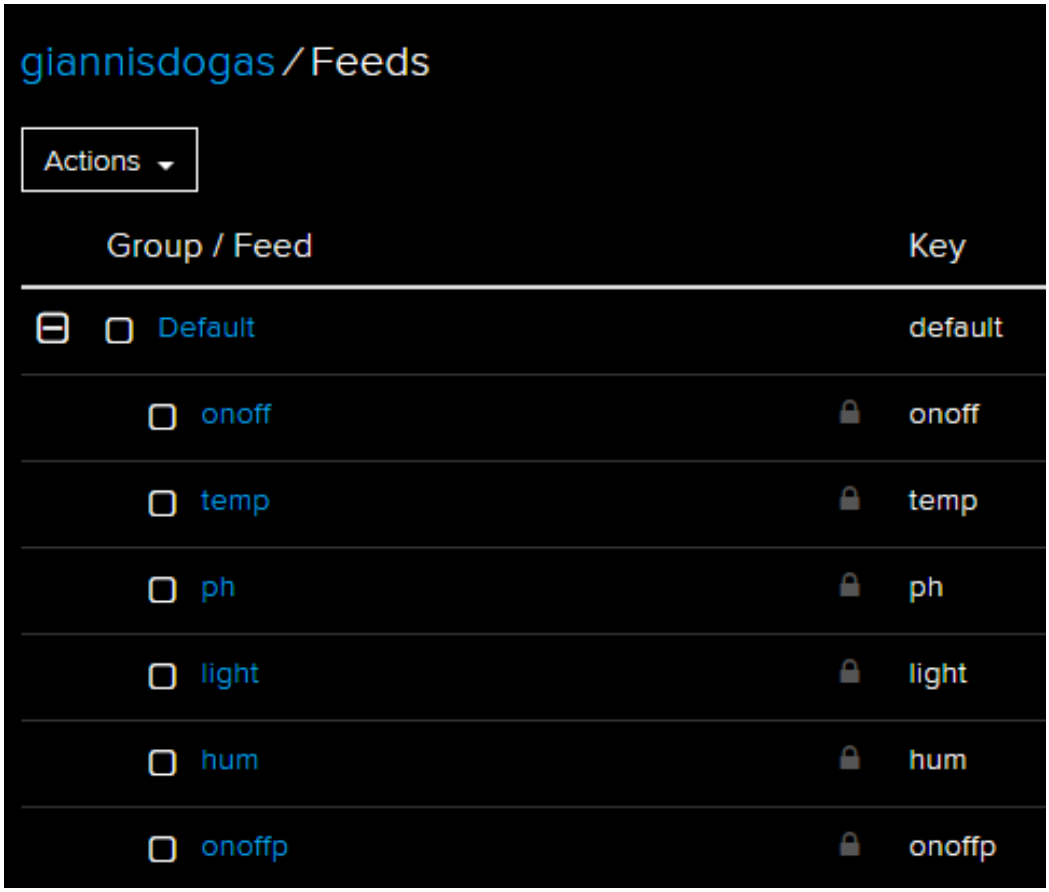
Εντός της mqttloop() πραγματοποιείται η αποστολή –το publish- των περιεχομένων των μεταβλητών temp, ph, light και hum, στον Broker. Επίσης πραγματοποιείται και ping στον server για διατήρηση της σύνδεσης.

```
void mqttloop(){

MQTT_connect();

...

//  Now we can publish stuff, yay!
//  Serial.print(F("\nSending values "));
//  Serial.println(temp);
//  Serial.println(ph);
//  Serial.println(light);
//  Serial.println(hum);
//  Serial.print("...");
//  //for debugging

  if (! thermokrasia.publish(temp)) {
    //Serial.println(F("Failed sending temp"));
  }
  else {
    //Serial.println(F("OK sending temp"));
  }
  if (! oxitita.publish(ph)) {
    //Serial.println(F("Failed sending ph"));
  }
  else {
    //Serial.println(F("OK sending ph"));
  }
  if (! fwteinotita.publish(light)) {
    //Serial.println(F("Failed sending light"));
  }
  else {
    //Serial.println(F("OK sending light"));
  }
```

```
  if (! igrasia.publish(hum)) {
    //Serial.println(F("Failed sending hum"));
  }
  else {
    //Serial.println(F("OK sending hum"));
  }

  // ping the server to keep the mqtt connection alive
  if(! mqtt.ping()) {
    //Serial.println(F("MQTT Ping failed."));
  }

  delay(200);

}
```

3.3 Υλοποίηση του IoT Dashboard στο Cloud της Adafruit

Στην πλατφόρμα Cloud της Adafruit υλοποιήθηκε Dashboard το οποίο εμφανίζει της πληροφορίες τις οποίες λαμβάνουμε από τους αισθητήρες, αλλά δίνει και την δυνατότητα απομακρυσμένου ελέγχου στους ενεργοποιητές (relays για έλεγχο αντλίας νερού και φωτισμού). Για την υλοποίηση του Dashboard, αρχικά απαιτείται η προσθήκη των Feeds που χρησιμοποιούμε, στον MQTT Broker. Προστέθηκαν τα Feeds τόσο για subscribe (feeds/onoff, feeds/onoffp), όσο και για publish (feeds/temp, feeds/ph, feeds/light, feeds/hum).

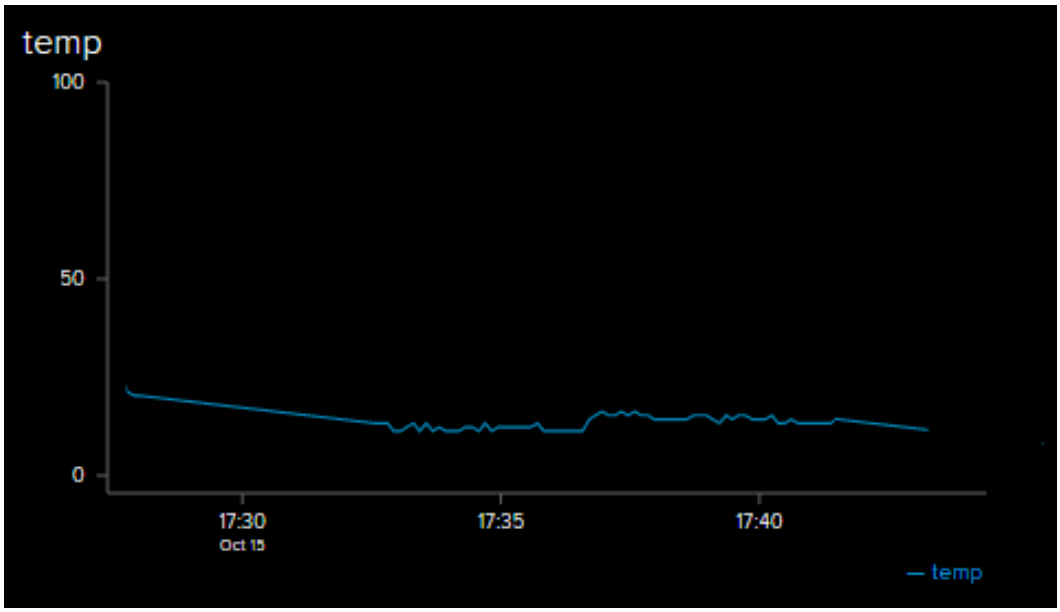


Εικόνα 40 - Τα υλοποιηθέντα Feeds στο Cloud της Adafruit

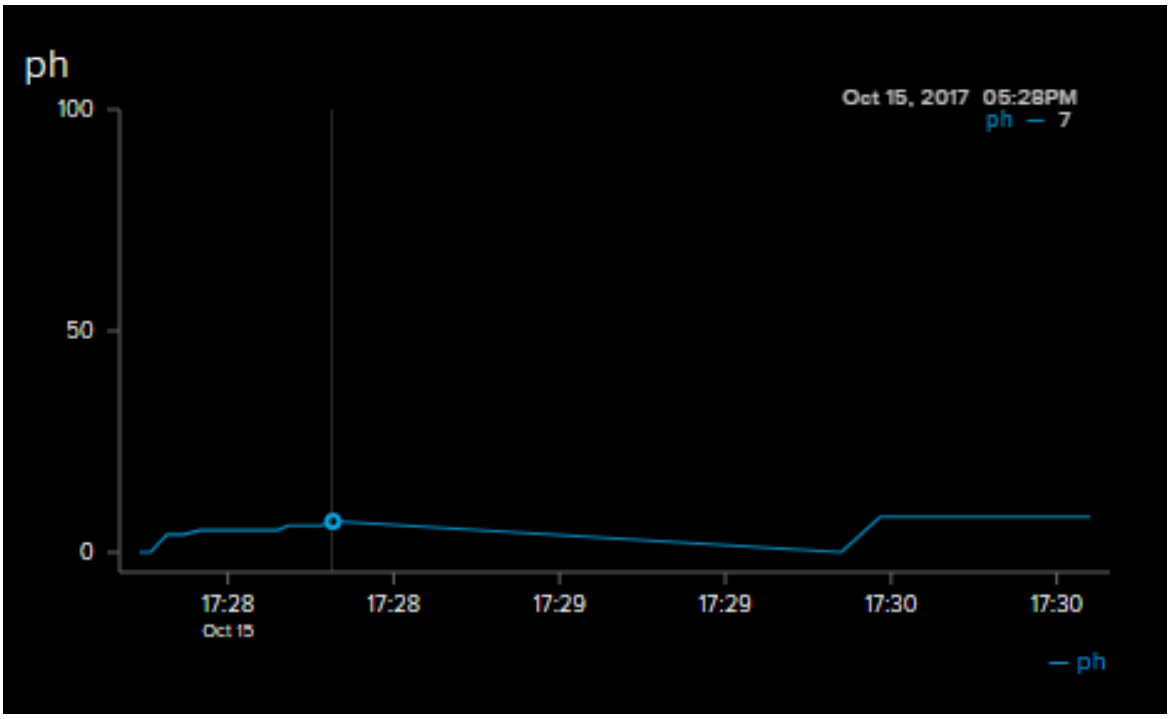
Για κάθε ένα από τα publish feeds, δημιουργήθηκε και το αντίστοιχο Gauge, το οποίο εμφανίζει την ένδειξη. Για κάθε ένα από τα subscribe feeds, δημιουργήθηκε και ο αντίστοιχος εικονικός διακόπτης για απομακρυσμένο έλεγχο των συσκευών.

Εικόνα 41 – Gauges για τα publish feeds, και διακόπτες για τα subscribe feeds

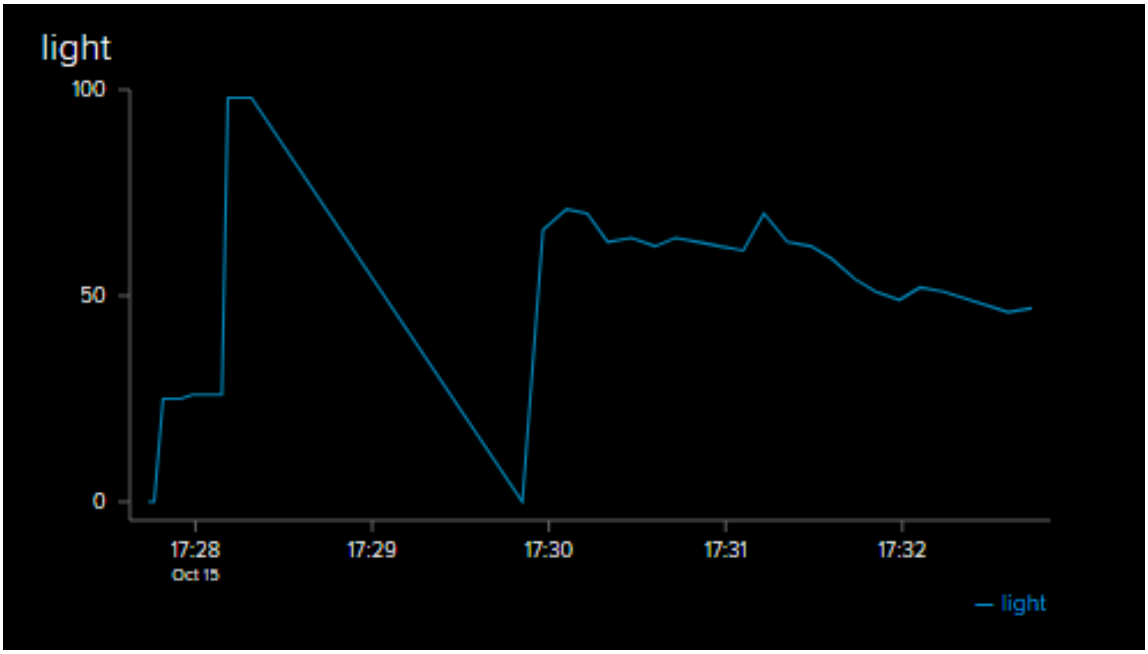
Επιπλέον, στο Dashboard προστέθηκαν και γραφήματα για να δίνεται στον χρήστη η δυνατότητα να βλέπει εύκολα πώς διαμορφώθηκαν οι μετρήσεις από τους αισθητήρες σε σύγκριση με τον χρόνο.



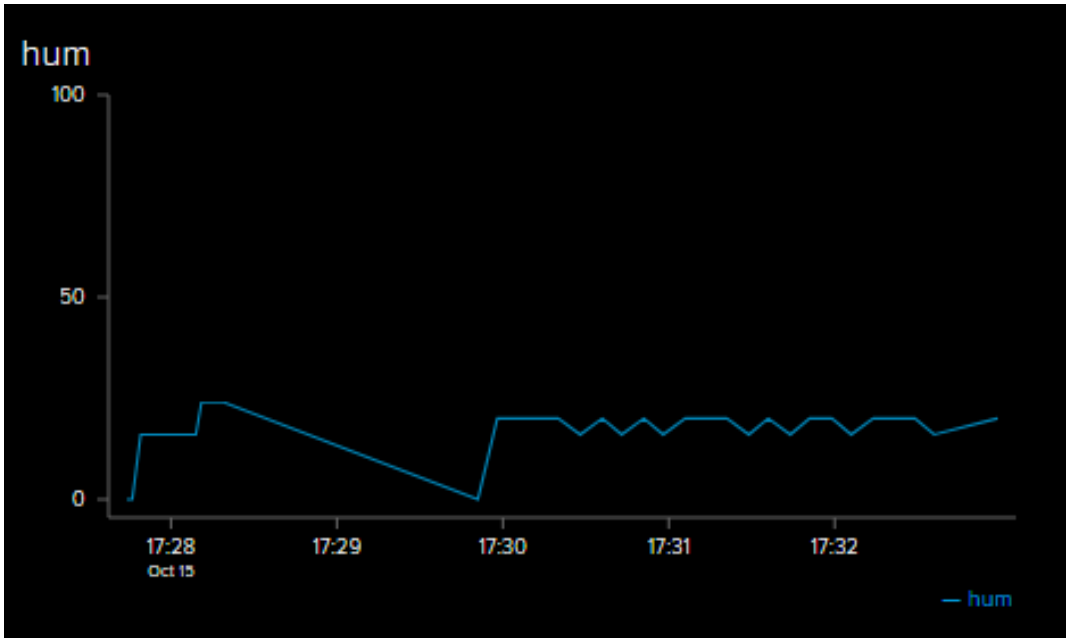
Εικόνα 42 - Διάγραμμα μετρήσεων θερμοκρασίας



Εικόνα 43 - Διάγραμμα μετρήσεων οξύτητας

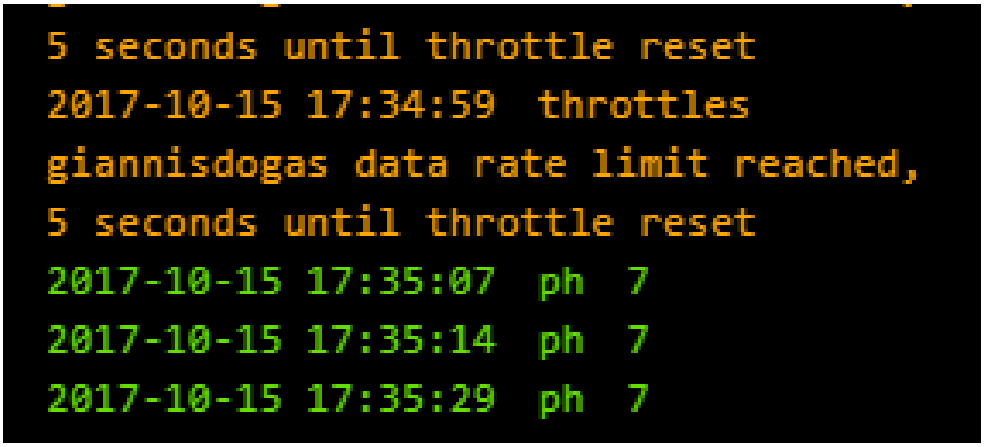


Εικόνα 44 - Διάγραμμα μετρήσεων φωτός

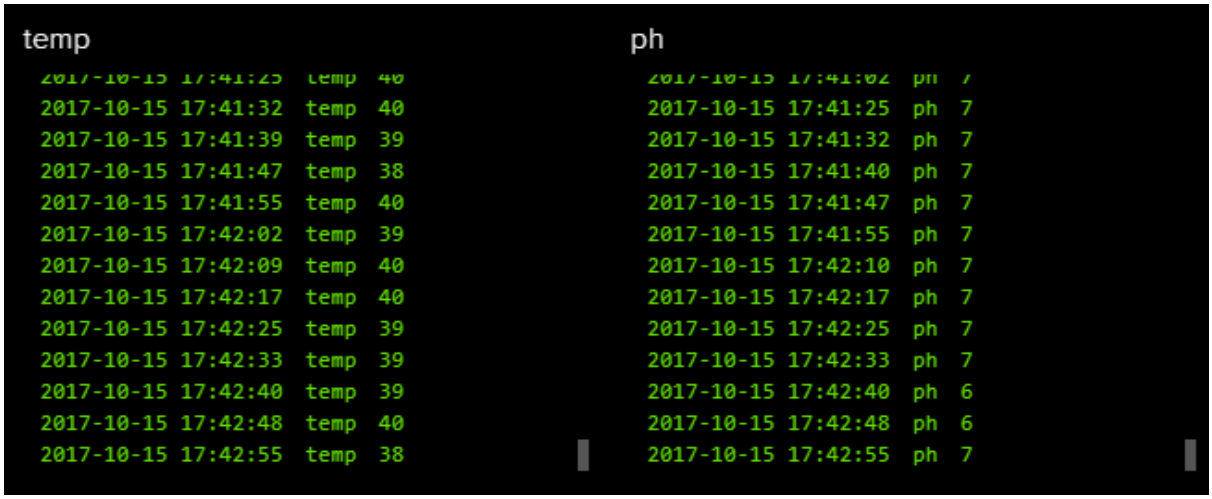


Εικόνα 45 - Διάγραμμα μετρήσεων υγρασίας

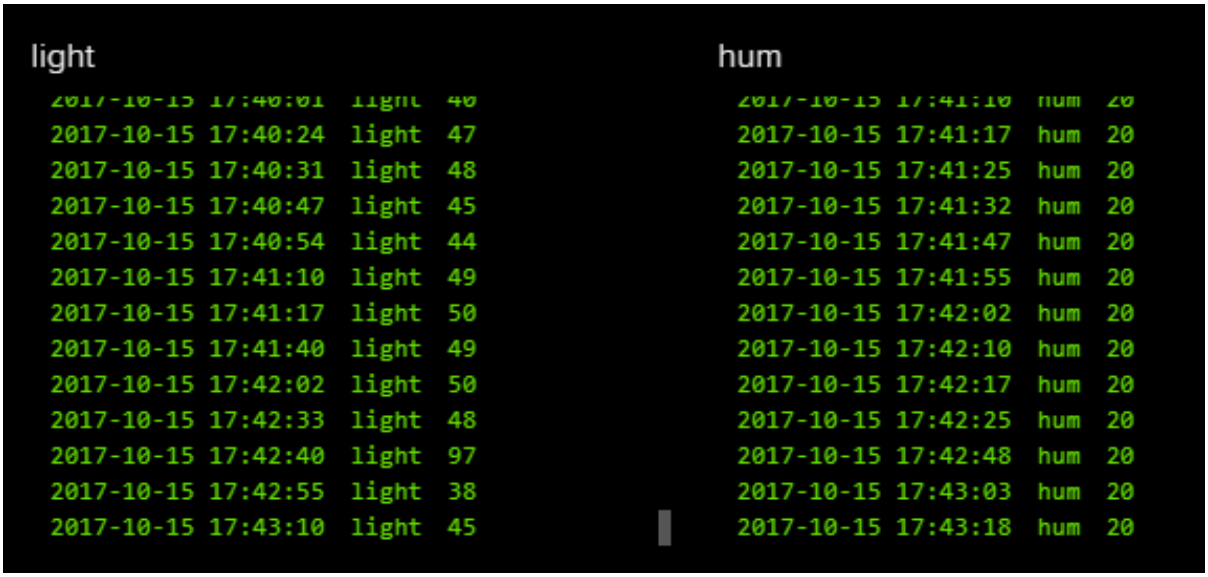
Τέλος, στο κάτω μέρος του Dashboard δίνεται το stream των πληροφοριών στην καθαρή μορφή του, μαζί με το αντίστοιχο timestamp για κάθε έναν αισθητήρα, αλλά και πληροφορίες για debugging σε περίπτωση που αυτό απαιτηθεί.



Εικόνα 46 – Stream Raw πληροφοριών στο Dashboard, αλλά και πληροφορίες Debugging

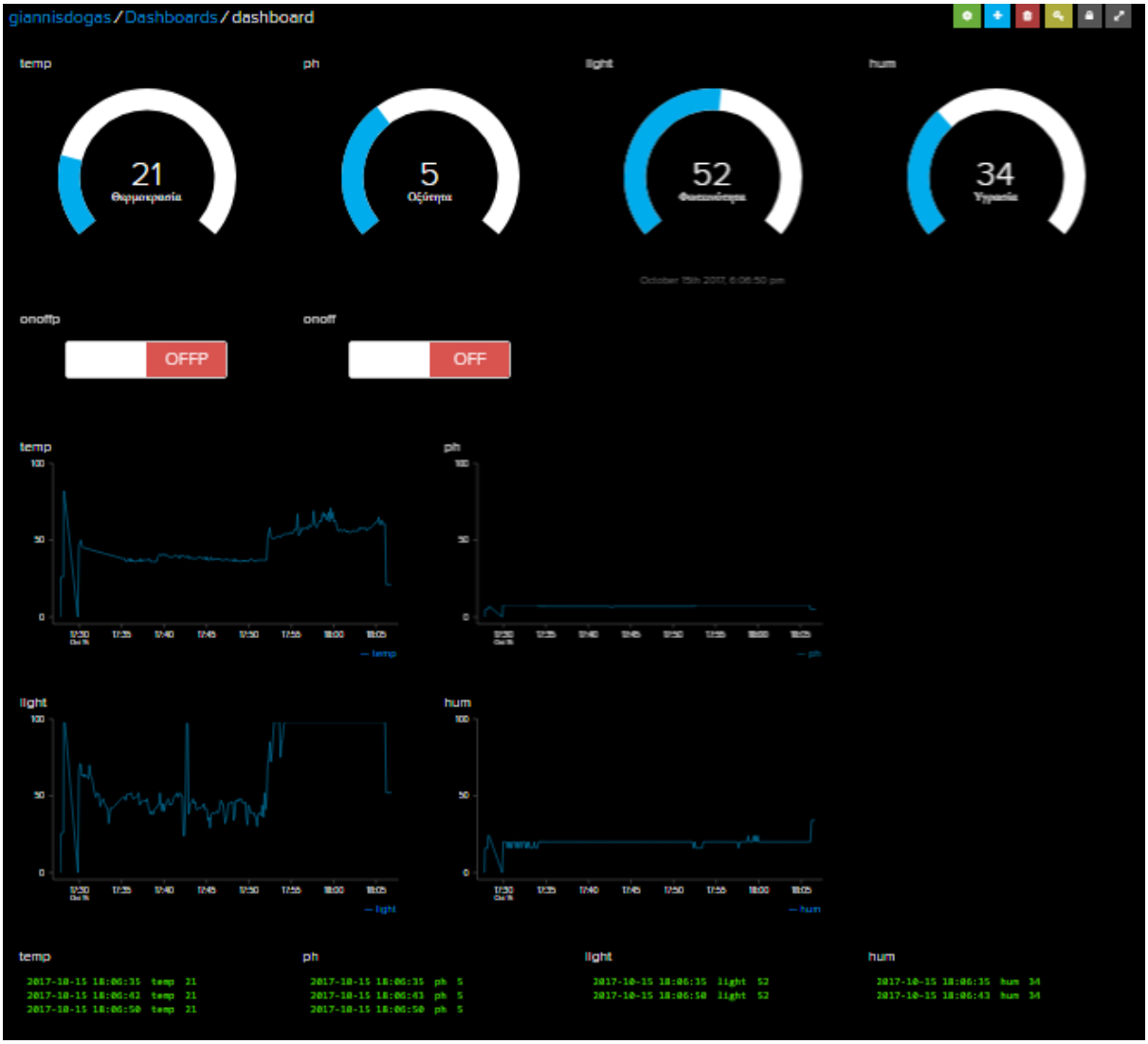


Εικόνα 47 – Stream πληροφοριών θερμοκρασίας και οξύτητας



Εικόνα 48 – Stream πληροφοριών φωτεινότητας και υγρασίας

Συνολικό overview του υλοποιηθέντος Dashboard, στο Cloud του Broker της Adafruit παρουσιάζεται παρακάτω. Αναφέρουμε ότι η παρούσα υλοποίηση του κεντρικού node/gateway είναι συμβατή με οποιονδήποτε MQTT 3, ή 3.1.1. Broker, συνεπώς και με τις περισσότερες υπηρεσίες Cloud.



Εικόνα 49 – Overview του πλήρους υλοποιηθέντος IoT Dashboard

4. Συμπεράσματα - Προτάσεις

Στην παρούσα εργασία μελετήθηκαν οι απαιτήσεις και προϋποθέσεις για την δημιουργία ενός πλήρους ασύρματου δικτύου αισθητήρων και ενεργοποιητών, μικρού εύρους ζώνης μεγάλης εμβέλειας, χαμηλού κόστους, με σκοπό την χρήση στην γεωργία. Βάσει των απαιτήσεων αυτών, υλοποιήθηκε σύστημα το οποίο να τις υπερκαλύπτει,

Χρησιμοποιήθηκε υλικό και λογισμικό ανοικτού κώδικα (open source) σε όλα τα κρίσιμα συστατικά του όλου συστήματος. Οι βιβλιοθήκες και το λογισμικό ανοικτού κώδικα που χρησιμοποιήθηκε χρειάσθηκε να τροποποιηθεί κατάλληλα και να επιλυθούν οι ασυμβατότητες που εντοπίσθηκαν κατά την ολοκλήρωσή τους σε ένα μια ενιαία εφαρμογή. Ειδικότερα:

- Υλοποιήθηκαν ασύρματα Client Nodes με συνδεσιμότητα LoRa, στα οποία συνδέθηκαν τυπικοί αισθητήρες για εφαρμογές γεωργίας (θερμοκρασίας, φωτός, οξύτητας και υγρασίας) και ενεργοποιητές αυτόματου ποτίσματος και φωτισμού. Η επιλογή του των αισθητήρων πραγματοποιήθηκε κυρίως βάσει των απαιτήσεων κόστους, χαμηλής κατανάλωσης ενέργειας, ακρίβειας, γραμμικότητας, εύρους, και ύπαρξης σχετικής βιβλιογραφίας.

- Υλοποιήθηκε ασύρματο Server Node με συνδεσιμότητα LoRa το οποίο αναλαμβάνει να συλλέξει όλα τα δεδομένα από τα Client Nodes και να τα ανεβάσει σε MQTT Broker. Η επιλογή των πλακετών ανάπτυξης, των μικροελεγκτών και των shields πραγματοποιήθηκαν με βασικά κριτήρια το κόστος, την επεκτασιμότητα, την επεξεργαστική ισχύ, τους περιορισμούς μνήμης, και την ύπαρξη σχετικής βιβλιογραφίας.

- Πραγματοποιήθηκε συγγραφή λογισμικού για τα client nodes, η οποία περιλαμβάνει ρουτίνες άντλησης, επεξεργασίας και κανονικοποίησης δεδομένων

από αισθητήρες, ρουτίνα αποστολής μέσω LoRa προς ένα κεντρικό server node, αλλά και ρουτίνα λήψης δεδομένων από αυτό.

- Επίσης συγγράφηκε λογισμικό για το server node, το οποίο περιλαμβάνει ρουτίνα λήψης/αποστολής δεδομένων από/προς τα client nodes καθώς και επικοινωνίας μέσω MQTT με έναν Broker. Στον Broker υλοποιήθηκαν Gauges, εικονικοί διακόπτες και γραφήματα για εμφάνιση των δεδομένων.

Το παρόν σύστημα υπερκαλύπτει τις ανάγκες αποστολής στοιχείων τηλεμετρίας από τυπικούς αισθητήρες γεωργίας, προς ένα κεντρικό Node και προώθηση αυτών προς το διαδίκτυο. Η επιλογή του LoRa για την δικτύωση των nodes καλύπτει τις ανάγκες του συστήματος σε εμβέλεια, ενώ ταυτόχρονα συμβάλλει στην μειωμένη κατανάλωση ενέργειας και επιπρόσθετα βασίζεται σε χαμηλής πολυπλοκότητας και κόστου πύλης δικτύου μεταφέροντας το κόστος ανάπτυξης και συντήρησης σε επίπεδο «νέφους-cloud». Η χρήση των νεότερων τεχνολογιών σε κάθε περίπτωση –όπως για παράδειγμα η χρήση του πρωτοκόλλου MQTT για επικοινωνία με το διαδίκτυο και αποστολή/λήψη δεδομένων προς/από το cloud αποτελούν τις βέλτιστες επιλογές για οποιοδήποτε εφαρμογή γεωργίας, εξασφαλίζοντας αξιόπιστη μετάδοση ακόμα και όταν δεν υπάρχει σταθερή σύνδεση στο διαδίκτυο. Στο σύστημα δύναται να προστεθούν δύο ακόμα αναλογικοί αισθητήρες στην παρούσα υλοποίηση, ή αν απαιτηθούν παραπάνω μπορεί να γίνει χρήση εξωτερικού ADC που επικοινωνεί με SPI με τον μικροελεγκτή. Αντίστοιχα, η χρήση του Abstraction Layer στην συγγραφή κώδικα που πραγματοποιήθηκε τον καθιστά συμβατό και με λοιπές Development Boards και μικροελεγκτές.

Για την εμπορική χρήση, διάθεση και επέκταση του συστήματος, κρίνονται σκόπιμες περαιτέρω τροποποιήσεις και σχετική έρευνα που αυτές απαιτούν.

Όσον αφορά τα Client Nodes, πρέπει να πραγματοποιηθεί εκ νέου επιλογή αισθητήρα οξύτητας καθότι η τακτική βαθμονόμηση που αυτός απαιτεί κρίνεται να είναι ασύμβατη με την εμπορική διάθεση ενός προϊόντος. Ταυτόχρονα, το σύνολο των αισθητήρων αλλά

και του κόμβου θα πρέπει να ελεγχθεί ότι δύναται να αντέξει τις καταπονήσεις του περιβάλλοντος και στην μακροχρόνια χρήση σε αυτό, το οποίο δεν ήταν εφικτό να δοκιμαστεί στο πλαίσιο της παρούσας εργασίας. Αντίστοιχα, το σύνολο του κόμβου απαιτείται να στεγανοποιηθεί/προστατευθεί από μηχανικές καταπονήσεις, όπου αυτό απαιτείται. Επίσης, περεταίρω έρευνα απαιτείται στην εκμετάλλευση των λειτουργιών χαμηλής κατανάλωσης ενεργείας του μικροελεγκτή και στην εύρεση του βέλτιστου duty cycle λειτουργίας δεδομένης της μονάδας παροχής ενέργειας που χρησιμοποιείται σε κάθε περίπτωση. Εάν υπάρχει συνδυασμός με ΑΠΕ και τεχνικές συγκομιδής ενέργειας, το duty cycle λειτουργίας θα μπορεί να μεταβάλλεται δυναμικά αναλόγως του επιπέδου φόρτισης των μπαταριών κάθε μέρα.

Όσον αφορά το Server Node, η υποστήριξη χιλιάδων Client Nodes οι οποίοι αποστέλλουν δεδομένα, θα απαιτήσει την αντικατάστασή του από Concentrator πολλών καναλιών, με μεγαλύτερο κόστος κατασκευής και στο κομμάτι του μικροελεγκτή/επεξεργαστή, και στο κομμάτι του ραδιοπομποδέκτη LoRa. Η επιλογή του MQTT middleware πρωτοκόλλου για την υλοποίηση της όλης εφαρμογής με μηνύματα του τύπου publish-subscribe αποτελεί την βέλτιστη επιλογή για την μαζική χρήση διαδικτυωμένων αισθητήρων για εφαρμογές Γεωργίας Ακριβείας χωρίς περιορισμούς στην γραμμική ανάπτυξη της προτεινόμενης υλοποίησης σε μεγάλης έκτασης και πυκνότητας κόμβων εφαρμογές. Αυτός είναι και ο λόγος επιλογής του ενώ χρησιμοποιείται από τις μεγαλύτερες πλατφόρμες ανάπτυξης cloud εφαρμογών όπως τη IBM Bluemix [28] και Microsoft Azure [29] καθώς και πλατφόρμες κοινωνικών δικτύων όπως το Facebook. [30]

ΒΙΒΛΙΟΓΡΑΦΙΑ - ΠΗΓΕΣ

[1] Denis Ilie-Ablachim, et al., «Monitoring device for culture substrate growth,» *University POLITEHNICA of Bucharest,* 2016.

[2] Thomas Zachariah, et al., «The Internet of Things Has a Gateway Problem,» *Electrical Engineering and Computer Science Department,* 2015.

[3] O. Vermesan & P. Friess, Internet of Things: Converging Technologies for Smart Environments and Integrated Ecosystems, River Publishers, 2013.

[4] Array of Things, «Array of Things GitHub,» 08 2017. [Ηλεκτρονικό]. Available: https://arrayofthings.github.io/. [Πρόσβαση 09 2017].

[5] T. Sohraby, et al., Wireless sensor networks: technology, protocols, and applications, John Wiley and Sons, 2007.

[6] K. A. Dimitrios Piromalis, «SensoTube: A Scalable Hardware Design Architecture,» *MDPI,* 2016.

[7] Jean-Paul Bardyn, et al., «IoT: The Era of LPWAN is starting now,» *Wireless and Sensing Business Unit, Semtech,* 2016.

[8] C. McClelland, «IoT for all,» 04 2017. [Ηλεκτρονικό]. Available: https://www.iotforall.com/iot-connectivity-comparison-lora-sigfox-rpma-lpwan-technologies/. [Πρόσβαση 07 2017].

[9] W. Wong, «Choices Abound for Long-Range Wireless IoT,» *Electronic Design,* 2016.

[10] Eli De Poorter, et al., «Sub-GHz LPWAN Network Coexistence, Management and Virtualization: An Overview and Open Research Challenges,» *Springer Science, Business Media New York,* 2017.

[11] H. H. Rashmi Sharan Sinha, et al., «A survey on LPWA technology: LoRa and NB-IoT,» *Science Direct,* 2017.

[12] Libelium, «The Future of Farming through the IoT Perspective,» *Beecham Research,* 2016.

[13] Terje Lassen, et al., «How can Long Range NarrowBand improve Agricultural Efficiency,» *Electronic Design,* 2015.

[14] DroneTimes JP, «DroneTimes,» 01 2017. [Ηλεκτρονικό]. Available: https://www.dronetimes.jp/articles/809. [Πρόσβαση 10 2017].

[15] Arduino LLC, «Arduino Uno Rev3,» 02 2017. [Ηλεκτρονικό]. Available: https://store.arduino.cc/arduino-uno-rev3. [Πρόσβαση 06 2017].

[16] Atmel, «ATmega328/P Introduction Feature,» 10 2016. [Ηλεκτρονικό]. Available: http://www.atmel.com/devices/atmega328p.aspx. [Πρόσβαση 07 2017].

[17] Arduino cc, «Arduino Yun,» 06 2017. [Ηλεκτρονικό]. Available: https://store.arduino.cc/arduino-yun. [Πρόσβαση 07 2017].

[18] Dragino Technology Co., LTD, «Dragino Wiki,» 10 2017. [Ηλεκτρονικό]. Available: http://wiki.dragino.com/index.php?title=Lora_Shield. [Πρόσβαση 10 2017].

[19] M. McCauley, «Airspayce,» 10 2017. [Ηλεκτρονικό]. [Πρόσβαση 10 2017].

[20] Kresimir Grgic, et al., «A Web-Based IoT Solution for Monitoring Data Using MQTT Protocol,» *Faculty of EE, CS, and IT Osijek University,* 2016.

[21] ISTSOS, «ISTSOS Website,» 05 2017. [Ηλεκτρονικό]. Available: http://istsos.org/en/trunk/doc/ws_mqtt.html. [Πρόσβαση 08 2017].

[22] MQTT org, «MQTT,» 11 2014. [Ηλεκτρονικό]. Available: http://mqtt.org/. [Πρόσβαση 08 2017].

[23] Adafruit, «Adafruit MQTT Library,» 08 2017. [Ηλεκτρονικό]. Available: https://github.com/adafruit/Adafruit_MQTT_Library. [Πρόσβαση 2017 09].

[24] Texas Instruments, «Lm35 - Texas Instruments,» August 2016. [Ηλεκτρονικό]. Available: www.ti.com/lit/ds/symlink/lm35.pdf. [Πρόσβαση 10 2017].

[25] DFRobot, «PH meter(SKU: SEN0161),» 27 06 2017. [Ηλεκτρονικό]. Available: https://www.dfrobot.com/wiki/index.php/PH_meter(SKU:_SEN0161). [Πρόσβαση 03 09 2017].

[26] EETech Media, diyhacking, «diyhacking,» 03 2017. [Ηλεκτρονικό]. [Πρόσβαση 07 2017].

[27] SENBA OPTICAL & ELECTRONIC CO.,LTD, «GL55 Series Photoresistor,» 04 2016. [Ηλεκτρονικό]. Available: akizukidenshi.com/download/ds/senba/GL55%20Series%20Photoresistor.pdf. [Πρόσβαση 06 2017].

[28] IBM, «Explore MQTT and the Internet of Things service on IBM Bluemix,» IBM Bluemix, 2016. [Ηλεκτρονικό]. Available: https://www.ibm.com/developerworks/cloud/library/cl-mqtt-bluemix-iot-node-red-app/index.html. [Πρόσβαση 09 2017].

[29] Microsoft, «Communicate with your IoT hub using the MQTT protocol,» Microsoft Docs, 2017. [Ηλεκτρονικό]. Available: https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-mqtt-support. [Πρόσβαση 2017].

[30] MQTT, «MQTT used by Facebook Messenger,» 2011. [Ηλεκτρονικό]. Available: http://mqtt.org/2011/08/mqtt-used-by-facebook-messenger. [Πρόσβαση 2017].

5. Παραρτήματα

5.1 Κώδικας Arduino Yun, κεντρικού Node / Gateway

```
#include <SPI.h>
#include <RH_RF95.h>
#include <Bridge.h>
#include <Console.h>
#include <BridgeClient.h>
#include "Adafruit_MQTT.h"
#include "Adafruit_MQTT_Client.h"

/************************ Adafruit.io Setup ************************/

#define AIO_SERVER      "io.adafruit.com"
#define AIO_SERVERPORT  1883
#define AIO_USERNAME    "giannisdogas"
#define AIO_KEY         "c542cb756c9349d99d963dcdbccc8a74"


/************ Global State ************/

// Create a BridgeClient instance to communicate using the Yun's bridge &
Linux OS.
BridgeClient client;

// Setup the MQTT client class by passing in the WiFi client and MQTT server
and login details.
Adafruit_MQTT_Client mqtt(&client, AIO_SERVER, AIO_SERVERPORT, AIO_USERNAME,
AIO_KEY);


/************* Feeds *************/
// Feed Paths
Adafruit_MQTT_Publish thermokrasia = Adafruit_MQTT_Publish(&mqtt,
AIO_USERNAME "/feeds/temp");
Adafruit_MQTT_Publish oxitita = Adafruit_MQTT_Publish(&mqtt, AIO_USERNAME
"/feeds/ph");
```

```
Adafruit_MQTT_Publish fwteinotita = Adafruit_MQTT_Publish(&mqtt, AIO_USERNAME
"/feeds/light");
Adafruit_MQTT_Publish igrasia = Adafruit_MQTT_Publish(&mqtt, AIO_USERNAME
"/feeds/hum");


// onoff kai onffp feeds gia elegxo energopoihtwn, antlias nerou k fwtismoy
Adafruit_MQTT_Subscribe onoffbutton = Adafruit_MQTT_Subscribe(&mqtt,
AIO_USERNAME "/feeds/onoff");
Adafruit_MQTT_Subscribe onoffbuttonp = Adafruit_MQTT_Subscribe(&mqtt,
AIO_USERNAME "/feeds/onoffp");

// Singleton instance of the radio driver
RH_RF95 rf95;

int led = 8;

uint32_t temp=0;
uint32_t ph=0;
uint32_t light=0;
uint32_t hum=0;

String incoming="";

void setup()
{
  pinMode(led, OUTPUT);
  Serial.begin(9600);
  //while (!Serial) ; // Wait for serial port to be available
  if (!rf95.init())
    Serial.println("init failed");

  Bridge.begin();

  // Setup MQTT subscription for onoff and onoffp feeds.
  mqtt.subscribe(&onoffbutton);
  mqtt.subscribe(&onoffbuttonp);

}
```

```
void loop()
{
  mqttloop();
  if (rf95.available())
  {
    // an iparxei minima
    uint8_t buf[RH_RF95_MAX_MESSAGE_LEN];
    uint8_t len = sizeof(buf);
    if (rf95.recv(buf, &len))
    {
      digitalWrite(led, HIGH);
//      RH_RF95::printBuffer("request: ", buf, len);
      Serial.print("got request: ");
      Serial.println((char*)buf);
      incoming=((char*)buf);
      Serial.println(incoming);
      long int intincoming = incoming.toInt();
      Serial.println(intincoming);
      //spliting the digits, gia na xexwrisw tis metriseis
      temp=intincoming/1000000;
      Serial.println(temp);
      ph=((intincoming-temp*1000000)/10000);
      Serial.println(ph);
      light=((intincoming-temp*1000000-ph*10000)/100);
      Serial.println(light);
      hum=(intincoming-temp*1000000-ph*10000-light*100);
      Serial.println(hum);
    }
    else
    {
      Serial.println("recv failed");
    }
  }

}



void MQTT_connect() {
  int8_t ret;
```

```
  // Stop if already connected.
  if (mqtt.connected()) {
    return;
  }

  Serial.print("Connecting to MQTT... ");

  while ((ret = mqtt.connect()) != 0) { // connect will return 0 for
connected
       Serial.println(mqtt.connectErrorString(ret));
       Serial.println("Retrying MQTT connection in 5 seconds...");
       mqtt.disconnect();
       delay(5000);  // wait 5 seconds
  }
  Serial.println("MQTT Connected!");
}

void mqttloop(){
MQTT_connect();

  // this is our 'wait for incoming subscription packets' busy subloop
  Adafruit_MQTT_Subscribe *subscription;
  while ((subscription = mqtt.readSubscription(3000))) {
    if (subscription == &onoffbutton) {
      Serial.print(F("Got: "));
      Serial.println((char *)onoffbutton.lastread);
      // Send a reply
      //uint8_t data[] = "And hello back to you";
      rf95.send((char *)onoffbutton.lastread, 8);
      rf95.waitPacketSent();
      Serial.println("Sent a reply");
    }
  }



//  //Now we can publish stuff!
//  Serial.print(F("\nSending values "));
//  Serial.println(temp);
```

```
//  Serial.println(ph);
//  Serial.println(light);
//  Serial.println(hum);
//  Serial.print("...");
//  //for debugging
  if (! thermokrasia.publish(temp)) {
    //Serial.println(F("Failed sending temp"));
  } else {
    //Serial.println(F("OK sending temp"));
  }
  if (! oxitita.publish(ph)) {
    //Serial.println(F("Failed sending ph"));
  } else {
    //Serial.println(F("OK sending ph"));
  }
  if (! fwteinotita.publish(light)) {
    //Serial.println(F("Failed sending light"));
  } else {
    //Serial.println(F("OK sending light"));
  }
  if (! igrasia.publish(hum)) {
    //Serial.println(F("Failed sending hum"));
  } else {
    //Serial.println(F("OK sending hum"));
  }

// ping the server to keep the mqtt connection alive
//  if(! mqtt.ping()) {
//    //Serial.println(F("MQTT Ping failed."));
//  }

  delay(200);

}
```

5.2 Κώδικας Arduino Uno, για client nodes

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

```
#include <RH_RF95.h>

//definitions for the PH Meter Sensor
#define SensorPin A2            //pH meter Analog output to Arduino Analog
Input 0
#define Offset 0.00            //calibration
#define LED 13
#define samplingInterval 20 //rithmos digmatolipsias
#define printInterval 800
#define ArrayLenth  40    //deigmata
int pHArray[ArrayLenth];   //Store the average value of the sensor feedback
int pHArrayIndex=0;


RH_RF95 rf95;


void setup()
{
  Serial.begin(9600);
  //while (!Serial) ; // Wait for serial port to be available
  if (!rf95.init())
  Serial.println("init failed");
  pinMode(3,OUTPUT); //gia sindesi fwtismoy p.x. se thermokipio
  pinMode(4,OUTPUT); //gia sindesi aftomatou potismatos

}

void loop()
{
  int temp=0;
  int ph=0;
  int light=0;
  int hum=0;
  String datastring="";
  char databuf[10];
  uint8_t dataoutgoing[10];
  Serial.println("Sending to rf95_server");
  temp=readTempData();
```

```
ph=readPhData();
light=readLightData();
hum=readHumData();

//Serial.println(temp);
//Serial.println(ph);
//Serial.println(light);
//Serial.println(hum);
long sendstring;
sendstring=((temp*1000000L)+(ph*10000L)+(light*100L)+hum);
//Serial.println(sendstring);
datastring =dtostrf(sendstring, 8, 0, databuf);
Serial.println(datastring);
strcpy((char *)dataoutgoing,databuf);
rf95.send(dataoutgoing, sizeof(dataoutgoing));

rf95.waitPacketSent();
// Now wait for a reply
uint8_t buf[RH_RF95_MAX_MESSAGE_LEN];
uint8_t len = sizeof(buf);

if (rf95.waitAvailableTimeout(3000))
{
  // Should be a reply message for us now
  if (rf95.recv(buf, &len))
 {
    Serial.print("got reply: ");
    Serial.println((char*)buf);
    //Serial.print("RSSI: ");
    //Serial.println(rf95.lastRssi(), DEC);
    String wordd=(char*)buf;
    Serial.println(wordd);
    if (wordd=="ON"){
    digitalWrite(3,HIGH);
    }
    else if (wordd=="OFF"){
    digitalWrite(3,LOW);
    }
  }
  else
```

```
    {
      Serial.println("recv failed");
    }
  }
  else
  {
    //Serial.println("No reply, is rf95_server running?");
    //debugging
  }
  delay(1000);
}

int readTempData(){
  int sum=0;
  int cel;

  for (int i=0;i<=19;i++){
    int tempADCreading=analogRead(A0);
    int mv = (tempADCreading/1024.0)*5000;
    cel = mv/10;
    //Serial.println(tempADCreading); //gia debugging
    //Serial.println(mv); //gia debugging
    //Serial.println(cel); //gia debugging
    sum+=cel;
  }
  //Serial.println(sum); //gia debugging
  cel=sum/20;

  //Serial.println(cel); //gia debugging
  return cel;
}


int readLightData(){
  int fwsreading=analogRead(A1)-50; //se lux (!)
  Serial.println(fwsreading);
```

```
  if (fwsreading>99){
    fwsreading=99; //just to make sure
  }
  return fwsreading;
}

int readPhData(){
static unsigned long samplingTime = millis();
  static unsigned long printTime = millis();
  static float pHValue,voltage;
  //thelw opwsdipote digmatolipsia, gia megaliteri akriveia
  if(millis()-samplingTime > samplingInterval)
  {
      pHArray[pHArrayIndex++]=analogRead(SensorPin);
      if(pHArrayIndex==ArrayLenth)pHArrayIndex=0;
      voltage = avergearray(pHArray, ArrayLenth)*5.0/1024;
      pHValue = 3.5*voltage+Offset;
      samplingTime=millis();
  }
  if(millis() - printTime > printInterval)   //Every 800 milliseconds, print
a numerical, convert the state of the LED indicator
  {
  Serial.print("Voltage:");
        Serial.print(voltage,2);
        Serial.print("    pH value: ");
  Serial.println(pHValue,2);
        digitalWrite(LED,digitalRead(LED)^1);
        printTime=millis();
  }
  return pHValue;
}

//deigmatolipsia gia megaliteri akriveia ston PH sensotr
double avergearray(int* arr, int number){
  int i;
  int max,min;
  double avg;
  long amount=0;
  if(number<=0){
    // Serial.println("Error number for the array to avraging!/n");
```

```
    return 0;
  }
  if(number<5){   //less than 5, calculated directly statistics
    for(i=0;i<number;i++){
      amount+=arr[i];
    }
    avg = amount/number;
    return avg;
  }else{
    if(arr[0]<arr[1]){
      min = arr[0];max=arr[1];
    }
    else{
      min=arr[1];max=arr[0];
    }
    for(i=2;i<number;i++){
      if(arr[i]<min){
        amount+=min;        //arr<min
        min=arr[i];
      }else {
        if(arr[i]>max){
          amount+=max;    //arr>max
          max=arr[i];
        }else{
          amount+=arr[i]; //min<=arr<=max
        }
      }//if
    }//for
    avg = (double)amount/(number-2);
  }//if
  return avg;
}



int readHumData(){

int humid=analogRead(A3);
humid=map(humid,550,0,0,100);
```

```
return humid;


}
```