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

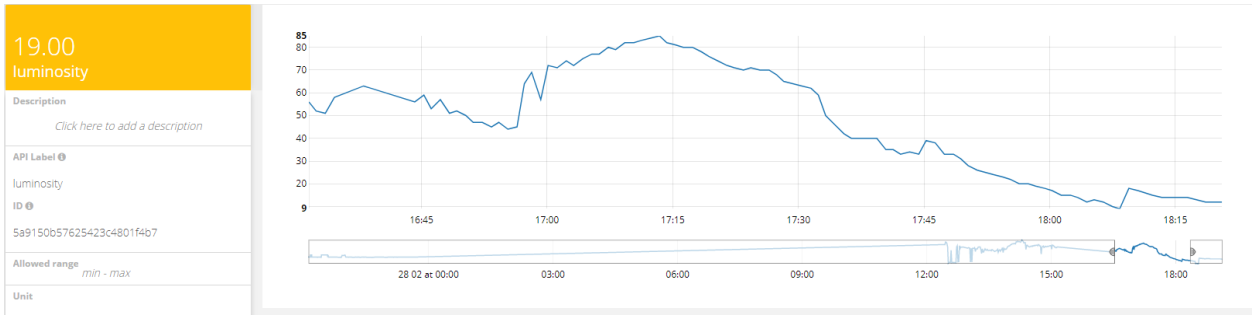
PIRAEUS UNIVERSITY of APPLIED SCIENCES
FACULTY OF ENGINEERING
Department of Electronics Engineering

Πρόγραμμα Μεταπτυχιακών Σπουδών
Διαδικτυωμένα Ηλεκτρονικά Συστήματα

Master of Science in
Internetworked Electronic Systems

ΜΕΤΑΠΤΥΧΙΑΚΗ ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Εφαρμογές έξυπνου φωτισμού σε τοπικό δίκτυο και προσαρμόσιμη λειτουργικότητα, βασισμένου σε τεχνολογία Ασύρματων Δικτύων Αισθητήρων για τους τοπικούς κόμβους και μικρού εύρους ζώνης – μεγάλης εμβέλειας για απομακρυσμένο έλεγχο



Μεταπτυχιακός Φοιτητής : Κοντράρος Μιχαήλ, Α.Μ. 0014

Επιβλέπων: Παπαγέωργας Παναγιώτης, Καθηγητής

ΑΙΓΑΛΕΩ, ΦΕΒΡΟΥΑΡΙΟΣ 2018

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

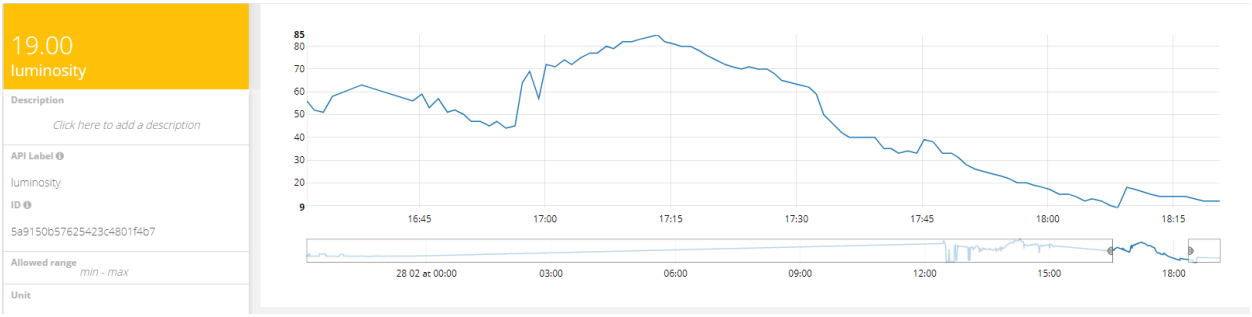
Διαδικτυωμένα Ηλεκτρονικά Συστήματα

Master of Science in

Internetworked Electronic Systems

MSc Thesis

Smart Lighting with enhanced services, applications with Intelligent and weather adaptive lighting in street light, based on local Wireless Sensor Networking technology and narrow bandwidth – long range networks for remote control



Student: Kontraros Michail, Reg. Nr. 0014

MSc Thesis Supervisor: Papageorgas Panagiotis, Professor

ATHENS-EGALEO, FEBRUARY 2018

ΠΕΡΙΛΗΨΗ

Αντικείμενο της παρούσας διπλωματικής είναι η μελέτη και η αξιολόγηση τεχνολογιών που μπορούν να χρησιμοποιηθούν σε εφαρμογές έξυπνου φωτισμού με στόχο την υλοποίηση πειραματικής πλατφόρμας χαμηλού κόστους και ανοικτού κώδικα που θα μπορούσε να χρησιμοποιηθεί από την ερευνητική κοινότητα. Αρχικά γίνεται μία εισαγωγή και παρουσιάζεται το θεωρητικό υπόβαθρο των υπαρχουσών τεχνολογιών αιχμής και των υλικών που χρησιμοποιούνται σε εφαρμογές φωτισμού ιδιαίτερα σε δρόμους (street lighting). Στην συνέχεια παρουσιάζεται μια συγκριτική μελέτη μεταξύ των ασυρμάτων πρότυπων που χρησιμοποιούνται και παραθέτονται τα συμπεράσματα από αυτά. Τέλος αναλύονται οι προδιαγραφές του συστήματος που σχεδιάσθηκε και η υλοποίηση του ανά επίπεδο. Στην πειραματική διάταξη που αναπτύχθηκε οι ασύρματοι κόμβοι (nodes) λαμβάνουν δεδομένα που έχουν σχέση με τον φωτισμό στην συνέχεια μέσω συγκεκριμένων σεναρίων μεταβάλλουν αυτόματα την φωτεινότητα, αλλά και μέσω ενεργοποιητών δύναται να μεταβάλλουν την κατάσταση ενός LED Engine. Ο απομακρυσμένος έλεγχος πραγματοποιείται με την χρήση διαδικτυακής τεχνολογίας του τύπου «νέφους» και συγκεκριμένα μέσω της πλατφόρμας Ubidots από οποιοδήποτε υπολογιστικό σύστημα με διαδικτυακή πρόσβαση και το οποίο διαθέτει τα κατάλληλα πιστοποιητικά προσβασιμότητας “credentials”.

ΛΕΞΕΙΣ – ΚΛΕΙΔΙΑ: Διαδίκτυο των Πραγμάτων, Συλλογή Δεδομένων, Διαδίκτυο των Πραγμάτων στον Έξυπνο Φωτισμό, Δίκτυα Μεγάλων Αποστάσεων, Έξυπνος Φωτισμός, Ασύρματα δίκτυα αισθητήρων, Lora, Dash7, Samsung Artik

ABSTRACT

The main objectives of this thesis are the study and the evaluation of technologies that are suitable for Smart Lighting applications and the development of an open-source cloud-based prototype platform for research experimentation. The first part of the thesis includes an introduction to the theoretical background of the technologies and the hardware implementations proposed from bibliography, as well as the state of the art technologies that are widely acceptable for street lighting monitoring and control. In the following sections, a comparison between the wireless standards we are using and the conclusions extracted is presented. Finally, the specifications of the system are presented with the components of the implementation designed, in a level-by-level analysis. In the prototype developed, the Edge Nodes are receiving data according to the implemented lighting control scenario. As a demonstration, we present a LED Engine dimming and status control application. The approach proposed makes remote control operation possible through the use of the cloud platform Ubidots from any device which has Internet access and with the appropriate credentials.

KEYWORDS: IoT, Data Collection, Internet of Things, IoT in Smart Lighting, Long Range Network, LoRa, Dash7, Smart Lighting, Wireless Sensor Networks, Samsung Artik

ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα να εκφράσω τις θερμές μου ευχαριστίες στον επιβλέποντα καθηγητή κ. Παναγιώτη Παπαγέωργα καθώς και στον συνεργάτη του κ. Πυρομάλη Δημήτριο για την υποστήριξή τους κατά την διάρκεια της εκπόνησης της παρούσας εργασίας. Τέλος θα ήθελα να ευχαριστήσω την οικογένειά μου και τους κοντινούς μου ανθρώπους για την ηθική και ψυχολογική υποστήριξη που μου παρείχαν όλο αυτό το διάστημα.

ΠΙΝΑΚΑΣ ΑΚΡΩΝΥΜΙΩΝ-ΣΥΝΤΟΜΟΓΡΑΦΙΩΝ

LoWPAN	Low-Power Wireless Personal Area Networks
6LoWPAN	IPv6 over Low-Power Wireless Personal Area Networks
GND	Ground
HTTP	Hypertext Transfer Protocol
IDE	Integrated Development Environment
IoT	Internet of Things
LED	Light Emitting Diode
D7A	Dash7
LoRa	Long Range
LPWAN	Low Power Wide Area Network
MAC	Media Access Control
MCU	Microcontroller Unit
MQTT	Message Queue Telemetry Transport
NB-IoT	NarrowBand Internet of Things
CoAP	Constrained Application Protocol
PWM	Pulse Width Modulation
SPI	Serial Peripheral Interface
I^2C	Inter-Integrated Circuit
TCP	Transmission Control Protocol
TCP/IP	Transmission Control Protocol/Internet Protocol
WiFi	Wireless Fidelity
WSN	Wireless Sensor Network
OSI	Open Systems Interconnection
OTA	Over The Air

ΠΕΡΙΕΧΟΜΕΝΑ

ΠΙΝΑΚΑΣ ΕΙΚΟΝΩΝ

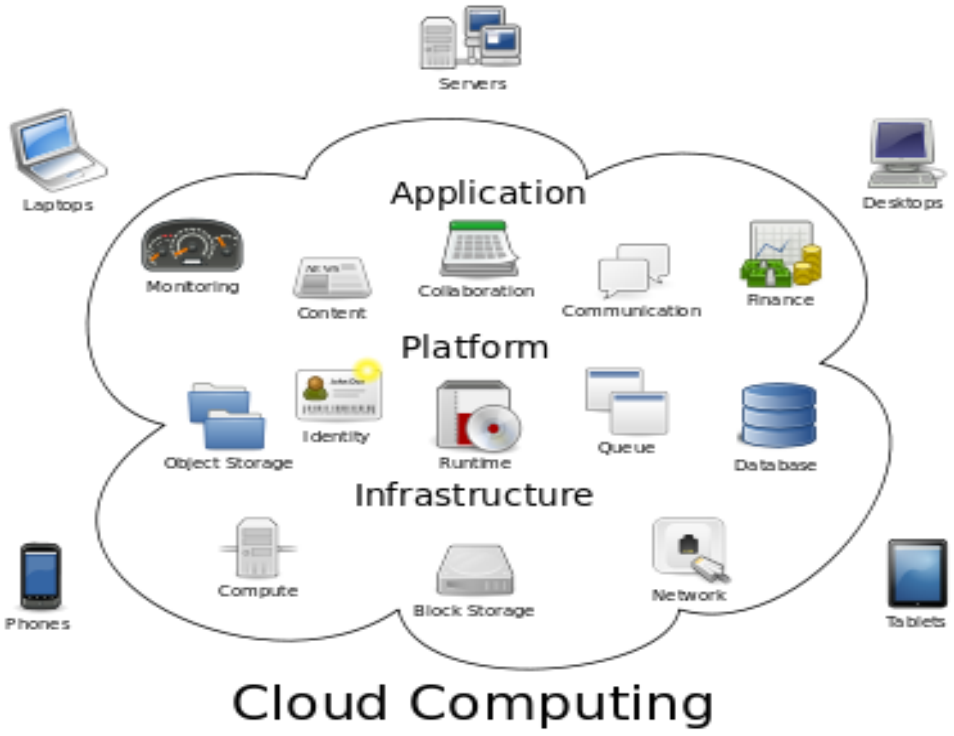
Κεφάλαιο 1

Εισαγωγή

1.1 Η έννοια του cloud computing

Η έννοια του cloud computing είναι πάρα πολύ δύσκολο να εξηγηθεί. Αρχικά θα μπορούσαμε να πούμε ότι με τον όρο cloud εννοούμε ένα σύννεφο με το οποίο προσπαθούμε να περιγράψουμε ένα απομακρυσμένο σύνολο αξιόπιστων υπηρεσιών στο οποίο και στηριζόμαστε, χωρίς όμως να μας ενδιαφέρει το πώς λειτουργεί αυτό στα ενδότερα του. Εννοιολογικά αυτό ονομάζεται utility ή grid computing.

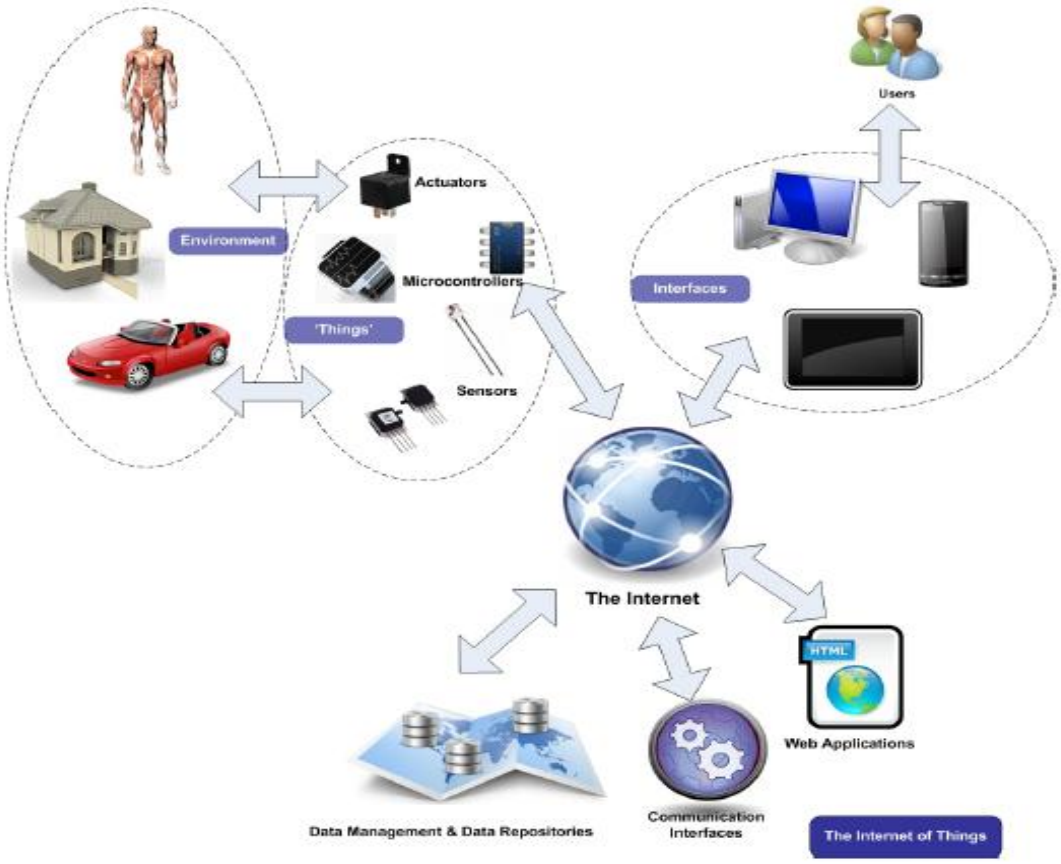
Το Cloud computing κληρονομεί τα χαρακτηριστικά του utility computing και επιπλέον παρέχει ένα δυναμικό και ελαστικό περιβάλλον διάθεσης υπηρεσιών το οποίο μπορεί να είναι ανθεκτικό σε ραγδαίες και γιγαντιαίας κλίμακας μεταβολές των συνθηκών του. Αυτό επιτυγχάνεται με τα εγγενή χαρακτηριστικά του, που είναι η αυτόματη ανάκαμψη, η αυτοεπιτήρηση και η αυτοδιαχείριση. Παρατηρώντας τα παραπάνω μπορούμε εύκολα να διαπιστώσουμε ότι πρόκειται για κάτι πολύ μεγαλύτερης κλίμακας και πλουσιότερο από το utility computing και την τεχνολογία του virtualization, το οποίο σε συνδυασμό με τεράστιες οικονομίες κλίμακας που προσφέρει, θα αποτελέσει τη νέα εποχή του cloud computing, [1].



Εικόνα 1 Δομή ενός τυπικού συστήματος Cloud Computing, [1].

1.2 Η έννοια του Internet of Things

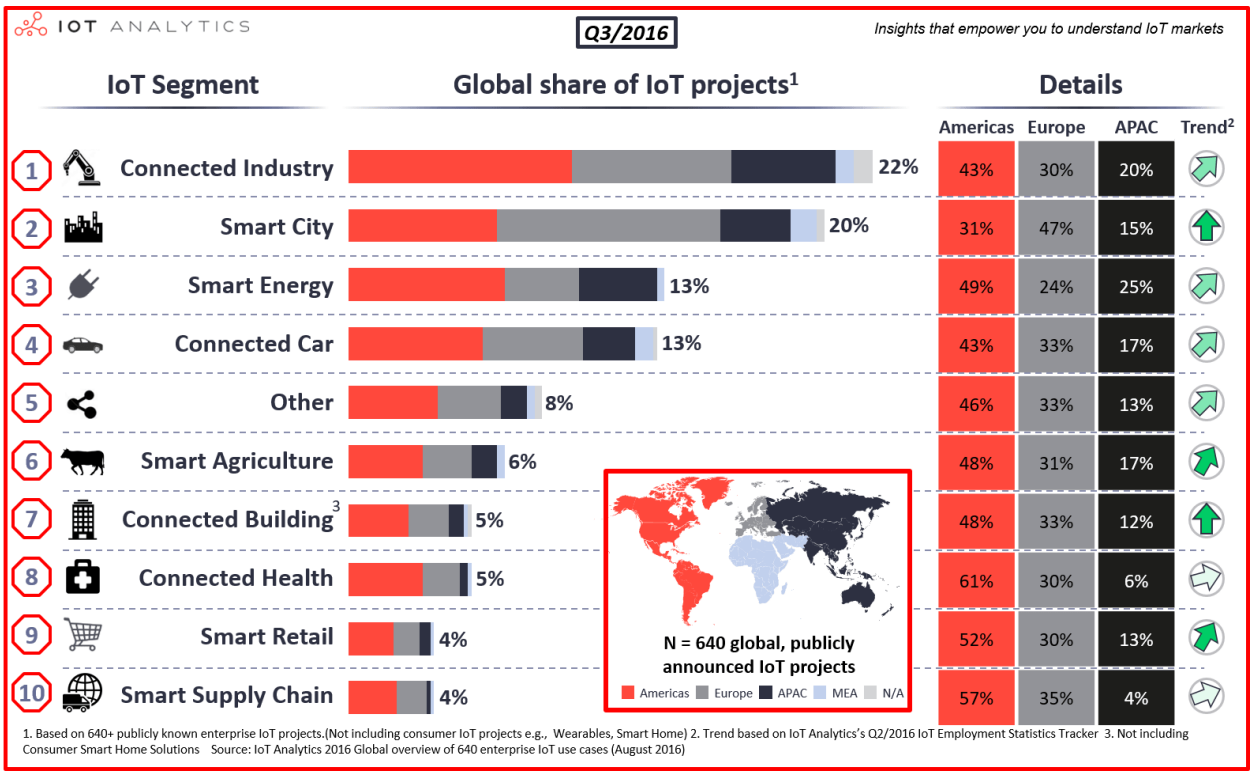
Το Internet of Things (Διαδίκτυο των Αντικειμένων) αναφέρεται στο παγκόσμιο δίκτυο που δίνει την δυνατότητα σε διατάξεις οι οποίες ανεξάρτητα πολυπλοκότητας μπορούν να επικοινωνούν μεταξύ τους συνεχώς κάνοντας χρήση κατάλληλων πρωτοκόλλων επικοινωνίας, να ανταλλάσσουν πληροφορίες μέσω του Διαδικτύου, να αποθηκεύουν και να επαναφέρουν δεδομένα, να ενημερώνουν το υλικο-λογισμικό τους (firmware) δημιουργώντας ένα έξυπνο, συνεχώς συνδεμένο στο Διαδίκτυο περιβάλλον (Cyber Physical). Αν και είναι σχετικά νέα σαν έννοια, υπάρχει μια ταχεία ανάπτυξη και υπάρχουν ήδη κάποιες πλατφόρμες οι οποίες είναι διαθέσιμες και μέσω αυτών μπορούμε να διαχειριστούμε-επεξεργαστούμε και να απεικονίσουμε δεδομένα που λαμβάνονται από διάφορα αισθητήρια. Μερικά παραδείγματα τέτοιων πλατφορμών είναι το ThingSpeak καθώς και το Ubidots στο οποίο θα εστιάσουμε στην παρούσα διπλωματική, [2].



Εικόνα 2 Δομή ενός τυπικού συστήματος Internet of Things [3]

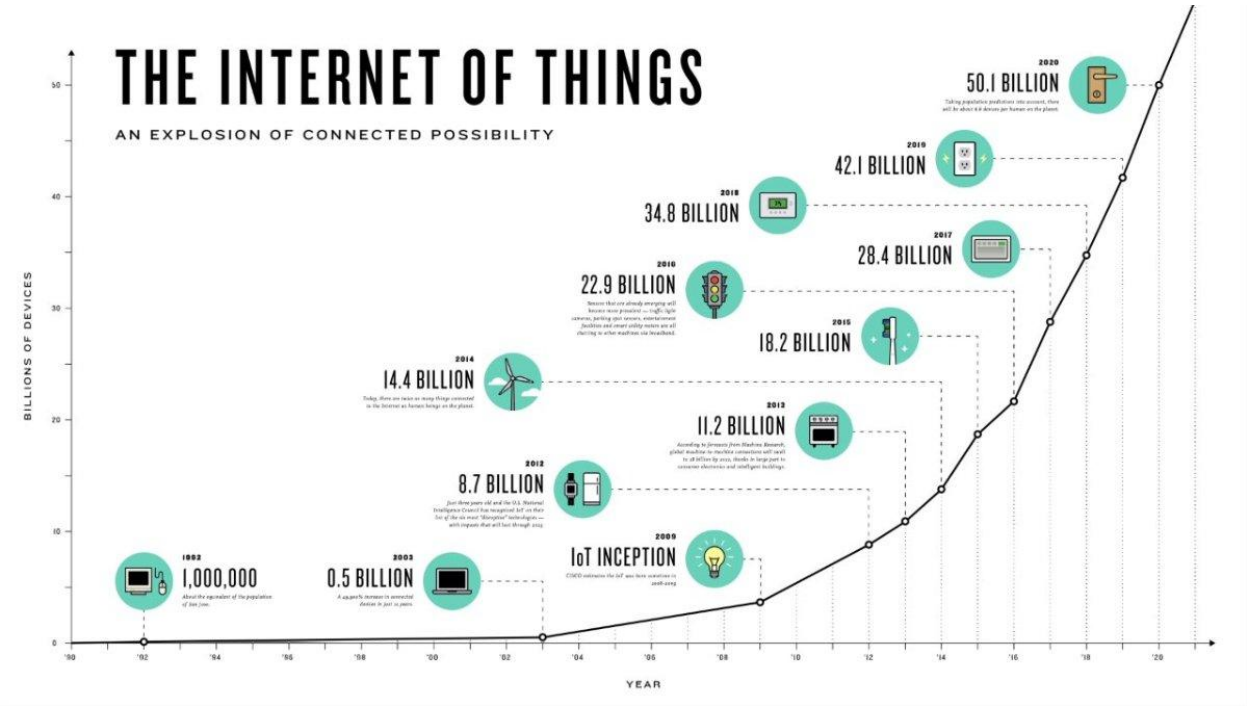
1.2.1 Τάσεις

Τα συστήματα που κάνουν χρήση του Internet of Things έχουν εισβάλει για τα καλά στην ζωή του ανθρώπου καθώς υφίσταται ταχεία ανάπτυξη και εξάπλωση των τεχνολογιών σε σχεδόν όλους του τομείς της αγοράς και των βιομηχανιών. Γενικά υπάρχει μία τάση να δημιουργηθούν νέες εφαρμογές που θα συγκεντρώνουν δεδομένα σε πραγματικό χρόνο από διαφορετικές σειρές από συνδεμένες συσκευές με στόχο να τα συνδυάσουν με άλλες πληροφορίες που έχουν αντληθεί από διαφορετικές πηγές, καθώς και η υλοποίηση συστημάτων που εκτός από ανάγνωση αισθητηρίων έχουν πια και αυτοματισμούς βάσει συγκεκριμένων σεναρίων οι οποίοι διαχειρίζονται ποικιλία ενεργοποιητών και θα είναι δυνατή η διαχείριση μέσω μίας cloud πλατφόρμας.



Εικόνα 3 Τάση ανάπτυξης εφαρμογών IoT ανά κατηγορία, [4].

Στο παρακάτω γράφημα εμφανίζεται πρόβλεψη που αναφέρει ότι μέχρι το έτος 2020 οι διασυνδεμένες συσκευές θα ξεπεράσουν τα 50.000.000.000.

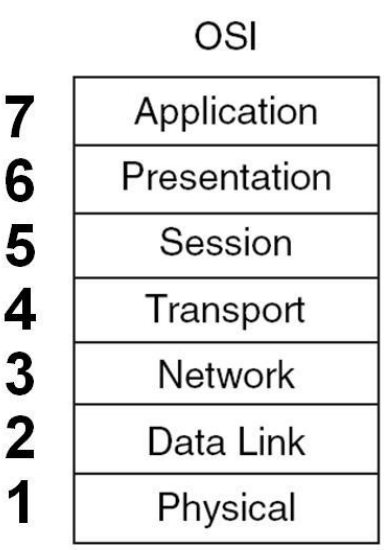


Εικόνα 4 Πρόβλεψη διασυνδεδεμένων συσκευών έως το 2020, [5].

1.3 Το μοντέλο αναφοράς OSI

Το μοντέλο της Διασύνδεσης Ανοιχτών Συστημάτων (OSI, Open Systems Interconnection) είναι ένα θεωρητικό μοντέλο που περιγράφει τον τρόπο με τον οποίο μπορούν να επικοινωνήσουν μεταξύ τους δύο οποιαδήποτε διαφορετικά συστήματα. Είναι γνωστό και ως μοντέλο των επτά επιπέδων.

Το μοντέλο OSI υποδιαιρεί τις λειτουργίες ενός τηλεπικοινωνιακού δικτύου σε επίπεδα, για το καθένα από τα οποία μπορεί να οριστεί κάποιο πρωτόκολλο σε μία συγκεκριμένη υλοποίηση. Το κάθε πρωτόκολλο υλοποιείται είτε σε υλικό είτε σε λογισμικό.



Function	Data Unit	Example Protocols	Devices
Interfaces between network and application software, and includes authentication services	Data	Email, http, FTP	Firewall, Server
Defines the format and organization of data, including encryption	Data	ASCII, JPEG	
Defines how to start, control and end conversations (sessions) between endpoints.	Data	NetBIOS	
Provides connection establishment between computers including connection establishment/termination, flow control, error recovery, and segmentation of Data	Segment	TCP/UDP	Router
Logical addressing, routing and path determination between devices	Packet	IP	Router
Formats data into frames for transmission across some physical medium. Includes the Media Access Control and Logical Link Control Sublayers.	Frame	Ethernet, Frame Relay, PPP	Switch, Wireless Access Point
Physical characteristics of the media being used, including electrical connectors, voltages, etc.	Bits	RJ45, 802.3, V.35	LAN Hub

Εικόνα 5 Μοντέλο Αναφοράς OSI, [6].

Τα επίπεδα του μοντέλου αναλύονται παρακάτω:

1. Φυσικό επίπεδο (Physical Layer)

   Το φυσικό επίπεδο αφορά τις φυσικές προδιαγραφές της επικοινωνίας. Σε αυτές περιλαμβάνονται τα ηλεκτρικά, μηχανικά και λειτουργικά χαρακτηριστικά των διασυνδέσεων των δύο υπολογιστικών συστημάτων. Μερικά παραδείγματα είναι οι προδιαγραφές των πρωτοκόλλων του RS-232, του Ethernet και του IEEE 802.11.

2. Ζεύξης Δεδομένων (Data Link Layer)

   Το επίπεδο ζεύξης δεδομένων παρέχει τα λειτουργικά και διαδικαστικά μέσα για τη μεταφορά δεδομένων από μια συσκευή σε μία άλλη. Οι μη ιεραρχημένες διευθύνσεις των συσκευών εδώ είναι οι φυσικές (π.χ. MAC διευθύνσεις), δηλαδή είναι προκαθορισμένες διευθύνσεις από το εργοστάσιο παραγωγής. Μερικά παραδείγματα είναι το Ethernet αλλά και το 802.11 για τα ασύρματα δίκτυα.

3. Επίπεδο δικτύου (Network Layer)

   Το επίπεδο δικτύου παρέχει τα λειτουργικά και διαδικαστικά μέσα για τη μεταφορά δεδομένων από τον αποστολέα προς τον παραλήπτη, μέσω ενός ή πολλαπλών δικτύων. Εκτελεί τις λειτουργίες δρομολόγησης και αναφέρει σφάλματα σχετικά με

την παράδοση των πακέτων. Οι δρομολογητές (router) λειτουργούν στο επίπεδο αυτό. Κλασσικό παράδειγμα πρωτοκόλλου είναι το IP (Internet Protocol).

4. Επίπεδο Μεταφοράς (Transport Layer)

   Στο Επίπεδο Μεταφοράς υλοποιείται το κανάλι επικοινωνίας μεταξύ των τερματικών κόμβων, μέσω του οποίου θα μεταβιβάζονται αξιόπιστα τα μηνύματά τους. Είναι υπεύθυνο για τον έλεγχο αξιοπιστίας ενός καναλιού, την κατάτμηση και αποτμηματοποίηση, καθώς και τον έλεγχο σφαλμάτων. Παραδείγματα πρωτοκόλλων που λειτουργούν σε αυτό το επίπεδο είναι το TCP και το UDP.

5. Επίπεδο Συνόδου (Session Layer)

   Σε αυτό το επίπεδο διενεργούνται όλες οι απαραίτητες λειτουργίες για την εγκαθίδρυση, την επίβλεψη και τον τερματισμό των συνόδων (sessions) μεταξύ των τελικών εφαρμογών. Επίσης είναι σε θέση να επιτρέπει ή απαγορεύει συγκεκριμένη παροχή υπηρεσίας, να αποκαθιστά νέα σύνδεση όταν η πρώτη για κάποιο λόγο διακοπεί και να επιτρέπει την επικοινωνία είτε αμφίδρομη είτε μονόδρομη.

6. Επίπεδο Παρουσίασης (Presentation Layer)

   Το επίπεδο αυτό έχει σχέση με την αναπαράσταση της πληροφορίας που μεταφέρεται από εφαρμογή σε εφαρμογή και έχει ως κύρια λειτουργία την εξασφάλιση της αναγνωσιμότητάς τους, ακόμα και μεταξύ κόμβων που χρησιμοποιούν διαφορετικές μορφές αναπαράστασης της πληροφορίας. Στο επίπεδο αυτό τα δεδομένα υφίστανται κρυπτογράφηση, συμπίεση, κωδικοποίηση MIME και όποια άλλη διαμόρφωση απαιτεί η μορφή δεδομένων ή ο σχεδιαστής του πρωτοκόλλου. Παραδείγματα αποτελούν η μετατροπή αρχείων από κώδικα EBCDIC σε κώδικα ASCII και η μετατροπή της δομής των δεδομένων σε μορφή XML ή αντίστροφα.

7. Επίπεδο Εφαρμογών (Application Layer)

   Το Επίπεδο Εφαρμογής παρέχει ένα σύνολο δικτυακών υπηρεσιών στις τελικές εφαρμογές των χρηστών. Στο επίπεδο αυτό γίνεται η διαχείριση των

κατανεμημένων εφαρμογών, η αποστολή του ηλεκτρονικού ταχυδρομείου και πολλά άλλα. Παραδείγματα πρωτοκόλλων που λειτουργούν στο επίπεδο εφαρμογών είναι το Telnet, το FTP, το SMTP και το HTTP, [7].

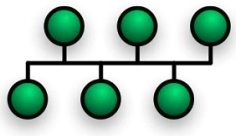
1.4 Τοπολογίες δικτύων

Οι κυριότερες τοπολογίες δικτύων που υπάρχουν είναι οι εξής:

- Τοπολογία διαύλου
- Τοπολογία δακτυλίου
- Τοπολογία αστέρα
- Κατανεμημένη τοπολογία
- Πλήρως κατανεμημένη τοπολογία

1.4.1 Τοπολογία διαύλου

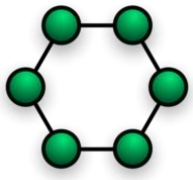
Στην τοπολογία διαύλου όλες οι συσκευές συνδέονται με ένα κεντρικό καλώδιο, το οποίο αποκαλείται bus. Αυτή η τοπολογία τείνει να εκλείψει λόγω της ευρείας ανάπτυξης της τοπολογίας αστέρα και των τοπολογιών κατανομής. Στο παρακάτω σχήμα απεικονίζεται ένα τοπικό δίκτυο διαύλου.



Εικόνα 6 Υλοποίηση τοπολογίας διαύλου.

1.4.2 Τοπολογία δακτυλίου

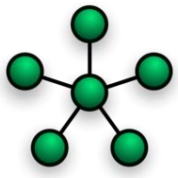
Στην τοπολογία δακτυλίου (ring) όλες οι συσκευές συνδέονται με μορφή ενός κλειστού βρόχου, έτσι ώστε κάθε συσκευή συνδέεται άμεσα με δύο άλλες συσκευές, ένα από κάθε πλευρά . Οι τοπολογίες δακτυλίων είναι σχετικά ακριβές και δύσκολο να εγκατασταθούν, αλλά προσφέρουν το υψηλό εύρος ζώνης και μπορούν να εκταθούν σε μεγάλες αποστάσεις. Παραδείγματα τέτοιων τοπολογιών αποτελούν το token ring και το FDDI.



Εικόνα 7 Υλοποίηση τοπολογίας δακτυλίου

1.4.3 Τοπολογία αστέρα

Στην τοπολογία αστέρα (Star) όλες οι συσκευές συνδέονται σε έναν κεντρικό κόμβο (coordinator) ο οποίος είναι υπεύθυνος για τον έλεγχο επικοινωνίας μεταξύ των απομακρυσμένων κόμβων (nodes). Τα δίκτυα τοπολογίας αστέρα είναι σχετικά εύκολο να εγκατασταθούν και να διαχειριστούν, και είναι η πιο κοινή τοπολογία που υπάρχει σήμερα, είτε αυτό αφορά δίκτυα αισθητήρων είτε δίκτυα υπολογιστών είτε οτιδήποτε άλλο. Στο παρακάτω σχήμα απεικονίζεται ένα τοπικό δίκτυο τοπολογίας αστέρα με πέντε υπολογιστές. [8]

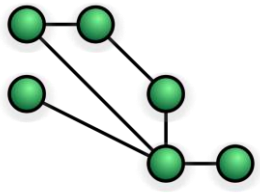


Εικόνα 8 Υλοποίηση τοπολογίας αστέρα

1.4.4 Τοπολογία κατανομής

Το επόμενο στάδιο πολυπλοκότητας είναι η κατανεμημένη τοπολογία (mesh topology), η οποία παίρνει ένα σύνολο τοπολογιών αστέρα και προσθέτει πλεονάζουσες συνδέσεις μεταξύ των διακοπτών. Δηλαδή δημιουργούνται εναλλακτικοί δρόμοι επικοινωνίας μεταξύ των υπολογιστών. Παράδειγμα τα τηλεφωνικά δίκτυα είναι μίξη της τοπολογίας αστέρα και της κατανεμημένης. Η αξιοπιστία είναι μεγάλη λόγω της δρομολόγησης πολλαπλής διαδρομής, [9]. Χαρακτηριστικά της κατανεμημένης τοπολογίας είναι τα εξής:

- Τα πακέτα πραγματοποιούν πολλαπλά άλματα (multiple hops) προκειμένου να φτάσουν στον προορισμό τους.
- Η εμβέλεια του δικτύου μπορεί να αυξηθεί κατακόρυφα καθώς οι κόμβοι γίνονται αυτόματα και επαναλήπτες (repeaters)
- Ελαχιστοποιεί τις νεκρές ζώνες.
- Σε περίπτωση που δεν είναι δυνατή η αποστολή δεδομένων από μία διαδρομή, ο κόμβος θα βρει εναλλακτική διαδρομή.
- Εύκολη προσθήκη και αφαίρεση συσκευών.
- Κάθε συσκευή προέλευσης μπορεί να επικοινωνήσει με την συσκευή προορισμού.
- Τα πρωτόκολλα δρομολόγησης είναι πιο πολύπλοκα από αυτά της τοπολογίας αστέρα.
- Μεγαλύτερο traffic λόγω μεγαλύτερης επικεφαλίδας από αυτά της τοπολογίας αστέρα.



Εικόνα 9 Υλοποίηση Τοπολογίας Mesh.

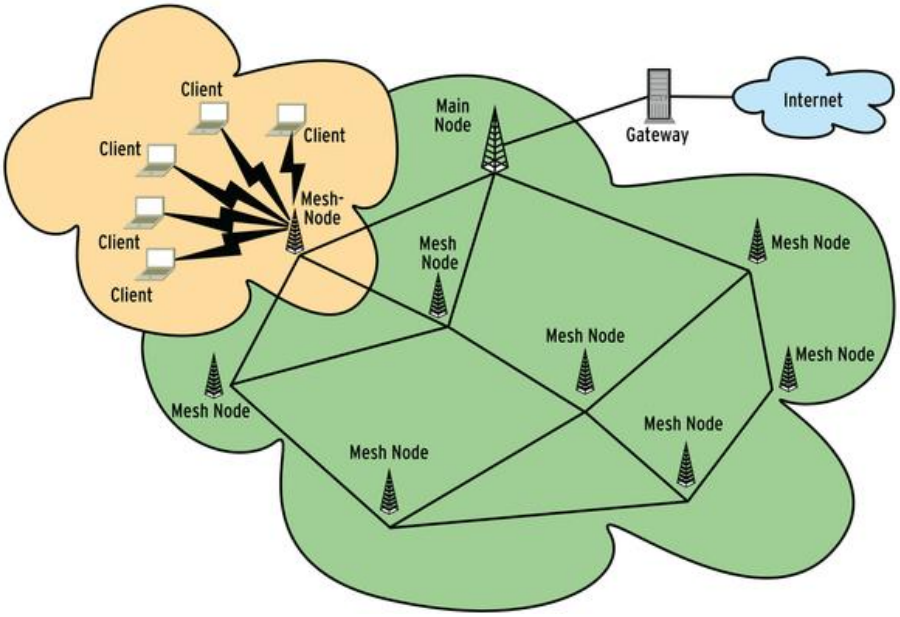
1.5 Ασύρματα Δίκτυα Αισθητήρων και τεχνολογίες (WSN)

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 φωτεινότητα κτλ. και δύναται 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, την παρακολούθηση των μηχανημάτων υγείας, για διαχείριση ενέργειας και πολλά άλλα.

Ένα Ασύρματο Δίκτυο Αισθητήρων αποτελείται από αριθμό κόμβων, όπου κάθε κόμβος συνδέετε σε έναν (η κάποιες φορές σε αρκετούς) αισθητήρες. Κάθε τέτοιος κόμβος του δικτύου αισθητήρων έχει χαρακτηριστικά μερικά κομμάτια: 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. Οι περιορισμοί σε μέγεθος και κόστος έχουν ως αποτέλεσμα αντίστοιχους περιορισμούς σε πόρους όπως ενέργεια, μνήμη, υπολογιστική ταχύτητα και στο εύρος ζώνης των επικοινωνιών. Η τοπολογία των αισθητήρων μπορεί να διαφέρει, αλλά συνήθως όταν αναφερόμαστε σε δίκτυα αισθητήρων μιλάμε για τοπολογία αστέρα ή τοπολογία κατανομής.

Στην τοπολογία αστέρα όλοι οι κόμβοι αποστέλλουν τα δεδομένα στον coordinator ο οποίος συνδέεται (ή υλοποιεί επιπρόσθετα) σε κάποια πύλη δικτύου (gateway) μέσω της οποίας υλοποιείται η διασύνδεση με το Διαδίκτυο και κατ' επέκταση με τον κατάλληλο server για επεξεργασία. Στην τοπολογία mesh τα δεδομένα αποστέλλονται από τους κόμβους σε κάποια πύλη και στην συνέχεια από εκεί μεταδίδονται για περαιτέρω επεξεργασία, αλλά σε περίπτωση που κάποιος κόμβος είναι εκτός εμβέλειας, είναι δυνατόν να σταλούν τα μηνύματα σε κάποιο 3ο κόμβο το οποίο θα τα προωθήσει με την σειρά του ώστε να παραληφθούν από το gateway. Αυτή η διαδικασία ονομάζεται αναπήδηση (hop). Ο αριθμός των hops αναφέρεται στον αριθμό των προωθήσεων που απαιτούνται για την τελική λήψη των δεδομένων από το Gateway.



Εικόνα 10 Τοπολογία κατανομής Ασύρματου Δικτύου Αισθητήρων, [10].

Αυτή η τοπολογία μπορεί αρχικά να φαίνεται αποδοτική καθώς αυξάνει την εμβέλεια, αλλά αυξάνει και την πολυπλοκότητα του συστήματος, αυξάνει την κίνηση του δικτύου και μειώνει την αυτονομία των Nodes που βρίσκονται κοντά στο gateway, καθώς θα πρέπει να γίνουν τα hops μέσω αυτών.

Παρακάτω γίνεται παρουσίαση τεχνολογιών που χρησιμοποιούνται σε ασύρματα δίκτυα αισθητήρων.

1.5.1 Sigfox

Η Sigfox είναι μία Γαλλική εταιρεία η οποία ιδρύθηκε το 2009. Ο τομέας της ήταν η υλοποίηση ασυρμάτων δικτύων για την διασύνδεση συσκευών χαμηλής ενέργειας, οι οποίοι πρέπει να στέλνουν, συνεχώς, δεδομένα χαμηλού όγκου.

Εικόνα 11 Λογότυπο του Sigfox, [11].

Το πρωτόκολλο Sigfox χρησιμοποιεί ιδιόκτητη τεχνολογία, διαμόρφωση ultra narrow band και χαμηλό ρυθμό μετάδοσης δεδομένων για την επίτευξη μεγαλύτερης εμβέλειας. Το sigfox χρησιμοποιεί την τοπολογία αστέρα και απαιτεί έναν κινητό χρήστη για την διαχείριση της μετάδοσης των πακέτων.

Συνεργάτες του Sigfox είναι μεγάλα ονόματα της παγκόσμιας αγοράς ηλεκτρονικών, όπως η Texas Instruments, η Silicon Labs και η ON Semiconductor.

Τέλος το Sigfox μπορεί να χρησιμοποιηθεί για την διασύνδεση συσκευών σε πολύ μεγάλη απόσταση αλλά και για την διασύνδεση συσκευών που βρίσκονται υπόγεια πχ. έξυπνους μετρητές σε φρεάτια. [11]

1.5.2 NarrowBand IoT (NB-IoT)

Το NB-IoT είναι ένα πρότυπο ασύρματης δικτύωσης για δίκτυα μεγάλων αποστάσεων χαμηλής κατανάλωσης ενέργειας. Σχεδιάστηκε από την 3GPP (3rd Generation Partnership Project) και επικεντρώνεται κυρίως στην κάλυψη εσωτερικών χώρων. Χαρακτηριστικά του η μεγάλη διάρκεια μπαταρίας, ο μεγάλος αριθμός κόμβων και όλα αυτά με όσο το δυνατόν χαμηλότερο κόστος. Χρησιμοποιεί διαμόρφωση DSSS και το ίδιο φάσμα συχνοτήτων με το LTE, το οποίο σημαίνει ότι όποια υλοποίηση πρέπει να έχει και την κατάλληλη άδεια. Επίσης για την βελτίωση των χαρακτηριστικών του και διότι δεν είναι αναγκαίο, ο ρυθμός μετάδοσης δεδομένων είναι σημαντικά χαμηλότερος από ότι το συμβατικό LTE, [12]. Επίσης το NB-IoT βρίσκεται ακόμη σε στάδιο ανάπτυξης, έχουν ήδη υλοποιηθεί κατάλληλες λύσεις σε επίπεδο υλικού από εταιρίες όπως η ublox και έχουν ξεκινήσει πειραματικές εφαρμογές σε διάφορες χώρες, ενώ πλήρης ανάπτυξη αυτών των υπηρεσιών αναμένεται το 2019.

Πίνακας 1 Χαρακτηριστικά NB IoT, [13].

	LTE Cat 1	LTE Cat 0	LTE Cat M1 (eMTC)	LTE Cat NB1 (NB-IoT)	EC-GSM-IoT
3GPP Release	Release 8	Release 12	Release 13	Release 13	Release 13
Downlink Peak Rate	10 Mbit/s	1 Mbit/s	1 Mbit/s	250 kbit/s	474 kbit/s (EDGE) 2 Mbit/s (EGPRS2B)
Uplink Peak Rate	5 Mbit/s	1 Mbit/s	1 Mbit/s	250 kbit/s (multi-tone) 20 kbit/s (single-tone)	474 kbit/s (EDGE) 2 Mbit/s (EGPRS2B)
Latency	50-100ms	not deployed	10ms-15ms	1.6s-10s	700ms-2s
Number of Antennas	2	1	1	1	1-2
Duplex Mode	Full Duplex	Full or Half Duplex	Full or Half Duplex	Half Duplex	Half Duplex
Device Receive Bandwidth	1.08 - 18 MHz	1.08 - 18 MHz	1.08 MHz	180 kHz	200 kHz
Receiver Chains	2 (MIMO)	1 (SISO)	1 (SISO)	1 (SISO)	1-2
Device Transmit Power	23 dBm	23 dBm	20 / 23 dBm	20 / 23 dBm	23 / 33 dBm

1.5.3 Dash7

Το Dash7 είναι ένα ανοικτό πρότυπο δικτύωσης αισθητήρων από ην Dash Alliance. Εναρμονίζεται με το ISO/IEC 18000-7 και παίζει σε 3 συχνότητες λειτουργίας οι οποίες είναι τα 433 MHz, τα 868 MHz και τα 915 MHz. Το dash7 όπως και το LoRa δημιουργήθηκε αρχικά για στρατιωτικούς σκοπούς αλλά στην πορεία χρησιμοποιήθηκε και για εμπορικές εφαρμογές, [14].

Εικόνα 12 Λογότυπο Dash7, [15].

Κύριο χαρακτηριστικό του, είναι η τεχνολογία δικτύωσης BLAST τα οποία είναι αρχικά των λέξεων, Bursty Light Asynchronous Transitive, [16]. Παρακάτω αναλύονται οι ορολογίες της.

- Bursty

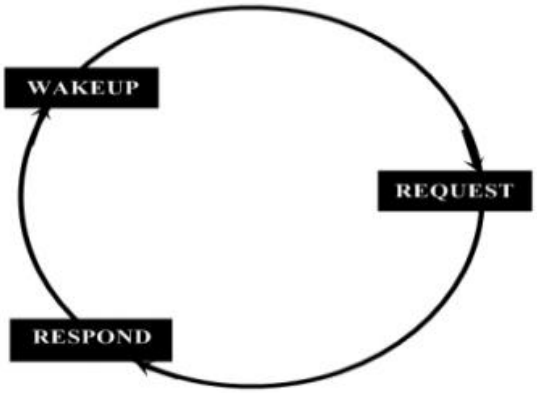
  Σημαίνει ότι η μεταφορά δεδομένων είναι ασυνεχής και δεν περιλαμβάνει περιεχόμενο όπως βίντεο, ήχο ή άλλα της ίδιας κατηγορίας.

- Light

  Σε συμβατικές εφαρμογές, το μέγεθος των πακέτων περιορίζεται στα 256 bytes. Η μετάδοση πολλαπλών ή συνεχών πακέτων μπορεί να επιτευχθεί, αλλά γενικά αποφεύγεται.

- Asynchronous

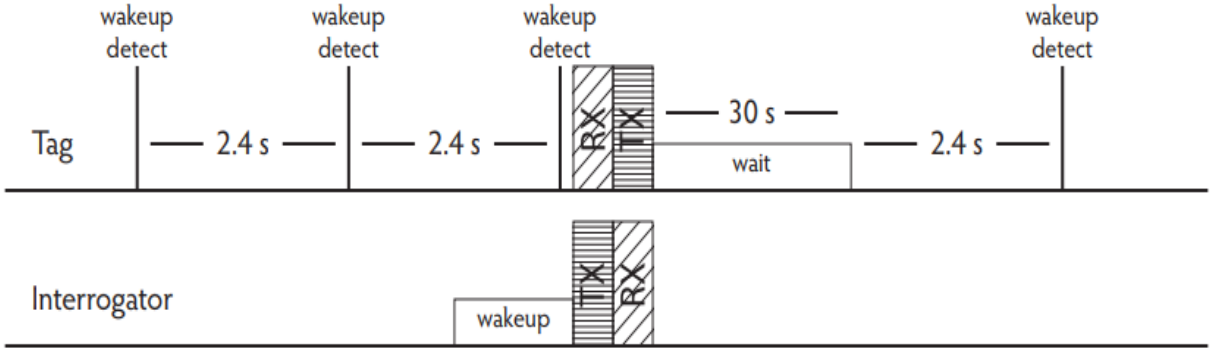
  Σημαίνει ότι η επικοινωνία είναι ασύγχρονη δηλαδή για την επικοινωνία μεταξύ δύο συσκευών δεν χρειάζεται σήμα χρονισμού, αλλά είναι στημένο με την μέθοδο wakeup-command-response.



Εικόνα 13 Διάγραμμα Μεθόδου Wakeup-Command-Response, [17].

Κατά την μέθοδο αυτή ο κόμβος (tag) ο οποίος είναι σε κατάσταση ύπνου επαναφέρεται σε συγκεκριμένα χρονικά διαστήματα. Εάν δεν λάβει κάποιο σήμα μεταπίπτει πάλι σε

κατάσταση ύπνου. Σε περίπτωση που λάβει κάποιο σήμα από τον interrogator υλοποιεί την επικοινωνία και στην συνέχεια πέφτει πάλι σε κατάσταση ύπνου.



Εικόνα 14 Χρονικό διάγραμμα υλοποίησης επικοινωνίας, [16].

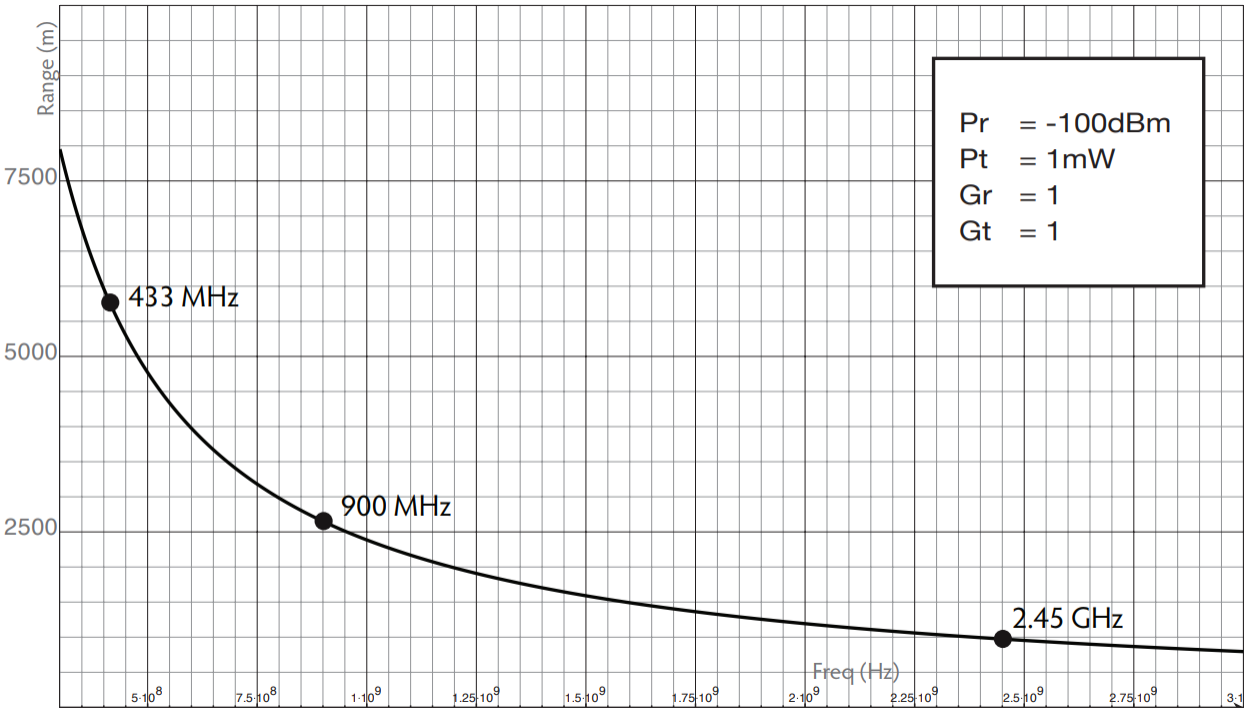
- Transitive

  Ένα δίκτυο συσκευών dash7 είναι κινητό ή μεταβατικό. Αυτό σημαίνει ότι σε σύγκριση με άλλες τεχνολογίες το dash7 είναι μεταφορτο-κεντρικό, έτσι δεν χρειάζεται οι συσκευές να διαχειρίζονται από μία σταθερή υποδομή (πχ σταθμούς βάσης).

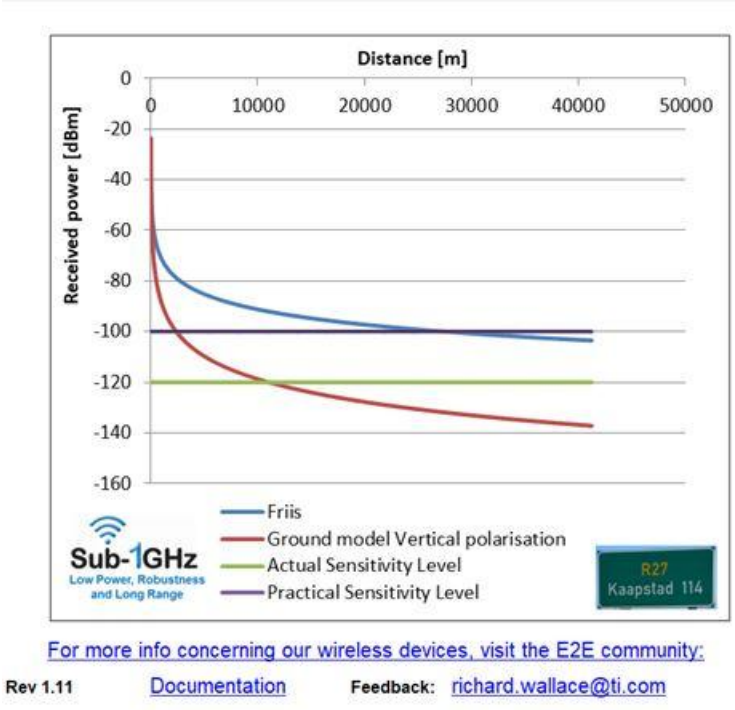
Οι συσκευές dash7 μπορούν να επικοινωνήσουν σε αποστάσεις μέχρι και 1,5 km, πέρα από αυτό όμως έχουν πραγματοποιηθεί δοκιμές από την Savi Technology στις οποίες η εμβέλεια ξεπέρασε τα 10 km. Γενικά για τον υπολογισμό της εμβέλειας ακολουθείται η εξίσωση του Friis [16], η οποία είναι:

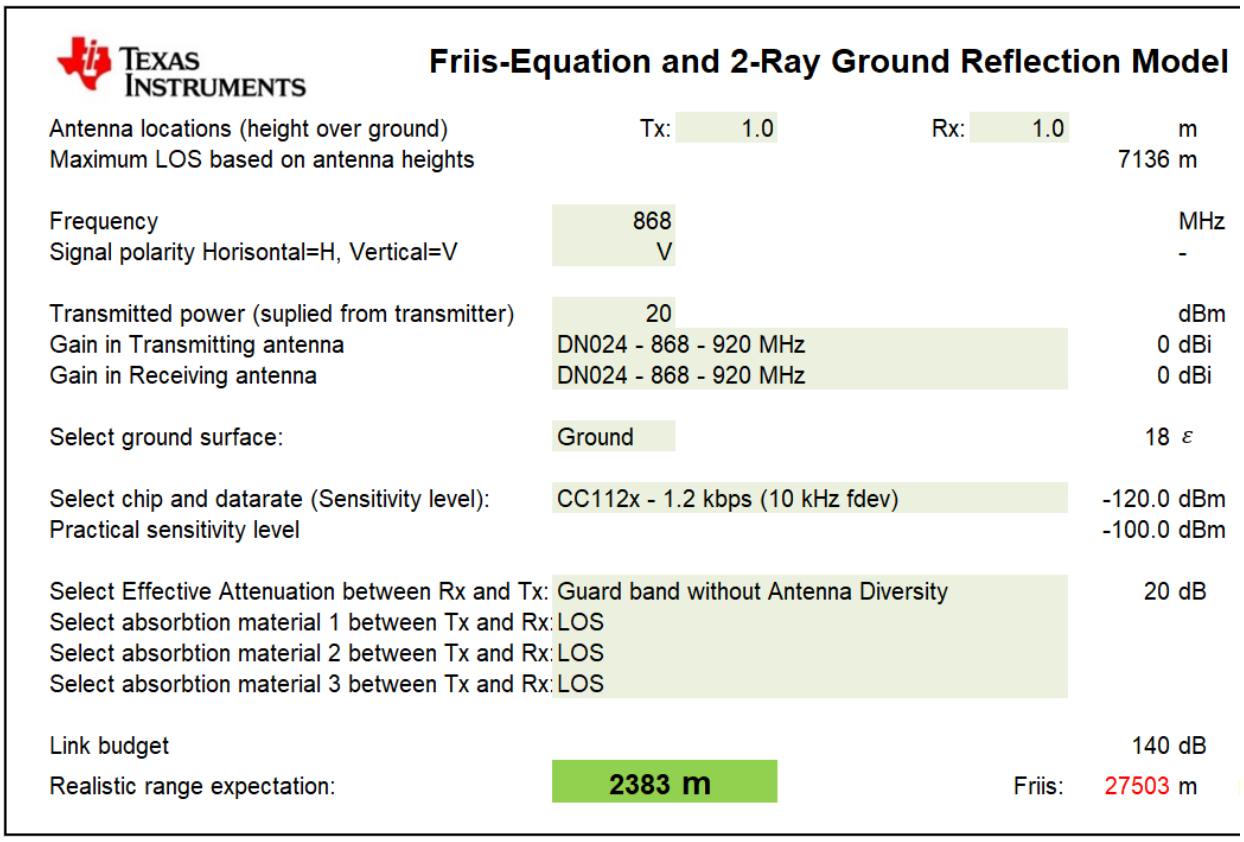
$$Range = \frac{1}{\frac{4\pi}{\lambda}\sqrt{\frac{Pr}{PtGtGr}}}$$

Παρακάτω μπορούμε να δούμε και γραφικά την εμβέλεια συναρτήσει της συχνότητας λειτουργίας, ενώ στην συνέχεια ένα παράδειγμα υπολογισμού της απόστασης με την βοήθεια εργαλείων της Texas Instruments



Εικόνα 15 Διάγραμμα εμβέλειας σε σχέση με την συχνότητα, [16].





Friis-Equation and 2-Ray Ground Reflection Model

Antenna locations (height over ground)	Tx: 1.0	Rx: 1.0		m
Maximum LOS based on antenna heights			7136	m
Frequency	868			MHz
Signal polarity Horisontal=H, Vertical=V	V			-
Transmitted power (suplied from transmitter)	20			dBm
Gain in Transmitting antenna	DN024 - 868 - 920 MHz		0	dBi
Gain in Receiving antenna	DN024 - 868 - 920 MHz		0	dBi
Select ground surface:	Ground		18	ε
Select chip and datarate (Sensitivity level):	CC112x - 1.2 kbps (10 kHz fdev)		-120.0	dBm
Practical sensitivity level			-100.0	dBm
Select Effective Attenuation between Rx and Tx:	Guard band without Antenna Diversity		20	dB
Select absorbtion material 1 between Tx and Rx:	LOS			
Select absorbtion material 2 between Tx and Rx:	LOS			
Select absorbtion material 3 between Tx and Rx:	LOS			
Link budget			140	dB
Realistic range expectation:	2383 m	Friis:	27503	m

Πίνακας 2 Θεωρητικός υπολογισμός εμβέλειας συναρτήσει των χαρακτηριστικών του συστήματος, [18].

Άλλα γνωρίσματα του Dash7 είναι η μεγάλη αυτονομία μπαταρίας, η ασφάλεια δεδομένων, αφού υποστηρίζει κλειδί κρυπτογράφησης AES 128 bit και ρυθμό μετάδοσης δεδομένων 200 Kbps. Αξιοσημείωτο είναι ότι τα σήματα είναι δυνατό να διαπεράσουν σκυρόδεμα και νερό και να ταξιδέψουν μεγάλες αποστάσεις με μικρή κατανάλωση ενέργειας, [19].

Πίνακας 3 Χαρακτηριστικά Dash7, [20].

Global Standard Used	Frequency bands	Channel width	Range	Maximum end node transmit power	Packet size	Data rate (uplink/downlink)	End node roaming allowed

DASH7 Alliance Protocol 1.x	433/868/915 MHz	25 kHz or 200 kHz	5 km	433 MHz: +10dBm 868/915 MHz: +27dBm	max. 256 bytes/packet	9.6 kbit/s, 55.55 kbit/s or 166.667 kbit/s / 9.6 kbit/s, 55.55 kbit/s or 166.667 kbit/s	Yes

1.5.4 Lora

Η LoRA Alliance είναι ένας οργανισμός μη κερδοσκοπικού χαρακτήρα ο οποίος δημιουργήθηκε με σκοπό να προωθήσει και να τυποποιήσει τεχνολογίες LPWAN.

Η LoRA Alliance έχει μέλη σε όλον τον κόσμο, περιλαμβάνοντας μεγάλες εταιρείες στον χώρο της τεχνολογίας και ιδρυτικά μέλη, εταιρείες παγκοσμίως γνωστές, όπως την IBM, την MicroChip, την Cisco, την Semtech, την Bouygues Telecom, την Singtel, την KPN,την Swisscom, την Fastnet και την Belgacom, [21].

Εικόνα 16 Το λογότυπο Lora, [22].

Το LoRa είναι ένας τύπος ασύρματης διαμόρφωσης ο οποίος δημιουργήθηκε για την επίτευξη συνδέσεων μεγάλης εμβέλειας. Το LoRa βασίζεται σε τεχνολογίες chrip εκτεταμένου φάσματος (Chirp Spread Spectrum-CSS), η οποία έχει την ίδια ενεργειακή απόδοση με την διαμόρφωση συχνότητας, αλλά επιτυγχάνει πολύ μεγάλη εμβέλεια. Η τεχνική αυτή χρησιμοποιεί ημιτονοειδές σήμα, του οποίου η συχνότητα αυξάνεται και μειώνεται ανάλογα με τον χρόνο. Αντίστοιχα ο χρόνος αυτός μεταφράζεται σε μετατόπιση συχνότητας στην έξοδο της ανάλυσης FFT. Έτσι το σύστημα γνωρίζοντας την μετατόπιση συχνότητας αλλά και την βασική συχνότητα Chirp, είναι δυνατό να αποδιαμορφώσει τα δεδομένα. Στο παρακάτω γράφημα εμφανίζονται δύο σύμβολα chirp. Το πρώτο σύμβολο είναι από την βασική συχνότητα Chirp, ενώ το δεύτερο μετατοπισμένο κατά ½ της διάρκειας του παλμού Chirp.

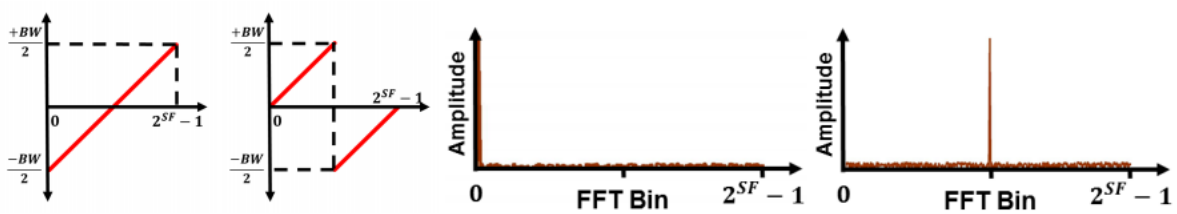


Fig. 4. **CSS modulation & Receiver FFT processing.** The two figures on the left shows two chirp symbols, where the second symbol is a cyclic shifted version of the first, offset by half the chirp duration. BW and SF are the bandwidth and spreading factor respectively. The time delays in the chirp translate to a peak in the FFT domain. The two plots on the right show the FFT output corresponding to the two symbols.

Εικόνα 17 Διαμόρφωση CSS και επεξεργασία FFT, [23].

Ο δέκτης για να αποδιαμορφώσει το σήμα αρχικά το πολλαπλασιάζει με την βασική Chirp κυματομορφή και έπειτα πραγματοποιεί ανάλυση FFT. Στα γραφήματα c και d μπορούμε να δούμε το αποτέλεσμα των 2 παραπάνω συμβόλων. Στο πρώτο διάγραμμα βλέπουμε ένα μέγιστο στην αρχή ενώ στο δεύτερο διάγραμμα στο μέσο του φάσματος. Από τον εντοπισμό αυτών τον μεγίστων είναι δυνατή η αποκωδικοποίηση των δεδομένων.

Όπως και τις υπόλοιπες μεθόδους εκτεταμένου φάσματος, η CSS χρησιμοποιεί ολόκληρό το εύρος ζώνης, για την μετάδοση ενός σήματος, κάνοντας το έτσι ανθεκτικό στον ηλεκτρονικό θόρυβο. Είναι αξιοσημείωτο να αναφέρουμε ότι η CSS διαμόρφωση παρουσιάζει ανθεκτικότητα στο φαινόμενο Doppler, που αποτελεί κλασικό πρόβλημα στις ασύρματες επικοινωνίες, [23].

Οι τεχνολογίες chirp χρησιμοποιούνταν κυρίως για στρατιωτικές εφαρμογές και εφαρμογές διαστημικής επικοινωνίας, λόγω της μεγάλης απόστασης και της ανθεκτικότητας στις ηλεκτρομαγνητικές παρεμβολές, ενώ το LoRA είναι η πρώτη υλοποίηση χαμηλού κόστους για εμπορικούς σκοπούς.

Το LoRAWAN είναι ένα πρωτόκολλο για την διαχείριση της επικοινωνίας μεταξύ πυλών LPWAN και συσκευών-κόμβων (node). Οι συσκευές μεταξύ τους έχουν ασύγχρονη επικοινωνία και μεταδίδουν δεδομένα όταν είναι αυτά διαθέσιμα. Το LoRaWAN ακολουθεί συνήθως τοπολογία αστέρα ή τοπολογία κατανομής, [24].

	Europe	North America	China	Korea	Japan	India
Frequency band	867-869MHz	902-928MHz	470-510MHz	920-925MHz	920-925MHz	865-867MHz
Channels	10	64 + 8 +8	In definition by Technical Committee	In definition by Technical Committee	In definition by Technical Committee	In definition by Technical Committee
Channel BW Up	125/250kHz	125/500kHz				
Channel BW Dn	125kHz	500kHz				
TX Power Up	+14dBm	+20dBm typ (+30dBm allowed)				
TX Power Dn	+14dBm	+27dBm				
SF Up	7-12	7-10				
Data rate	250bps- 50kbps	980bps-21.9kpbs				
Link Budget Up	155dB	154dB				
Link Budget Dn	155dB	157dB				

Εικόνα 18 Προδιαγραφές LoRaWAN ανά περιοχή, [24].

1.6 Πρωτόκολλα επικοινωνίας

1.6.1 Πρωτόκολλο HTTP

Από την στιγμή που μιλάμε για πρωτόκολλα επικοινωνίας είναι αναγκαίο να γίνει αναφορά στο HTTP καθώς είναι το πιο διαδεδομένο πρωτόκολλο που υπάρχει.

Το Πρωτόκολλο Μεταφοράς Υπερκειμένου (HyperText Transfer Protocol, HTTP) αποτελεί το κύριο πρωτόκολλο που χρησιμοποιείται στους φυλλομετρητές του Παγκοσμίου Ιστού για να μεταφέρει δεδομένα ανάμεσα σε έναν διακομιστή (server) και έναν πελάτη (client), [25].

1.6.1.1 Μέθοδοι αίτησης του HTTP

Αν και το HTTP πρωτόκολλο σχεδιάστηκε για χρήση στον Ιστό, υποστηρίζει λειτουργίες που είναι πιο γενικές απ' ότι απαιτείται. Οι λειτουργίες αυτές ονομάζονται μέθοδοι. Κάθε

αίτηση αποτελείται από μία ή περισσότερες γραμμές κειμένου ASCII.Η πρώτη λέξη της πρώτης γραμμής της αίτησης είναι το όνομα της ζητούμενης μεθόδου.

Παρακάτω παρουσιάζονται συνοπτικά οι ενσωματωμένες μέθοδοι αίτησης του πρωτοκόλλου HTTP:

- GET

Η μέθοδος GET ζητά από το διακομιστή να στείλει τη σελίδα. Η σελίδα κωδικοποιείται κατάλληλα σε μορφή Multipurpose Internet Mail Extensions (MIME). Η πιο συνήθης μορφή της μεθόδου GET είναι η εξής:

- GET όνομα_αρχείου HTTP/1.1

όπου το όνομα_αρχείου προσδιορίζει το όνομα του πόρου που πρέπει να προσκομιστεί και το 1.1 είναι η έκδοση του πρωτοκόλλου που χρησιμοποιείται.

- HEAD

Η μέθοδος HEAD ζητά μόνο την κεφαλίδα του μηνύματος, χωρίς τη πραγματική σελίδα. Η μέθοδος αυτή μπορεί να χρησιμοποιηθεί για τη συλλογή πληροφοριών για λόγους δεικτοδότησης ή απλώς και μόνο για τον έλεγχο εγκυρότητας μιας διεύθυνσης URL.

- POST

Η μέθοδος POST χρησιμοποιείται κατά την υποβολή φορμών. Όπως και η μέθοδος GET, η POST περιέχει μια διεύθυνση URL αλλά αντί να ανακτά απλώς τη σελίδα μεταφέρει δεδομένα στον διακομιστή όπως για παράδειγμα τα περιεχόμενα της φόρμας. Έπειτα ο διακομιστής διαχειρίζεται αυτά τα δεδομένα ανάλογα με το URL. Τέλος, η μέθοδος επιστρέφει μια σελίδα που δείχνει το αποτέλεσμα.

- PUT

Η μέθοδος PUT είναι η αντίστροφη της GET, δηλαδή αντί να διαβάζει τη σελίδα, γράφει τη σελίδα. Η μέθοδος αυτή κάνει εφικτή την κατασκευή μιας συλλογής ιστοσελίδων σε έναν απομακρυσμένο διακομιστή. Το σώμα της αίτησης περιέχει τη σελίδα. Μπορεί να

κωδικοποιείται μέσω του MIME, οπότε οι γραμμές που ακολουθούν την PUT μπορεί να περιέχουν κεφαλίδες Content-Type και πιστοποίησης ταυτότητας, ώστε να αποδείξουν ότι ο αιτών έχει πραγματικά την άδεια να εκτελέσει τη ζητούμενη λειτουργία.

- DELETE

Η μέθοδος DELETE καταργεί τη σελίδα ή τουλάχιστον δηλώνει ότι ο διακομιστής Ιστού έχει συμφωνήσει να καταργήσει τη σελίδα. Όπως και με την PUT και σε αυτή τη μέθοδο παίζουν μεγάλο ρόλο η πιστοποίηση της ταυτότητας και της άδειας εκτέλεσης της λειτουργίας.

- TRACE

Η μέθοδος TRACE χρησιμοποιείται για αποσφαλμάτωση. Ζητά από τον διακομιστή να επιστρέψει την αίτηση. Η μέθοδος αυτή είναι χρήσιμη όταν η επεξεργασία των αιτήσεων δεν γίνεται σωστά και ο πελάτης θέλει να δει ποια αίτηση έλαβε πραγματικά ο διακομιστής.

- CONNECT

Η μέθοδος CONNECT επιτρέπει στον χρήστη να πραγματοποιήσει σύνδεση με έναν διακομιστή Ιστού μέσω μιας ενδιάμεσης συσκευής, για παράδειγμα μέσω μια κρυφής μνήμης Ιστού.

- OPTIONS

Η μέθοδος OPTIONS παρέχει έναν τρόπο ώστε ο πελάτης να στέλνει ερωτήματα στον διακομιστή σχετικά με μια σελίδα και να λαμβάνει τις μεθόδους και τις κεφαλίδες που μπορούν να χρησιμοποιηθούν με αυτή τη σελίδα, [25].

1.6.2 Πρωτόκολλο MQTT

Το MQTT είναι ένα πρωτόκολλο ανταλλαγής μηνυμάτων πάνω από το TCP/IP. Τα αρχικά του σημαίνουν Message Queue Telemetry Transport. Βασίζεται στο μοντέλο επικοινωνίας

publish/subscribe, και χαρακτηριστικά του όπως απλός κώδικας, ευχρηστία εφαρμογής κ.α. το κάνουν ιδανικό για Machine to Machine επικοινωνίες και γενικά για IoT εφαρμογές.

1.6.2.1 Ιστορικά στοιχεία.

Το MQTT αναπτύχθηκε από τους Dr. Andy Stanford- Clark της εταιρείας IBM και τον Arlen Nipper της εταιρείας Eurotech (πρώην Arcom), το 1999. Πρωταρχικός τους στόχος τους ήταν να δημιουργήσουν ένα πρωτόκολλο για την παρακολούθηση σωληνώσεων πετρελαίου μέσω δορυφορικής σύνδεσης το οποίο θα ήταν χαμηλής κατανάλωσης ενέργειας καθώς και εύρους ζώνης. Οι αρχικές προδιαγραφές του ήταν οι εξής:

- Απλότητα εφαρμογής
- Quality of Service
- Χαμηλή κατανάλωση ενέργειας
- Χαμηλό εύρος ζώνης
- Δυνατότητα συνεχούς σύνδεσης.

Αρχικά ήταν ιδιωτικής χρήσης αλλά από την έκδοση 3.1 το 2010. Η εταιρεία IBM το διέθεσε δωρεάν έτσι ώστε ο καθένας να μπορεί να το χρησιμοποιείσαι σε εφαρμογές του.

Αξιοσημείωτο παράδειγμα υλοποίησης με το πρωτόκολλο MQTT είναι το Facebook Messenger, [26].

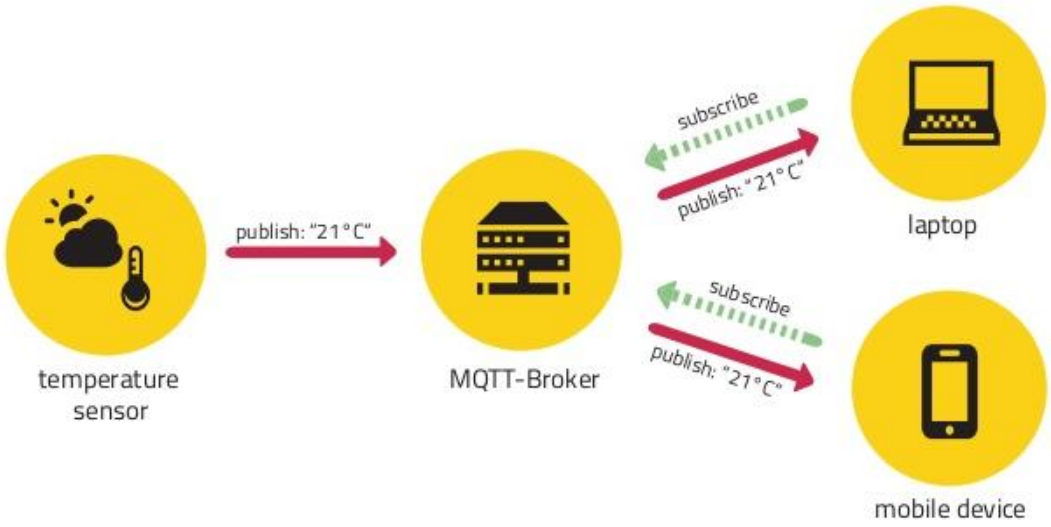
1.6.2.2 Χαρακτηριστικά πρωτόκολλου MQTT

Τα κύρια χαρακτηριστικά του πρωτόκολλου MQTT είναι τα εξής:

- Μοντέλο publish/subscribe

  Στο Μοντέλο publish/subscribe σε αντίθεση με το κοινό client-server οι συσκευές δεν επικοινωνούν απευθείας. Εδώ ο πελάτης δεν απαιτείται να γνωρίζει την ύπαρξη του παραλήπτη. Υπάρχει ενδιάμεσα ένας broker ο οποίος θα μπορούσε να θεωρηθεί ως «μεσάζοντας» ανάμεσα στον πελάτη και τον παραλήπτη. Η δουλειά του broker είναι να παραλαμβάνει όλα τα μηνύματα και να τα προωθεί κατάλληλα ώστε συγκεκριμένα

μηνύματα, να φτάνουν στους συνδρομητές οι οποίοι αιτούνται γι' αυτά(ενδιαφέρονται για το αιτούμενο θέμα- topic). Το πλεονέκτημα σε σχέση με τις υλοποιήσεις HTTP Post είναι ότι δεν απαιτείται ο Client να κάνει request για δεδομένα, αλλά ο server μπορεί να στέλνει δεδομένα όποτε απαιτείται μέσω μίας συνεχούς ανοικτής σύνδεσης TCP ελαχιστοποιώντας τον όγκο δεδομένων με την μεταφορά μόνο του payload.



Εικόνα 19 Δομή συστήματος publish/subscribe, [27].

Ουσιαστικά με αυτό το μοντέλο πετυχαίνουμε 3 χαρακτηριστικά:

1. ***Χωρική αποσύνδεση.*** Ο publisher και ο subscriber δεν γνωρίζουν ο ένας την ύπαρξη του άλλου.
2. ***Χρονική αποσύνδεση.*** Ο publisher και ο subscriber δεν χρειάζεται να τρέχουν την ίδια χρονική στιγμή.
3. ***Αποσύνδεση συγχρονισμού.*** Όλα τα μέρη (publisher, subscriber, broker) δεν διακόπτουν την λειτουργία τους κατά την αποστολή ή παραλαβή μηνυμάτων. [28]

- ***Quality of Service***

Το QoS θα μπορούσε να θεωρηθεί ως μία συμφωνία μεταξύ αποστολέα και παραλήπτη για το πόσο εγγυημένη είναι η παράδοση ενός μηνύματος.

Το MQTT ορίζει τρία επίπεδα quality of service για την παράδοση των μηνυμάτων, τα οποία αναλύονται ως εξής:

1. Επίπεδο ποιότητας 0

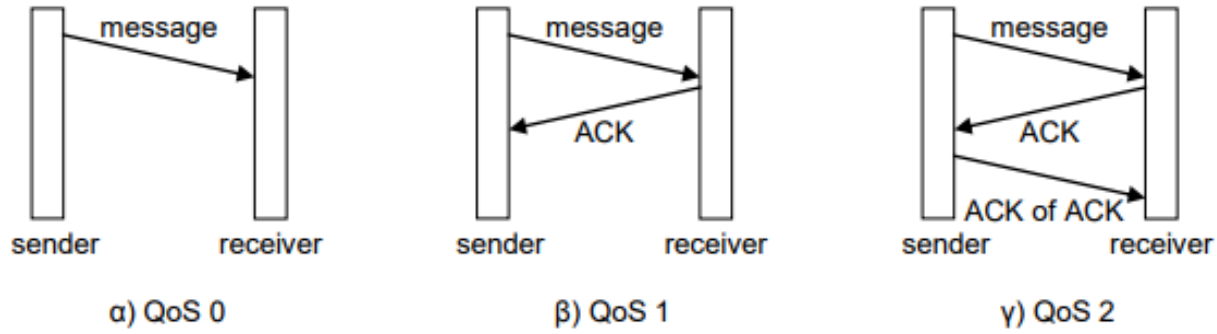
   Παράδοση το πολύ μία φορά. Στην περίπτωση αυτή τα δεδομένα αποστέλλονται μία φορά και δεν λαμβάνεται καμία επιβεβαίωση παράδοσης. Τα μηνύματα παραδίδονται σύμφωνα με την βέλτιστη προσπάθεια του πρωτόκολλου TCP. Αυτό το επίπεδο χρησιμοποιείτε συνήθως όταν δεν υπάρχουν απαιτήσεις για αξιοπιστία αποστολής δεδομένων μεταξύ αποστολέα και παραλήπτη.

2. Επίπεδο ποιότητας 1

   Παράδοση το πολύ μία φορά. Στην περίπτωση αυτή, ο αποστολέας αποθηκεύει το μήνυμα μετά την αποστολή έως ότου πάρει publish acknowledge (PUBACK message). Σε περίπτωση αποτυχίας σύνδεσης ή το μήνυμα δεν παραληφθεί μέσα σε συγκεκριμένο χρονικό διάστημα, ο αποστολέας ξαναστέλνει το μήνυμα.

3. Επίπεδο ποιότητας 2

   Παράδοση ακριβώς μία φορά. Στην περίπτωση αυτή εξασφαλίζεται ότι το μήνυμα θα παραδοθεί στον παραλήπτη. Είναι ο πιο ασφαλής τρόπος αλλά και ο πιο αργός γιατί αυξάνεται η κίνηση του δικτύου ανά μήνυμα. Σε αυτό το επίπεδο γίνονται 2 ανταλλαγές μηνυμάτων. Αν publisher στείλει δεύτερή φορά ένα μήνυμα, ο broker θα προωθήσει μόνο ένα. Αυτό το επίπεδο ποιότητας χρησιμοποιείται όταν η πολλαπλή λήψη ίδιων μηνυμάτων δεν είναι αποδεκτή, [29].



Εικόνα 20 Επίπεδα ποιότητας υπηρεσιών του πρωτοκόλλου MQTT, [29].

- Μικρή επιβάρυνση στο δίκτυο

Η επιβάρυνση στο δίκτυο είναι ελάχιστη καθώς το μέγεθος της επικεφαλίδας του μηνύματος είναι 2 bytes.

- Παρακρατημένα μηνύματα (Retained Messages)

Ο διακοσμητής διατηρεί το μήνυμα, ακόμα και μετά την αποστολή σε όλους τους συνδρομητές. Οπότε εάν ένας συνδρομητής αιτηθεί για το ίδιο topic τα μηνύματα που είχαν παρακρατηθεί θα αποσταλούν και σε αυτόν.

- Τελευταία ευχή και διαθήκη (Last Will & Testament)

Μηχανισμός ενημέρωσης των συνδρομητών για την περίπτωση απροσδόκητης αποσύνδεσης. Με αυτή την υπηρεσία ο client πληροφορεί τον server για το μήνυμα που θα πρέπει να δημοσιευτεί σε περίπτωση διακοπής. Χρήσιμο χαρακτηριστικό για ποικιλία εφαρμογών όπως αποτυχία αισθητήρων κ.τ.λ.

1.6.3 Πρωτόκολλο CoAP

Το CoAP (Constrained Application Protocol) είναι ένα πρωτόκολλο επιπέδου εφαρμογής που προορίζεται για χρήση σε διαδικτυακές συσκευές περιορισμένων πόρων, όπως οι κόμβοι WSN.

Έχει σχεδιαστεί ώστε να μεταφράζεται εύκολα σε HTTP για απλοποιημένη ενοποίηση με το διαδίκτυο και για την κάλυψη εξειδικευμένων απαιτήσεων, όπως είναι η υποστήριξη πολλαπλής διανομής (multicast support), πολύ χαμηλή επιβάρυνση (overhead) και απλότητα. Η πολλαπλή εκπομπή (multicast support), το χαμηλό overhead και η απλότητα είναι πολύ σημαντικά χαρακτηριστικά για το Διαδίκτυο των Πραγμάτων (IoT) και για M2M (machine-to-machine) συσκευές, οι οποίες τείνουν να είναι ενσωματωμένες και να έχουν έτσι πολύ λιγότερη μνήμη καθώς και χαμηλή κατανάλωση από αυτές που έχουν οι παραδοσιακές συσκευές διαδικτύου. Ως εκ τούτου, η αποτελεσματικότητα είναι πολύ σημαντική. Το CoAP μπορεί να λειτουργήσει στις πιο πολλές συσκευές που υποστηρίζουν το UDP (User Datagram Protocol) πρωτόκολλο, [30].

Η ομάδα εργασίας CoRE σχεδίασε το CoAP με βάση τα ακόλουθα χαρακτηριστικά, δίνοντας έμφαση στην ανάλυση της πολυπλοκότητας του πρωτοκόλλου:

- URI (Unirform Resource Identifier) και υποστήριξη περιεχομένου.
- Υποστήριξη για την ανακάλυψη πόρων που παρέχονται από γνωστές CoAP υπηρεσίες.
- Απλή αποθήκευση προσωρινής μνήμης (caching) βασισμένη στην μέγιστη ηλικία (max aged based).
- Απλή εγγραφή (subscription) για έναν πόρο, και ως αποτέλεσμα την προώθηση ειδοποιήσεων (push notifications).

1.6.4. Σύγκριση μεταξύ πρωτοκόλλων

1.6.4.1 CoAP vs MQTT

Η αλήθεια είναι ότι η επιλογή μεταξύ αυτών των δύο πρωτόκολλων είναι δύσκολη. Από την μία πλευρά έχουμε το MQTT το οποίο είναι ιδανικό για επικοινωνία συσκευών μέσα σε ένα Wide Area Network λόγω του μοντέλου publish/subscribe. Επίσης το MQTT είναι

μέρος του Azure και του Amazon Services με αποτέλεσμα να είναι εύκολο να υιοθετηθεί σε υπάρχουσες εφαρμογές από τους developers, [31].

Από την άλλη πλευρά έχουμε το CoAP με μεγάλο πλεονέκτημα την υποστήριξη με το HTTP. Στην περίπτωση που υπάρχει προεγκατεστημένο σύστημα που είναι web service-based, το CoAP είναι μια καλή επιλογή. Επίσης το CoAP χρησιμοποιεί συνδέσεις UDP. Αυτό έχει ως αποτέλεσμα να είναι γρήγορη και με μικρότερο bandwidth η αποστολή δεδομένων προς πολλαπλούς clients, αλλά συγχρόνως η σχεδίαση του είναι πιο δύσκολη για την υλοποίηση της σύνδεσης, σε σύγκριση με το MQTT, το οποίο είναι TCP.

Γενικά για εφαρμογές οι οποίες θα κάνουν συνδέσεις σε υλοποιήσεις τύπου web server τότε το CoAP είναι καλή επιλογή αλλά για περιπτώσεις που ο κόμβος χρησιμοποιείτε κυρίως για αποστολή δεδομένων σε κάποια cloud εφαρμογή, το MQTT φαίνεται να αποτελεί την ιδανική επιλογή, [31].

Πέρα από αυτό και τα δύο πρωτόκολλα παρέχουν επιλογές κρυπτογράφησης και QoS καθώς και πολλά άλλα χαρακτηριστικά.

	CoAP	MQTT
Communications Model	Request-Response, or Pub-Sub	Pub-Sub
RESTful	Yes	No
Transport Layer Protocol	Preferably UDP; TCP can be used	Preferably TCP; UDP can be used (MQTT-S)
Header	4 Bytes	2 Bytes
Number of message types	4	16
Messaging	Asynchronous and Synchronous	Asynchronous
Application Reliability	2 Levels	3 Levels
Security	IPSEC or DTLS	Not defined in standard
Intermediaries	Yes	Yes (MQTT-S)

Πίνακας 4 Συγκριτικό CoAP με MQTT, [32].

1.6.4.2 HTTP vs. MQTT

1.6.4.2.1 Σχεδιασμός και μετάδοση δεδομένων

Όπως προαναφέραμε το HTTP είναι το πιο διαδεδομένο πρωτόκολλο που υπάρχει, αυτό όμως δεν το κάνει και το πιο ιδανικό. Το MQTT κερδίζει συνεχώς έδαφος και δεν μπορεί να κριθεί ως ασήμαντος αντίπαλος. Παρακάτω θα αναλύσουμε τις διαφορές που έχουν τα δύο πρωτόκολλα και γιατί θεωρείτε ιδανικό για εφαρμογές IoT.

Αρχικά το HTTP ακολουθεί το μοντέλο Post/Request σε αντίθεση με το MQTT που ακολουθεί το μοντέλο Publish/Subscribe. Αυτό δίνει ένα σημαντικό πλεονέκτημα υπέρ του MQTT γιατί δεν απαιτείται ο Client να κάνει request για δεδομένα, αλλά ο server μπορεί να στέλνει δεδομένα όποτε απαιτείται, μέσω μίας συνεχούς ανοικτής σύνδεσης TCP.

Ένα επιπλέον προτέρημα είναι ότι το MQTT είναι δεδομένο-κεντρικό σε αντίθεση με το HTTP το οποίο είναι έγγραφο-κεντρικό. Αυτό έχει ως αποτέλεσμα το MQTT να είναι πιο ελαφρύ. Επίσης για τον ίδιο λόγο, είναι δυνατό να γίνει διαχείριση της ενέργειας, με αποτέλεσμα να το κάνει ιδανικό για εφαρμογές χαμηλής κατανάλωσης ενέργειας όπως σε δίκτυα αισθητήρων και ενεργοποιητών.

1.6.4.2.2 Αποστολή δεδομένων. Ταχύτητα και Αξιοπιστία

Το MQTT είναι πιο γρήγορο από το HTTP για διάφορους λόγους. Αρχικά όπως προαναφέραμε παραπάνω, το HTTP ακολουθεί το μοντέλο Post/Request σε αντίθεση με το MQTTP που ακολουθεί το μοντέλο Publish/Subscribe, οπότε δεν είναι απαραίτητο να υλοποιείτε σύνδεση κάθε φορά που πρέπει να γίνει αποστολή δεδομένων. Επίσης η επικεφαλίδα του MQTT είναι πολύ μικρή και το μέγεθος του πακέτου δεδομένων μπορεί να είναι και 2 μόλις bytes. Στο HTTP αυτό είναι παραμετροποιήσιμο. Αυτό μπορεί αρχικά να θεωρείται πλεονέκτημα καθώς καθιστά εύκολη την ανάγνωση από ανθρώπους, ταυτόχρονα όμως είναι περιττό, και για περιπτώσεις εφαρμογών με περιορισμένο εύρος ζώνης μπορεί να θεωρηθεί σαν μειονέκτημα.

Όσον αφορά την αξιοπιστία το MQTT προσφέρει ποικιλία επιλογών, όπως QoS, Last will & Testament και Retained messages. Το HTTP δεν προσφέρει τέτοιου είδους επιλογές, [33].

Πίνακας 5 Συγκριτικό MQTT με HTTP, [34].

Features	MQTT	HTTP
Full Form	Message Queue Telemetry Transport	Hyper Text Transfer Protocol
Design Methodology	The protocol is data centric.	The protocol is document centric.
Architecture	It has publish/subscribe architecture. Here devices can publish any topics and can also subscribe for any topics for any updates.	It has request/response architecture.
Complexity	simple	more complex
Data security	YES	NO, hence HTTPS is used to provide data security.
Upper layer protocol	It runs over TCP.	It runs over UDP.
message size	small, it is binary with 2Byte header.	Large, it is in ASCII format.
Service levels	3	1
Libraries	30KB C, 100KB Java	Large
Port number	1883	80 or 8080
Data distribution	1 to 0/1/N	one to one only

1.6.4.2.3 Συμπεράσματα

Το MQTT είναι εύκολο στην χρήση. Είναι ιδανικό ως προς τον χρόνο απόκρισης, την κατανάλωση ενέργειας και το εύρος ζώνης. Επίσης παρέχει επιλογές για περιπτώσεις διακοπτόμενης επικοινωνίας, κάτι που το HTTP δεν το κάνει.

Το HTTP έχει μεγάλη επεκτασιμότητα. Αλλά για εφαρμογές IoT, οι οποίες στο σύνολό τους δεν έχουν μεγάλες απαιτήσεις σε εύρος ζώνης, αλλά υπάρχει απαίτηση για την αξιοπιστία της μετάδοσης, το MQTT φαίνεται να είναι καταλληλότερο .

Κεφάλαιο 2

Πειραματική Υλοποίηση

2.1 Θεωρητικό υπόβαθρο συστήματος

2.1.1 Γενικά στοιχεία φωτισμού

Οι τεχνολογίες φωτισμού χωρίζονται στις παρακάτω κατηγορίες:

1. **Φωτισμός εργασιών (task lighting)**

Ο φωτισμός εργασιών αναφέρετε στις εφαρμογές αύξησης φωτεινότητας για την επίτευξη μίας συγκεκριμένης ενέργειας. Κλασικό παράδειγμα μίας τέτοιας εφαρμογής είναι η ρύθμιση του φωτεινότητας σε οθόνες.

2. **Φωτισμός έμφασης (Accent lighting)**

Ο φωτισμός έμφασης αναφέρετε στις εφαρμογές τις οποίες το φως εστιάζεται σε μία συγκεκριμένη περιοχή ή αντικείμενο. Κλασικά παραδείγματα τέτοιων εφαρμογών είναι η έμφαση αντικειμένων όπως έργων τέχνης αλλά και ο φωτισμός δρόμων.

3. **Γενικός φωτισμός (General lighting)**

Ο γενικός φωτισμός αφορά όλες τις υπόλοιπες εφαρμογές οι οποίες δεν μπορούν να αποδοθούν σε μία από τις παραπάνω κατηγορίες, [35].

Όσον αφορά την ενέργεια, σύμφωνα με μελέτες το 19% της παγκόσμιας κατανάλωσης ενέργειας χρησιμοποιείται σε εφαρμογές φωτισμού. Αντίστοιχα το 6% των συνολικών εκπομπών άνθρακα, [36].

Λόγω της μεγάλης ζήτησης ενέργειας υπάρχει μεγάλη απαίτηση δημιουργίας τεχνολογιών οι οποίες μειώνουν την ζήτηση αυτή. Οι εφαρμογές έξυπνου φωτισμού είναι ένας αποδοτικός τρόπος μείωσης της απαιτούμενης ενέργειας επιτρέποντας στον χρήστη την απομακρυσμένη διαχείριση της κατάστασης λειτουργίας των συστημάτων φωτισμού αλλά και της φωτεινότητας. Η δυνατότητες αυτές όχι μόνο μειώνουν την κατανάλωση ενέργειας

αλλά παρέχουν και ένα επίπεδο άνεσης και ευκολίας προς τον χρήστη καθώς αυτός έχει την δυνατότητα επιλογής του επιπέδου φωτεινότητας.

Οι εφαρμογές έξυπνου φωτισμού συνήθως περιλαμβάνουν κάποια σενάρια διαχείρισης φωτεινότητας και της κατάστασής του. Αυτό γίνεται συνήθως είτε στην απλή του μορφή κάνοντας χρήση χρονικών μοντέλων είτε κάνοντας χρήση αισθητήρων, [35].

Σύμφωνα με μελέτες, η χρήση συστημάτων που συμπεριλαμβάνουν αισθητήρια και αυτόματη ρύθμιση φωτεινότητας, το ποσοστό εξοικονόμησης ενέργειας αγγίζει το 32% σε σύγκριση με τα συμβατικά συστήματα φωτισμού, ακόμα και αν αυτό έχει δυνατότητα απόδοσης 50% περισσότερο από τα συμβατικά συστήματα, [37].

Ένα ολοκληρωμένο σύστημα φωτισμού αποτελείται από ένα σύνολο αισθητήρων, συνήθως αισθητήρα φωτεινότητας και κίνησης, μία μονάδα ελέγχου, η οποία είναι συνήθως κάποιος μικροελεγκτής χαμηλής κατανάλωσης ενέργειας, και ένα σύστημα ενεργοποιητών (actuators) για την διαχείριση του λαμπτήρα. Αυτό μπορεί να είναι από ένα απλό διακοπτικό έως ένα ολοκληρωμένο σύστημα διαχείρισης ενέργειας. Τέλος αποτελείται από έναν λαμπτήρα, ο οποίος βασίζεται συνήθως σε κάποιο LED Engine, [35].

2.1.2 Τι σημαίνει ο όρος “Smart Lighting”

Οι τεχνολογίες “Smart Lighting” έχουν σχεδιαστεί κυρίως για την βελτίωση της απόδοσης ισχύος ενός συστήματος φωτισμού. Αυτές μπορεί να περιλαμβάνουν αυτοματοποιημένα συστήματα τα οποία πραγματοποιούν μετατροπές βάσει συνθηκών λειτουργίας, όπως εξωτερικός φωτισμός, κίνηση στον χώρο και πολλά άλλα, [35].

2.1.3 Γιατί Smart Lighting; Πλεονεκτήματα

Οι υπάρχουσες τεχνολογίες χειροκίνητου φωτισμού αποτελούν μία μεγάλη δαπάνη για τον ετήσιο προϋπολογισμό μίας χώρας. Ένα χειροκίνητο σύστημα ανοιγοκλείνει τους λαμπτήρες βάσει συγκεκριμένου χρονικού διαγράμματος, συνήθως ανάβουν το απόγευμα και σβήνουν το πρωί. Αυτό έχει ως αποτέλεσμα την άσκοπη κατανάλωση ενέργειας σε

σημεία τα οποία δεν έχουν ανάγκη για φωτισμό είτε ολοσχερώς είτε ως ένα επίπεδο. Τα κύρια μειονεκτήματα ενός τέτοιου συστήματος είναι τα εξής:

- Χειροκίνητη εναλλαγή κατάστασης ON/OFF λαμπτήρων,
- Μεγαλύτερη κατανάλωση και λόγω τεχνολογίας λαμπτήρων αλλά και του τρόπου διαχείρισης τους,
- Αδυναμία μεταβολής φωτεινότητας,
- Μεγαλύτερες απαιτήσεις σε εργατικό δυναμικό, [38].

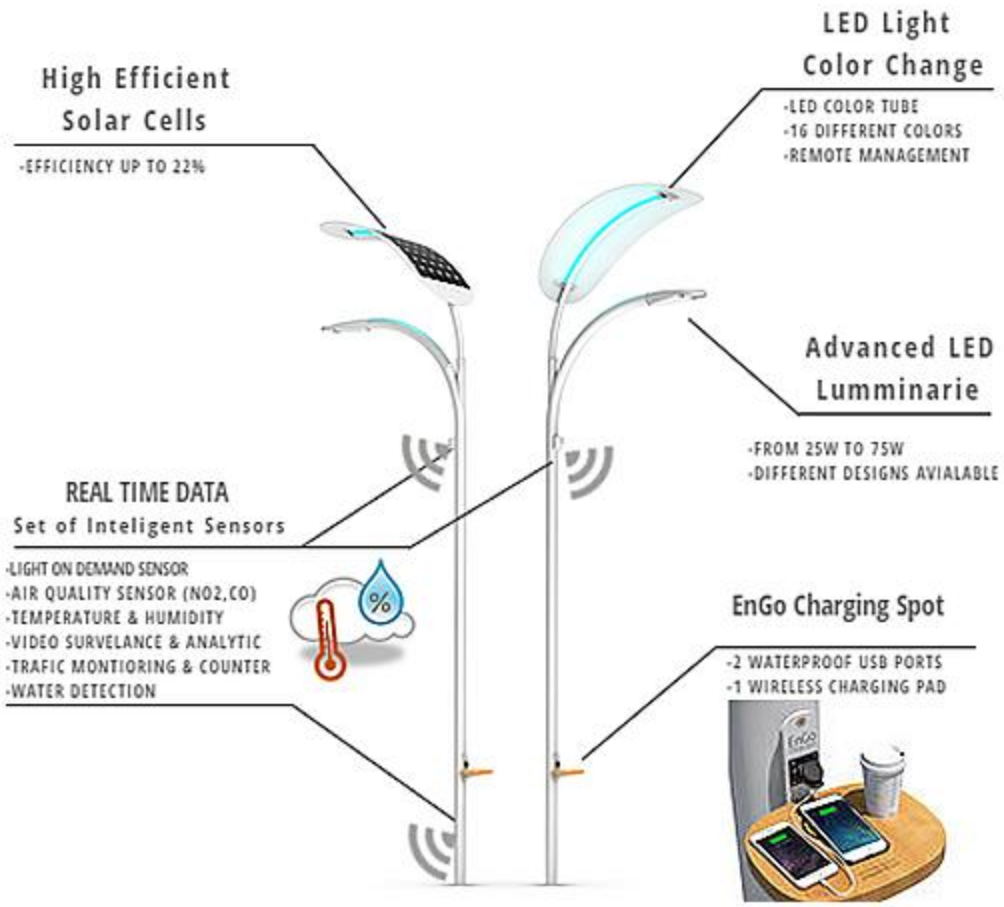
Η χρήση των τεχνολογιών έξυπνου φωτισμού βελτιώνουν κατά πολύ τον τρόπο της σύγχρονης ζωής. Με τις τεχνολογίες έξυπνου φωτισμού είναι δυνατόν να εξοικονομηθεί ενέργεια, μειώνοντας έτσι όχι μόνο το κόστος ενέργειας αλλά και το κόστος συντήρησης. Τα ποσοστά αυτά αυξάνονται σε περίπτωση χρήσης συστημάτων έξυπνου φωτισμού. Λόγω των χαμηλότερων απαιτήσεων ενέργειας, αυτό σημαίνει αυτόματα και μείωση των εκπομπών άνθρακα καθώς μειώνετε και η παραγωγή ενέργειας. Επίσης η αξιοποίηση έξυπνων συστημάτων ελέγχου μπορεί να μειώσει μέχρι και τα ποσοστά εγκληματικότητας σε μία περιοχή κάνοντας έτσι τους δρόμους πιο ασφαλείς. Επίσης το ίδιο σύστημα είναι δυνατόν να μπορεί να αυξήσει γρήγορα την αποτελεσματικότητα του φωτισμού και τη διαχείριση της κυκλοφορίας. Τέλος συνοψίζοντας τα παραπάνω με την χρήση τεχνολογιών έξυπνου φωτισμού έχουμε τα παρακάτω πλεονεκτήματα:

- Σημαντική μείωση κατανάλωσης ενέργειας
- Μείωση στο κόστος συντήρησης
- Μειωμένες εκπομπές άνθρακα
- Απαίτηση μια μικρότερο εργατικό δυναμικό
- Αυτόματη διαχείριση λαμπτήρων βάσει συγκεκριμένου προγράμματος-σεναρίων περιοχής
- Δυνατότητα αλλαγής σεναρίων ανάλογα το προφίλ της περιοχής, [38].

2.1.4 Υλοποιήσεις έξυπνου φωτισμού

Πόλεις και εταιρείες παγκοσμίως έχουν αρχίσει να αναγνωρίζουν τα πλεονεκτήματα της υλοποίησης συστημάτων έξυπνου φωτισμού. Αυτές οι καινοτόμες λύσεις φωτισμού επιτρέπουν στις επιχειρήσεις και τις κυβερνήσεις των χώρων να μειώσουν το ενεργειακό κόστος καθώς και να τις εφαρμόσουν ένα ευρύ φάσμα νέων λύσεων. Δεδομένα όπως φωτεινότητας, κίνησης, περιβαλλοντολογική μόλυνση, αλλά και επιπλέον χαρακτηριστικά όπως απομακρυσμένη διαχείριση και σταθμοί φόρτισης, είναι διαθέσιμα μέσω των υλοποιήσεων αυτών.

Λόγω της περιβαλλοντολογικής μόλυνσης η Ευρωπαϊκή Ένωση έχει θέσει στόχους μείωσης των εκπεμπόμενων ρύπων και της μείωσης κατανάλωσης ενέργειας μέχρι το 2020. Βάσει αυτού όλα τα κράτη-μέλη της Ε.Ε. θα πρέπει να υλοποιήσουν έργα και μεθόδους για την επίτευξη των παραπάνω. Ένα αξιόλογο παράδειγμα είναι αυτό της Ολλανδίας και συγκεκριμένα το Amsterdam Smart City Program (ASC), το οποίο ξεκίνησε το 2009. Το πρόγραμμα διαιρείται σε πολλούς τομείς ανάλογα τον τομέα διαχείρισης, όπως διαχείριση αποβλήτων, το geo locating και φυσικά το street lighting. Σχετικά με τον έξυπνο φωτισμό, για το ASC έχουν υλοποιηθεί έργα με εξοπλισμό της ENGOPlanet, [39].

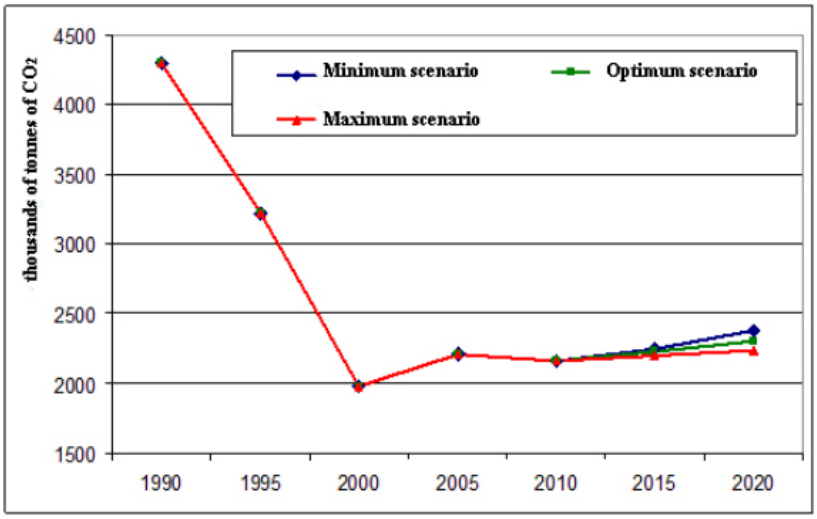


Εικόνα 21 Σύστημα έξυπνου φωτισμού της ENGOPlanet, [40].

2.1.5 Έξυπνη πόλη και περιβάλλον

Η χρήση έξυπνων συστημάτων στις πόλεις δεν ευνοεί μόνο τους χρήστες αλλά και το ίδιο το περιβάλλον. Κάνοντας χρήση συστημάτων διαχείρισης ενέργειας, προστατεύουμε το περιβάλλον καθώς επιτυγχάνονται μειωμένες εκπομπές άνθρακα από την μείωση της απαιτούμενης ενέργειας σε κομβικές υπηρεσίες για μία πόλη, την διαχείριση του φωτισμού, την διανομή του νερού κ.α.

Σύμφωνα με μελέτες για το projectRPA "Rīgas gaisma" το οποίο υλοποιήθηκε στην πρωτεύουσα της Λετονίας, την Ρίγα, η υλοποίηση μόνο των συστημάτων έξυπνου φωτισμού επέφερε μείωση της κατανάλωσης ενέργειας κατά 400 MWh και μείωση των εκπομπών άνθρακα κατά 150 τόνους, σε ετήσια βάση, [41].



Εικόνα 22 Διάγραμμα μείωσης των εκπεμπόμενων ρύπων με΄τα την υλοποίση έργων έξυπνης πόλης μεταξυ 1990-2020, [42].

2.2 Προδιαγραφές πειραματικής διάταξης

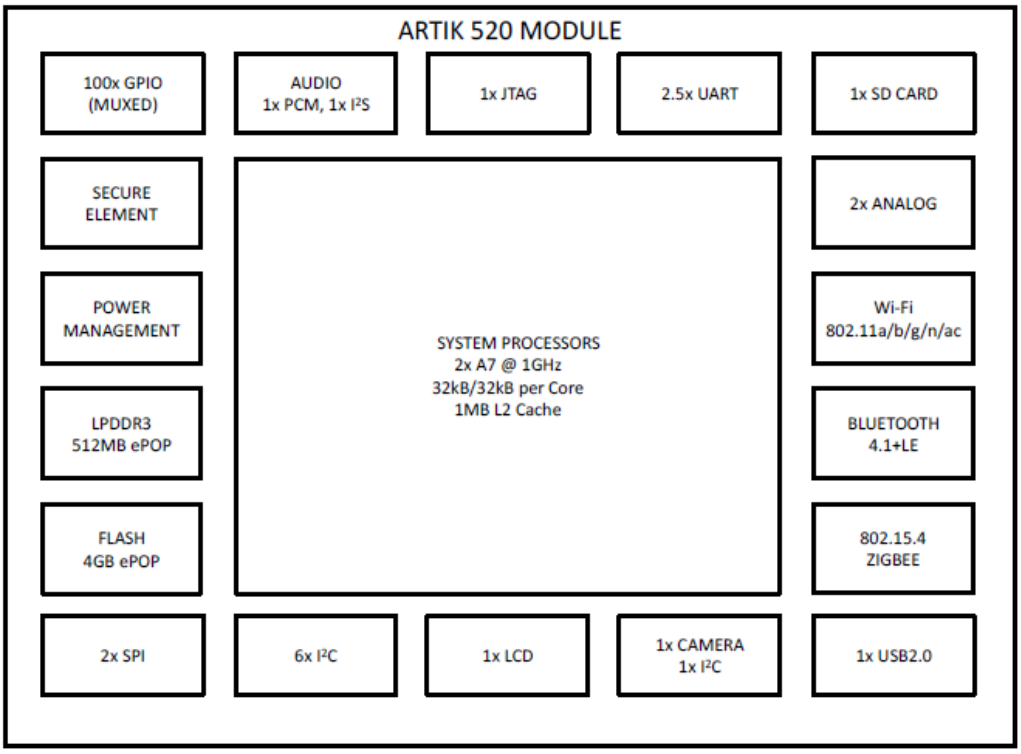
Για την κατασκευή συστήματος με δίκτυο αισθητήρων είναι απαραίτητη η υλοποίηση 2 διατάξεων. Μία διάταξη η οποία θα διαβάζει και διαχειρίζεται τα led engines και στην συνέχεια στέλνει τα απαραίτητα δεδομένα προς τον διαχειριστή και μία διάταξη η οποία θα αναλάβει τον ρόλο του συγκεντρωτή δεδομένων. Σε αυτή την διάταξη όλοι οι κόμβοι θα αποστέλλουν τις κατάλληλες πληροφορίες προς το Gateway, και στην συνέχεια όλα θα αποστέλλονται προς το διαδίκτυο σε πλατφόρμα από την οποία θα μπορούμε να απεικονίσουμε αλλά και να διαχειριστούμε την χρήση των led engines. Για την υλοποίηση και των δύο διατάξεων είναι απαραίτητη η χρήση πλακετών ανάπτυξης. Για την περίπτωση του Gateway, θα πρέπει να υποστηρίζει ποικιλία ασύρματων τεχνολογιών. Σε περίπτωση που κάποια απαιτούμενη τεχνολογία δεν υποστηρίζεται, θα πρέπει να είναι δυνατό να ενσωματωθεί στο σύστημα με χρήση άλλων πρωτόκολλων επικοινωνίας.

Οι αισθητήρες επιλέχθηκαν έτσι ώστε να είναι όλοι ψηφιακοί με κοινό πρωτόκολλο, κατά προτίμηση σε παράλληλη συνδεσμολογία, με σκοπό να μειωθούν οι απαραίτητες ψηφιακές

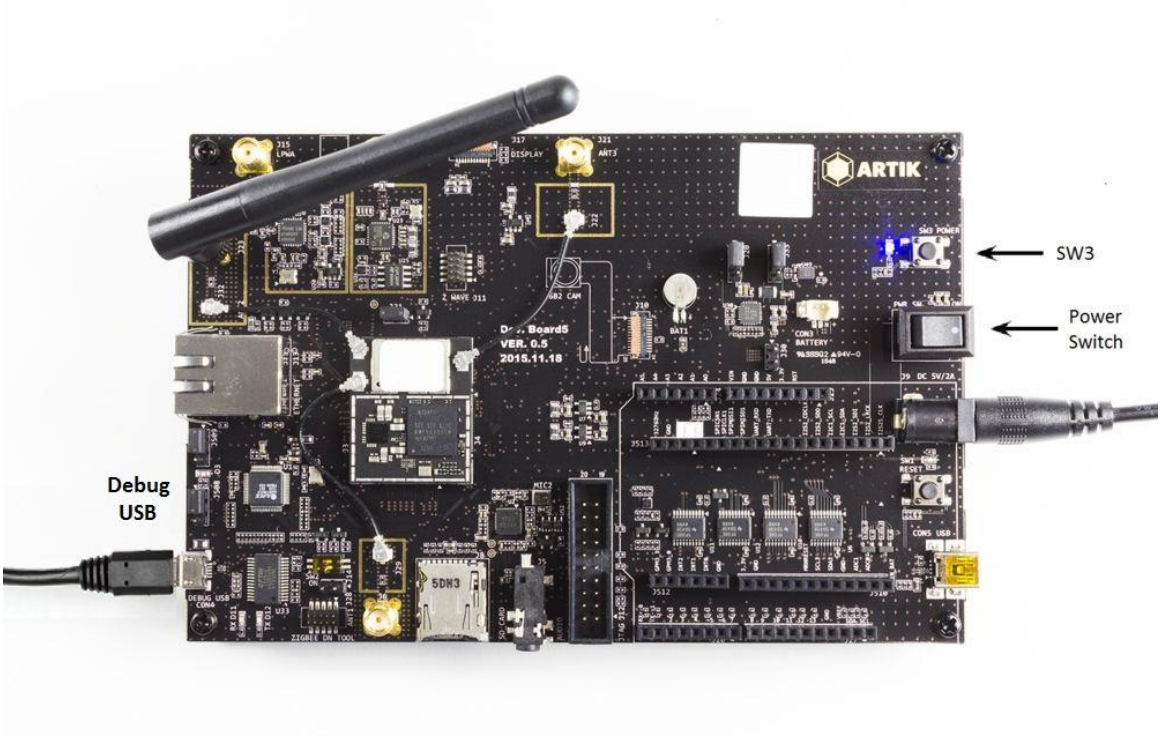
είσοδοι των αναπτυξιακών. Επίσης βασικό ρόλο για την επιλογή τους έπαιξαν οι παράμετροι κόστους, ακρίβειας, ευαισθησίας αλλά και διαθεσιμότητας βιβλιοθηκών.

2.2.1 Gateway Artik 520

Σαν διαδικτυακή πυλη του ασύρματου δίκτυού αισθητήρων, επιλέχτηκε το Samsung Artik 520. Το Artik 520 είναι ένα ολοκληρωμένο System in Module το οποίο περιλαμβάνει διπύρηνο Arm Cortex A7 επεξεργαστή, 512 MB μνήμη DRAM και μία ποικιλία πρότυπων επικοινωνίας όπως Wifi, Bluetooth, Zigbee και άλλα. Όλα αυτά μαζί με το λειτουργικό του σύστημα το οποίο είναι Linux και συγκεκριμένα διανομή Fedora, το κάνουν μία τέλεια επιλογή για εφαρμογές IoT, [43].



Εικόνα 23 Μπλοκ διάγραμμα περιφεριακών Artik 520, [43].



Εικόνα 24 Artik 520, [36].

Το Artik 520 περιλαμβάνει επίσης σε ένα κομμάτι του, αναπτυξιακό το οποίο μπορεί να προγραμματιστεί από το IDE του Arduino, αλλά και να δεχθεί shields αυτού, [44].

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 αλλά και για την ύπαρξη δυνατότητας της επέκτασης του συστήματος από την ποικιλία ασύρματων τεχνολογιών που διαθέτει.

2.2.2 Arduino

Για τα nodes των αισθητήρων επιλέχθηκε η πλατφόρμα Arduino Uno. Αποτελείται απο μία πλακέτα ανάπτυξης βασισμένη στον μικροελεγκτή ATMega328P χαμηλού κόστους, και ανοιχτού υλικού (open hardware).

Ο προγραμματισμός της γίνεται μέσω της γλώσσας προγραμματισμού Wire, μιας «απλοποιημένης» C++ έκδοσης. Η πλατφόρμα Arduino Uno διαθέτει 14 ψηφιακές εξόδους/εισόδους και 6 αναλογικές εισόδους. Κάποιες από αυτές έχουν συγκεκριμένες

ιδιαιτερότητες. Για παράδειγμα η ψηφιακές πόρτες 0 και 1 χρησιμοποιούνται για σειριακή επικοινωνία. Βασική προϋπόθεση για την ενεργοποίηση της είναι η εκτέλεση της εντολής Serial.Begin (Baud-Rate) στην void setup του προγράμματος. Επίσης οι πόρτες 3, 5, 6, 9, 10 και 11 μπορούν να χρησιμοποιηθούν για την δημιουργία PWM παλμών. Τέλος διαθέτει ενσωματωμένο Led στην πόρτα 13.

Οι αναλογικές είσοδοι δύναται να μετρήσουν τάσεις 0-5 V από προεπιλογή. Εάν απαιτείται, είναι δυνατόν το εύρος να αλλάξει σε 0-3,3V ή 0-Vreference όπου η τάση αναφοράς-Vreference μπορεί να είναι οποιαδήποτε επιθυμητή τάση μέχρι 5V.

Εικόνα 25 Arduino Uno, [45].

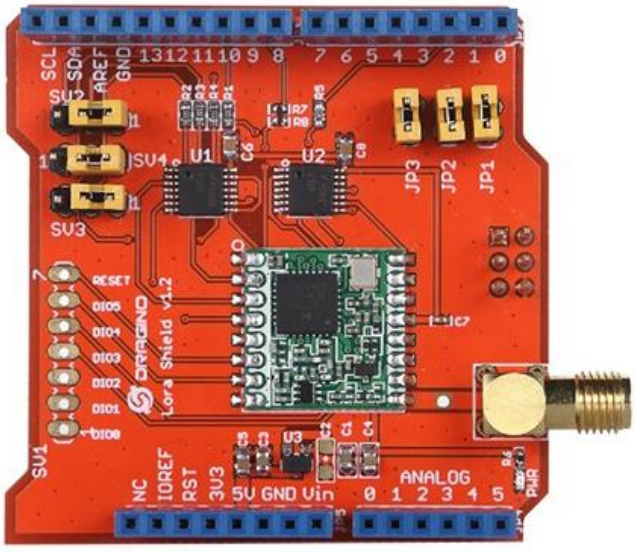
Η τροφοδοσία του μπορεί να γίνει από διάφορες πηγές ενώ το εύρος τάσης εισόδου είναι από 6 έως 20V. Αν και υποστηρίζει το παραπάνω εύρος τάσεων και επειδή το regulator που διαθέτει είναι γραμμικό, γενικά προτείνεται η τάση τροφοδοσίας να μην υπερβαίνει τα 12 V για την αποφυγή αύξησης θερμοκρασίας.

Λόγω του ανοιχτού υλικού και της τεράστιας απήχησης του Arduino, έχουν κατασκευαστεί για αυτό ποικιλία πλακετών επέκτασης-shield αυξάνοντας έτσι το εύρος εφαρμογών που μπορεί να χρησιμοποιηθεί.

Ο Arduino Uno επιλέχθηκε για Node λόγω της απλότητας υλοποίησης, της ύπαρξης κατάλληλων shield ασύρματων τεχνολογιών και το κόστος του αλλά και για την επάρκεια εισόδων/ εξόδων, [45].

2.2.2.1 Lora Shield

Για την χρήση της ασύρματης τεχνολογίας LoRa, χρησιμοποιήθηκε η πλακέτα επέκτασης Dragino σε συνδυασμό με ένα Arduino Uno. Η συγκεκριμένη πλακέτα είναι ευρέως διαδεδομένη, ενώ το κόστος της δεν ξεπερνά τα 40 ευρώ στην Ελλάδα.



Εικόνα 26 LoRa Shield, [46].

Η υλοποίηση του βασίζεται στα ολοκληρωμένα SX1276/SX1278 της Semtech και υπάρχει διαθέσιμη στις συχνότητες 915MHz, 868 MHZ και 433 MHZ. Η επικοινωνία με την πλατφόρμα Arduino γίνεται μέσω SPI και έχει ευαισθησία -148 dBm και ισχύ εκπομπής 20 dBm, [47].

Για τον προγραμματισμό του χρησιμοποιήθηκε η έτοιμη βιβλιοθήκη Lora του Sandeep Mistry από το GitHub. Επιλέχθηκε λόγω της απλότητας της ως προς τον προγραμματισμό. Για την χρήση της LoRa.h θα πρέπει κατ' ελάχιστον να υπάρχει ο παρακάτω κώδικας στο πρόγραμμα, [48].

```
#include <SPI.h>
#include <LoRa.h>
void setup() {
Serial.begin(9600);
if (!LoRa.begin(868E6)) {
Serial.println("Starting LoRa failed!");
while (1);
}
}
```

Παράδειγμα κώδικά αρχικοποίησης Dragino Shield

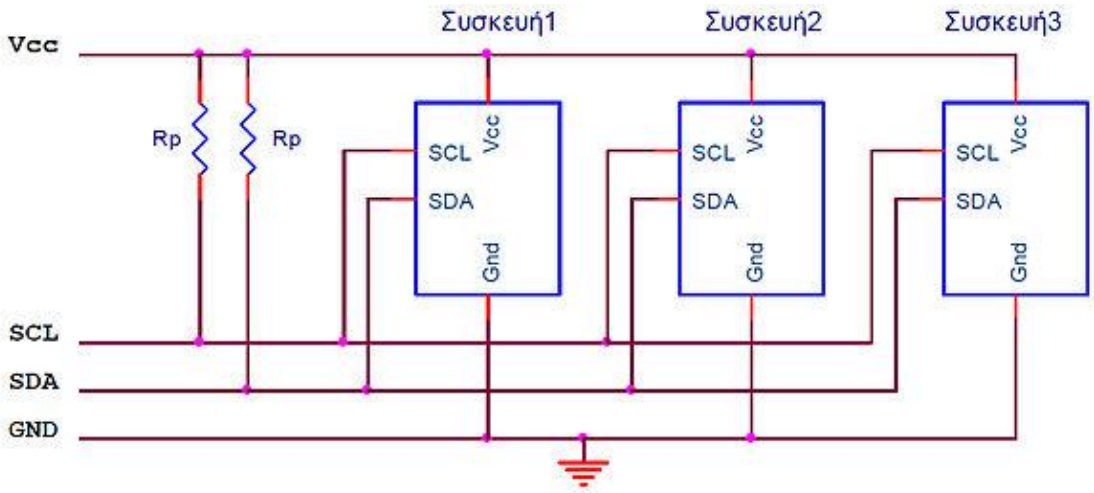
```
void loop() {
// send packet
LoRa.beginPacket();
LoRa.print("hello ");
LoRa.endPacket();
delay(5000);
}
```

```
void loop() {
  int packetSize = LoRa.parsePacket();
  if (packetSize) {
    // received a packet
    Serial.print("Received packet '");
    // read packet
    while (LoRa.available()) {
      Serial.print((char)LoRa.read());
    }
}
```

Παραδείγματα αποστολής και λήψης δεδομένων με Dragino Shield

2.2.2.2 Ο δίαυλος I²C

Ο δίαυλος I²C είναι ένας σειριακός δίαυλος που δημιουργήθηκε από τη Philips (τώρα NXP) και χρησιμοποιείται για την σύνδεση περιφερειακών μικρής ταχύτητας σε μητρικές κάρτες, ενσωματωμένα συστήματα, κινητά τηλέφωνα ή άλλες ηλεκτρονικές συσκευές.



Εικόνα 27 Παράδειγμα σύνδεσης συσκευών I2C, [49].

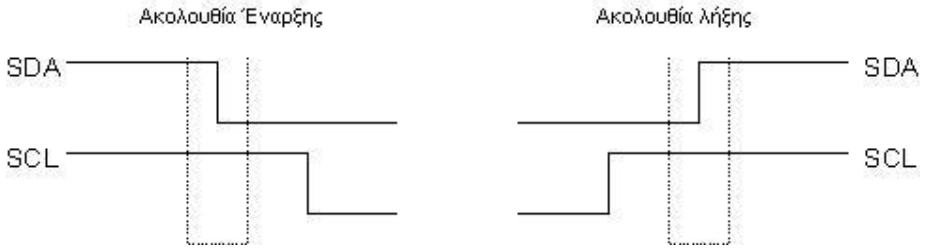
Για τη μεταφορά των *δεδομένων* χρησιμοποιεί μόνο δύο καλώδια, τα SCL και SDA. Η γραμμή SCL είναι η γραμμή ρολογιού, ενώ η SDA είναι η γραμμή δεδομένων. Οι γραμμές αυτές συνδέονται σε όλες τις συσκευές, που υπάρχουν πάνω στο δίαυλο I²C. Τυπικές τάσεις που χρησιμοποιούνται στο δίαυλο είναι τα +5V ή +3,3V

Ο μέγιστος αριθμός συσκευών, που μπορούν να συνδεθούν στον δίαυλο, περιορίζεται από τον αριθμό των διαθέσιμων διευθύνσεων, αλλά και από τη συνολική χωρητικότητα του διαύλου, η οποία π.χ. για τον standard mode, δεν πρέπει να υπερβαίνει τα 400pF. Η απαίτηση αυτή περιορίζει τις πρακτικές αποστάσεις επικοινωνίας.

Οι συσκευές στον δίαυλο I²C είναι είτε Κύριοι (Masters) είτε Υποτελείς (Slave). Η Master συσκευή είναι αυτή που ελέγχει και οδηγεί τη γραμμή ρολογιού SCL. Οι Slave συσκευές είναι αυτές που ανταποκρίνονται στις συσκευές Master. Σε έναν δίαυλο μπορεί να είναι συνδεμένες πολλές Master και πολλές Slaves συσκευές. Και οι Master και οι Slave συσκευές μπορούν να μεταφέρουν δεδομένα στον δίαυλο, αλλά μόνο οι Master συσκευές ελέγχουν την μεταφορά.

Όταν μία Master συσκευή επιθυμεί να επικοινωνήσει με μία Slave συσκευή, ξεκινά στέλνοντας στον δίαυλο μία ακολουθία έναρξης (start sequence). Η ακολουθία έναρξης,

είναι μία από τις δύο ειδικές ακολουθίες που ορίζονται στο δίαυλο I²C. Η άλλη είναι η ακολουθία λήξης (stop sequence). Οι ακολουθίες έναρξης και λήξης διαφέρουν στο ότι είναι οι μοναδικές περιπτώσεις στις οποίες επιτρέπεται να αλλάζει η γραμμή δεδομένων (SDA), ενόσω η γραμμή ρολογιού (SCL) είναι σε κατάσταση λογικού 1 (high). Όταν μεταφέρονται δεδομένα, η γραμμή SDA πρέπει να παραμένει σταθερή και να μην αλλάζει όσο η γραμμή ρολογιού είναι high. Οι ακολουθίες έναρξης και λήξης σημαδεύουν την έναρξη και τη λήξη μιας μεταφοράς με μία slave συσκευή. Δηλαδή ο δίαυλος θεωρείται ότι είναι απασχολημένος, μετά από μία ακολουθία έναρξης και ελεύθερος λίγο χρόνο μετά την ακολουθία λήξης, [49].



Εικόνα 28 Ακολουθίες Έναρξης Λήξης, [49].

2.2.2.3 Διευθυνσιοδότηση συσκευών του I²C διαύλου

Σε όλες τις slave συσκευές που συνδέονται στον δίαυλο, έχει αποδοθεί ένας αριθμός σαν διεύθυνση. Οι master συσκευές δεν είναι απαραίτητο να έχουν διεύθυνση, εκτός εάν υπάρχουν πολλές master συσκευές στον δίαυλο (περιβάλλον Multi-master). Οι master συσκευές μπορούν να διαλέξουν αυθαίρετα μία από τις συνδεμένες slave για επικοινωνία, χρησιμοποιώντας τη διεύθυνσή της. Οι διευθύνσεις των συσκευών του I²C διαύλου είναι είτε 7 bit (θεωρητικά έως 128 συσκευές στο δίαυλο), είτε 10 bit (θεωρητικά έως 1024 συσκευές στο δίαυλο) ή ακόμη και 16 bit (θεωρητικά 65536 συσκευές στο δίαυλο), [49].

2.3 Οι αισθητήρες του συστήματος

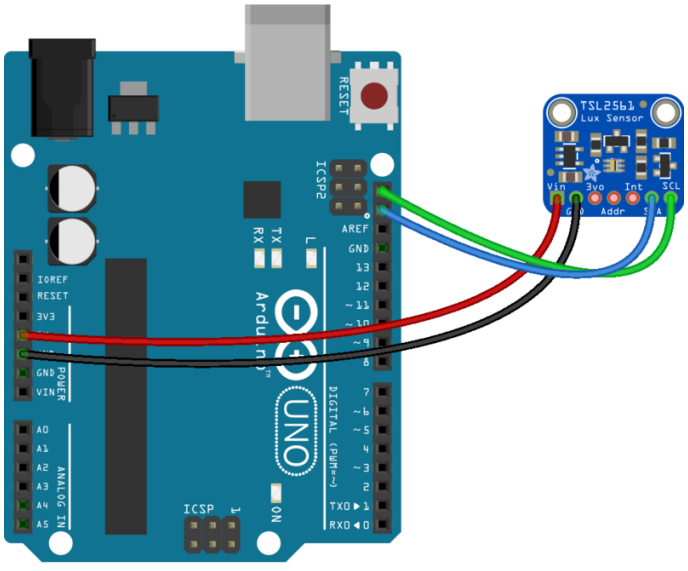
2.3.1 Ο αισθητήρας φωτεινότητας TSL2561

Για την μέτρηση της φωτεινότητας επιλέχθηκε ο αισθητήρας TSL2561. Πρόκειται για έναν προηγμένο ψηφιακό αισθητήρα, χαμηλού κόστους. Σε σύγκριση άλλους αισθητήρες χαμηλού κόστους, αυτός είναι πιο ακριβής, ενώ είναι δυνατό να ρυθμιστεί για διαφορετικές κλίμακες κέρδους / χρονισμού.

Το εύρος μέτρησης κυμαίνεται από 0.1 έως 17000+ Lux, σε πραγματικό χρόνο. Αξιοσημείωτο είναι ότι περιέχει διόδους υπέρυθρων εκτός από ορατού φάσματος. Αυτό σημαίνει ότι μπορεί να πραγματοποιήσει χωριστή μέτρηση για το φάσμα υπερύθρων και για το ορατό φως.

Ο TSL2561 διαθέτει επικοινωνία I2C ενώ χρησιμοποιεί μία από τις διευθύνσεις 0x29, 0x39, 0x49. Ο ορισμός της διεύθυνσης γίνεται ανάλογα την κατάσταση του pin address (3.3v, 0v ή floating).

Τέλος, η τάση τροφοδοσίας του αισθητήρα είναι 3,3v, η κατανάλωση του 0,24 mA ενώ για την σύνδεση του αισθητήρα με Arduino υπάρχει διαθέσιμη βιβλιοθήκη στο Github.



Εικόνα 29 Συνδεσμολογία αισθητήρα TSL2561, [50].

Για την υλοποίηση προγράμματος για μέτρηση φωτεινότητας απαιτείται αρχικά η παραμετροποίηση του αισθητήρα και στην συνέχεια η εκκίνηση της διαδικασίας κάνοντας χρήση της εντολής tsl.begin(), [50], [51]. Για την λήψη μέτρησης φωτεινότητας εκτελείται η παρακάτω ρουτίνα:

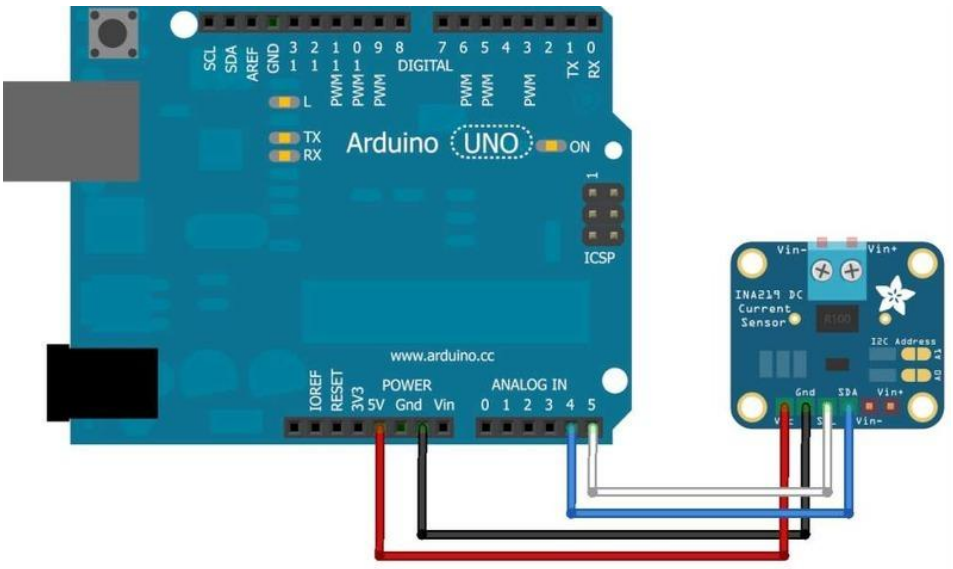
```
void loop(void)
{
  /* Get a new sensor event */
  sensors_event_t event;
  tsl.getEvent(&event);
   /* Display the results (light is measured in lux) */
  if (event.light)
  {    Serial.print(event.light); Serial.println(" lux");
  }
   delay(250);
}
```

2.3.2 Ο αισθητήρας ρεύματος INA219 της Adafruit

Για την μέτρηση του ρεύματος χρησιμοποιήθηκε ο αισθητήρας INA219. Πρόκειται για έναν αισθητήρα με ακρίβεια 1%.

Ένας τελεστικός ενισχυτής υψηλής ακρίβειας μετράει την πτώση τάσης από μία αντίσταση shunt 0.1Ω 1%. Το μέγιστο ρεύμα που μπορεί να μετρήσει μέχρι 3.2A. Επίσης με τον ADC 12 bit που διαθέτει η ανάλυση για το εύρος των ±3.2A είναι 0.8 mA ενώ εάν γίνει αντικατάσταση της αντίστασης shunt μπορεί να αλλάξει το scale βάσει των χαρακτηριστικών της.

Ο INA219 διαθέτει επικοινωνία I2C ενώ χρησιμοποιεί μία από τις διευθύνσεις 0x40, 0x41, 0x44 και 0x45. Ο ορισμός της διεύθυνσης γίνεται με βραχυκύκλωμα ορισμένων επαφών στην πλακέτα του αισθητήρα.



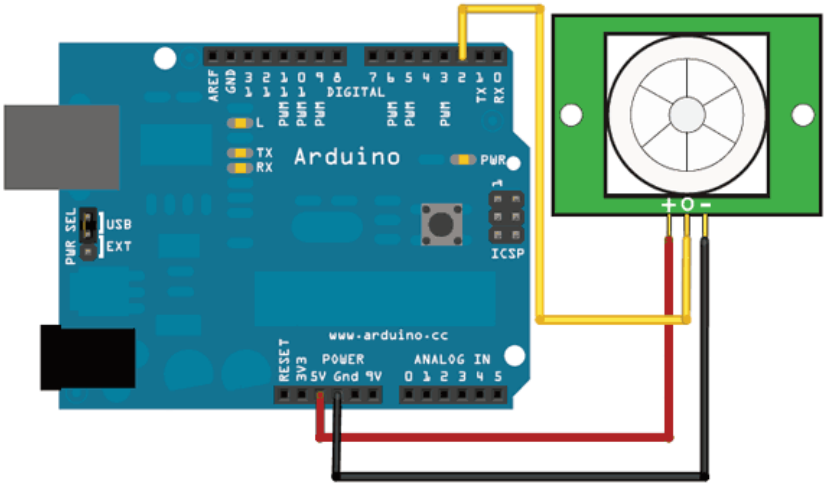
Εικόνα 30 Συνδεσμολογία αισθητήρα INA219, [52].

Για τον προγραμματισμό του αισθητήρα χρησιμοποιήθηκε η έτοιμη βιβλιοθήκη της Adafruit. Για την πραγματοποίηση μέτρησης ρεύματος με τον αισθητήρα INA219 δεν απαιτείται αρχικοποίηση, [52], [53], Παράδειγμα μέτρησης ρεύματος περιγράφεται παρακάτω:

```
void loop(void)
{
  float current_mA = 0;
  current_mA = ina219.getCurrent_mA();
    Serial.print("Current:       ");
Serial.print(current_mA);
Serial.println(" mA");
  Serial.println("");
  delay(2000);
}
```

2.3.3 Ο ανιχνευτής κίνησης HC-SR501

Για την ανίχνευση κίνησης επιλέχθηκε ο αισθητήρας HC-SR501. Έχει την δυνατότητα να ανιχνεύσει την κίνηση μέσα σε ένα δωμάτιο σε εμβέλεια έξι μέτρων. Ο αισθητήρας διαθέτει δύο μεταβλητές αντιστάσεις (trimmer) όπου μπορείτε να ρυθμίσετε την ευαισθησία και τον χρόνο ενεργοποίησης του από την στιγμή που θα ανιχνεύσει την κίνηση. Η τάση λειτουργίας του αισθητήρα είναι 5 - 12 VDC, ενώ σε περίπτωση εντοπισμού κίνησης, ενεργοποιείτε έξοδος τάσης 3.3V. Η συνολική κατανάλωση του αισθητήρα είναι μικρότερη από 60μA.



Εικόνα 31 Συνδεσμολογία αισθητήρα HC-SR501.

Για την υλοποίηση προγράμματος για τον εντοπισμό κίνησης δεν απαιτείται κάποια βιβλιοθήκη καθώς ο αισθητήρας αποτελείται μόνο από μία ψηφιακή έξοδο 5V, [54]. Για την ανάγνωση της κατάστασης του αισθητήρα απαιτείται η χρήση της παρακάτω εντολής:

```
digitalRead(PirSensorPin);
```

2.3.4. Η Πλατφόρμα Ubidots

Για την απεικόνιση των δεδομένων και για την χρήση των ενεργοποιητών θα πρέπει να βρεθεί πλατφόρμα η οποία θα είναι ικανή να διαχειριστεί των αριθμό των απαιτούμενων

αισθητήρων και ενεργοποιητών. Θα πρέπει το documentation να είναι επαρκές και κατά προτίμηση να υποστηρίζει το πρωτόκολλο επικοινωνίας MQTT.

Για την συγκέντρωση των δεδομένων επιλέχθηκε η ελεύθερη έκδοση της πλατφόρμας Ubidots. Στην συγκεκριμένη έκδοση μπορούμε να εισάγουμε μέχρι 20 συσκευές ενώ υποστηρίζει και MQTT. Στο Education Plan, η πρώτη συσκευή είναι δωρεάν ενώ για τις υπόλοιπες απαιτείται πληρωμή με credits. Το Education Plan του Ubidots περιλαμβάνει 5000 credits δωρεάν ενώ εάν επιθυμούμε την αγορά επιπλέον, κοστίζουν 5$ ανά 1000 credits.

Για την χρήση της πλατφόρμας χρησιμοποιήθηκε της βιβλιοθήκης PubSubClient. Για την αποστολή δεδομένων προς τον broker του Ubidots απαιτείται να γνωρίζουμε τα παρακάτω:

- Το Token του λογαριασμού μας.
- Το topic του αισθητήρα/ ενεργοποιητή.

Το Token είναι διαθέσιμο στο προφίλ του λογαριασμού μας ενώ το topic είναι το όνομα της συσκευής.

Η εισαγωγή δεδομένων αισθητήρα αλλά και για τις εντολές των ενεργοποιητών, γίνεται με τον παρακάτω τρόπο.

{"Sensor1": 10, "Sensor2": 50}

Επίσης υπάρχει η δυνατότητα να αποστείλουμε επιπλέον παραμέτρους όπως στίγμα GPS, κείμενο και timestamp. Παρακάτω παραθέτονται παράδειγμα αποστολής τέτοιου τύπου δεδομένων.

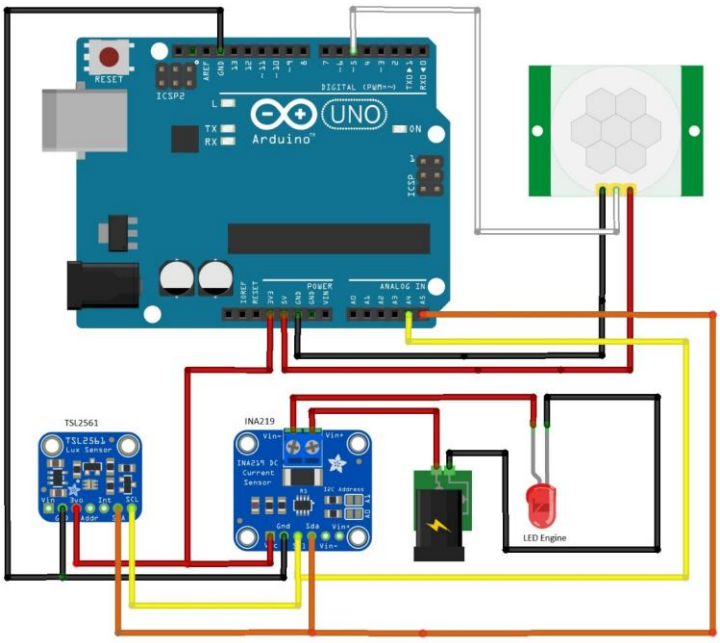
{"temperature": {"value":10, "timestamp": 1464661369000, "context":{"lat":-6.2, "lng":75.4, "my-key":"hello there"}}, "humidity": 50}

Κεφάλαιο 3

Υλοποίηση συστήματος

3.1 Υλοποίηση των Nodes του συστήματος

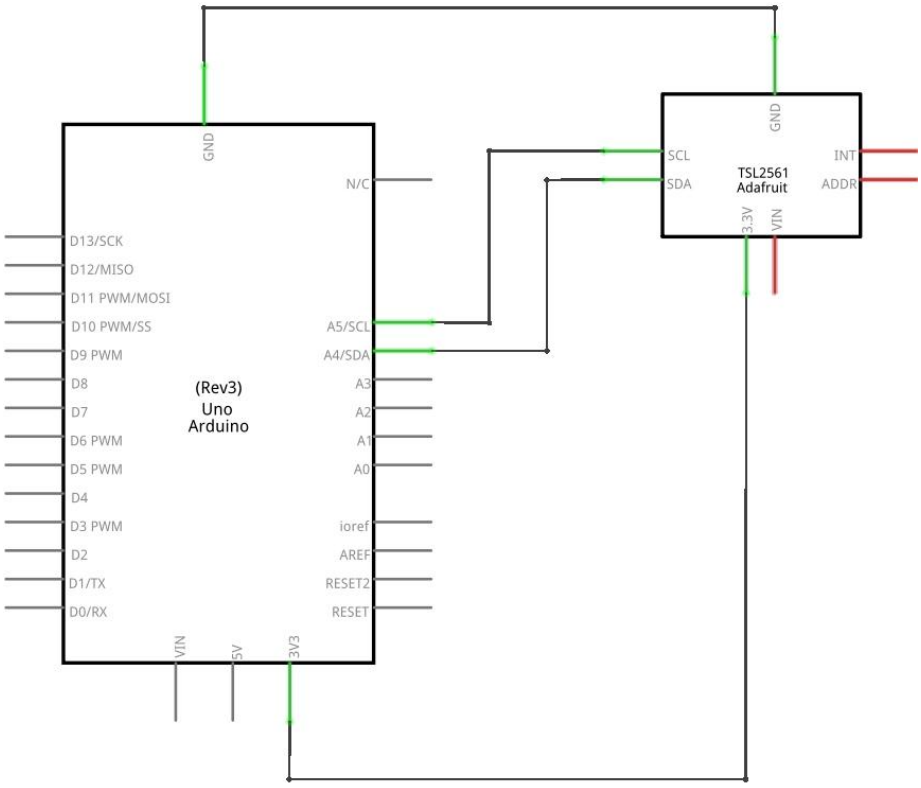
Για την υλοποίηση του συστήματος κάθε Client Node θα χρησιμοποιηθεί η πλαφόρμα Arduino Uno με LoRa Shields. Από αισθητήρια θα συνδεθούν ο αισθητήρας φωτεινότητας, ο αισθητήρας ρεύματος και ο ανιχνευτής κίνησης. Τα δεδομένα φωτεινότητας θα χρησιμοποιούνται ώστε να είναι δυνατή η αυτόματη ρύθμιση της φωτεινότητας του Led Engine ενώ οι μετρήσεις ρεύματος και από τον ανιχνευτή κίνηση, θα χρησιμοποιούνται ως αναφορά για την χρήση της λάμπας και της κίνησης στην περιοχή. Το πρωτόκολλο επικοινωνίας των αισθητήρων κίνησης και φωτεινότητας είναι το I2C οπότε συνδέονται παράλληλα στις κατάλληλες εισόδους. Για την σύνδεση του αισθητήρα κίνησης απαιτείται χρήση μίας ψηφιακής εισόδου.



Εικόνα 32 Σχηματικό σύνδεσης του Node.

3.1.1 Σύνδεση του αισθητήρα φωτεινότητας και ανάγνωση των δεδομένων του

Για την σύνδεση του αισθητήρα φωτεινότητας απαιτείται τροφοδοσία του αισθητήρα και η σύνδεση των pin SDA και SDK με τον Arduino. Η μέτρηση που λαμβάνει ο Arduino δεν χρειάζεται κάποια επεξεργασία (με εξαίρεση την πιθανή βαθμονόμηση του αισθητήρα) και μπορεί να αξιοποιηθεί άμεσα.



Εικόνα 33 Σχηματικό σύνδεσης αισθητήρα φωτεινότητας.

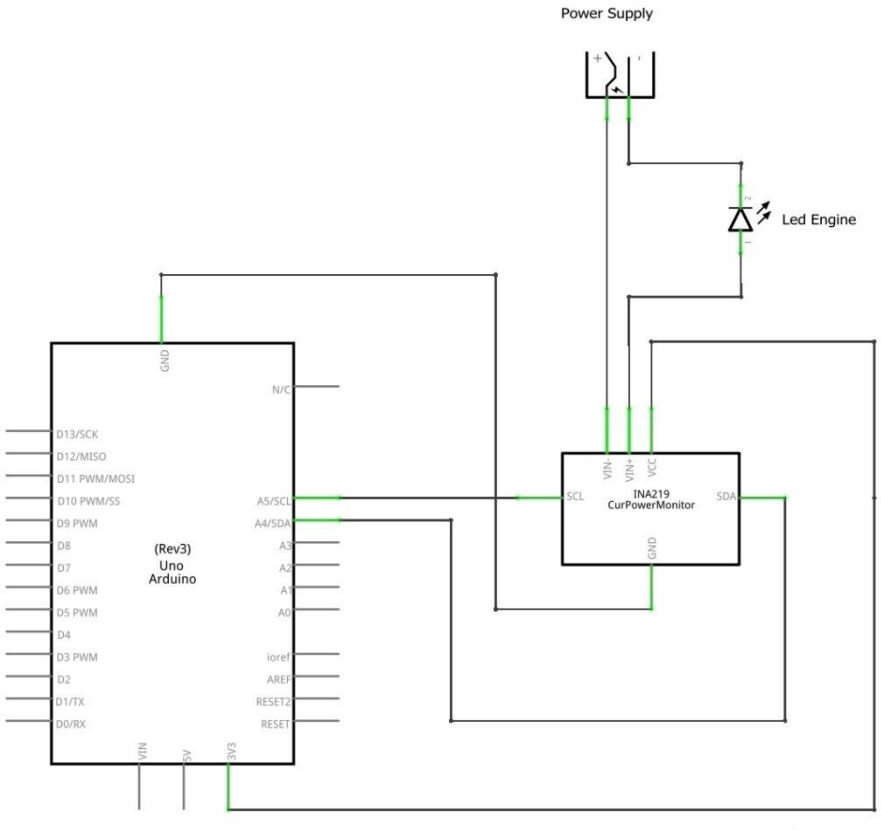
Για την εκκίνηση και την ανάγνωση των δεδομένων του αισθητήρα αναπτύχθηκαν οι συναρτήσεις displaySensorDetails(), configureSensor() και η LuxMetering().

Η πρώτη συνάρτηση αφορά την εμφάνιση των λεπτομερειών του αισθητήρα στην σειριακή θύρα. Η συνάρτηση configureSensor αφορά την παραμετροποίηση του αισθητήρα και περιλαμβάνει τις παραμέτρους gain και setintegrationtime. Η πρώτη αφορά το κέρδος του αισθητήρα και έχει επιλεχθεί η επιλογή auto ώστε να είναι μεταβαλλόμενο ανάλογα την

φωτεινότητα. Η δεύτερη αφορά τον χρόνο μέτρησης. Όσο περισσότερος χρόνος μέτρησης, τόσο μεγαλύτερη ακρίβεια μέτρησης. Έχει επιλεχθεί ο χρόνος των 13 ms. Καθώς δεν μας ενδιαφέρει η μεγάλη ακρίβεια μέτρησης για την συγκεκριμένη εφαρμογή. Τέλος μέσω της συνάρτησης LuxMetering() παίρνουμε την μέτρηση φωτεινότητας και την καταχωρούμε στην μεταβλητή lux.

3.1.2 Σύνδεση του αισθητήρα ρεύματος και ανάγνωση των δεδομένων του

Για την σύνδεση του αισθητήρα ρεύματος απαιτείται τροφοδοσία του αισθητήρα και η σύνδεση των pin SDA και SDK με τον Arduino. Η μέτρηση που λαμβάνει ο Arduino δεν χρειάζεται κάποια επεξεργασία (με εξαίρεση την πιθανή βαθμονόμηση του αισθητήρα) και μπορεί να αξιοποιηθεί άμεσα.

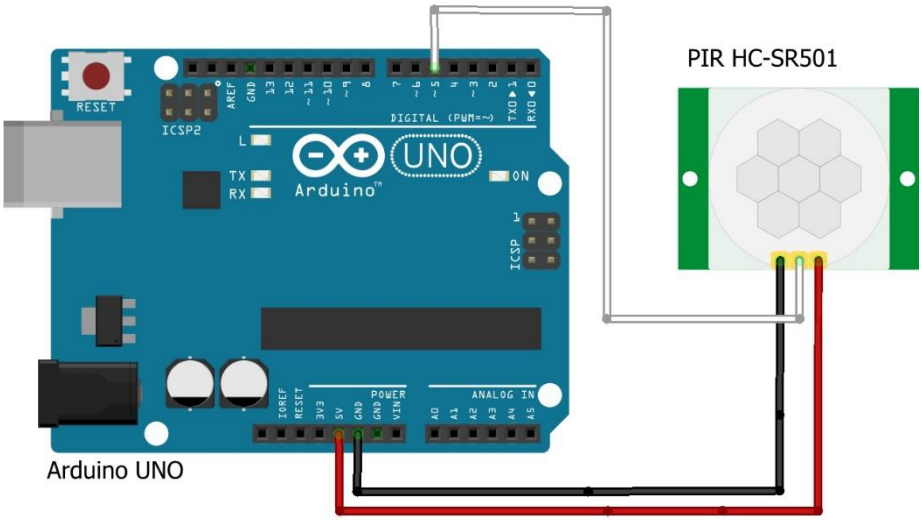


Εικόνα 34 Σχηματικό σύνδεσης αισθητήρα ρεύματος.

Για την εκκίνηση και την ανάγνωση των δεδομένων του αισθητήρα αναπτύχθηκε η συνάρτηση CurrentMetering(), στην οποία απλά η μέτρηση ρεύματος μετράτε μέσω της εντολής ina219.getCurrent_mA() και καταχωρείτε στην μεταβλητή current.

3.1.3 Σύνδεση του αισθητήρα κίνησης και ανάγνωση των δεδομένων του

Για την σύνδεση του αισθητήρα κίνησης απαιτείται τροφοδοσία του αισθητήρα και η σύνδεση σε μία ψηφιακή είσοδο του Arduino.



Εικόνα 35 Σχηματικό σύνδεσης αισθητήρα κίνησης.

Ο αισθητήρας κίνησης χρησιμοποιείται ως μετρητής κίνησης. Οι εναύσεις του αισθητήρα καταγράφονται σε έναν αθροιστή και αποστέλλονται για αξιολόγηση από τον χρήστη. Για την σύνδεση του αισθητήρα αναπτύχθηκε η συνάρτηση MotionDetect(). Η συνάρτηση αυτή καταγράφει συνεχώς την κατάσταση της ψηφιακής εισόδου που είναι συνδεμένος ο αισθητήρας. Σε περίπτωση που ο Arduino δώσει λογικό 1 στην έξοδο ενώ στον προηγούμενο κύκλο προγράμματος ήταν λογικό 0, ο αθροιστής αυξάνετε. Αυτή η μέθοδος ονομάζεται εντοπισμός θετικής παρυφής.

3.2.1 Αποστολή δεδομένων από το LoRa Client στο LoRa Gateway

Για την αποστολή των δεδομένων έχει υλοποιηθεί η συνάρτηση LoraSend(). Η συνάρτηση καλείτε ανά ορισμένο χρονικό διάστημα. Μέσα σε αυτή αρχικά στέλνουμε τα δεδομένα στην σειριακή θύρα του Arduino για λόγους αποσφαλμάτωσης. Στην συνέχεια εκτελούμε την εντολή LoRa.beginPacket() η οποία ορίζει την εκκίνηση της αποστολής του πακέτου δεδομένων. Έπειτα μέσω της εντολής LoRa.print() στέλνουμε τα δεδομένα. Τέλος για να ορίσουμε την λήξη του πακέτου εκτελούμε την εντολή LoRa.endPacket() ενώ θέτουμε τον αθροιστή του αισθητήρα κίνησης 0 ώστε να αρχίσει την καταμέτρηση από την αρχή. Στην συνέχεια εκτελείται η συνάρτηση Lorareceive() με την οποία λαμβάνονται οι εντολές από τους actuators. Για εντοπισμό της λήψης των δεδομένων εκτελείται η εντολή LoRa.parsePacket() στην οποία καταχωρείται το μέγεθος του πακέτου. Στην συνέχεια αν και εφόσον υπάρχουν διαθέσιμα πακέτα, εμφανίζονται στην σειριακή θύρα και καταχωρούνται στην μεταβλητή toggle ώστε να γίνει ανάγνωση της κατάστασης του actuator.

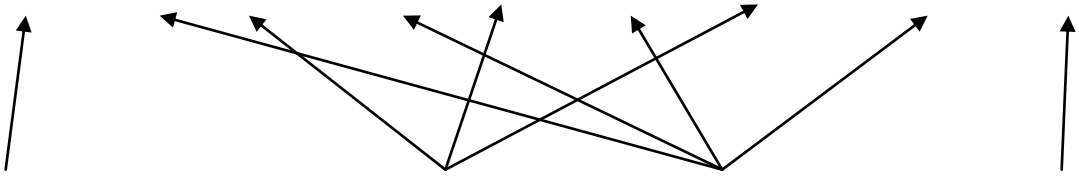
3.2 Υλοποίηση του Gateway του συστήματος

3.2.2 Λήψη δεδομένων από το LoRa Gateway στο LoRa Client

Το LoRa Gateway έχει προγραμματιστεί ώστε να εξυπηρετεί δύο εργασίες. Η πρώτη είναι η λήψη των δεδομένων από τον κόμβο και η αποστολή τους προς την κεντρική πύλη σειριακά, αλλά και η ανάγνωση των εντολών από τον broker για την αποστολή τους προς τους κόμβους. Η αποστολή και η λήψη των δεδομένων μέσω LoRa είναι ακριβώς ίδια με του LoRa Client.

Για την αποστολή των δεδομένων προς το Artik γίνεται απλώς εκτελώντας την εντολή Serial.print(). Για την αποστολή δεδομένων έχει υλοποιηθεί μία “comma separated” αριθμοσειρά η οποία αναγνωρίζεται μέσω συναρτήσεων στο Artik κατηγοριοποιεί τα δεδομένα κατάλληλα. Η αριθμοσειρά είναι της παρακάτω μορφής.

<<«Lux_Number»,«Current_Number»,«Motion_Number»,«Brightness_Number»>



Start Marker Comma Separators Data End Marker

Για την λήψη των δεδομένων του actuator λόγω της απλότητας του συστήματος χρησιμοποιήθηκε μία ψηφιακή είσοδος.

3.2.3 Αποστολή δεδομένων του LoRa Gateway στο Artik και αποστολή προς τον Broker

Για την λήψη των δεδομένων στο Artik Cloud χρησιμοποιήθηκε η σειριακή θύρα. Για την χρήση της σειριακής υλοποιήθηκαν οι συναρτήσεις recvWithStartEndMarkers(),parseData() και η showParsedData().

Η πρώτη συνάρτηση είναι υπεύθυνη για τον εντοπισμό των σημείων εκκίνησης και τερματισμού της αριθμοσειράς. Η συνάρτηση ελέγχει τα δεδομένα που διαβάζονται από την σειριακή θύρα και σε περίπτωση που εντοπίσει το σημείο εκκίνησης (Στην περίπτωση μας το σύμβολο “<”), αποθηκεύει τα επόμενα δεδομένα σε έναν προσωρινό πίνακα char. Η συνάρτηση εντοπίζει το τέλος της αριθμοσειράς μόλις εντοπίσει το σημείο τερματισμού (Σύμβολο “>”). Μετά από αυτό το σημείο εκτελείται η συνάρτηση ParseData(). Η συνάρτηση αυτή διαβάζει την προσωρινή μεταβλητή που είναι αποθηκευμένα τα δεδομένα του κύκλου προγράμματος, και έπειτα τα διαχωρίζει σε ξεχωριστές μεταβλητές. Τα δεδομένα ακολουθούν comma separated λογική και η σειρά τους είναι ορισμένη από τον προγραμματισμού του κόμβου.

Μετά την κατηγοριοποίηση των δεδομένων είναι δυνατή η αποστολή προς τον Broker. Για την αποστολή των δεδομένων έχουν υλοποιηθεί η συνάρτηση sendData(). Για την αποστολή των δεδομένων θα πρέπει αυτά να τροποποιηθούν βάσει των προδιαγραφών

του broker. Για να γίνει αυτό αρχικά μετατρέπουμε τα δεδομένα σε string και τα επεξεργαζόμαστε κατάλληλα. Στην συνέχεια τα μετατρέπουμε σε πίνακα char και τα στέλνουμε στο αντίστοιχο topic μέσω της client.publish() όπου client ο ορισμένος server από το πρόγραμμα. Αντίστοιχα για την λήψη δεδομένων από τον broker χρησιμοποιούμε την συνάρτηση client.subscribe().

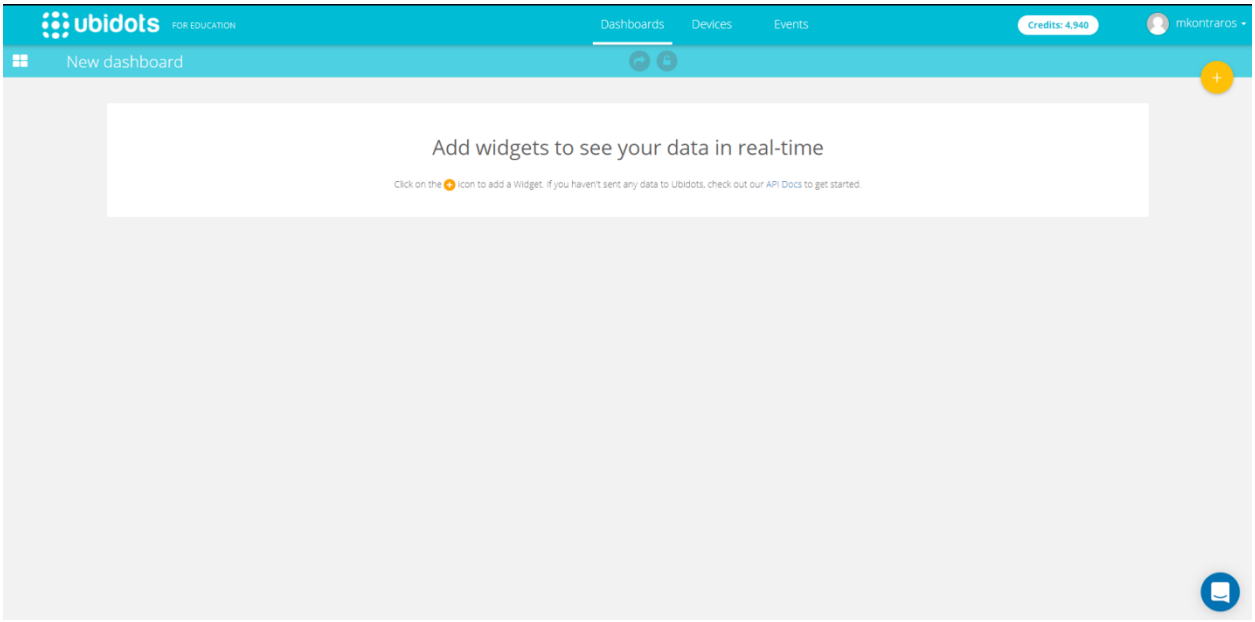
3.2.4 Λήψη δεδομένων του Broker προς το Artik Gateway

Για την λήψη των δεδομένων στο Artik από τον Broker θα πρέπει να γίνει subscribe στα Topics που έχουν δημιουργηθεί σε αυτόν. Για την συγκεκριμένη εργασία έχει φτιαχτεί ένα Topic με όνομα Toggle Light και έχει ως σκοπό την διακοπή λειτουργίας του Led Engine. Η διαδικασία της υλοποίησης θα αναλυθεί σε επόμενο κεφάλαιο. Για την λήψη των δεδομένων έχει υλοποιηθεί η συνάρτηση callback() ενώ χρησιμοποιήθηκε και η έτοιμη συνάρτηση client.subscribe("Topic").

3.3 Υλοποίηση του Ubidots

3.3.1 Υλοποίηση του Dashboard

Το πρώτο στάδιο υλοποίησης αφορά την δημιουργία νέου λογαριασμού. Μετά την δημιουργία του λογαριασμού μας εμφανίζετε ένα κενό Dashboard. Από αυτή την στιγμή είμαστε έτοιμοι να καταχωρήσουμε δεδομένα. Πηγαίνουμε στο παράθυρο Devices και πατάμε add device. Στην συνέχεια επιλέγουμε ένα όνομα συσκευής, το οποίο θα είναι και το topic στο οποίο θα κάνουμε publish τα δεδομένα, [55].



Εικόνα 36 Dashboad του Ubidots.

Με την δημιουργία νέας συσκευής μπορούμε είτε να θέσουμε την διάταξη μας σε λειτουργία και να δημιουργηθούν όλες οι μεταβλητές αυτόματα, είτε να τις δημιουργήσουμε μόνοι μας.

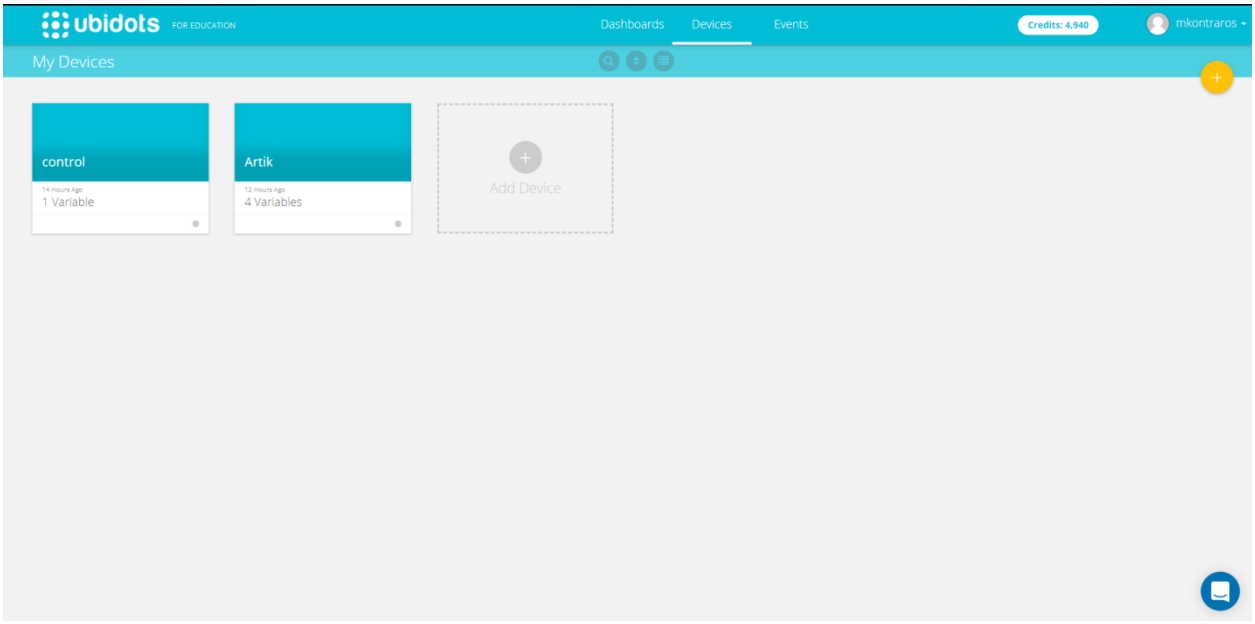
Επιλέγουμε add new variable για να τις δημιουργήσουμε. Οι αισθητήρες που υπάρχουν στο υλοποιημένο σύστημα είναι οι εξής:

- Luminosity
- Brightness
- Pir_sensor
- Current

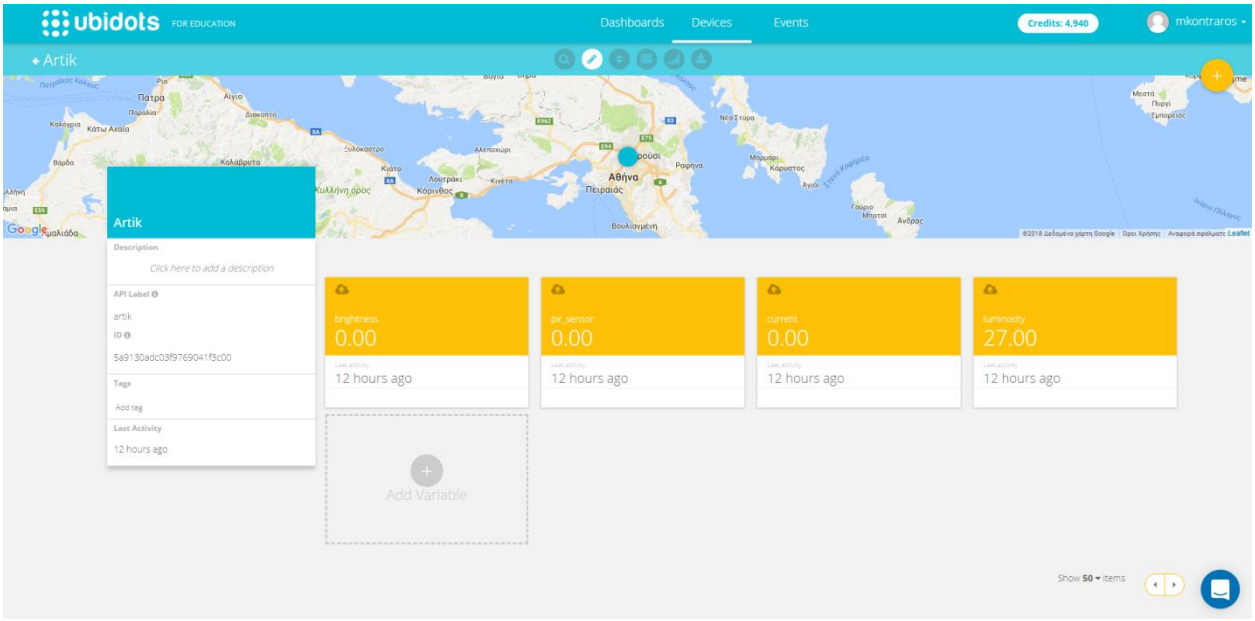
Στη συνέχεια δημιουργούμε μία νέα συσκευή και μία νέα μεταβλητή για να την χρησιμοποιήσουμε ως ενεργοποιητή. Ο ενεργοποιητής που δημιουργήθηκε είναι ο εξής:

- Toggle_Light

Επίσης μπορούμε στον χάρτη να ορίσουμε το στίγμα GPS των συσκευών.



Εικόνα 37 Δημιουργία νέων συσκευών.

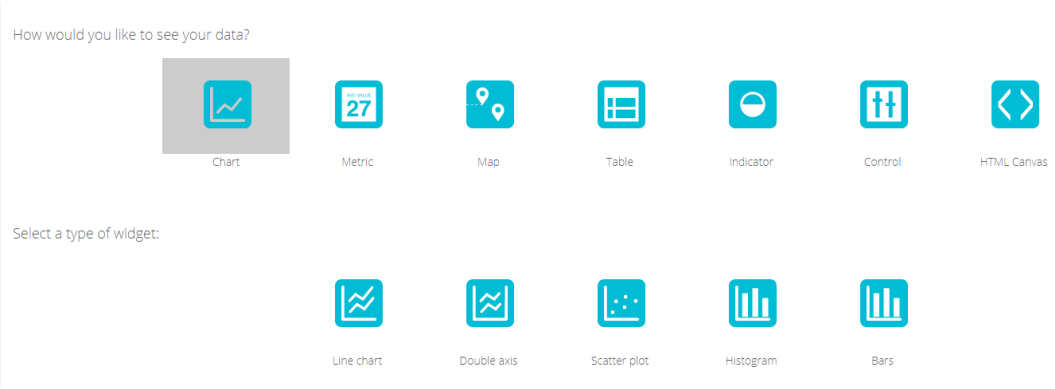


Εικόνα 38 Δημιουργία νέων μεταβλητών.

Σύμφωνα με το documentation του Ubidots δεν απαιτείται καμία ρύθμιση, για τον τύπο των αισθητηρίων, στις μεταβλητές. Οπότε οι μεταβλητές αυτές μπορεί να είναι είτε μετρήσεις είτε

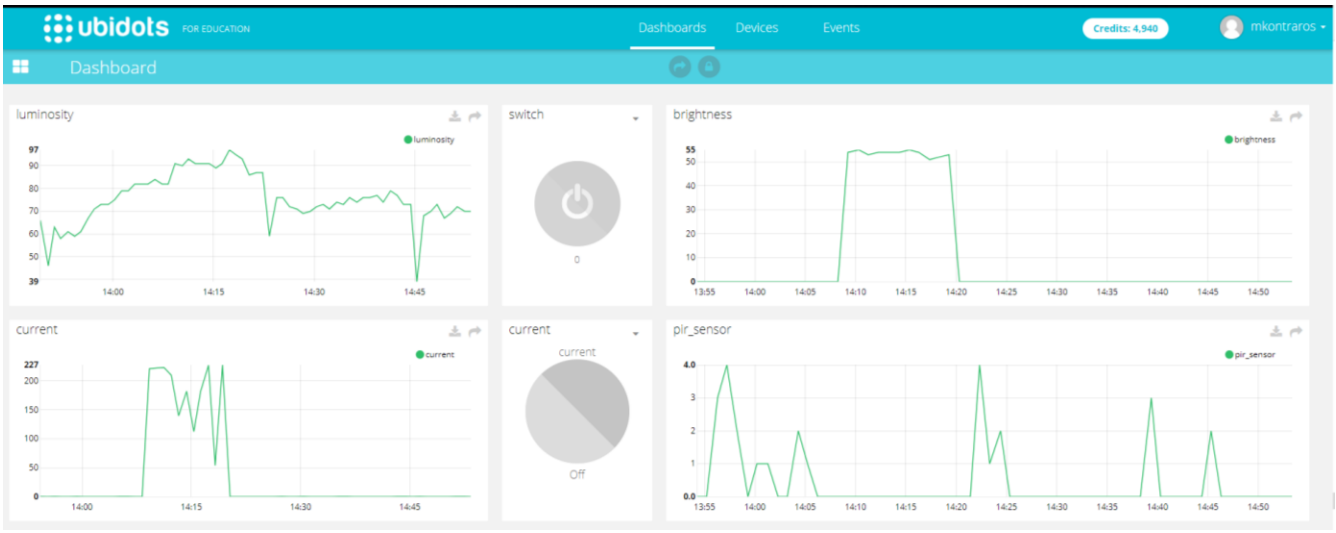
στίγμα είτε οτιδήποτε άλλο. Η υλοποίηση της σωστής υποδομής για την απεικόνιση των μετρήσεων γίνεται στο Dashboard. Αυτό γίνεται μέσω των Widgets.
Αφού ολοκληρώσαμε την καταχώρηση των συσκευών συνεχίζουμε με την υλοποίηση του DashBoard. Το περιβάλλον είναι πλήρως παραμετροποιήσημο το οποίο σημαίνει ότι εκτός από την επιλογή του τύπου απεικόνισης μπορούμε να ορίσουμε την θέση και το μέγεθος των Widgets. Επίσης μπορούμε να ορίσουμε widget τα οποία είναι συνδεμένα με την ίδια μεταβλητή αλλά είναι σχεδιασμένα για άλλου τύπου εργασία, [55].

Μερικές επιλογές widgets από την ποικιλία που μας προσφέρει το Ubidots φαίνεται στην παρακάτω εικόνα.



Εικόνα 39 Δημιουργία νέου Widget.

Με την δημιουργία των Widgets έχουμε ένα λειτουργικό περιβάλλον εργασίας. Το υλοποιημένο dashboard φαίνεται στην παρακάτω εικόνα.



Εικόνα 40 Dashboard υλοποίησης.

3.3.2 Υλοποίηση Events

Από την στιγμή που έχουμε υλοποιήσει το διάταξη αποστολής και λήψης δεδομένων από την πλατφόρμα, είναι δυνατό να προγραμματίσουμε γεγονότα (events) τα οποία αφορούν την εφαρμογή. Μέσω των Event θα μπορούσαμε να παρακολουθήσουμε τις μετρήσεις μίας μεταβλητής και ανάλογα των μετρήσεων να πραγματοποιήσουμε ενέργειες, όπως η αποστολή email ή SMS με συναγερμούς, να αλλάξουμε την κατάσταση των ενεργοποιητών και πολλά άλλα, [**55**].

Η υλοποίηση των event ακολουθεί την λογική If (statement) then (do).

Η πλατφόρμα μας δίνει τις παρακάτω επιλογές ελέγχου.

- Μέθοδος σύγκρισης

  Σε αυτή την μέθοδο ελέγχονται οι μετρήσεις των μεταβλητών. Και συγκρίνονται με μία τιμή threshold που έχει ορίσει ο χρήστης

- Μέθοδος αδράνειας

  Σε αυτή την μέθοδο ελέγχεται η αδράνεια της μεταβλητής και συγκρίνεται με το χρονικό όριο το οποίο έχει επιβάλει ο χρήστης.

- Μέθοδος τοποθεσίας

Είναι η μέθοδος γεωφραγμού. Ελέγχει εάν μία μεταβλητή τοποθεσίας εισέλθει ή εξέλθει από ένα χώρο τον οποίο έχει επιβάλει ο χρήστης.

Για την υλοποίηση των event μας δίνονται οι παρακάτω επιλογές:

- Αποστολή SMS
- Αποστολή E-mail
- Ειδοποίηση σε χρήστη της εφαρμογής Telegram
- Webhook
- Ρύθμιση μεταβλητής

Κεφάλαιο 4

Αποτελέσματα-Συμπεράσματα

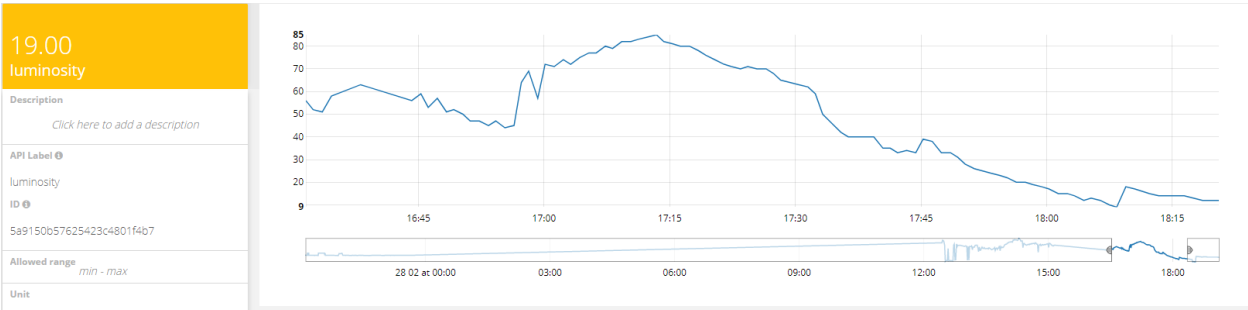
4.1 Απεικόνιση δεδομένων, συμπεράσματα και μελλοντικές ενέργειες

Στην παρούσα εργασία υλοποιήθηκε δίκτυο ασυρμάτων αισθητήρων και ενεργοποιητών χαμηλού κόστους με εφαρμογές πάνω σε συστήματα έξυπνου φωτισμού. Βασικός στόχος ήταν, οι διατάξεις που υλοποιήθηκαν να είναι χαμηλού κόστους για την χρήση τους σε εφαρμογές smart city και ιδιαίτερα street lighting, από την ερευνητική κοινότητα, σε επίπεδο πειραματισμού.

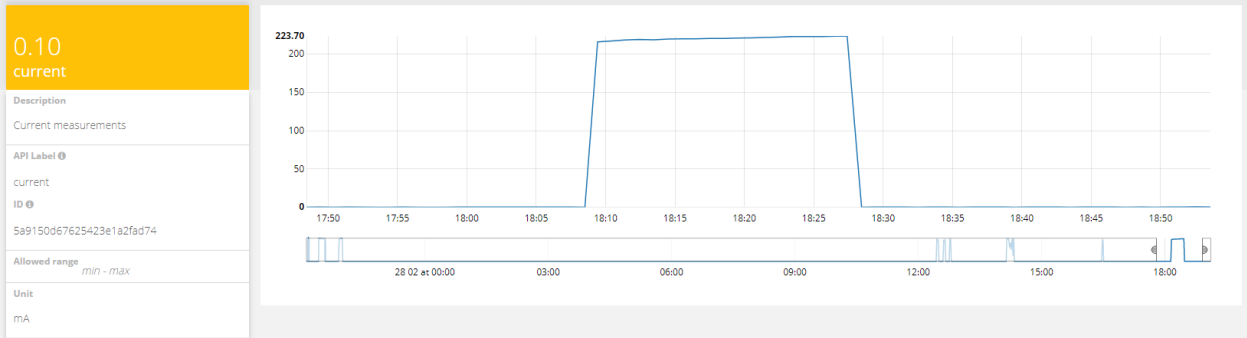
Υλοποιήθηκε Gateway το οποίο δύναται να υποστηρίξει ποικιλία ασύρματων τεχνολογιών όπως zigbee, z-wave, WiFi και sigfox ενώ υλοποιήθηκε συνδεσιμότητα LoRa κάνοντας χρήση της πλατφόρμας Arduino και μετάδοσης των δεδομένων μέσω της σειριακής θύρας.

Επίσης υλοποιήθηκε ασύρματο Node το οποίο ήταν υπεύθυνο για την συλλογή των απαραίτητων δεδομένων, την αυτόματη ρύθμιση φωτεινότητας αλλά και την λήψη και εκτέλεση εντολών από το broker.

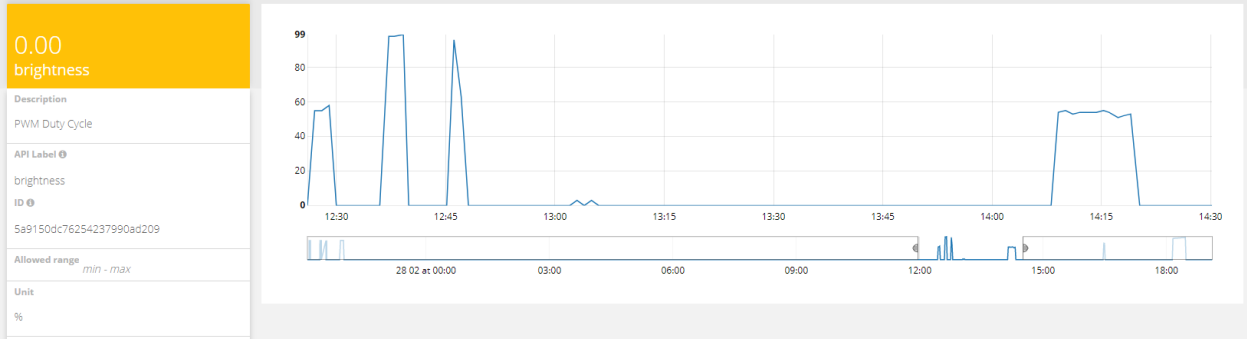
Μετά την υλοποίηση του όλων των σταδίων του project είναι δυνατή η ενεργοποίηση του συστήματος για την συλλογή των δεδομένων. Μέσω του Dashboard μπορούμε να δούμε τις καταγεγραμμένες μετρήσεις ενώ είναι υλοποιήθηκαν event που αφορούν την αδράνεια της διάταξης και μεταβάλλουν την κατάσταση του ενεργοποιητή βάσει φωτεινότητας. Παρακάτω παραθέτονται γραφήματα από τους 4 αισθητήρες.



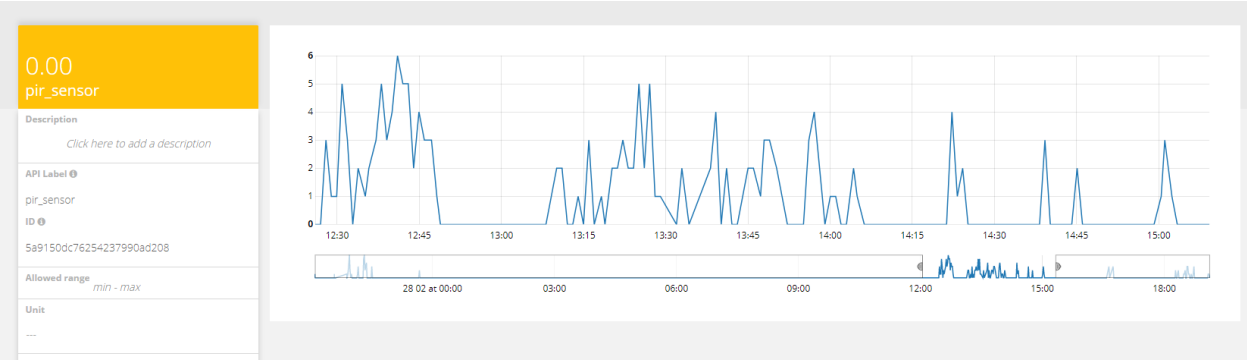
Εικόνα 41 Δεδομένα φωτεινότητας.



Εικόνα 42 Δεδομένα κατανάλωσης ρεύματος.



Εικόνα 43 Δεδομένα αυτόματης ρύθμισης Duty Cycle παλμών PWM.



Εικόνα 44 Δεδομένα εντοπισμού κίνησης.

Κάνοντας χρήση των δεδομένων από τους αισθητήρες είναι δυνατή η δημιουργία προφίλ δεδομένων για το σημείου εγκατάστασης. Έπειτα κάνοντας χρήση των προφίλ, είναι

δυνατή η βέλτιστη ρύθμιση της φωτεινότητας ανά περιοχή, αλλά και για την λήψη στρατηγικών αποφάσεων όσον αφορά την βελτίωση των σεναρίων φωτισμού.

Η παρούσα υλοποίηση έχει μεγάλες προοπτικές ανάπτυξης. Μερικές προτάσεις για θεματολογία ανάπτυξης της, θα μπορούσε να οριστεί η βελτίωση του κώδικα μεταβολής φωτεινότητας. Στην συγκεκριμένη υλοποίηση η φωτεινότητα μεταβάλλεται με χρήση της συνάρτησης «map». Η συνάρτηση υπολογισμού είναι η εξής:

$$pwmvalue = \frac{(x - in_{min}) * (out_{max} - out_{min})}{(in_{max} - in_{\min})} + out_{min}$$

όπου:

- pwmvalue το duty cycle των παλμών pwm [0,255]
- x: η τιμή της φωτεινότητας της παρούσα στιγμή
- in_{min} : η επιλεγμένη ελάχιστη τιμή της φωτεινότητας
- in_{min} : η επιλεγμένη μέγιστη τιμή της φωτεινότητας
- out_{min} : η επιλεγμένη ελάχιστη τιμή που δύναται να πάρει η μεταβλητή pwmvalue
- out_{min}: η επιλεγμένη μέγιστη τιμή που δύναται να πάρει η μεταβλητή pwmvalue

Η χρήση της συνάρτησης ενδέχεται να είναι ιδανική αλλά για βέλτιστα αποτελέσματα θα πρέπει να γίνει αξιοποίηση των δεδομένων φωτεινότητας, όταν αυτά είναι επαρκή, ώστε να είμαστε σε θέση να επιλέξουμε την βέλτιστη συνάρτηση υπολογισμού.

Ένα άλλο σενάριο βελτίωσης αφορά την μεταβολή των μεταβλητών φωτεινότητας μέσω OTA(Over The Air). Γενικά οι υλοποιήσεις με LoRa δεν προτείνουν προγραμματισμό OTA λόγω χαμηλής ταχύτητας και θεμάτων ασφαλείας. Αλλά ακόμα και σε αυτή την περίπτωση είναι δυνατός ο «εικονικός» προγραμματισμός κάνοντας χρήση ενός topic το οποίο θα παρεμβάλει μία μεταβλητή στην συνάρτηση μεταβολής και θα αυξομειώνει την τιμή της φωτεινότητας αναλογικά. Σε αυτή την περίπτωση η συνάρτηση μεταβολής θα είναι η εξής:

$$pwmvalue = chval * \left(\frac{(x - in_{min}) * (out_{max} - out_{min})}{(in_{max} - in_{\min})} + out_{min} \right)$$

όπου:

- chval: τιμή ορισμένη από τον διαχειριστή για την αυξομείωση της pwmvalue.

Έτσι μέσω της τιμής chval θα μπορεί ο διαχειριστής, μετά την αξιοποίηση των δεδομένων φωτεινότητας, να αλλάζει την συνάρτηση μεταβολής φωτεινότητας, αναλογικά, με σκοπό το βέλτιστο αποτέλεσμα.

Επιπλέον όσον αφορά την κατανάλωση ενέργειας, αφού μιλάμε για μία υλοποίηση η οποία δεν έχει μεγάλες απαιτήσεις ως προς την αμεσότητα αποστολής δεδομένων, είναι δυνατό η αποστολή των δεδομένων να μην γίνεται την στιγμή της πραγματοποίησης μέτρησης από το σύστημα, αλλά να γίνεται η αποθήκευση τους προσωρινά στον κόμβο και την αποστολή τους ανά προγραμματισμένα χρονικά διαστήματα, συνολικά. Για την υλοποίηση αυτή θα πρέπει να προστεθεί RTC module στο οποίο θα καταγράφεται η ώρα και να γίνουν οι απαραίτητες αλλαγές στο format του payload των αισθητήρων ώστε να αναγράφεται η ώρα μέτρησης σε μορφή timestamp. Αυτό θα είχε ως μειονέκτημα την μείωση της συχνότητας λήψης δεδομένων στην πλατφόρμα αλλά το αποτέλεσμα θα ήταν η κατακόρυφη μείωση της κατανάλωσης ενέργειας καθώς η αποστολή των μετρήσεων θα γίνεται μερικές φορές την ημέρα βάσει προγράμματος, ενώ η ανάλυση των μετρήσεων θα παρέμενε η ίδια.

Παραρτήματα

Παράρτημα Α

Κώδικας κόμβου με συνδεσιμότητα LoRa

```
/*
  Created 28 February 2018
  By Kontraros Michail
*/
#include <Adafruit_INA219.h>
#include <SPI.h>
#include <LoRa.h>
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_TSL2561_U.h>

Adafruit_INA219 ina219;
Adafruit_TSL2561_Unified tsl = Adafruit_TSL2561_Unified(TSL2561_ADDR_FLOAT,
12345);

/**************************************************************************/
/*
   Variable assignment
*/
/**************************************************************************/

uint32_t currentFrequency;  //variable required in INA219 library
int pirCount = 0;           //counter variable for motion detection
int lastpirState = 0;       // previous state of the pirsensor

int pirSensor = 5;          // pirSensor Pin input
int pirstate;               // variable for reading the PirSensor status
int ledctrl = 3;            // PWM output for LED Engine Dimming
int pirresult;

int pwmvalue;               // Variable of PWM value
int pwmpercent;             //PWM value in %
float currentresult = 0.0;  //variable of result of current sensor
float luxresult = 0.0;      //variable of result of lux sensor

long previousMillis;        // Variable required for delay function
long interval = 60000;      //time interval of data transmission

String datastring = "";     //data to send
char databuf[10];

char loraread;
String toggle;

/**************************************************************************/
/*
    Displays some basic information on the Lux sensor from the unified
    sensor API sensor_t type of TSL2561, during initialization
*/
```

```
/**************************************************************************/

void displaySensorDetails(void)
{
  sensor_t sensor;
  tsl.getSensor(&sensor);
  Serial.println("------------------------------------");
  Serial.print  ("Sensor:       ");
  Serial.println(sensor.name);
  Serial.print  ("Driver Ver:   ");
  Serial.println(sensor.version);
  Serial.print  ("Unique ID:    ");
  Serial.println(sensor.sensor_id);
  Serial.print  ("Max Value:    ");
  Serial.print(sensor.max_value);
  Serial.println(" lux");
  Serial.print  ("Min Value:    ");
  Serial.print(sensor.min_value);
  Serial.println(" lux");
  Serial.print  ("Resolution:   ");
  Serial.print(sensor.resolution);
  Serial.println(" lux");
  Serial.println("------------------------------------");
  Serial.println("");
  delay(500);
}

/**************************************************************************/
/*
    Configures the gain and integration time for the TSL2561
*/
/**************************************************************************/

void configureSensor(void)
{

  tsl.enableAutoRange(true);            /* Auto-gain ... switches
automatically between 1x and 16x */

  tsl.setIntegrationTime(TSL2561_INTEGRATIONTIME_13MS);      /* fast but low
resolution */

  /* Update these values depending on what you've set above! */
  Serial.println("------------------------------------");
  Serial.print  ("Gain:         "); Serial.println("Auto");
  Serial.print  ("Timing:       "); Serial.println("13 ms");
  Serial.println("------------------------------------");
}

/**************************************************************************/
/*
    Arduino setup function (automatically called at startup)
*/
/**************************************************************************/

void setup()
{
```

```
  // initialize the pirSensor pin as an input
  // initialize the ledctrl pin as an output

  pinMode(pirSensor, INPUT);
  pinMode(ledctrl, OUTPUT);

  Serial.begin(115200);
  while (!Serial) ; // Wait for serial port to be available

  Serial.println("Light Sensor Test"); Serial.println("");

  /* Initialise the Lux sensor */
  if (!tsl.begin())
  {
    /* There was a problem detecting the ADXL345 ... check your connections
*/
    Serial.print("Ooops, no TSL2561 detected ... Check your wiring or I2C
ADDR!");
    while (1);
  }

  /* Display some basic information on the Lux sensor */
  displaySensorDetails();

  /* Setup the sensor gain and integration time */
  configureSensor();

  /* We're ready to go! */
  Serial.println("");



/**************************************************************************/
  /*
      Lora module initialisation goes here
  */

/**************************************************************************/

  Serial.println("Init LoRa Module");
  if (!LoRa.begin(868E6)) {
    Serial.println("Starting LoRa failed!");
    while (1);
  }



/**************************************************************************
******/
  /*
      INA219 module initialization
      By default the initialization will use the largest range (32V, 2A).
However
      you can call a setCalibration function to change this range (see
comments).
  */
```

```
/*************************************************************************
****/


  // To use a slightly lower 32V, 1A range (higher precision on amps):
  //ina219.setCalibration_32V_1A();
  // Or to use a lower 16V, 400mA range (higher precision on volts and amps):
  //ina219.setCalibration_16V_400mA();

  //Serial.println("Measuring voltage and current with INA219 ...");
}

/*************************************************************************
****/
/*
     Starting main loop function
*/
/*************************************************************************
****/

void loop() {

  //initiate the LoRa Receive procedure, for reading actions from gateway
  LoRarecieve();

  //Reading values from sensors
  // Serial.println("Reading values from sensors");
  currentresult = CurrentMetering();
  luxresult = LuxMetering();
  MotionDetect();
  //Reading toggle state of led engine function according to command
  if (toggle == "0")
  {
    // Serial.println("Switch off command initiated");
    pwmvalue = 0;
  }

  Else if (toggle == "1")
  {
    // Serial.println("Switch on command initiated");
    LightAutoControl(luxresult);
  }

  analogWrite(ledctrl, pwmvalue);
  pwmpercent = map(pwmvalue, 1, 255, 0, 100);   // change pwm value in %
  pwmpercent = constrain(pwmpercent, 0, 100);   // and limits range of pwm
values to between 0 and
  //100 to prevent exceeding the limits



/*************************************************************************
***/
  /* Delay function of Lora module. Delay Standard function stops the
progress of
```

```
    till delay time exceeds. Using this program keeps running till time
passes
    If time passes data are transmitted
  */

/*********************************************************************
**/

  unsigned long currentMillis = millis();

  if (currentMillis - previousMillis >= interval) {
    previousMillis = currentMillis;

    LoRaSend();

  }
}

void MotionDetect()
{
  // read the state of the pirSensor
  int pirresult = digitalRead(pirSensor);
  // Serial.println(pirresult);

  if (pirresult != lastpirState) {
    // if the state has changed, increment the counter
    if (pirresult == HIGH) {
      // if the current state is HIGH then the state went from off to on:
      pirCount++;
      // Serial.println("on");
      Serial.print("number of Motion Detects: ");
      Serial.println(pirCount);
    } else {
      // if the current state is LOW then the state went from on to off:
      //Serial.println("off");

    }
  }
  lastpirState = pirresult;


}

float CurrentMetering() {

  ina219.begin();


  float shuntvoltage = 0;
  float busvoltage = 0;
  float current_mA = 0;

  float loadvoltage = 0;

  shuntvoltage = ina219.getShuntVoltage_mV();
  busvoltage = ina219.getBusVoltage_V();
  current_mA = ina219.getCurrent_mA();
```

```
  loadvoltage = busvoltage + (shuntvoltage / 1000);
  /*
    Serial.print("Bus Voltage:   "); Serial.print(busvoltage);
Serial.println(" V");
    Serial.print("Shunt Voltage: "); Serial.print(shuntvoltage);
Serial.println(" mV");
    Serial.print("Load Voltage:  "); Serial.print(loadvoltage);
Serial.println(" V");
    Serial.print("Current:       "); Serial.print(current_mA);
Serial.println(" mA");
    Serial.println("");

    Serial.print("Current value is ");
    Serial.println(current_mA);*/

  return current_mA;
}


float LuxMetering()
{
  /* Get a new sensor event */
  sensors_event_t event;
  tsl.getEvent(&event);

  /* Display the results (light is measured in lux) */
  if (event.light)
  {
    //        Serial.print("Lux value is ");
    //        Serial.println(event.light);
  }
  else
  {
    /* If event.light = 0 lux the sensor is probably saturated
       and no reliable data could be generated! */
    //Serial.println("Sensor overload");
  }
  // delay(250);

  return event.light;
}


void LightAutoControl(float luxresult)
{
  pwmvalue = map(luxresult, 1, 200, 255, 0);    //Map the pwm value bettween
0 and 255 regarding the Luminosity
  pwmvalue = constrain(pwmvalue, 0, 255);       // and limits range of pwm
values  between 0 and 255 to prevent
  //exceeding the limits
  //max value of sensor is 17000

  //  Serial.print("PWM value from Lux Calculation is ");
  //  Serial.println(pwmvalue);

  return pwmvalue;
}
```

```
void LoRarecieve()
{
  // try to parse packet
  int packetSize = LoRa.parsePacket();
  if (packetSize) {
    // received a packet
    Serial.println("Received packet from Gateway");
    // read packet
    while (LoRa.available()) {
      loraread = ((char)LoRa.read());
      // Serial.println(loraread);

      toggle = String(loraread);
      Serial.print("Toggle Pin set to ");
      Serial.println(toggle);
      Serial.println("");
      Serial.println(pirresult);

    }
  }
}

void LoRaSend()
{
  Serial.println("Sending Data now");
  Serial.print("Current= ");
  Serial.println(currentresult);
  Serial.print("Lux= ");
  Serial.println(luxresult);
  Serial.print("Motion Detected Number= ");
  Serial.println(pirCount);
  Serial.print("Brightness level(%)= ");
  Serial.println(pwmpercent);
  Serial.println("");


  LoRa.beginPacket();
  LoRa.print("<");
  LoRa.print(luxresult);
  LoRa.print(",");
  LoRa.print(currentresult);
  LoRa.print(",");
  LoRa.print(pirCount);
  LoRa.print(",");
  LoRa.print(pwmpercent);
  LoRa.print(">");
  LoRa.endPacket();
  pirCount = 0;
}
```

Παράρτημα Β

Κώδικας δέκτη δεδομένων από κόμβο LoRa

```
/*
  Created 28 February 2018
  By Kontraros Michail
*/
#include <SPI.h>
#include <LoRa.h>

const int togglepin = 5;
int toggleState = 0;
boolean newData = false;

unsigned long previousMillis = 0;     // will store last time LoRa received
data
const long interval = 5000;           // interval at which receive data from
Artik Gateway

//====================================================================

void setup() {
  Serial.begin(115200);
  while (!Serial);

  Serial.println("Init LoRa Receiver");
  // Declare Frequency
  if (!LoRa.begin(868E6)) {
    Serial.println("Starting LoRa failed!");
    while (1);
  }
  //Declare that togglepin is input
  pinMode(togglepin, INPUT);

}

//====================================================================

void loop() {

//Check for Toggle command in specific interval
  unsigned long currentMillis = millis();

  if (currentMillis - previousMillis >= interval) {

    // save the last time LoRa received data
    previousMillis = currentMillis;

    toggleState = digitalRead(togglepin);

    LoRa.beginPacket();
    LoRa.print(toggleState);
    LoRa.endPacket();
```

```
    // Serial.println("");
    //    Serial.print("Toggle State is: ");
    //    Serial.print(toggleState);
  }
  // try to parse packet
  int packetSize = LoRa.parsePacket();

  if (packetSize) {
    // received a packet
        Serial.println("");

    Serial.println("Received the following packet");
   // Serial.println("");
    // read packet
    while (LoRa.available()) {
      Serial.print((char)LoRa.read());

    }
    // print RSSI of packet
    //    Serial.print("' with RSSI ");
    //    Serial.println(LoRa.packetRssi());
  }
}
```

Παράρτημα Γ

Κώδικας Artik Gateway

```
/*
  Created 28 February 2018
  By Kontraros Michail
*/

#include <Ethernet.h>
#include <PubSubClient.h>


#define ACTUATOR1_NAME   "Toggle_Light"
//#define ACTUATOR2_NAME   "ACTUATOR2_NAME"
//#define ACTUATOR3_NAME   "ACTUATOR3_NAME"

#define sec             5000

char clientId[]          = "My_Artik";
char mqttuser[]          = "A1E-UcDgvuCssO02l6pVfprWCOVU3C0FWn";
char mqttpass[]          = "A1E-UcDgvuCssO02l6pVfprWCOVU3C0FWn";


char sensorTopic[]    = "/v1.6/devices/Artik";
char actuator1Topic[] = "/v1.6/devices/control/switch";


// Ubidots MQTT server
char server[]            = "things.ubidots.com";

char message_buff_sensor[100];

int toggle = 0;
const int togglepin = 11;

char message_buffer[150];

EthernetClient apiClient;

//Serial Parsing params
const byte numChars = 20;
char receivedChars[numChars];
char tempChars[numChars];        // temporary array for use when parsing

// variables to hold the parsed data
//char messageFromPC[numChars] = {0};
int lux = 0;
float current = 0.0;
int pirsensor = 0;
int pwmpercent = 0;

boolean newData = false;
```

```
// handles message arrived on subscribed topic
void callback(char* topic, byte* payload, unsigned int length)
{
  Serial.print("Message arrived [");
  Serial.print(topic);
  Serial.print("] ");
  for (int i = 0; i < length; i++) {
    Serial.print((char)payload[i]);
  }
  Serial.println();

  // here is the value from the actuator
  DebugSerial.print(topic);
  DebugSerial.print(" value: ");
  // DebugSerial.println(string_actuator_value);
  DebugSerial.println(payload[53]);

  // you can use it here accordingly just add or remove based on how manny
actuators you have
  //int actuator_value = payload[53].toInt();

  if (String(topic) == String(actuator1Topic)) {
    //    DebugSerial.print("Toggle_Light: ");
    //    DebugSerial.println(actuator_value);
    toggle = payload[53];

    //  } else if ((String(topic) == String(actuator2Topic))) {
    //    setlevel = actuator_value;
    //
    //    //    DebugSerial.print("Brightness_Control: ");
    //    //    DebugSerial.println(actuator_value);
    //
    //  } else if ((String(topic) == String(actuator3Topic))) {
    //
    //    mctrl = actuator_value;

    //    DebugSerial.print("Manual_Control: ");
    //    DebugSerial.println(actuator_value);

  }
  // client.loop();

}

boolean conn = false;

PubSubClient client(server, 1883, callback, apiClient);

void setup()
{
  // init serial for debugging
  Serial.begin(115200);
  DebugSerial.begin(115200);
  pinMode(togglepin, OUTPUT);


  DebugSerial.println("\nStarting connection to server...");
```

```
  if (client.connect(clientId, mqttuser, mqttpass)) {
    //add or remove based on the number of actuators
    client.subscribe(actuator1Topic);
    //    client.subscribe(actuator2Topic);
    //    client.subscribe(actuator3Topic);
    DebugSerial.println("Successfuly connected and running!");
  } else {
    DebugSerial.println("Connection problem");
  }
}

void loop()

{

  DebugSerial.print("The toggle state is: ");
  DebugSerial.println(toggle);
  DebugSerial.println("");

  if (toggle == 49)
  { digitalWrite(togglepin, HIGH);
  }
  else if (toggle == 48)
  {
    digitalWrite(togglepin, LOW);
  }


  recvWithStartEndMarkers();
  if (newData == true) {
    conn = true;

    strcpy(tempChars, receivedChars);
    // this temporary copy is necessary to protect the original data
    //   because strtok() used in parseData() replaces the commas with \0
    parseData();
    showParsedData();
    newData = false;
  }

  //if client it's not connected or disconnects here we try to reconnect
  if (!client.connected()) {
    DebugSerial.println("reconnecting ...");
    client.connect(clientId, mqttuser, mqttpass)
    //delay(3*sec);
    sendData();

    //add or remove based on the number of actuators
    client.subscribe(actuator1Topic);
    //    client.subscribe(actuator2Topic);
    //    client.subscribe(actuator3Topic);
  }
  sendData();

  // MQTT client loop processing
  client.loop();
```

```
  conn = false;
}

void sendData(void)
{

  if (conn)
  {

    String pubString_sensor = "{\"Pir_Sensor\": " + String(pirsensor) +
",\"Current\": " + String(current) + ",\"Luminosity\": " + String(lux) +
",\"Brightness\": " + String(pwmpercent) + "}" ;

    pubString_sensor.toCharArray(message_buff_sensor,
pubString_sensor.length() + 1);
    DebugSerial.print(pubString_sensor);
    client.publish(sensorTopic, message_buff_sensor);

    conn = false;

  }
  // MQTT client loop processing
  client.loop();
}

//================================================================================

void recvWithStartEndMarkers() {
  static boolean recvInProgress = false;
  static byte ndx = 0;
  char startMarker = '<';
  char endMarker = '>';
  char rc;

  while (Serial.available() > 0 && newData == false) {
    rc = Serial.read();

    if (recvInProgress == true) {
      if (rc != endMarker) {
        receivedChars[ndx] = rc;
        ndx++;
        if (ndx >= numChars) {
          ndx = numChars - 1;
        }
      }
      else {
        receivedChars[ndx] = '\0'; // terminate the string
        recvInProgress = false;
        ndx = 0;
        newData = true;
      }
    }

    else if (rc == startMarker) {
      recvInProgress = true;
    }
  }
```

```
}

//============================================================================

void parseData() {      // split the data into its parts

  char * strtokIndx; // this is used by strtok() as an index

  strtokIndx = strtok(tempChars, ","); // this continues where the previous
call left off
  lux = atoi(strtokIndx);     // convert this part to an integer

  strtokIndx = strtok(NULL, ","); // this continues where the previous call
left off
  current = atoi(strtokIndx);     // convert this part to an integer

  strtokIndx = strtok(NULL, ",");
  pirsensor = atoi(strtokIndx);     // convert this part to a float

  strtokIndx = strtok(NULL, ","); // this continues where the previous call
left off
  pwmpercent = atoi(strtokIndx);     // convert this part to an integer
}

//============================================================================

void showParsedData() {
  DebugSerial.println("Time to parse data");
  DebugSerial.print("Lux is ");
  DebugSerial.println(lux);
  DebugSerial.print("Current is  ");
  DebugSerial.println(current);
  DebugSerial.print("Pir is ");
  DebugSerial.println(pirsensor);
  DebugSerial.print("Brightnes Set Level is at ");
  DebugSerial.print(pwmpercent);
  DebugSerial.println(" %");
  DebugSerial.println("");

}
```

Βιβλιογραφία

[1] «Cloud Computing,» 19 December 2017. [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/Cloud_computing.

[2] «Internet of Things,» 25 December 2017. [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/Internet_of_things.

[3] C. Doukas, Building Internet of Things with the Arduino, USA: CreateSpace Independent Publishing Platform, 2012.

[4] «IOT Analytics,» [Ηλεκτρονικό]. Available: https://iot-analytics.com/top-10-iot-project-application-areas-q3-2016/.

[5] K. SETRAG, «The IoT Magazine,» 29 July 2017. [Ηλεκτρονικό]. Available: https://theiotmagazine.com/iot-digital-transformation-in-insurance-209cac56f496.

[6] J. Rinehart. [Ηλεκτρονικό]. Available: https://jjrinehart.wordpress.com/tag/osi-model/.

[7] «Μοντέλα Αναφοράς OSI,» 26 October 2017. [Ηλεκτρονικό]. Available: https://el.wikipedia.org/wiki/%CE%9C%CE%BF%CE%BD%CF%84%CE%AD%CE%BB%CE%BF_%CE%B1%CE%BD%CE%B1%CF%86%CE%BF%CF%81%CE%AC%CF%82_OSI.

[8] «Τοπολογία δικτύου,» 3 November 2017. [Ηλεκτρονικό]. Available: https://el.wikipedia.org/wiki/%CE%A4%CE%BF%CF%80%CE%BF%CE%BB%CE%BF%CE%B3%CE%AF%CE%B1_%CE%B4%CE%B9%CE%BA%CF%84%CF%8D%CE%BF%CF%85#%CE%94%CE%B9%CE%B1%CF%8D%CE%BB%CE%BF%CF%85.

[9] «Mesh networking,» Wikipedia, 17 December 2017. [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/Mesh_networking.

[10] «NNC3,» [Ηλεκτρονικό]. Available: http://www.nnc3.com/mags/LM10/Magazine/Archive/2009/98/056-059_mesh/article.html.

[11] «Sigfox,» Wikipedia, 21 October 2017. [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/Sigfox.

[12] «NarrowBand IOT,» Wikipedia, 5 December 2017. [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/NarrowBand_IOT.

[13] «Wikipedia,» [Ηλεκτρονικό]. Available: http://e2e.ti.com/support/wireless_connectivity/proprietary_sub_1_ghz_simpliciti/f/156/t/375556.

[14] D. Piromalis, K. Arvanitis και N. Sigrimis, «A DASH7-based WSN Node with IEEE 1451 Compatibility for Precision Farming Applications,» σε *Sustainable Agriculture through ICT innovation*, Torino, 2013.

[15] «Dash7-Alliance,» [Ηλεκτρονικό]. Available: http://www.dash7-alliance.org/.

[16] J. Norair, «Introduction to DASH7 Technologies,» τόμ. 1, p. 22, 2009.

[17] M. Arsalan, A. Umair και V. K. Verma, «Dash7: Performance,» *IOSR Journal of Electronics and Communication Engineering (IOSRJECE),* τόμ. 2, αρ. 5, 2012.

[18] «Texas Instruments,» Texas Instruments, [Ηλεκτρονικό]. Available: http://e2e.ti.com/support/wireless_connectivity/proprietary_sub_1_ghz_simpliciti/f/156/t/375556.

[19] «DASH7,» Wikipedia, 19 December 2017. [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/DASH7.

[20] «Wikipedia,» [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/DASH7.

[21] «LoRa Alliance,» [Ηλεκτρονικό]. Available: https://www.lora-alliance.org.

[22] [Ηλεκτρονικό]. Available: https://learn.adafruit.com/alltheiot-transports/lora-sigfox.

[23] V. Talla, M. Hessar, B. Kellogg, A. Najafi, J. R. Smith και S. Gollakota, «LoRa Backscatter: Enabling The Vision of Ubiquitous Connectivity,» *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies,* τόμ. 1, αρ. 3, 2017.

[24] «Lora Alliance,» November 2015. [Ηλεκτρονικό]. Available: https://www.lora-alliance.org/what-is-lora.

[25] «Πρωτόκολλο Μεταφοράς Υπερκειμένου,» Wikipedia, 5 May 2017. [Ηλεκτρονικό]. Available: https://el.wikipedia.org/wiki/%CE%A0%CF%81%CF%89%CF%84%CF%8C%CE%BA%CE%BF%CE%BB%CE%BB%CE%BF_%CE%9C%CE%B5%CF%84%CE%B1%CF%86%CE%BF%CF%81%CE%AC%CF%82_%CE%A5%CF%80%CE%B5%CF%81%CE%BA%CE%B5%CE%B9%CE%BC%CE%AD%CE%BD%CE%BF%CF%85.

[26] «MQTT,» Wikipedia, 13 December 2017. [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/MQTT.

[27] «HiveMQ,» [Ηλεκτρονικό]. Available: https://www.hivemq.com/blog/how-to-get-started-with-mqtt.

[28] Χ. Σ. Χρίστου, *Πλατφόρμα για Απομακρυσμένο Έλεγχο Οικίας,* Athens: ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ, 2016.

[29] Η. Π. Θεοδώρα, *Πλατφόρμα απομακρυσμένου ελέγχου οχήματος βασισμένου σε διαδικτυακή τεχνολογία,* Athens: ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ, 2013.

[30] «Constrained Application Protocol,» Wikipedia, 18 October 2017. [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/Constrained_Application_Protocol.

[31] «IoT Agenda,» 4 May 2017. [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/Constrained_Application_Protocol.

[32] C. Pereira και A. Aguiar, «Towards Efficient Mobile M2M Communications: Survey and Open Challenges,» *Sensors,* τόμ. 14, αρ. 10, 2014.

[33] «Medium,» [Ηλεκτρονικό]. Available: https://medium.com/mqtt-buddy/mqtt-vs-http-which-one-is-the-best-for-iot-c868169b3105.

[34] «RF Wireless World,» [Ηλεκτρονικό]. Available: http://www.rfwireless-world.com/Terminology/MQTT-vs-HTTP.html.

[35] «Smart Lighting,» Wikipedia, 17 November 2017. [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/Smart_lighting.

[36] A. Bahga και V. Madisetti, Internet of Things: A Hands-On Approach, Arshdeep Bahga, 2014.

[37] A. D. Galasiu και G. R. Newsham, «Energy savings due to occupancy sensors and personal controls: a pilot field study,» σε *11th European Lighting Conference*, Κωνσταντινούπολη, 2009.

[38] B. Abinaya, S. Gurupriya και M. Pooja, «IOT BASED SMART AND ADAPTIVE,» σε *Second International Conference On Computing and Communications Technologies(ICCCT'17)*, Chennai, Tamil Nadu, 2017.

[39] «Amsmartedam City,» [Ηλεκτρονικό]. Available: https://amsterdamsmartcity.com/products/smart-solar-street-lights.

[40] «EnGoPlanet,» 16 December 2017. [Ηλεκτρονικό]. Available: https://www.engoplanet.com/single-post/price-components-Solar-operated-Street-Light.

[41] Whyte Jeanette, T. Hart, K. Yanrong, Z. Lei, C. Cai, G. Yuming, L. Hao, C. Ying και J.

Xiaxia, Comparative Study of Smart Cities in Europe and China, 2014.

[42] *Riga in partnership for development of smart cities,* Tartu: Step Up, 2014.

[43] Samsung, «Samsung Artik,» Samsung, [Ηλεκτρονικό]. Available: https://www.artik.io/modules/artik-520/.

[44] Samsung, «Artik.io,» Samsung, [Ηλεκτρονικό]. Available: https://developer.artik.io/documentation/developer-guide/ide/arduino.html.

[45] Arduino, «Arduino,» Arduino, [Ηλεκτρονικό]. Available: https://www.arduino.cc/.

[46] «Seeedstudio,» [Ηλεκτρονικό]. Available: https://www.seeedstudio.com/Dragino-LoRa-Shield-support-868M-frequency-p-2651.html.

[47] Dragino, «Lora Dragino,» [Ηλεκτρονικό]. Available: http://wiki.dragino.com/index.php?title=Lora_Shield.

[48] sandeepmistry, «Github,» [Ηλεκτρονικό]. Available: https://github.com/sandeepmistry/arduino-LoRa.

[49] «Wikipedia,» 30 January 2017. [Ηλεκτρονικό]. Available: https://el.wikipedia.org/wiki/I%C2%B2C.

[50] Adafruit, «Adafruit,» Adafruit, [Ηλεκτρονικό]. Available: https://www.adafruit.com/product/439.

[51] microbuilder, «Github,» [Ηλεκτρονικό]. Available: https://github.com/adafruit/Adafruit_TSL2561.

[52] Adafruit, «Adafruit,» Adafruit, [Ηλεκτρονικό]. Available: https://www.adafruit.com/product/904.

[53] Driverblock, «Github,» [Ηλεκτρονικό]. Available: https://github.com/adafruit/Adafruit_INA219.

[54] Adafruit, «Adafruit,» Adafruit, [Ηλεκτρονικό]. Available: https://learn.adafruit.com/pir-passive-infrared-proximity-motion-sensor/overview.

[55] Ubidots, « Ubidots,» [Ηλεκτρονικό]. Available: https://ubidots.com/docs/