ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΦΥΣΙΚΗΣ
ΔΙΑΤΜΗΜΑΤΙΚΟ ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
ΣΤΟΝ ΗΛΕΚΤΡΟΝΙΚΟ ΑΥΤΟΜΑΤΙΣΜΟ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

“Καταγραφή και Μεταφορά Δεδομένων για το Internet of Things”

Ευάγγελος Β. Χριστόπουλος

Επιβλέπουσα: **Άννα Τζανακάκη,** Επίκουρη Καθηγήτρια

ΑΘΗΝΑ

Οκτώβριος 2018

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

“Καταγραφή και Μεταφορά Δεδομένων για το Internet of Things”

Ευάγγελος Β. Χριστόπουλος

Α.Μ.: 2014529

ΕΠΙΒΛΕΠΟΝΤΕΣ: **Άννα Τζανακάκη,** Επίκουρη Καθηγήτρια

ΕΞΕΤΑΣΤΙΚΗ ΕΠΙΤΡΟΠΗ: **Άννα Τζανακάκη,** Επίκουρη Καθηγήτρια
Διονύσης Ρέΐσης, Αναπληρωτής Καθηγητής
Έκτορας Νισταζάκης, Αναπληρωτής Καθηγητής

ΠΕΡΙΛΗΨΗ

Σκοπός της παρούσας εργασίας είναι η καταγραφή μετρήσεων με χρήση συστήματος αισθητήρων οι οποίοι έχουν εγκατασταθεί σε πλακέτες Arduino, καθώς και της μεταφοράς, αποθήκευσης και επεξεργασίας των δεδομένων.

Αρχικά, περιγράφεται η αρχιτεκτονική του Internet of Things και οι δυνατότητες του. Το επόμενο κεφάλαιο περιέχει τις υπάρχουσες τεχνολογίες που χρησιμοποιήθηκαν για την υλοποίηση της εργασίας. Αναλύεται ο όρος Big Data και παρατίθενται τα βασικά του χαρακτηριστικά, παρουσιάζεται η διαδικασία της ανάλυσης των δεδομένων, γίνεται πλήρης περιγραφή του MQTT πρωτοκόλλου και παρουσιάζονται οι πλακέτες Arduino Uno και Wemos D1. Στο τέταρτο κεφάλαιο σχεδιάζεται και περιγράφεται το σύστημα καταγραφής και μεταφοράς δεδομένων για το Internet of Things όπως υλοποιήθηκε στην παρούσα εργασία. Στη συνέχεια, αναλύεται ο τρόπος με τον οποίο γίνεται η καταγραφή των μετρήσεων από τις πλακέτες και η αποστολή των δεδομένων. Παρουσιάζεται ο τρόπος σύνδεσης των αισθητήρων και ο τρόπος επικοινωνίας των πλακετών με τον Server. Το επόμενο κεφάλαιο περιέχει τους τρόπους διαχείρισης των δεδομένων. Περιγράφονται μέθοδοι ανάλυσης και οπτικοποίησης τους. Η εργασία ολοκληρώνεται με τα συμπεράσματα που εξήχθησαν και με προτάσεις μελλοντικών επεκτάσεων.

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: Internet of Things, Arduino, MQTT, Big Data, Data Analysis

ABSTRACT

The purpose of this paper is to record measurements using a sensor system installed on Arduino boards, as well as data transfer, storage and processing.

Initially, the Internet of Things architecture and its capabilities are described. The next chapter contains the existing technologies used to implement the work. The Big Data is analyzed and its basic features are presented, the data analysis process, a full description of the MQTT protocol and the Arduino Uno and Wemos D1 boards are presented. In the fourth chapter, the measurement recording system and transfer system is designed and described as implemented in this paper. Next, we analyze how the measurements are recorded from the boards and the data are sent to the Server. Here's how to connect the sensors and how the boards communicate with the Server. The next chapter contains ways of managing data. Methods of analysis and visualization are described. The master thesis concludes with the conclusions drawn and with proposals for future extensions.

KEYWORDS: Internet of Things, Arduino, MQTT, Big Data, Data Analysis

Αφιερώνω την Εργασία αυτή στη σύζυγό μου, Ηρώ.

ΕΥΧΑΡΙΣΤΙΕΣ

Για τη διεκπεραίωση της παρούσας Διπλωματικής Εργασίας, θα ήθελα να ευχαριστήσω την επιβλέπουσα επ. καθ. Άννα Τζανακάκη για τη συνεργασία και την πολύτιμη συμβολή της στα πλαίσια της εκπόνησης της διπλωματικής μου εργασίας στο Μεταπτυχιακό Δίπλωμα Εξειδίκευσης στον Ηλεκτρονικό Αυτοματισμό.

Παράλληλα, θα ήθελα να ευχαριστήσω τον συνάδελφό μου Νικόλαο Γκάτζιο για την βοήθεια του, την υποστήριξή του, αλλά και για την ώθηση που μου έδωσε στην ολοκλήρωση της παρούσας εργασίας. Επίσης θα ήθελα να ευχαριστήσω τη σύζυγό μου, Ηρώ, για τη στήριξη που έδειξε κατά τη διάρκεια της υλοποίησης και εν γένει των σπουδών μου.

Πίνακας Περιεχομένων

Λίστα Εικόνων

Κεφάλαιο 1: Εισαγωγή

1.1 Πρόλογος – Σκοπός

Σκοπός της παρούσας εργασίας είναι η καταγραφή μετρήσεων με χρήση συστήματος αισθητήρων οι οποίοι έχουν εγκατασταθεί σε πλακέτες Arduino, καθώς και της μεταφοράς, αποθήκευσης και επεξεργασίας των δεδομένων.

Αρχικά, στα πρώτα κεφάλαιο παρουσιάζονται οι υπάρχουσες τεχνολογίες οι οποίες χρησιμοποιήθηκαν για την υλοποίηση του σεναρίου χρήσης των αισθητήρων.

Πιο συγκεκριμένα, στο δεύτερο κεφάλαιο της εργασίας γίνεται η περιγραφή του όρου Internet of Things. Παρουσιάζεται η αρχιτεκτονική του IoT και οι δυνατότητες και τα προβλήματα που δημιουργούνται από την ανάπτυξή του. Παράλληλα, περιγράφονται οι τρόποι με τους οποίους το IoT αναμένεται να επηρεάσει κάποιους βασικούς τομείς της καθημερινής μας ζωής.

Στη συνέχεια, στο επόμενο κεφάλαιο της εργασίας εισάγεται ο όρος Big Data και παρατίθενται τα βασικά χαρακτηριστικά που τα διέπουν. Παράλληλα, παρουσιάζεται η Sitewhere πλατφόρμα, μια IoT πλατφόρμα ανοιχτού λογισμικού για την αποστολή και την αποθήκευση των δεδομένων.

Έπειτα γίνεται πλήρης περιγραφή του MQTT Broker, του πρωτόκολλο που χρησιμοποιήθηκε για την ανταλλαγής μηνυμάτων μεταξύ IoT συσκευών. Αναφέρονται τα πλεονεκτήματα του συγκεκριμένου πρωτοκόλλου, ενώ τέλος παρουσιάζονται τα επίπεδα ασφαλείας που διαθέτει. Στο τέλος του κεφαλαίου, παρουσιάζονται οι πλακέτες Arduino Uno και Wemos D1. Περιγράφονται τα χαρακτηριστικά τους, γίνεται μια ιστορική αναδρομή και παρουσιάζεται το περιβάλλον προγραμματισμού των πλακετών αυτών.

Στο τέταρτο κεφάλαιο σχεδιάζεται και περιγράφεται το σύστημα καταγραφής και μεταφοράς δεδομένων μέσω IoT. Παρουσιάζονται οι συσκευές που χρησιμοποιήθηκαν και ο τρόπος λειτουργίας τους. Στο σημείο αυτό θα πρέπει να αναφερθεί οτι το σύστημα καταγραφής και μεταφοράς δεδομένων λειτούργησε την περίοδο από 10/9/17 μέχρι 20/3/18. Στη περίοδο αυτή έγιναν μετρήσεις ανά ένα λεπτό από κάθε πλακέτα και τα δεδομένα αποθηκεύτηκαν στην βάση δεδομένων.

Στο επόμενο κεφάλαιο της εργασίας αναλύεται ο τρόπος με τον οποίο γίνεται η καταγραφή των θερμοκρασιών από τις πλακέτες και η αποστολή των δεδομένων στον MQTT Broker. Περιγράφεται ο τρόπος σύνδεσης των αισθητήρων στις πλακέτες, η διαδικασία σύνδεσής των πλακετών με τον MQTT Broker και ο προγραμματισμός τους.

Το τελευταίο κεφάλαιο περιλαμβάνει μεθόδους και τεχνικές αποθήκευσης, συλλογής, ανάλυσης και επεξεργασίας των δεδομένων. Παρουσιάζει την διαδικασία αποθήκευσης των δεδομένων στην InfluxDB βάση δεδομένων. Παράλληλα, μέσω προγραμμάτων python που κατασκευάστηκαν, γίνεται ανάκτηση δεδομένων και οπτικοποίηση τους σε διαγράμματα.

Η εργασία ολοκληρώνεται με τα συμπεράσματα που εξήχθησαν και με προτάσεις μελλοντικών επεκτάσεων. Τέλος, παρουσιάζεται το Παράρτημα Α, στον οποίο βρίσκονται όλα τα τμήματα κώδικα που δημιουργήθηκαν.

Κεφάλαιο 2: Internet of Things

Το Internet of Things (IoT) ή Διαδίκτυο των Πραγμάτων είναι η επερχόμενη εξέλιξη του Διαδικτύου των υπηρεσιών. Μέχρι τώρα, η πληροφορία που δημιουργείτο και μεταφερόταν μέσα από το Διαδίκτυο, ήταν αποτέλεσμα της αλληλεπίδρασης του ανθρώπου με υπηρεσίες που προσφέρει το Διαδίκτυο. Αυτό που αλλάζει πλέον, είναι ότι αυτόνομες συσκευές-αντικείμενα (IoT συσκευές) μπορούν να παράγουν, να δεχτούν και να επεξεργαστούν πληροφορία την οποία διακινούν μέσα στο Διαδίκτυο.

2.1 Ορισμός του IoT

Αν και δεν υπάρχει ένας ακριβής πρότυπος ορισμός του Internet of Things, παρακάτω παρουσιάζονται διάφοροι ορισμοί του.

Το Διαδίκτυο των Πραγμάτων είναι ένα δίκτυο όχι μόνο υπολογιστών αλλά και διασυνδεδεμένων αντικειμένων τα οποία περιέχουν ενσωματωμένους αισθητήρες, ηλεκτρονικά συστήματα, λογισμικά, με δυνατότητα σύνδεσης, τα οποία μπορούν να ανταλλάσσουν δεδομένα. [1].

Με τον όρο Διαδίκτυο των Πραγμάτων αναφερόμαστε σε ένα σύστημα αλληλένδετων υπολογιστικών συσκευών, μηχανικών και ψηφιακών συσκευών, αντικειμένων, ζώων ή ανθρώπων που διαθέτουν μοναδικά αναγνωριστικά στοιχεία και τη δυνατότητα μεταφοράς δεδομένων μέσω δικτύου χωρίς να απαιτείται αλληλεπίδραση μεταξύ ανθρώπου προς άνθρωπο ή ανθρώπου προς υπολογιστή [2]. Γεγονός που οδηγεί στην επέκταση της συνδεσιμότητας του Διαδικτύου πέρα από τις συνηθισμένες συσκευές (υπολογιστές, smartphones, tablets), σε ένα εύρος συσκευών – αντικειμένων τα οποία χρησιμοποιούν την ενσωματωμένη τεχνολογία για την επικοινωνία και αλληλεπίδραση τους με το εξωτερικό περιβάλλον [3].

Ο όρος αναπτύχθηκε το 1999, με αρχικό σκοπό την ανάπτυξη της RFID τεχνολογίας, η οποία χρησιμοποιείται για την ταυτοποίηση ανθρώπων ή αντικειμένων μέσω ραδιοσυχνοτήτων. Ωστόσο το IoT έγινε ευρέως γνωστό μετά από μια δεκαετία και έφτασε σε εμπορικά επίπεδα το 2014. Ένας απλός ορισμός του θα μπορούσε να είναι: «*Αισθητήρες ενσωματωμένοι σε φυσικά αντικείμενα, τα οποία είναι συνδεδεμένα ενσύρματα ή ασύρματα σε ένα δίκτυο*» [4].

2.2 Εξέλιξη& Εφαρμογές IoT

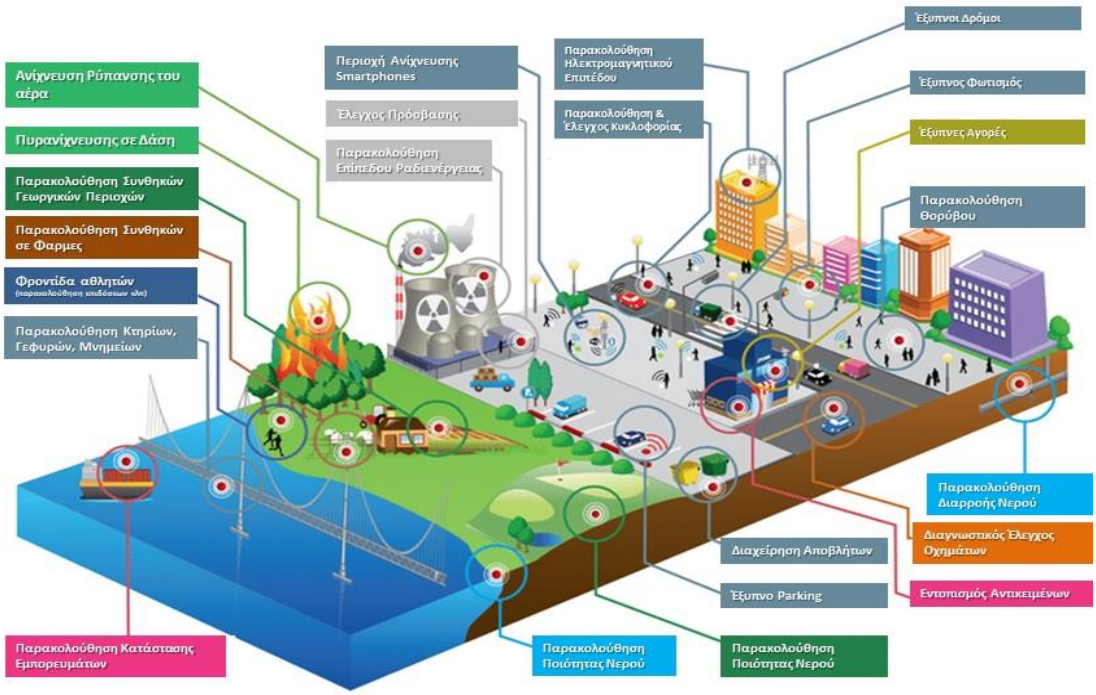
Η γρήγορη διάδοση του Internet of Things οφείλεται στην αυξανόμενη παραγωγή μειωμένου κόστους συσκευών που διαθέτουν δυνατότητα σύνδεσης καθώς και προηγμένους αισθητήρες.

Για παράδειγμα σε ένα έξυπνο σπίτι, συσκευές όπως το ψυγείο, ο φωτισμός, ο κλιματισμός έχουν την δυνατότητα σύνδεσης τους στο διαδίκτυο με αποτέλεσμα να μπορούν να ελεγχθούν από απόσταση χρησιμοποιώντας το κινητό ή τον υπολογιστή. Παράλληλα ορισμένες από αυτές έχουν την δυνατότητα της αυτενέργειας. Για παράδειγμα μια έξυπνη λάμπα μπορεί να ενεργοποιείται μόνη της όταν αντιληφθεί ότι ο φωτισμός στο περιβάλλον είναι χαμηλός.

Έξυπνες πόλεις πλέον χρησιμοποιούν τεχνολογίες IoT για την μείωση της κυκλοφορίας και τον αυτοματοποιημένο φωτισμό των κοινόχρηστων χώρων. Παράλληλα έχουν εγκαταστήσει έξυπνη τεχνολογία στάθμευσης και αισθητήρες για την παρακολούθηση της ποιότητας του αέρα και του θορύβου.

Το Internet of Things αποτελεί μια τεχνολογία που φέρνει τεράστιες αλλαγές στις καθημερινές μας συνήθειες, στις υπηρεσίες υγείας, στις αγορές και στη βιομηχανία. Τα δεδομένα που συγκεντρώνονται, χρησιμοποιούνται για τον έλεγχο και την βελτίωση της απόδοσης, τον εντοπισμό και την πρόβλεψη των αναγκών των ανθρώπων και των οργανισμών πριν καν αυτές εκδηλωθούν.

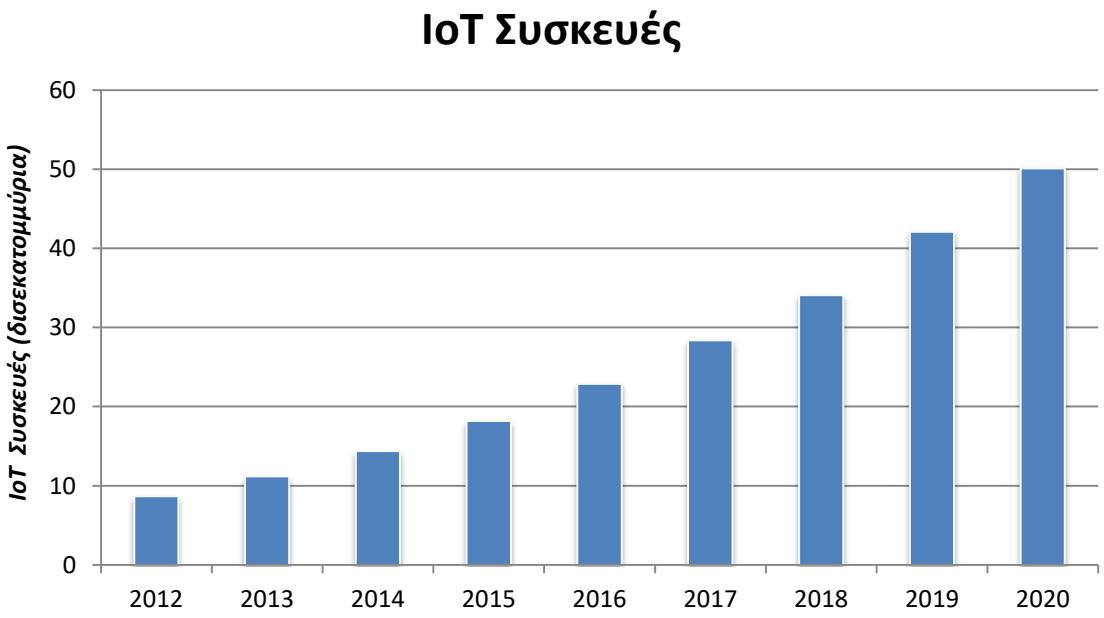
Οι εφαρμογές του Internet of Things είναι πρακτικά απεριόριστες. Αυτοματοποίηση λειτουργιών σε σπίτια και κτίσματα, οι έξυπνες πόλεις, λειτουργίες σχετικές με την φροντίδα υγείας, την αυτοκίνηση και την λιανική πώληση και την βιομηχανία είναι μόνο ορισμένες εφαρμογές του. Στην εικόνα που ακολουθεί παρουσιάζονται ενδεικτικά ορισμένες εφαρμογές του IoT ανά τομέα ανθρώπινης δράσης.



Εικόνα 1 - Ανάπτυξη του IoT [5]

Η ανάπτυξη του IoT είναι τεράστια. Ο όγκος των δεδομένων που δημιουργήθηκαν από IoT συσκευές το 2014 παγκοσμίως, έφτασαν τα 134.5 ZB (περίπου 11.2 ZB ανά μήνα) ενώ η πρόβλεψη για το 2019 είναι ότι θα ξεπεράσει τα 507.5 ZB. [6]

Το IoT αναπτύσσεται με εκθετικό ρυθμό. Το 2016 οι συνδεδεμένες συσκευές ήταν περίπου 25 δισεκατομμύρια, ενώ το 2020 προβλέπεται ότι θα ξεπεράσουν τα 50 δισεκατομμύρια, κάτι το οποίο σημαίνει ότι σε κάθε άνθρωπο θα αντιστοιχούν περίπου 8 IoT συσκευές. [7]



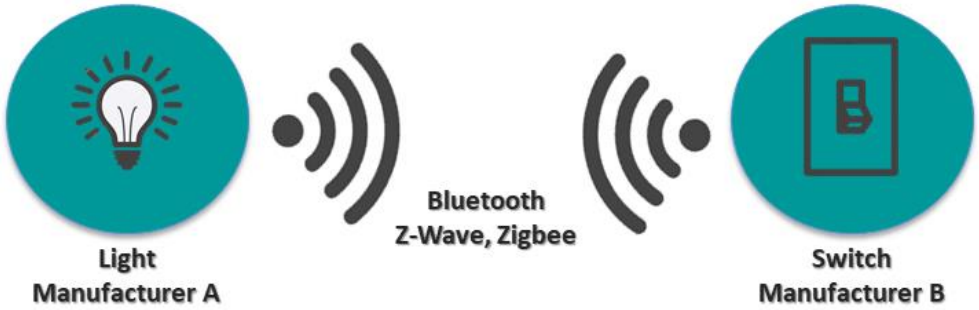
Εικόνα 2 – Ανάπτυξη του Internet of Things[8]

2.3 Μοντέλα Επικοινωνίας του IoT

Στις αρχές του 2015 το Internet Architecture Boars (Συμβούλιο Αρχιτεκτονικής του Διαδικτύου) κυκλοφόρησε ένα κατευθυντήριο αρχιτεκτονικό έγγραφο για την σύνδεση και επικοινωνία των συσκευών, το οποίο αναλύει ένα πλαίσιο τεσσάρων κοινών μοντέλων επικοινωνίας που χρησιμοποιείται από συσκευές IoT.

Device to Device Communications

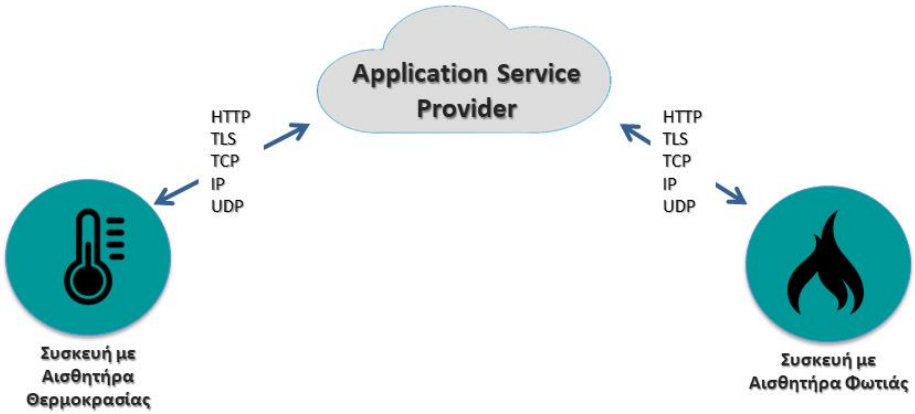
Το μοντέλο επικοινωνίας Device to Device εφαρμόζεται μεταξύ δύο ή περισσότερων συσκευών οι οποίες επικοινωνούν μεταξύ τους απευθείας. Αυτές οι συσκευές μπορούν να συνδεθούν και να επικοινωνήσουν σε πολλούς τύπους δικτύων και τις περισσότερες φορές χρησιμοποιούν πρωτόκολλά όπως Bluetooth, Z-Wave, ή ZigBee. [9] Αυτό το μοντέλο επιτρέπει ανταλλαγή μικρών πακέτων μεταξύ της επικοινωνίας των συσκευών, κάτι που συνεπάγεται χαμηλό Data rate.



Εικόνα 2 - Device to Device Μοντέλο Επικοινωνίας

Device to Cloud Communications

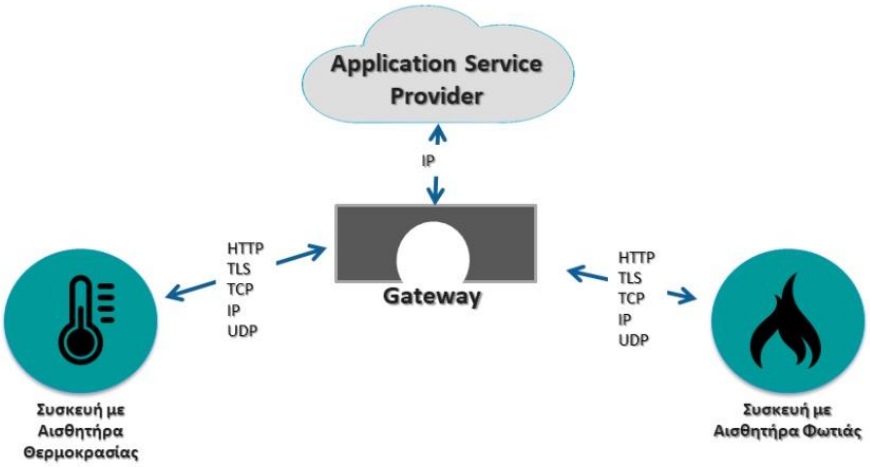
Στο μοντέλο επικοινωνίας Device to Cloud η κάθε συσκευή συνδέεται απευθείας σε μια διαδικτυακή υπηρεσία Cloudόπως ένας πάροχος υπηρεσιών εφαρμογής ο οποίος λαμβάνει τα δεδομένα και διαχειρίζεται την κίνηση τους [10]. Η σύνδεση των συσκευών γίνεται μέσω Ethernet, WiFi ή GSM δίκτυο.



Εικόνα 3 - Device to Cloud Μοντέλο Επικοινωνίας

Device to Gateway Model

Το μοντέλο επικοινωνίας Device to Gateway μοιάζει αρκετά με το Device to Cloud μοντέλο. Η βασική διαφορά είναι ότι κάθε συσκευή δεν συνδέεται άμεσα με την διαδικτυακή υπηρεσία Cloudαλλά μέσω μιας υπηρεσίας Local Gateway η οποία λειτουργεί ως ενδιάμεση πύλη για την επίτευξη της σύνδεσης μεταξύ συσκευής ή συσκευών και Cloud. Μέσω του Local Gateway προσφέρεται μεγαλύτερη ασφάλεια στη σύνδεση και δυνατότητα χρήση διαφορετικών τεχνολογιών και πρωτοκόλλων.[11]

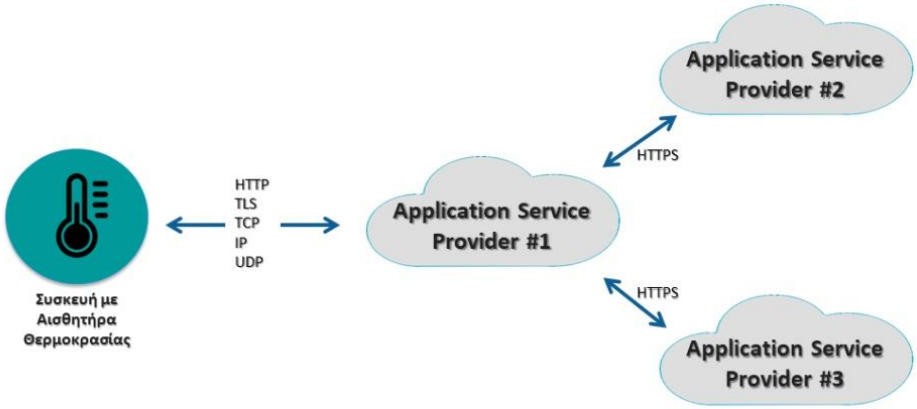


Εικόνα 4 - Device to Gateway Μοντέλο Επικοινωνίας

Μοντέλο Back-EndData-Sharing

Οι συσκευές IoT στέλνουν τα δεδομένα τους μόνο σε μια διαδικτυακή υπηρεσία Cloud εφαρμογών. Ωστόσο συχνά οι χρήστες απαιτούν τη δυνατότητα εξαγωγής και ανάλυσης δεδομένων σε συνδυασμό με δεδομένα από άλλες πηγές. Το μοντέλο επικοινωνίας Back-End Data-sharing στηρίζεται σε μια αρχιτεκτονική η οποία δίνει την δυνατότητα στους χρήστες να εξάγουν και να αναλύουν τα δεδομένα κάθε συνδεδεμένης συσκευής από μια υπηρεσία Cloud, χωρίς αναγκαστικά να είναι η υπηρεσία Cloud στην οποία η συσκευή έστειλε τα δεδομένα. [12]

Με το συγκεκριμένο μοντέλο ξεπερνιούνται οι περιορισμοί της single device to cloud σύνδεσης. Τα δεδομένα ενώ στέλνονται σε μια υπηρεσία cloud, είναι προσβάσιμα και από άλλες συνδεδεμένες υπηρεσίες Cloud.



Εικόνα 5 - Μοντέλο Back-End Data-Sharing Μοντέλο Επικοινωνίας

2.4 Πλεονεκτήματα του IoT & Προβληματισμοί

Με την χρήση συσκευών IoT επιτυγχάνεται δυναμικός έλεγχος και αυτοματοποιημένη ανάδραση στην καθημερινή μας ζωή. Παράλληλα, αυξάνει την αποτελεσματικότητα και οδηγεί σε εξοικονόμηση χρόνου και ενέργειας. Ο άνθρωπος μπορεί να συνδέει και να ελέγχει καθημερινές συσκευές του σπιτιού του και της καθημερινότητας του (π.χ. air-conditions, πρίζες, λάμπες, έλεγχος κατάστασης αυτοκινήτου, έλεγχος κτιριακών εγκαταστάσεων κλπ.) απευθείας από το τηλέφωνο, την τηλεόραση ή τον υπολογιστή του. Κατά συνέπεια, η βέλτιστη αξιοποίηση πόρων και ενέργειας οδηγεί σε εξοικονόμηση χρημάτων.

Ο μεγάλος όγκος δεδομένων επιτρέπει την σωστή πρόβλεψη μελλοντικών δεδομένων και καταστάσεων. Ένα σύστημα χρησιμοποιώντας IoT συσκευές μπορεί να γνωρίζει την μελλοντική κατάσταση που θα βρεθεί και να λάβει ανάλογες αποφάσεις.

Από την άλλη πλευρά, η διασύνδεση τόσων συσκευών και υπηρεσιών δημιουργεί ορισμένους προβληματισμούς.

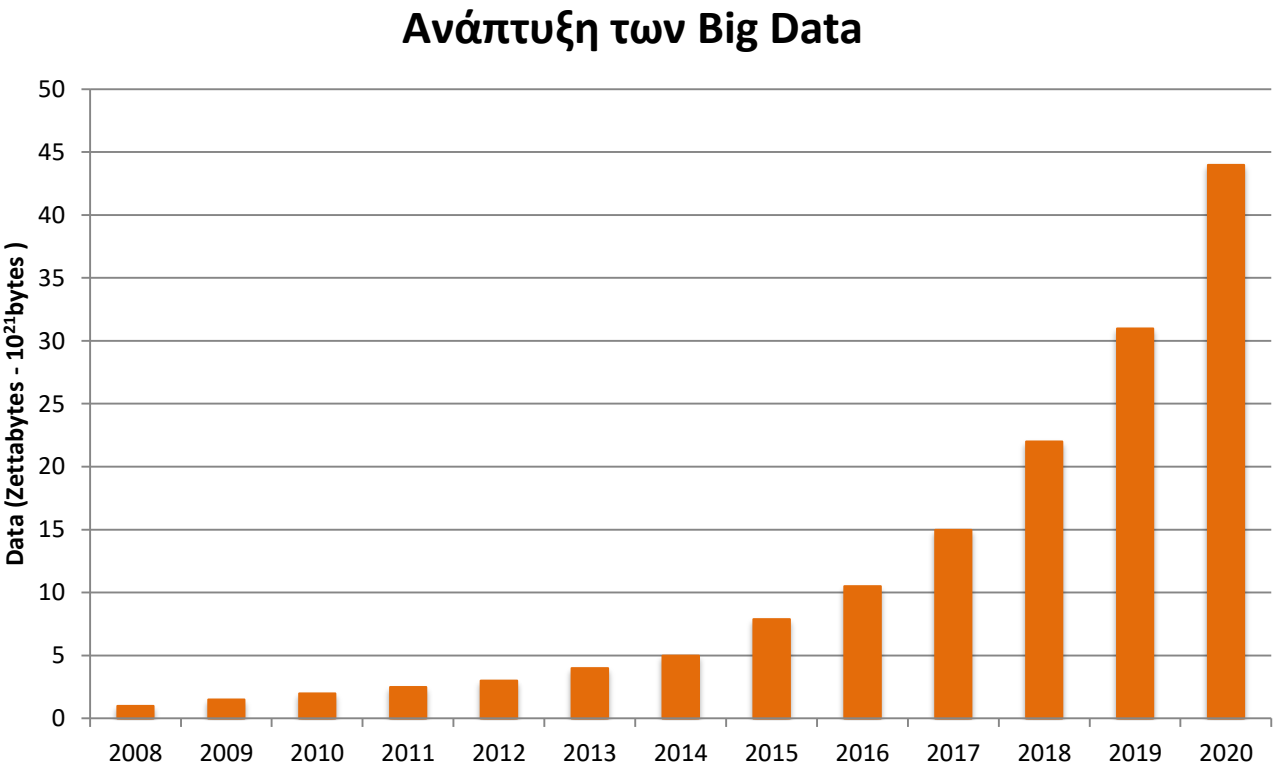
Όπως έχει ήδη αναφερθεί, οι IoT συσκευές αυξάνονται εκθετικά, το IoT να γίνεται ολοένα και περισσότερο αναπόσπαστο κομμάτι της ζωής μας, κάτι που οδηγεί στη δημιουργία μιας νέας οντότητας μεταξύ της ανθρώπινης κοινωνίας και των φυσικών συστημάτων. Πρέπει να δεχτούμε ότι η τεχνολογία παίρνει σημαντική θέση στον έλεγχο της ζωής. Η ζωή μας θα ελέγχεται όλο και περισσότερο από την τεχνολογία, και θα εξαρτάται από αυτήν.

Παράλληλα, η αποθήκευση και η διαχείριση του τεράστιου όγκου δεδομένων που παράγουν οι IoT συσκευές είναι μεγάλο κεφάλαιο. Η ασφάλεια των δεδομένων καθώς και η προστασία προσωπικών δεδομένων είναι ένα μεγάλο ζήτημα για το IoT. Επιπλέον, το IoT αποτελείται από τεράστιο αριθμό ετερογενών συσκευών, γεγονός που το καθιστά ευάλωτο και επιρρεπές σε επιθέσεις, ενώ δημιουργείται και το ζήτημα της συμβατότητας των διάφορων IoT συσκευών.

Κεφάλαιο 3: Θεωρητικό Υπόβαθρο και Υπάρχουσες Τεχνολογίες

3.1 Big Data

Ο όγκος των δεδομένων αυξάνεται με εκθετικούς ρυθμούς. Για να συνειδητοποιήσουμε αυτή την ραγδαία αύξηση αρκεί να αναλογιστούμε ότι τα παραγόμενα δεδομένα παγκοσμίως το 2005 ήταν 130 exabytes (10^{18} bytes), το 2015 αυξήθηκαν σε 7.910 exabytes το 2015, ενω οι προβλέψεις για το 2020 προβλέπεται να ξεπεράσουν τα 44 zettabytes (10^{21} bytes). [13]



Εικόνα 6 - Ανάπτυξη Big Data

3.1.1 Ορισμός των Big Data

Ο όρος Big Data χρησιμοποιείται είτε για να προσδιορίσουμε ένα σύνολο δεδομένων μεγάλου όγκου είτε σε σύνθετα δεδομένα, όπου η συνήθης επεξεργασία τους είναι ανεπαρκής. Ορισμένες πηγές αναφέρουν ότι μία συλλογή δεδομένων χαρακτηρίζεται ως Big Data όταν το μέγεθος τους είναι της τάξεως των petabytes (10^{15}bytes), exabytes (10^{18} bytes) η zettabytes (10^{21} bytes). Ωστόσο μια τέτοια ποσοτικοποίηση για τον ορισμό του όρου Big Data, θα μας οδηγούσε σε λάθος συμπεράσματα, αφού ο όγκος δεν είναι το μόνο σημαντικό χαρακτηριστικό των Big Data.

Για το λόγο αυτό Big Data θεωρείται μία συλλογή δεδομένων τόσο μεγάλη ή τόσο ετερογενής και σύνθετη ως προς τη δομή της, ώστε τα παραδοσιακά λογισμικά επεξεργασίας δεδομένων είναι αδύνατον να τη διαχειριστούν.

3.1.2 Χαρακτηριστικά των Big Data

Ένας από τους πιο γνωστούς ορισμούς για τα Big Data διατυπώθηκε πολύ εύστοχα από την Gartner με βάση το αποκαλούμενο Μοντέλο 3V. Σύμφωνα με αυτόν, τρία είναι τα κύρια χαρακτηριστικά των μεγάλων δεδομένων: ο όγκος (Volume), η ταχύτητα (Velocity) και η ποικιλομορφία (Variety). Ωστόσο το συγκεκριμένο μοντέλο επεκτάθηκε σε Μοντέλο 4V λαμβάνοντας υπόψη το χαρακτηριστικό της εγκυρότητας (Veracity). [14]

Παρακάτω, αναλύονται όλα τα χαρακτηριστικά τα οποία χρησιμοποιούνται ως τρόποι διάκρισης και κατηγοριοποίησης των Big Data.

Όγκος (Volume)

Η ποσότητα των παραγόμενων και αποθηκευμένων δεδομένων. Το μέγεθος των δεδομένων καθορίζει την αξία και τη δυνητική γνώση αλλά και το αν θα μπορούν να θεωρηθούν BigData ή όχι.

Ποικιλία (Variety)

Ο τύπος και η φύση των δεδομένων είναι ένα βασικό χαρακτηριστικό τους. Βοηθάει τα άτομα που το αναλύουν να χρησιμοποιούν αποτελεσματικά την προκύπτουσα γνώση.

Ταχύτητα (Velocity)

Η ταχύτητα με την οποία τα δεδομένα καταφθάνουν, αποθηκεύονται, επεξεργάζονται και τέλος αναλύονται ώστε να παρθούν αποφάσεις βάσει αυτών.

Εγκυρότητα (Veracity)

Η ποιότητα των δεδομένων μπορεί να ποικίλει σημαντικά, επηρεάζοντας την ακριβή ανάλυση τους. Τα παραδοσιακά συστήματα διαχείρισης δεδομένων αποδέχονται την εγκυρότητα των δεδομένων τους και αρκετές φορές λαμβάνουν υπόψη τους δεδομένα τα οποία δεν είναι ούτε ακριβή ή ούτε ορθά.

Λόγω των πολλών και διαφορετικών μορφών των BigData, η ποιότητα και η ακρίβεια είναι λιγότερο ελεγχόμενες παράμετροι. Η κύρια υπόσχεση των Big Data είναι η λήψη καλύτερων αποφάσεων βασιζόμενοι σε δεδομένα.

Αξία (Value)

Η πρόσβαση σε Big Data πρέπει να μπορεί να μετατραπεί σε αξία, αλλιώς δεν είναι χρήσιμη. Η αξία είναι ένα σημαντικό χαρακτηριστικό των Big Data το οποίο μετράται τόσο με χρήση δεικτών, όσο και με στατιστικές αναλύσεις.

Μεταβλητότητα (Variability)

Η μεταβλητότητα των Big Data παρατηρείται όταν τα δεδομένα ποικίλουν σημαντικά, επηρεάζοντας την ακριβή ανάλυση και αξιοποίησης τους.

3.2 Ανάλυση Δεδομένων

Ο μεγάλος όγκος δεδομένων που δημιουργείται καθημερινά δεν θα μπορούσε να δώσει χρήσιμες πληροφορίες για την λήψη αποφάσεων, χωρίς πρώτα να επεξεργαστεί. Η ανάλυση των δεδομένων είναι πιο επιτακτική από ποτέ. Μέσω της διαδικασίας αυτής, μπορούν να εξαχθούν σημαντικά συμπεράσματα από μεγάλου όγκου δεδομένα.

Η Ανάλυση Δεδομένων [15] (Data Analysis), αποτελεί μια διαδικασία απόκτησης, παρατήρησης, επεξεργασίας και μοντελοποίησης των δεδομένων με στόχο την εξεύρεση χρήσιμης πληροφορίας για την υποστήριξη διαφόρων ειδών λήψεων αποφάσεων. Στην ουσία, είναι ένα σύνολο μεθόδων και τεχνικών με σκοπό την εξαγωγή γνώσης και πληροφορίας μέσα από μοτίβα που προκύπτουν από τα δεδομένα.

Οι τεχνικές ανάλυσης δεδομένων μπορούν να ομαδοποιηθούν στις παρακάτω κατηγορίες:

Data fusion (συγχώνευση δεδομένων): Είναι οι τεχνικές που χρησιμοποιούνται για την συλλογή και συγχώνευση δεδομένων από διαφορετικές πηγές με σκοπό την περαιτέρω επεξεργασία τους.

Data mining («εξόρυξη» δεδομένων): Είναι οι τεχνικές που εξετάζουν τα δεδομένα για τον εντοπισμό τυχών έγκυρων, χρήσιμων και κατανοητών ακολουθιών – μοτίβων (patterns) ή σχέσεων μεταξύ των δεδομένων ώστε να επιτευχθεί γρηγορότερα και αποτελεσματικότερα η εξαγωγή συμπερασμάτων.

Optimization (βελτιστοποίηση): Είναι οι τεχνικές που χρησιμοποιούνται για την βελτίωση της απόδοσης του συστήματος από το οποίο προέρχονται τα δεδομένα, σύμφωνα με ένα πλήθος παραμέτρων.

Visualization (απεικόνιση): Είναι οι τεχνικές μέσω των οποίων τα αποτελέσματα της ανάλυσης των δεδομένων αναπαρίστανται σε εικόνες και διαγράμματα. Οι τεχνικές αυτές είναι χρήσιμες αφού δίνουν μία πιο εύκολη και κατανοητή μορφή στα δεδομένα.

3.2.1 Εξόρυξη Δεδομένων

Οι τεχνικές Εξόρυξης Δεδομένων (Data Mining), χρησιμοποιούν ένα μεγάλο αριθμό αλγόριθμων και επιτρέπουν μέσω της εξέτασης των δεδομένων, τον εντοπισμό μοτίβων, τα

οποία ενδεχομένως οδηγούν σε κάποιο αποτέλεσμα και κατ' επέκταση στη λήψη αποφάσεων [16].

Οι αλγόριθμοι αυτοί χρησιμοποιούν ορισμένες από τις παρακάτω διαδικασίες :

Classification (Ταξινόμηση): ταξινόμηση των δεδομένων ή των συμβάντων σύμφωνα με κατηγορίες.

Clustering (Ομαδοποίηση): ομαδοποίηση δεδομένων που ακολουθούν παρόμοια patterns.

Prediction (Μοντέλα πρόβλεψης): εντοπισμός πιθανών σχέσεων μεταξύ των στοιχείων, βάσει μοντέλων πρόβλεψης ή ανάλυση παλινδρόμησης (Regression Analysis).

Association (Σχέσεις μεταξύ των δεδομένων): εντοπισμός σχέσεων μεταξύ των τιμών των δεδομένων από μια ή περισσότερες συλλογές.

Anomaly detection (Εντοπισμό ανωμαλιών): εντοπισμός των ακραίων τιμών ή περιπτώσεις διακοπής μίας ακολουθίας μέσα σε ένα data set.

Περίληψη: καταγραφή και παρουσίαση των διαφόρων χαρακτηριστικών ή patterns που εμφανίζονται σε μια ή περισσότερες συλλογές δεδομένων.

3.2.2 Απεικόνιση Δεδομένων

Η δημιουργία μεγάλου όγκου δεδομένων έφερε την ανάγκη για τρόπους απεικόνισης τους ώστε να γίνονται αντιληπτές οι σχέσεις και οι δομές που κρύβονται μέσα σε αυτά. Η διαδικασία της μετατροπής των αριθμητικών δεδομένων σε εικόνα ονομάζεται Οπτικοποίηση Δεδομένων (Data Visualization) [17].

Η διαδικασία αυτή έχει σκοπό την ερμηνεία των πληροφοριών και των σχέσεων που περιέχονται σε αυτά. Εκμεταλλεύεται το γεγονός ότι ο άνθρωπος αντιλαμβάνεται πολύ πιο εύκολα και γρήγορα την ύπαρξη μιας πληροφορίας όταν τα δεδομένα παρουσιάζονται ως σημεία ή σχήματα στο χώρο παρά ως αριθμητικές τιμές σε έναν πίνακα.

Η οπτικοποίηση δεδομένων εφαρμόζεται εύκολα σε παρατηρήσεις που εξαρτώνται από δύο ή τρείς μεταβλητές αφού κάθε μεταβλητή αντιστοιχεί σε μια διάσταση του χώρου. Σε περιπτώσεις που υπάρχουν περισσότερες από τρεις μεταβλητές, η διαδικασία οπτικοποίησης γίνεται πιο σύνθετο πρόβλημα. Οι μεταβλητές δεν μπορούν να οπτικοποιηθούν στις τρεις διαστάσεις του χώρου, όπου γίνονται αντιληπτές από το ανθρώπινο μάτι. Για τον λόγο αυτό οι πολλές διαστάσεις κωδικοποιούνται με τέτοιο τρόπο ώστε να μπορούν να αναπαρασταθούν στη συνέχεια στο δισδιάστατο ή στο τρισδιάστατο χώρο.

3.3 Internet of Things Platforms

3.3.1 Γενικά

Η ανάγκη των ανθρώπων να παρακολουθούν δεδομένα από απόσταση, ταυτόχρονα, από διάφορους τύπους αισθητήρων και σε πραγματικό χρόνο, ενώ μπορούν να επεξεργάζονται και να αναλύουν τα δεδομένα αυτά για την δημιουργία χρήσιμων συμπερασμάτων, ανάγκασε πολλές εταιρίες στον χώρο του IoT και του Υπολογιστικού Νέφους να υιοθετήσουν IoT πλατφόρμες. Οι πλατφόρμες αυτές είναι εργαλεία τα οποία δίνουν την δυνατότητα στους χρήστες να μπορούν να παρακολουθούν, να αποθηκεύουν, να ανακτούν, και να επεξεργάζονται μεγάλου όγκου διαφορετικά δεδομένα από πολλούς και διαφορετικούς αισθητήρες.

3.3.2 SiteWhere Platforms

Στην παρούσα εργασία χρησιμοποιήθηκε η πλατφόρμα SiteWhere [18], η οποία είναι ανοιχτού κώδικα και προσφέρει όλες τις δυνατότητες παρακολούθησης και αντιμετώπισης συμβάντων, διαμοίρασης και αποθήκευσης δεδομένων. Είναι ένα ολοκληρωμένο σύστημα διαχείρισης στο οποίο μπορούν να συνδεθούν και να ελεγχθούν ένας μεγάλος αριθμός συσκευών. Η επικοινωνία με τις συσκευές είναι αμφίδρομη και ασύγχρονη ή σε πραγματικό χρόνο, επιτρέποντας στο SiteWhere να διαχειρίζεται λύσεις που απαιτούν πολύ μεγάλο αριθμό συσκευών και κύκλων συμβάντων.

Χρησιμοποιεί αρκετές δημοφιλείς λύσεις ανοιχτού κώδικα που έχεις ως αποτέλεσμα την ενοποίηση αλλά κυρίως την απλοποίηση και ασφάλεια λειτουργιών. Ορισμένες από τις ενσωματωμένες λύσεις που περιέχει είναι [19]:

- MongoDB, η προεπιλεγμένη βάση δεδομένων
- Mosquitto, ο προεπιλεγμένος MQTT Broker
- InfluxDB, Βάση Δεδομένων για δεδομένα χρονοσειράς
- Apache Spark, για κατανεμημένη επεξεργασία μεγάλου όγκου δεδομένων
- Twilio client, για ειδοποιήσεις SMS μέσω του Cloud
- Eclipse Californium, για ανταλλαγή μηνυμάτων CoAP

Το SiteWhere υποστηρίζει ένα πλήθος καθιερωμένων πρωτοκόλλων IoT (MQTT, AMQP, STOMP) παρέχοντας παράλληλα δημοσιευμένα API(Application Programming Interface) για κάθε εφαρμογή συγκεκριμένων αναγκών. Μπορεί επίσης να επικοινωνεί με εξωτερικές υπηρεσίες Cloud όπως το dweet.io για εξωτερικά μηνύματα και το InitialState για προηγμένη οπτικοποίηση δεδομένων.

3.4 MQTT

Το MQTT (Message Queuing Telemetry Transport) είναι πρωτόκολλο μετάδοσης μηνυμάτων πάνω από το TCP/IP, με τη χρήση ενός μηχανισμού (publish/subscribe), όπου ο ενδιάμεσος (MQTT Broker) μπορεί να μοιράζει τα μηνύματα σε περισσότερους από έναν αποδέκτες [20].

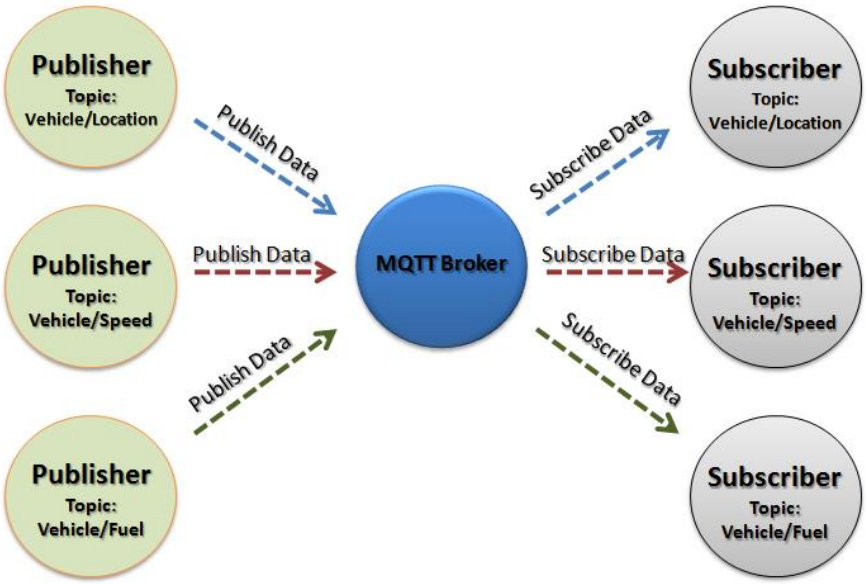
Είναι ένα πρωτόκολλο ανταλλαγής μηνυμάτων μεταξύ συσκευών σε low bandwidth για την μεταφορά δεδομένων χωρίς καθυστερήσεις. Η επιδίωξη του σχεδιασμού του είναι η ελαχιστοποίηση του εύρους ζώνης του δικτύου καθώς και των απαιτήσεών σε πόρους .

3.4.1 MQTT Broker, Client και η σύνδεση τους

Το MQTT επιτρέπει στις συσκευές να στέλνουν δημόσια πληροφορίες σχετικά με ένα θέμα (topic). Σε αντίθεση με το μοντέλο client-server όπου οι δύο συσκευές επικοινωνούν απευθείας, η επικοινωνία μεταξύ clients γίνεται μέσω του MQTT Broker. Χιλιάδες συσκευές (clients) μπορούν να συνδέονται στον MQTT Broker, είτε για να στέλνουν, είτε για να λαμβάνουν δεδομένα, είτε και για τα δύο συγχρόνως [21].

Κάθε client δεν γνωρίζει τις συσκευές που είναι συνδεδεμένες στον MQTT Broker, ενώ ο Broker γνωρίζει την ύπαρξη όλων των clients που είναι συνδεδεμένοι σε αυτόν. Οι clients που στέλνουν δεδομένα δεν χρειάζεται να γνωρίζουν οτιδήποτε για τους λήπτες, αφού την διαχείριση και προώθηση των μηνυμάτων ανήκει αποκλειστικά στον MQTT Broker.

Για παράδειγμα ένα όχημα στέλνει δεδομένα για την τοποθεσία, την ταχύτητα και το επίπεδο καυσίμου του χρησιμοποιώντας τρία διαφορετικά topics. Τα δεδομένα μεταφέρονται στον MQTT Broker και από εκεί μοιράζονται στις συσκευές που έχουν εγγραφεί στα αντίστοιχα topics.



Εικόνα 7 – Clients & MQTT Broker

Ένας client μπορεί να είναι από ένας μικροελεγκτής μέχρι ένα πλήρες υπολογιστικό σύστημα και είτε να στέλνει, είτε να λαμβάνει δεδομένα από τον MQTT Broker. Η σύνδεσή κάθε client με τον MQTT Broker μπορεί να γίνεται μέσα από οποιοδήποτε δίκτυο.

3.4.2 Publish/Subscribe/Usubscribe

Το κύριο μοντέλο που χρησιμοποιείται για την επικοινωνία είναι το publish και/ή το subscribe. Κάθε client για να δημοσιεύσει οποιαδήποτε πληροφορία, συνδέεται με τον MQTT Broker και χρησιμοποιώντας ένα συγκεκριμένο topic, αποστέλλει την πληροφορία στον MQTT Broker (**Publish**). Από το σημείο αυτό, και αφού ο MQTT Broker λάβει το μήνυμα, όλοι οι clients που έχουν εγγραφεί στο συγκεκριμένο topic (**Subscribe**), λαμβάνουν την πληροφορία [19].

Για να σταματήσει ένας client να δέχεται πληροφορίες από τον MQTT Broker, για ένα συγκεκριμένο topic στο οποίο είχε ήδη εγγραφεί, αρκεί να διαγράψει το subscription αυτό. (**Unsubscribe**).

3.4.3 Τα Οφέλη του Πρωτόκολλου MQTT

Το MQTT είναι ένα ελαφρύ πρωτόκολλο. Τα μηνύματα που ανταλλάσσονται έχουν μικρό μέγεθος κάτι που επιτυγχάνεται τόσο με τα μικρού όγκο δεδομένα που μεταφέρονται σε κάθε μήνυμα, αλλά και με τις μικρού μεγέθους κεφαλίδες που έχουν αυτά. Το κάθε μήνυμα MQTT έχει κεφαλίδα μεγέθους που ποικίλει από μόλις 2 μέχρι 5 bytes. Τα πρώτα 4 bits αντιπροσωπεύουν τον τύπο του μηνύματος και τα επόμενα 4 bits περιέχουν συγκεκριμένα control flags.

Επίσης, θα πρέπει να αναφερθεί ότι το πρωτόκολλο MQTT είναι ασύγχρονο. Δεν απαιτείται η ταυτόχρονη λειτουργία όλων των clients και του MQTT Broker.

Τέλος, ένα μεγάλο πλεονέκτημα του Πρωτόκολλου MQTT είναι η κλιμάκωσή του. Όλες οι λειτουργίες του MQTT Broker είναι υψηλά παραλληλοποιήσιμες, με αποτέλεσμα να επιτρέπει την κλιμάκωση στις εφαρμογές του.

3.4.4 Ασφάλεια σύνδεσης.

Το πρωτόκολλο MQTT δεν προσφέρει υψηλά επίπεδα ασφάλειας. Παρόλα αυτά, υπάρχουν ορισμένοι μηχανισμοί που θα μπορούσαν να καταστήσουν το πρωτόκολλο πιο ασφαλές [22].

Ένας από τους βασικότερους μηχανισμούς είναι η **πιστοποίηση** κατά την σύνδεση του client με τον Broker. Με την δυνατότητα αυτή, μόνο οι πιστοποιημένοι clients μπορούν να συνδεθούν με τον MQTT Broker και κατ' επέκταση να γίνουν είτε publishers είτε subscribers.

Άλλος ένα μηχανισμός είναι η **εξουσιοδότηση** του client. Με αυτό τον τρόπο μπορούμε να εξουσιοδοτήσουμε συγκεκριμένους clients για το αν και σε ποια topics θα έχουν πρόσβαση, αλλά και για το αν θα μπορούν να γίνουν publishers ή subscribers σε συγκεκριμένα topics.

Όλα τα MQTT μηνύματα μεταφέρονται από το client στον MQTT Broker ή αντίστροφα μέσα στο δίκτυο. Σε όλη τη διαδρομή τα μηνύματα αυτά μπορούν εύκολα να διαβαστούν και να τύχουν κακόβουλης επεξεργασίας, αφού στέλνονται σε απλό κείμενο. Για την αποφυγή τέτοιων κινδύνων μπορούν να χρησιμοποιηθούν πρωτόκολλα **κρυπτογράφησης** όπως το TLS (Transport Layer Security) και SSL (Secure Sockets Layer). Με τα συγκεκριμένα πρωτόκολλα τα δεδομένα μεταφέρονται κρυπτογραφημένα και η σύνδεση μεταξύ σε client και MQTT μπορεί να θεωρηθεί ασφαλής.

3.5 Arduino & Sensors

Για την υλοποίηση της παρούσας εργασίας και πιο συγκεκριμένα για την καταγραφή δεδομένων, επιλέχθηκαν οι πλακέτες Arduino Uno και Wemos συνοδευόμενες από τους αντίστοιχους αισθητήρες.

Το Arduino είναι μία open source πλακέτα με ενσωματωμένο μικροελεγκτή η οποία έχει την δυνατότητα να συνδέεται με μονάδες εισόδου και εξόδου, είτε αναλογικές, είτε ψηφιακές. Είναι μια πλακέτα εύκολη στην χρήση και εξίσου εύκολη στο προγραμματισμό της. Το Arduino μπορεί να χρησιμοποιηθεί για την ανάπτυξη ανεξάρτητων διαδραστικών αντικειμένων αλλά και να συνδεθεί με υπολογιστή μέσω προγραμμάτων Processing, Pure data, Max/Msp [23].

Θετικό στοιχείο του Arduino, είναι η υλική επεκτασιμότητα, αφού μπορούμε να τοποθετήσουμε πάνω από αυτήν μία ή περισσότερες πλακέτες (shields) της εταιρείας αλλά και πλακέτες που μπορούμε να σχεδιάσουμε μόνοι μας (prototype).

3.5.1 Ιστορία πλακέτας Arduino

Η πλακέτα Arduino δημιουργήθηκε προκειμένου να κατασκευαστεί μια συσκευή για τον έλεγχο προγραμμάτων διαδραστικών σχεδίων από μαθητές, η οποία θα κόστιζε πολύ πιο φθηνά από άλλα πρωτότυπα συστήματα διαθέσιμα ως εκείνη την περίοδο.

Από το 2005, όπου και δημιουργήθηκε η πρώτη πλακέτα Arduino, έχουν κατασκευαστεί αρκετά μοντέλα, ορισμένα από τα οποία αναφέρονται παρακάτω [24]. Το καθένα διαθέτει τα δικά του χαρακτηριστικά, που το καθιστούν ιδανικό για συγκεκριμένες χρήσεις. Η ποικιλία αυτή φιλοδοξεί στην κάλυψη των διαφορετικών αναγκών των χρηστών τους.

Έτος	Μοντέλο	Μικροελεκτής
2006	Arduino Mini	ATmega168
2007	LilyPad Arduino	ATmega168V ή ATmega328V
2008	Arduino Nano	ATmega328
2010	Arduino Uno	ATmega328P
2010	Arduino Mega2560	ATmega2560
2011	Arduino Ethernet	ATmega328
2012	Arduino Leonardo	Atmega32U4
2012	Arduino Due	ATSAM3X8E
2012	Arduino Micro	ATmega32U4
2013	Arduino Yún	Atmega32U4, Atheros AR9331
2015	Arduino 101	Intel® Curie™ module two tiny cores, x86 &ARC
2016	Arduino / Genuino MKR1000	ATSAMW25

3.5.2 Χαρακτηριστικά Πλακέτας Arduino Uno

Το Arduino Uno περιέχει τον μικροελεγκτή ATmega338, ενώ διαθέτει 14 ψηφιακούς και 6 αναλογικούς ακροδέκτες [25]. Η επικοινωνία της πλακέτας και του υπολογιστή γίνεται μέσα από τον Serial over USB ελεγκτή. Ο ελεγκτής αυτός, προσφέρει μια σειριακή επικοινωνία μεταξύ της πλακέτας και του υπολογιστή, είτε για την μεταφορά των προγραμμάτων που κατασκευάζονται στον υπολογιστή προς την πλακέτα, είτε για αμφίδρομη επικοινωνία της πλακέτας με τον υπολογιστή μέσα από το πρόγραμμα την ώρα που εκτελείται.

Εικόνα 8 - Arduino Uno

Υπάρχουν τρείς τρόποι τροφοδοσίας της πλακέτας: είτε μέσω της USB σύνδεσης, είτε από μια υποδοχή 2.1mm που διαθέτει, είτε από το Vin ακροδέκτη(5V) σε συνδυασμό με τον ακροδέκτη γείωσης. Με τον δεύτερο και τρίτο τρόπο πρέπει να χρησιμοποιήσουμε τροφοδοτικό ή μπαταρία. Η τροφοδοσία θα πρέπει να κυμαίνεται μεταξύ 7 και 12 V. Τροφοδοσία μικρότερη των 7V προκαλεί αστάθεια, ενώ μεγαλύτερη προκαλεί υπερθέρμανση.

Οι 14 ψηφιακοί ακροδέκτες μπορούν να χρησιμοποιηθούν είτε σαν είσοδοι, είτε σαν έξοδοι. Εξαρτάται από το πώς θα οριστούν στο πρόγραμμα. Λειτουργούν στα 5V με δυνατότητα παροχής ή λήψης 40mAρεύματος. Οι 6 αναλογικοί ακροδέκτες διαθέτουν 10bit ανάλυσης, οπότε μπορούν να αποτυπώσουν 1024 διαφορετικές τιμές. Χρησιμοποιούνται κυρίως για εισαγωγή αναλογικών σημάτων.

Η πλακέτα Arduino Uno διαθέτει τρεις διαφορετικές μνήμες: Flash, SRAM, EEPROM. Η μνήμη Flashέχει μέγεθος 32KB. Τα 2KB δεν είναι διαθέσιμα, αφού χρησιμοποιούνται από το firmware του Andruino, το οποίο είναι αναγκαίο για την εγκατάσταση του προγράμματος στην πλακέτα. Στα 30KB που είναι διαθέσιμα αποθηκεύεται το πρόγραμμα αφού πρώτα μεταγλωττιστεί στον υπολογιστή. Η μνήμη EEPROM έχει μέγεθος 1KB και χρησιμοποιείται για τις εγγραφές ή τις αναγνώσεις δεδομένων από το πρόγραμμα. Τέλος η μνήμη SRAM έχει μέγεθος 2KB χρησιμοποιείται από το πρόγραμμα για την αποθήκευση μεταβλητών και διάφορων άλλων δομών δεδομένων.

3.5.3 Χαρακτηριστικά Πλακέτας Wemos

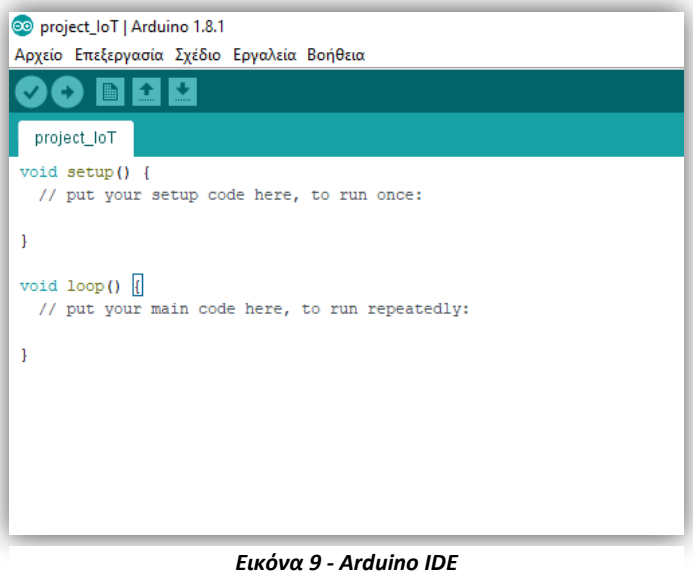
Η πλακέτα Wemos D1 [26] περιέχει τον μικροελεγκτή ESP-8266EX. Η πλακέτα Wemos λειτουργεί όπως η πλακέτα Arduino Uno και μπορεί να δεχθεί τα ίδια shields, τους ίδιους αισθητήρες και τις ίδιες μονάδες εισόδου/εξόδου με αυτά που μπορεί να δεχθεί η πλακέτα Arduino Uno. Ωστόσο, υπάρχουν ορισμένες διαφορές στην συσχέτιση των ακροδεκτών τους, με αποτέλεσμα προγράμματα που έχουμε σχεδιάσει για Arduino Uno, να χρειάζονται αλλαγές στους ακροδέκτες εισόδου και εξόδου.

Ένα βασικό πλεονέκτημα της Wemos είναι ότι δεν χρειάζεται κάποιο επιπλέον shield για να συνδεθεί με το Διαδίκτυο, αφού έχει ήδη ενσωματωμένο Wifi adapter.

3.5.4 Προγραμματισμός – Arduino IDE

Μέχρι την δημιουργία του πρώτου Arduino (2005) η διαδικασία σχεδίασης και προγραμματισμού των μικροελεγκτών ήταν πολύπλοκή και απαιτούσε εξαιρετικά μεγάλη γνώση και εξειδίκευση. Πλέον τα Arduino συνοδεύονται με λογισμικό προγραμματισμού που είναι χαρακτηριστικό για μια επαγγελματική κοινότητα που δεν είναι απαραίτητο να γνωρίζει το H/W των μικροελεγκτών.

Το περιβάλλον στο οποίο συντάσσεται ο κώδικας, ο οποίος στη συνέχεια θα ανέβει στη πλακέτα, ονομάζεται Arduino IDE [27]. Πρόκειται για ένα εύχρηστο και φιλικό ολοκληρωμένο περιβάλλον ανάπτυξης προγραμμάτων στο οποίο δίνεται η δυνατότητα της σύνταξης, της μεταγλώττισης αλλά και του ανεβάσματος τους προγράμματος στην πλακέτα. Μάλιστα, οι δύο τελευταίες λειτουργίες μπορούν να πραγματοποιηθούν με το πάτημα μόλις ενός κουμπιού.Το Arduino IDE συνοδεύεται με μια μεγάλη βιβλιοθήκη λογισμικού, γεγονός που καθιστά αρκετές λειτουργίες εισόδου ή εξόδου πολύ πιο εύκολες.



Εικόνα 9 - Arduino IDE

Παράλληλα, δεδομένου ότι η πλακέτα Arduino χρησιμοποιεί μικροελεγκτές Atmel, το περιβάλλον ανάπτυξης της Atmel και το AVR Studio μπορεί επίσης να χρησιμοποιηθεί για την ανάπτυξη λογισμικού για την πλακέτα Arduino.

Η γλώσσα προγραμματισμού που χρησιμοποιείται είναι η Wiring, η οποία είναι μια παραλλαγή της γλώσσας C/C++ για μικροελεγκτές AVR όπως ο ATmega. Οι εντολές και οι συναρτήσεις που χρησιμοποιούνται είναι ίδιες με αυτές της C με την ίδια σύνταξη, ίδιου τύπου δεδομένων και ίδιων τελεστών. Επίσης, υπάρχουν κάποιες επιπλέον εντολές και συναρτήσεις για την διαχείριση του hardware της πλακέτας Arduino. Μερικές από αυτές περιγράφονται παρακάτω:

	Είδος	**Περιγραφή**
LOW	Σταθερά	Έχει την τιμή 0 (είναι αντίστοιχη του λογικού false)

HIGH	Σταθερά	Έχει την τιμή 1 (αντίστοιχη του λογικού true).
INPUT	Σταθερά	Έχει την τιμή 0 (αντίστοιχη του λογικού false).
OUTPUT	Σταθερά	Έχει την τιμή 1 και είναι αντίστοιχη του λογικού true.
pinMode(*pin*, *mode*)	Εντολή	Ορίζει αν το συγκεκριμένο ψηφιακό pin θα λειτουργεί ως pin εισόδου ή pin εξόδου ανάλογα με την τιμή που δίνεται στην παράμετρο mode(INPUT ή OUTPUT αντίστοιχα).
digitalWrite(*pin*, *pinstatus*)	Εντολή	Ορίζει την κατάσταση (HIGH ή LOW) στο συγκεκριμένο ψηφιακό pin.
digitalRead(*pin*)	Συνάρτηση	Επιστρέφει την κατάσταση του συγκεκριμένου ψηφιακού pin (0 για LOW και 1 για HIGH) εάν είναι pin εισόδου.
analogReference(*type*)	Εντολή	Η παράμετρος type λαμβάνει τις τιμές DEFAULT, INTERNAL ή EXTERNAL για να καθορίσει την τάση αναφοράς (Vref) των αναλογικών εισόδων (5V, 1.1V ή η εξωτερική τάση με την οποία τροφοδοτείται το pin AREF αντίστοιχα)
analogRead(*pin*)	Συνάρτηση	Επιστρέφει την τιμή από 0 εώς 1023 (ακέραια), ανάλογα με την τάση που τροφοδοτείται το συγκεκριμένο pin αναλογικής εισόδου στην κλίμακα 0 ως Vref.
analogWrite(*pin*, *value*)	Εντολή	Ορίζει το συγκεκριμένο ψηφιακό pin σε κατάσταση ψευδοαναλογικής εξόδου (PWM). Η παράμετρος value καθορίζει το πλάτος του παλμού σε σχέση με την περίοδο του παραγόμενου σήματος στην κλίμακα από 0 ως 255 (π.χ. με value 127, το πλάτος του παλμού είναι ίσο με μισή περίοδο).
millis()	Συνάρτηση	Μετρητής που επιστρέφει το χρονικό διάστημα σε ms από την στιγμή που άρχισε η εκτέλεση του προγράμματος.
delay(*time*)	Εντολή	Σταματά προσωρινά την ροή του προγράμματος για time ms. Η παράμετρος time είναι unsigned long (από 0 ως 2^32). Όλες οι συναρτήσεις των οποίων η εκτέλεση ενεργοποιείται από interrupt θα εκτελεστούν κανονικά κατά την διάρκεια του delay.
attachInterrupt(*interrupt*, *function*,*triggermode*)	Εντολή	Θέτει σε λειτουργία το συγκεκριμένο interrupt, ώστε να ενεργοποιεί την συνάρτηση function, κάθε φορά που ικανοποιείται η συνθήκη που ορίζεται από την παράμετρο triggermode: • LOW (ενεργοποίηση όταν η κατάσταση του pin που αντιστοιχεί στο συγκεκριμένο interrupt γίνει LOW) • RISING (όταν από LOW γίνει HIGH) • FALLING (όταν από HIGH γίνει LOW) • CHANGE (όταν αλλάξει κατάσταση γενικά)
detachInterrupt(*interrupt*)	Εντολή	Απενεργοποιεί το συγκεκριμένο interrupt.
noInterrupts()	Εντολή	Σταματά προσωρινά την λειτουργία όλων των interrupt

interrupts()	Εντολή	Επαναφέρει την λειτουργία των interrupt που διακόπηκε προσωρινά από μια εντολή noInterrupts.
Serial.begin(*datarate*)	Μέθοδος κλάσης	Θέτει τον ρυθμό μεταφοράς δεδομένων του σειριακού interface
Serial.println(*data*)	Μέθοδος κλάσης	Στέλνει τα δεδομένα μέσω του σειριακού interface. Η παράμετρος data μπορεί να είναι είτε αριθμός είτε αλφαριθμητικό.

Η δομή κάθε πρόγραμμα που κατασκευάζεται για πλακέτα Arduino αποτελείται τουλάχιστον από τις δυο βασικές συναρτήσεις: void setup(){}, void loop(){}.

Η πρώτη συνάρτηση setup() εκτελείται μια φορά, στην αρχή κατά την εκτέλεση του προγράμματος, ενώ η συνάρτηση loop() εκτελείται επαναληπτικά. Ο αριθμός των επαναλήψεων στη μονάδα του χρόνου εξαρτάται από το χρόνο που χρειάζεται η συνάρτηση για την εκτέλεση όλων των εντολών που βρίσκονται μέσα σε αυτή.

Κεφάλαιο 4: Σχεδιασμός και υλοποίηση του συστήματος αισθητήρων

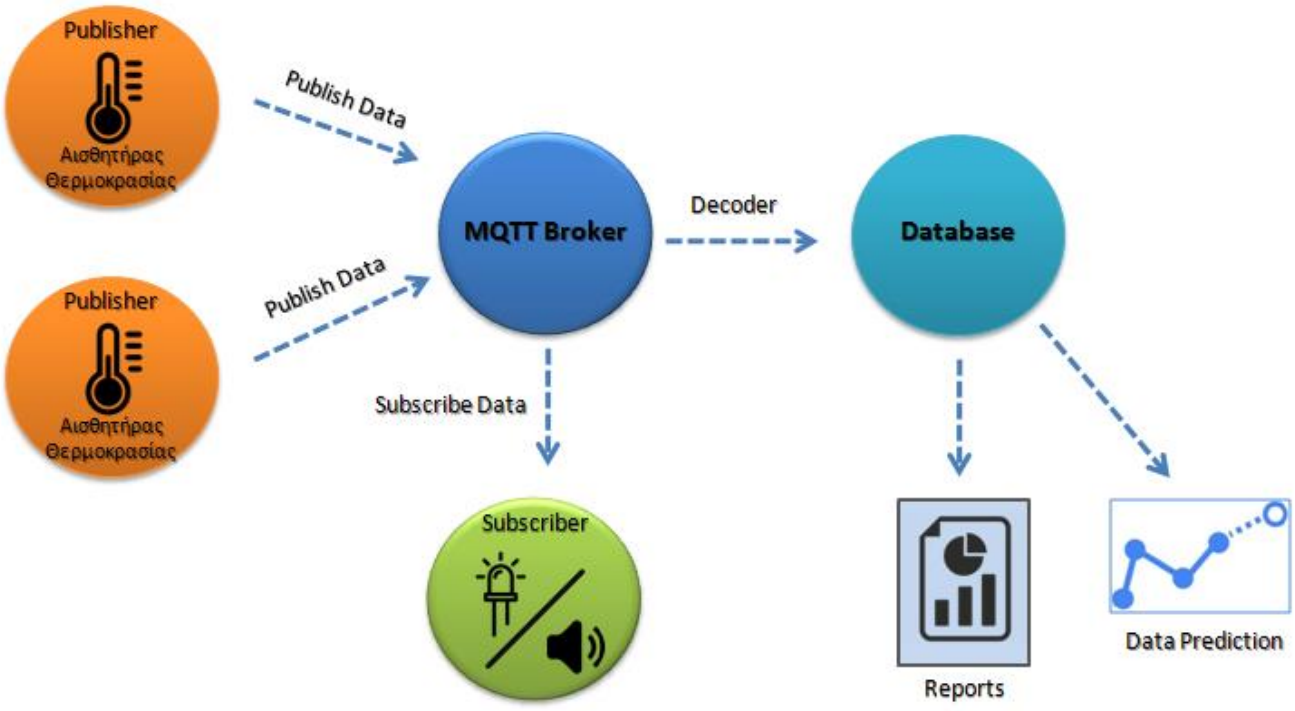
4.1 Περιγραφή σεναρίου χρήσης συστήματος αισθητήρων

Στη παρούσα εργασία δημιουργείται ένα ολοκληρωμένο σύστημα καταγραφής, μεταφοράς, καταχώρησης, οπτικοποίησης και ανάλυσης δεδομένων.

Πιο συγκεκριμένα, αισθητήρες θερμοκρασίας ελέγχουν την εσωτερική και εξωτερική θερμοκρασία ενός χώρου και στέλνουν τα δεδομένα σε έναν MQTT Broker ο οποίος προωθεί τα δεδομένα στις εγγεγραμμένες συσκευές, αλλά και σε βάση δεδομένων όπου αποθηκεύονται για μελλοντική χρήση (επεξεργασία/ανάλυση/αξιοποίηση).

Μια εγγεγραμμένη στον MQTT Broker συσκευή ελέγχει τις τιμές που λαμβάνει, και σε περίπτωση που αυτές ξεπεράσουν ή πέσουν κάτω από συγκεκριμένα όρια, ένας βομβητής και ένα LED ειδοποιεί ότι υπάρχουν τιμές εκτός ορίων.

Παράλληλα, ανακτούνται τα δεδομένα από την βάση, δημιουργούνται διαγράμματα για την οπτικοποίησή τους και τέλος με συγκεκριμένες μεθόδους αναλύσεις, αξιοποιούνται τα δεδομένα και δημιουργούνται πρόβλεψης για μελλοντικές μετρήσεις.



Εικόνα 10–Σενάριο συστήματος χρήσης αισθητήρων

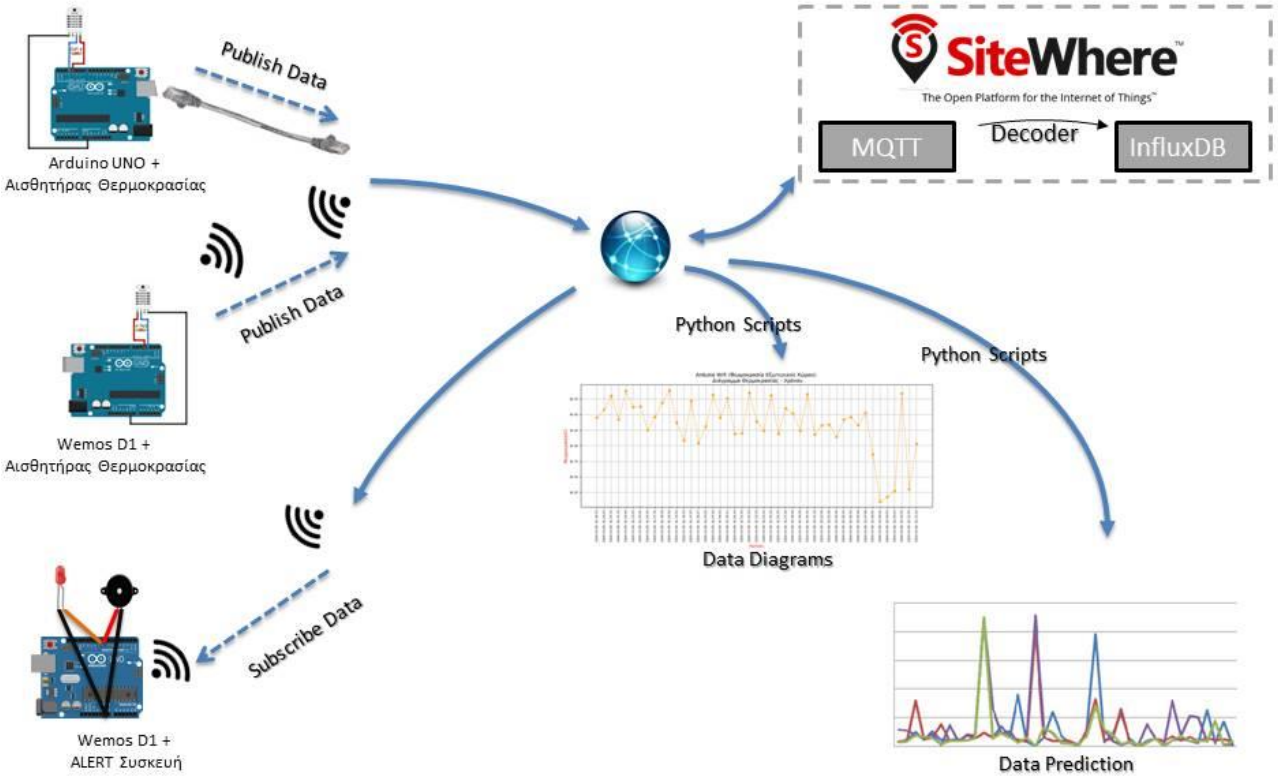
4.2 Υλοποίηση

Για την υλοποίηση του παραπάνω σεναρίου χρησιμοποιήθηκαν ως Publisher συσκευές, πλακέτες Arduino Uno και Wemos. Στις πλακέτα έχει εγκατασταθεί ένας αισθητήρας θερμοκρασίας και οι τιμές των θερμοκρασιών στέλνονται περιοδικά, στον MQTT Broker της IoT πλατφόρμας SiteWhere.

Η πλατφόρμα SiteWhere, όπως έχουμε αναφέρει, περιέχει MQTT Broker αλλά και βάση δεδομένων INFLUXDB. Έτσι όλα τα δεδομένα που στέλνονται από τις πλακέτες, λαμβάνονται από τον MQTT Broker και στην συνέχεια, αποκωδικοποιούνται και αποθηκεύονται στην βάση δεδομένων. Παράλληλα, τα δεδομένα μοιράζονται από τον MQTT Broker σε μια πλακέτα η οποία λειτουργεί σαν subscriber συσκευή. Η πλακέτα αυτή χρησιμοποιείται σαν ALERT συσκευή, έχει εγκατεστημένο ένα βομβητή και ένα LED, τα οποία ενεργοποιούνται, όταν οι τιμές που λαμβάνει είναι μη αποδεκτές.

Τέλος με χρήση της γλώσσας προγραμματισμού Python, ανακτούμε τα δεδομένα από την βάση, δημιουργούμε διαγράμματα αλλά και χρησιμοποιώντας αλγόριθμο πρόβλεψης χρονοσειράς κάνουμε πρόβλεψη για μελλοντικά δεδομένα.

Στην συνέχεια παρουσιάζεται ολοκληρωμένα το σύστημα καταγραφής, μεταφοράς, αποθήκευσης, ανάκτησης, οπτικοποίησης και ανάλυσης δεδομένων



Εικόνα 11 - Σύστημα καταγραφής, μεταφοράς, καταχώρησης, οπτικοποίησης και ανάλυσης δεδομένων

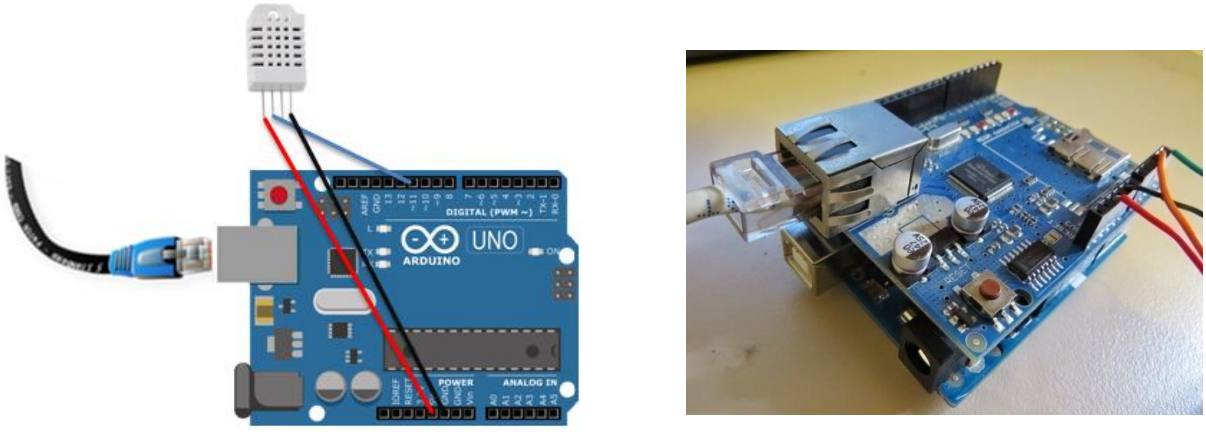
Με την υλοποίηση του παραπάνω συστήματος, γίνεται φανερό ότι ένα τέτοιο μοντέλο μπορεί εύκολα να κλιμακωθεί. Το μικρό κόστος σε επίπεδο hardware, ο εύκολος τρόπος προγραμματισμού του αλλά και ότι σε κάθε πλακέτα μπορούν να εγκατασταθούν μια σειρά από αισθητήρες, οδηγεί στο συμπέρασμα ότι ένα τέτοιο σύστημα με την κατάλληλη παραμετροποίηση μπορεί εύκολα να χρησιμοποιηθεί σε όλους τους τομείς της ανθρώπινης δράσης.

Κεφάλαιο 5: Καταγραφή και Μεταφορά Δεδομένων

Στο παρόν κεφάλαιο αναλύεται ο τρόπος με τον οποίο γίνεται η καταγραφή των θερμοκρασιών από τις πλακέτες, η επεξεργασία και η αποστολή των δεδομένων στον και από τον MQTT Broker. Παρουσιάζονται όλες οι βιβλιοθήκες που χρησιμοποιήθηκαν κατα τον προγραμματισμό των πλακετών ώστε να λειτουργήσουν σαν publisher συσκευές.

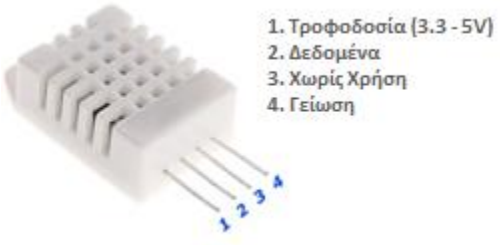
5.1 Καταγραφή Μετρήσεων

Για τις εσωτερικές μετρήσεις χρησιμοποιήθηκε η πλακέτα Arduino Uno με εγκατεστημένο τον αισθητήρα θερμοκρασίας (και υγρασίας) DHT22 και το Ethernet Shield για την επικοινωνία της πλακέτας με τον MQTT.



Εικόνα 12 - Arduino Uno - Ethernet Shield - DHT22

Το DHT-22 είναι ένας βασικός, χαμηλού κόστους, αισθητήρας που χρησιμοποιείτε για την εύρεση της σχετικής υγρασίας και θερμοκρασίας στον χώρο. Στο εσωτερικό του κρύβει έναν αισθητήρα υγρασίας και ένα θερμίστορ (μεταβλητή αντίσταση που η τιμή της μεταβάλλεται ανάλογα με την θερμοκρασία της ατμόσφαιρας) ελέγχοντας τον αέρα που το περιβάλει. Διαθέτει τέσσερις ακροδέκτες, από τους οποίους ο πρώτος χρησιμοποιείται για την τροφοδοσία (3-5V), ο δεύτερος για την αποστολή των δεδομένων και ο τέταρτος για την γείωση.



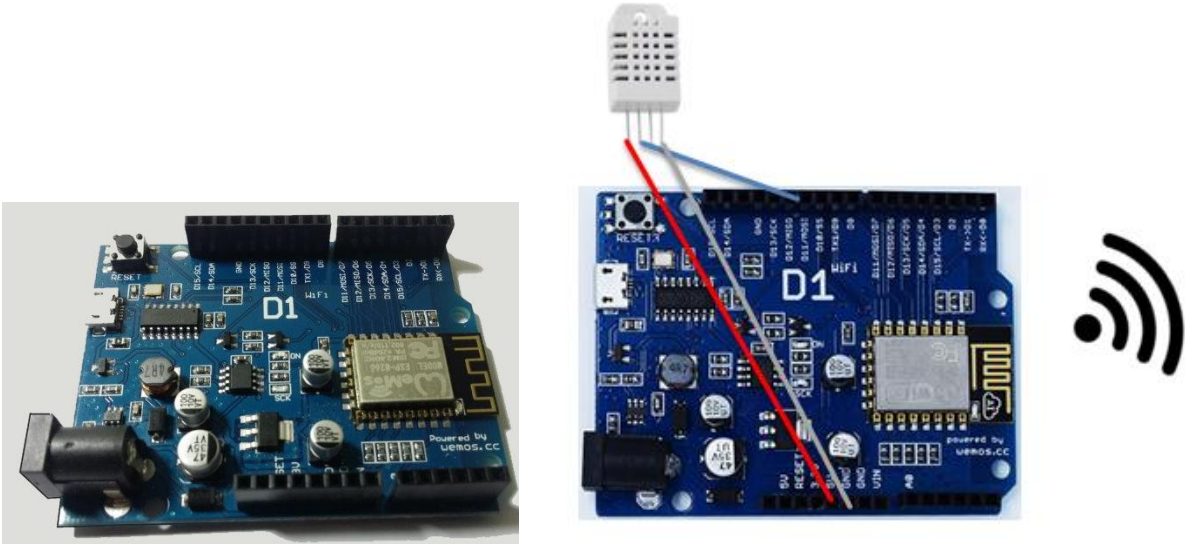
Εικόνα 13 - Αισθητήρας DHT-22

Τα τεχνικά χαρακτηριστικά του αισθητήρα:

	Υγρασία	Θερμοκρασία
Τάση	3,3 – 5 V	3,3 – 5 V
Έξοδος	Ψηφιακή	Ψηφιακή
Αισθητήρας	Πολυμερής Πυκνωτής	DS18B20
Εύρος	0 – 100% Rh	-40 – 125 $C°$
Ακρίβεια	2 – 5 %	± 0,5 $C°$
Δειγματοληψία	0,5 Hz	0,5 Hz

Εικόνα 14 - Τεχνικά Χαρακτηριστηκά DHT-22

Για τις εξωτερικές μετρήσεις χρησιμοποιήθηκε η πλακέτα Wemos στην οποία εγκαταστάθηκε ο Αισθητήρας θερμοκρασίας και υγρασίας DHT-22. Η επικοινωνία της πλακέτας με τον MQTT Broker γίνεται όπως έχουμε προαναφέρει με τον ενσωματωμένο Wifi adapter που περιέχει.



Εικόνα 15 - Wemos

Η καταγραφή των τιμών θερμοκρασίας και υγρασίας γίνονται με περίοδο ενός λεπτού. Στη συνέχεια, κάθε πλακέτα, ελέγχει την σύνδεση της με τον MQTT Broker και στέλνει τα δεδομένα στον MQTT Broker.

Σε επίπεδο προγραμματισμού της κάθε πλακέτας, για την καταγραφή της θερμοκρασίας χρησιμοποιήθηκε η βιβλιοθήκη DHT.h. Η σύνδεση του αισθητήρα με το Arduino Uno και το Wemos γίνεται μέσω μιας ψηφιακής εισόδου στην οποία εισέρχεται τάση η οποία έχει σταθμιστεί ανάλογα με την θερμοκρασία του περιβάλλοντος. Η τάση αυτή, με την βοήθεια της βιβλιοθήκης, μετατρέπεται σε τιμή η οποία αντιστοιχεί στην θερμοκρασία.

Χρησιμοποιήθηκαν οι παρακάτω συναρτήσεις, κλάσεις και μεταβλητές από την βιβλιοθήκη DHT.h:

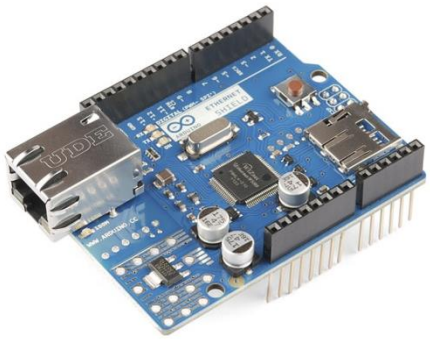
Εντολή	Περιγραφή
dht22.read22(pin)	Διαβάζει τις τιμές που καταγράφει ο αισθητήρας DHT22. Όπου pin τοποθετείται ο αριθμός του ακροδέκτη στον οποίο έχει συνδεθεί το data pin του αισθητήρα DHT22.

Ο κώδικας με τον οποίο γίνεται η σύνδεση του αισθητήρα με την πλακέτα Arduino καθώς και οι μετρήσεις παρουσιάζεται στο **παράρτημα Α**

5.2 Σύνδεση Arduino

Για την επικοινωνία της πλακέτας Arduino Uno με τον MQTT χρησιμοποιείται το Ethernet Shield. Μέσω του Ethernet Shield επιτυγχάνεται η ενσύρματη επικοινωνία με το Διαδίκτυο. Τοποθετείται πάνω στις υποδοχές του Arduino, δεν απαιτεί επιπλέον τροφοδοσία αφού χρησιμοποιεί την τροφοδοσία του Arduino ενώ παράλληλα από πάνω του μπορεί να τοποθετηθεί κάποιο επιπρόσθετο shield.

Περιλαμβάνει μια θύρα Rj45 μέσω της οποίας γίνεται η σύνδεση και μια θύρα υποδοχής SD, η οποία μπορεί να χρησιμοποιηθεί για την αποθήκευση αρχείων με σκοπό το διαμοιρασμό μέσω δικτύου, ωστόσο δεν χρησιμοποιείται στην παρούσα εργασία.



Εικόνα 16 - Ethernet Shield

Σε επίπεδο προγραμματισμού της πλακέτας Arduino Uno, χρησιμοποιείται η βιβλιοθήκη Ethernet.h. Μέσω της συγκεκριμένης βιβλιοθήκης το Ethernet Shield επιτυγχάνει να συνδέσει την πλακέτα με το Διαδίκτυο. Αρχικά ορίζεται η MAC διεύθυνση και μια στατική IP του Ethernet shield και η πλακέτα, προσπαθεί να συνδεθεί με το Διαδίκτυο. Το Ethernet Shield υποστηρίζει τα πρωτόκολλα TCP και UDP και μπορεί να έχει 4 ταυτόχρονες συνδέσεις. Μόλις η σύνδεση επιτευχθεί, χρησιμοποιώντας το πρωτόκολλο NTP (Network Time Protocol), η πλακέτα συνδέεται με τον εξυπηρετητή ntp.grnet.gr και συγχρονίζει την ώρα της.

Στη συνέχεια, με την βοήθεια της βιβλιοθήκης PubSubClient.h δημιουργείται ένας MQTT Client ο οποίος επιχειρεί να συνδεθεί με τον MQTT Broker. Η βιβλιοθήκη αυτή, παρέχει ένα πρόγραμμα-πελάτη για τη αποστολή μηνυμάτων ή εγγραφή σε topics για τη λήψη

μηνυμάτων, με ένα MQTT Broker. Η κάθε publisher συσκευή αποστέλει τα δεδομένα στο topic “DataManagement/input/arduino_vc”.

Ο συγκεκριμένος Broker, έχει IP 195.134.79.240 και επιτρέπει την κίνηση στο Port 1338. Μόλις επιτευχθεί η σύνδεση, η πλακέτα είναι έτοιμη να στέλνει ή να λαμβάνει δεδομένα.

Παράλληλα για τον προγραμματισμό της πλακέτας Wemos ακολουθούμε την ίδια διαδικασία με την μόνη διαφορά την σύνδεση της πλακέτας με το Διαδίκτυο. Η σύνδεση γίνεται ασύρματα, γεγονός που επιτυγχάνεται με την χρήση της βιβλιοθήκης ESP8266WiFi.h.

Χρησιμοποιήθηκαν οι παρακάτω συναρτήσεις, κλάσεις και μεταβλητές από τις βιβλιοθήκες Ethernet.h και PubSubClient.h:

Εντολή	**Περιγραφή**
Ethernet.begin(mac, ip)	Αρχικοποιεί τη βιβλιοθήκη Ethernet και τις ρυθμίσεις του δικτύου. Η βιβλιοθήκη Ethernet υποστηρίζει την δυνατότητα απόκτησης διεύθυνσης IP μέσω DHCP με την χρήση της κλήσης Ethernet.begin(mac).
EthernetUDP.begin(localPort)	Αρχικοποιεί μια UDP σύνδεση στην τοπική localPort
PubSubClient(server, port, callback, EthernetClient):	Δημιουργείτε ο Constructor για την αρχικοποίηση της βιβλιοθήκης. Το callback είναι μια συνάρτηση που καλείται για την λήψη δεδομένων από τον broker.
MQTTClient.connect(clientID):	Συνδέει τον συγκεκριμένο MQTTClient στον MQTT Broker με το αναγνωριστικό clientID
MQTTClient.loop():	Η συνάρτηση αυτή καλείται τακτικά ώστε να διατηρείται η σύνδεση του MQTTClient με τον Broker.
MQTTClient.subscribe(topic):	Με την κλήση της συνάρτησης, ο MQTTClient εγγράφεται στο συγκεκριμένο topic

Ο κώδικας με τον οποίο γίνεται η σύνδεση του Arduino Uno με το Διαδίκτυο και τον MQTT Broker παρουσιάζεται στο **παράρτημα Α**

5.3 Μεταφορά Δεδομένων στον MQTT

Η μεταφορά των δεδομένων από τις πλακέτες προς τον MQTT Broker και αντίστροφα γίνεται με την βοήθεια JSON. Το JSON είναι μία μορφή δεδομένων σε απλή μορφή κειμένου. Είναι ένας εύκολος τρόπος αποτύπωσης δεδομένων ο οποίος βοηθάει στην αποφυγή πολύπλοκων διαδικασιών για την ανάγνωση, δημιουργία, ανάλυση και αποστολής δεδομένων. Είναι εύκολο τόσο για τις μηχανές να το αναλύσουν και να το δημιουργήσουν όσο και για τους ανθρώπους να το διαβάσουν. Βασίζεται σε ένα υποσύνολο της Java Script Programming Language, Standard ECMA-262 3rd Edition - December 1999.

Παρά το γεγονός ότι η ονομασία του JSON (JavaScript Object Notation) περιέχει τη λέξη Javascript, δεν είναι απαραίτητα σχετικό με τη γλώσσα αυτή. Το JSON δηλαδή μπορεί να χρησιμοποιηθεί σε ένα μεγάλο εύρος γλωσσών προγραμματισμού όπως C, C++, C#, Java, JavaScript, Perl, Python κ.α.

Ένα αντικείμενο τύπου JSON είναι μια σειρά από ένα ή περισσότερα ζεύγη ονομάτων-τιμών. Το όνομα είναι τύπου string ενώ η τιμή μπορεί να είναι τύπου: String, Boolean, number, object, array, άδεια τιμή (null). Το αντικείμενο ξεκινάει με ένα αριστερό υποστήριγμα (brace) ({) και τελειώνει με ένα δεξί (}). Κάθε όνομα στο ζεύγος ακολουθείται από μία άνω κάτω τελεία (:) και τα ζεύγη ονομάτων τιμών χωρίζονται με κόμμα (,). Εκτός από τα ζεύγη ονομάτων-τιμών, ένα JSON μπορεί να περιέχει και πινάκα.

Στη συνέχεια παρουσιάζεται το JSON όπως αυτό δημιουργείται από τις πλακέτες και στέλνεται στον MQTT Broker.

JSON : {“id”:TIMH, “value”: TIMH, “ Timestamp”:TIMH}

Το πεδίο id παίρνει την τιμή “arduinoE” αν τα δεδομένα προέρχονται από την πλακέτα για τις εσωτερικές μετρήσεις ή “arduinoW” αν τα δεδομένα προέρχονται από την πλακέτα για τις εξωτερικές μετρήσεις. Το πεδίο value παίρνει τιμές της θερμοκρασίες που δίνουν οι αισθητήρες θερμοκρασίας. Τέλος στο πεδίο time δημιουργείται μια χρονοσφραγίδα τις μορφής ***ΕΕΕΕ:ΜΜ:ΗΗΩΩ:ΛΛ:ΔΔ***

Παρακάτω παρουσιάζεται το JSON message που έστειλε η πλακέτα εσωτερικής μέτρησης στον MQTT Broker, την χρονική στιγμή 2018-01-13 15:49:59 +0200 όπου η θερμοκρασία ήταν 24.57 °C

{"id":"arduinoE","value":"24.57","Timestamp":"2018-01-13 15:49:59 +0200"}

Σε επίπεδο προγραμματισμού της κάθε πλακέτας, χρησιμοποιείται η η βιβλιοθήκη ArduinoJson.h για την μετατροπή των δεδομένων σε JSON μηνύματα και αντίστροφα. Οι publisher συσκευές χρησιμοποιούν την βιβλιοθήκη για να μετατρέψουν τα δεδομένα (μετρήσεις, ώρα, id συσκευής) σε JSON μήνυμα. Αντίστροφα, η subscriber συσκευή μετατρέπει το JSON μήνυμα που λαμβάνει απο τον MQTT Broker σε δεδομένα και τα επεξεργάζεται.

Χρησιμοποιήθηκαν οι παρακάτω συναρτήσεις, κλάσεις και μεταβλητές απο την βιβλιοθήκη ArduinoJson.h :

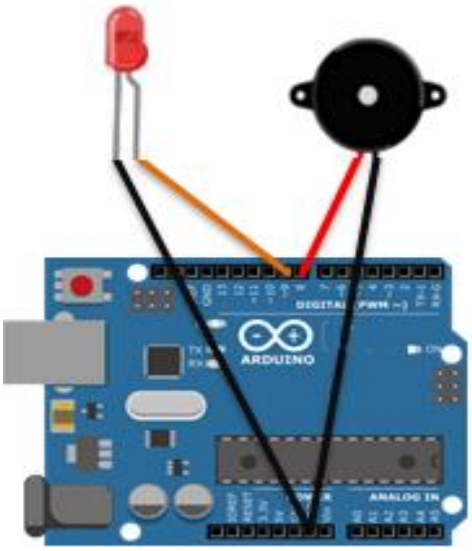
Εντολή	**Περιγραφή**
StaticJsonBuffer<200> jsonBuffer	Δεσμεύει μνήμη στην SRAM του Arduino, για τη δημιουργία του JSON πίνακα. Η τιμή ‘200’ ορίζει το μέγεθος της κατειλημμένης μνήμης σε bytes.
JsonObject& root = jsonBuffer.parseObject(json)	Η συνάρτηση parseObject() κατανέμει τα δεδομένα από το json string σε ένα JsonObject δια της αναφοράς. Αυτό σημαίνει ότι τα δεδομένα δεν αντιγράφονται από το jsonBuffer στο jsonObject, αλλά το

	jsonObject περιέχει αναφορές προς το jsonBuffer σχετικά με το που μπορεί να βρει το κάθε δεδομένο. Όπου JsonObject είναι η ρίζα του JSON δέντρου

Ο κώδικας με τον οποίο δημιουργείται το JSON μήνυμα από τα δεδομένα και αντίστροφα παρουσιάζεται στο **παράρτημα Α**

5.4 Μεταφορά Δεδομένων από τον MQTT

Στη παρούσα εργασία, παράλληλα, έχει χρησιμοποιηθεί μια πλακέτα ως subscriber συσκευή στον MQTT Broker. Η συγκεκριμένη πλακέτα λειτουργεί ως ALERT συσκευή, έχει εγκατασταθεί ένας βομβητή και ένα LED λαμπάκι, τα οποία ενεργοποιούνται όταν οι τιμές της εξωτερικής θερμοκρασίας που λαμβάνει απο τον MQTT Broker είναι κάτω από 0 ή πάνω από 40. Τα δεδομένα τα λαμβάνει ως JSON μηνύματα, τα αποκωδικοποιεί και στην συνέχεια επεξεργάζεται τις τιμές.



Εικόνα 17 - Arduino ως Subscriber

Η πλακέτα χρησιμοποιεί την βιβλιοθήκη PubSubClient.h, δημιουργείται ένας MQTTClient ο οποίος επιχειρεί να συνδεθεί με τον MQTT Broker. Έχει εγγραφεί στο ίδιο topic (DataManagement/input/arduino_vc) με τις publisher συσκευές και λαμβάνει όλα τα δεδομένα που στέλνονται. Παράλληλα χρησιμοποιεί την βιβλιοθήκη ArduinoJson.h για την αποκωδικοποίηση των JSON μηνυμάτων που λαμβάνει.

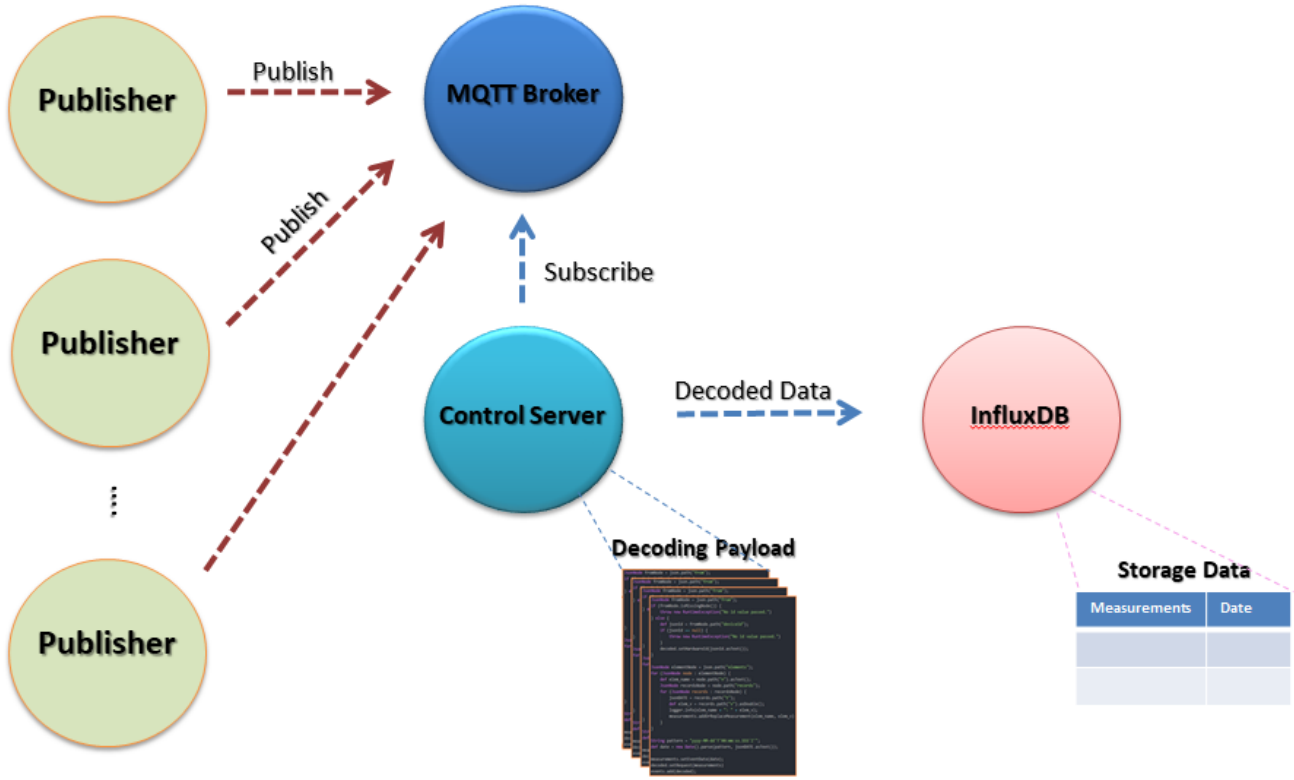
Ο κώδικας με τον οποίο έχει προγραμματιστεί η συγκεκριμένη subscriber συσκευή παρουσιάζεται στο **παράρτημα Α**

Κεφάλαιο 6: Διαχείριση Δεδομένων

Στο προηγούμενο κεφάλαιο ασχοληθήκαμε με την παραγωγή μετρήσεων από τις πλακέτες Arduino και την μεταφορά των δεδομένων αυτών στον MQTT. Το κεφάλαιο αυτό αποτελεί τη φυσική συνέχεια του προηγούμενου και περιλαμβάνει μεθόδους και τεχνικές αποθήκευσης, συλλογής, ανάλυσης και επεξεργασίας των δεδομένων.

6.1 Καταχώρηση Δεδομένων σε Βάση Δεδομένων

Οι μετρήσεις δημοσιεύονται απο τις publisher συσκευές στον MQTT με JSON μηνύματα με συγκεκριμένα θέματα. Στη συνέχεια ένας διακομιστής ελέγχου, ο οποίος είναι συνδεδεμένος με τον MQTT και έχει ήδη εγγραφεί σε συγκεκριμένα θέματα, λαμβάνει τα μηνύματα και σύμφωνα με το θέμα των μηνυμάτων που δέχεται, τα αποκωδικοποιεί και αποθηκεύει τις μετρήσεις κατάλληλα στη βάση δεδομένων. Με την μέθοδο αυτή ο διακομιστής ελέγχου μπορεί να δέχεται διαφορετικά μηνύματα από έναν ή περισσότερους publisher (μέσω του MQTT) και ελέγχοντας το θέμα του μηνύματος, αποκωδικοποιεί το μήνυμα και το αποθηκεύει.



Εικόνα 18 - Η ροή των δεδομένων απο τους Publisher ως την Βάση Δεδομένων

Συγκεκριμένα, ο Decoder μεταμορφώνει το JSON μήνυμα σε συμβολοσειρά η οποία μετατρέπεται σε ένα JsonNode. Το JsonNode είναι ένα δέντρο κόμβων στο οποίο κάθε κόμβος περιέχει την πληροφορία που θα αποθηκευτεί στη βάση. Για το λόγο αυτό, η δομή των δεδομένων, θα πρέπει να είναι γνωστή από πριν, ώστε ο decoder να μπορεί να αποκωδικοποιήσει σωστά (ανάλογα με το θέμα του μηνύματος) και στη συνέχεια να αποθηκεύσει τα δεδομένα στη βάση.

Στο σημείο αυτό θα πρέπει να αναφερθεί οτι ο Decoder της παρούσας εργασίας δημιουργήθηκε από τον συνάδελφο κ. Νικόλαο Γκάτζιο και παρουσιάζεται στο **παράρτημα Α.**

InfluxDB

Για την αποθήκευση των δεδομένων χρησιμοποιείται η βάση δεδομένων InfluxDB. Η InfluxDB είναι μια βάση δεδομένων χρονοσειρών ανοικτού κώδικα που αναπτύχθηκε από την ομάδα InfluxData. Είναι γραμμένο στην γλώσσα προγραμματισμού Go και βελτιστοποιείται για γρήγορη, υψηλής απόδοσης αποθήκευση και ανάκτηση δεδομένων χρονοσειρών σε πεδία όπως η παρακολούθηση λειτουργιών, οι μετρήσεις εφαρμογών, τα δεδομένα αισθητήρων του Internet of Things και οι αναλύσεις πραγματικού χρόνου. Επίσης, παρέχει υποστήριξη για επεξεργασία δεδομένων στο εργαλείο Graphite.

6.2 Συλλογή Δεδομένων

Μία από τις βασικότερες ενέργειες για να φτάσουμε στην επεξεργασία των δεδομένων, είναι η ανάκτηση-συλλογή των δεδομένων από την βάση. Στην παρούσα εργασία η ανάκτηση των δεδομένων γίνεται με διάφορες τεχνικές από την βάση στην οποία έχουν αποθηκευτεί οι μετρήσεις.

6.2.1 Python Scripts

Η συλλογή δεδομένων απο την InfluxDB βάση μπορεί να επιτευχθεί με την χρήση μιας γλώσσας προγραμματισμού. Στη συγκεκριμένη εργασία επιλέχθηκε η γλώσσα Python, μέσω της οποίας δημιουργήθηκαν scripts τα οποία επικοινωνούν με την βάση. Ένας InfluxDB-Python client αλληλεπιδρά με την InfluxDB βάση, συνδέεται, στέλνει τα ερωτήματα (queries) και τα αποτελέσματα επιστρέφουν και αποθηκεύονται σε αρχείο.

Σε επίπεδο προγραμματισμού χρησιμοποιείται το API InfluxDBClient. Το συγκεκριμένο API δημιουργεί έναν Client ο οποίος συνδέεται στην βάση IoT_test (IP 195.134.79.240) και επιτρέπει την κίνηση στο Port 8086, αφού πρώτα γίνει αυθεντικοποίηση (username/password).

```
host='195.134.79.240'
port=8086
user = 'Vangelis'
password = 'arduino2017'
dbname = 'IoT_test'
```

Μόλις επιτευχθεί η σύνδεση, εκτελείται το query, συλλέγονται τα δεδομένα και αποθηκεύονται τοπικά στο αρχείο D:\Measurements.csv.

```
client = InfluxDBClient(host, port, user, password, dbname, verify_ssl=False)
result = client.query(query)
```

Το query που εκτελέστηκε για να ανακτηθούν όλες οι μετρήσεις που έχουν αποθηκευτεί στην βάση, φαίνεται παρακάτω.

```
query = 'select  time as T, "mx:temp_arduinoE" as E,"mx:temp_arduinoW" as W from events  order by time DESC '
```

Στην εικόνα που ακολουθεί παρουσιάζεται η μορφή του συγκεκριμένου αρχείου. Κάθε εγγραφή αποτελείται από τρία πεδία: τον χρόνο που έγινε η μέτρηση, συνοδευόμενο είτε με την μέτρηση της μιας πλακέτας, είτε της άλλης (ανάλογα σε ποια από τις δύο ανήκει η μέτρηση την συγκεκριμένη χρονική στιγμή).

Measurements.csv - Notepad

File Edit Format View Help

Time,	ArduinoE,	ArduinoW
2017-12-11T09:06:29Z,	17.09,	None
2017-12-11T09:06:32Z,	None,	13.34
2017-12-11T09:07:29Z,	17.56,	None
2017-12-11T09:07:32Z,	None,	13.55
2017-12-11T09:08:29Z,	17.78,	None
2017-12-11T09:08:32Z,	None,	13.51
2017-12-11T09:09:30Z,	17.62,	None
2017-12-11T09:09:33Z,	None,	13.61
2017-12-11T09:10:30Z,	17.21,	None
2017-12-11T09:10:33Z,	None,	13.52
2017-12-11T09:11:31Z,	17.03,	None
2017-12-11T09:11:35Z,	None,	13.35
2017-12-11T09:12:31Z,	17.31,	None
2017-12-11T09:12:36Z,	None,	13.19
2017-12-11T09:13:31Z,	17.07,	None
2017-12-11T09:13:36Z,	None,	13.08
2017-12-11T09:14:32Z,	17.38,	None
2017-12-11T09:14:36Z,	None,	13.06
2017-12-11T09:15:32Z,	17.98,	None
2017-12-11T09:15:37Z,	None,	13.63
2017-12-11T09:16:34Z,	17.9,	None
2017-12-11T09:16:38Z,	None,	13.86

Εικόνα 19 - Δεδομένα του αρχείου Measurements.cvs

Ο κώδικας με τον οποίο πραγματοποιείται η σύνδεση με την βάση και η ανάκτηση των δεδομένων παρουσιάζεται στο **παράρτημα Α.**

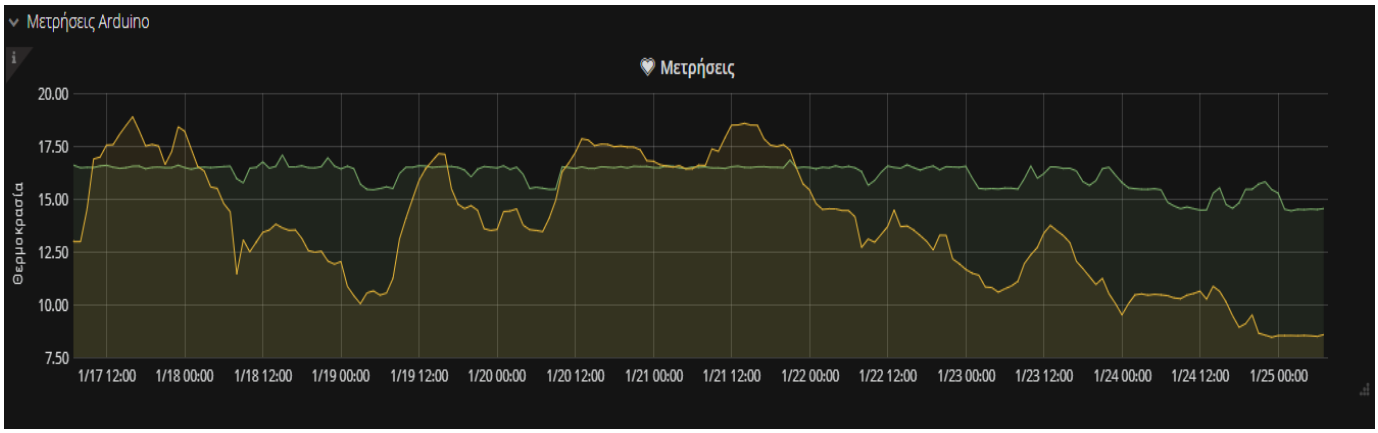
6.2.2 Οπτικοποίηση Δεδομένων

Η οπτικοποίηση δεδομένων, όπως έχει ήδη αναφερθεί, είναι η τεχνική με την οποία τα δεδομένα αναπαρίστανται σε σχηματική μορφή, με σκοπό την ποιοτική κατανόηση των πληροφοριών που εμπεριέχουν. Η οπτική παρουσίαση βοηθάει στην καλύτερη κατανόηση, για ανακάλυψη γνώσης και εξερεύνηση των δεδομένων.

Grafana

Ένα εργαλείο οπτικοποίησης που χρησιμοποιήθηκε είναι η Grafana, μια πλατφόρμα ανοιχτού κώδικα που λειτουργεί με την InfluxDB βάση.

Δημιουργεί γρήγορα και ευέλικτα client-side γραφήματα με πολυάριθμες επιλογές. Επίσης αξιολογεί συνεχώς τα δεδομένα και δημιουργεί notifications και alerts ανάλογα με τους κανόνες που έχουμε θέσει. Ένα από τα πλεονεκτήματα στο σχεδιασμό της Grafana, είναι η ανακατασκευή των γραφημάτων στην πλευρά του πελάτη, μειώνοντας τον υπολογιστικό φόρτο του εξυπηρετητή. Τέλος, επιτρέπει την ταυτόχρονη σύνδεση με πολλές πηγές.



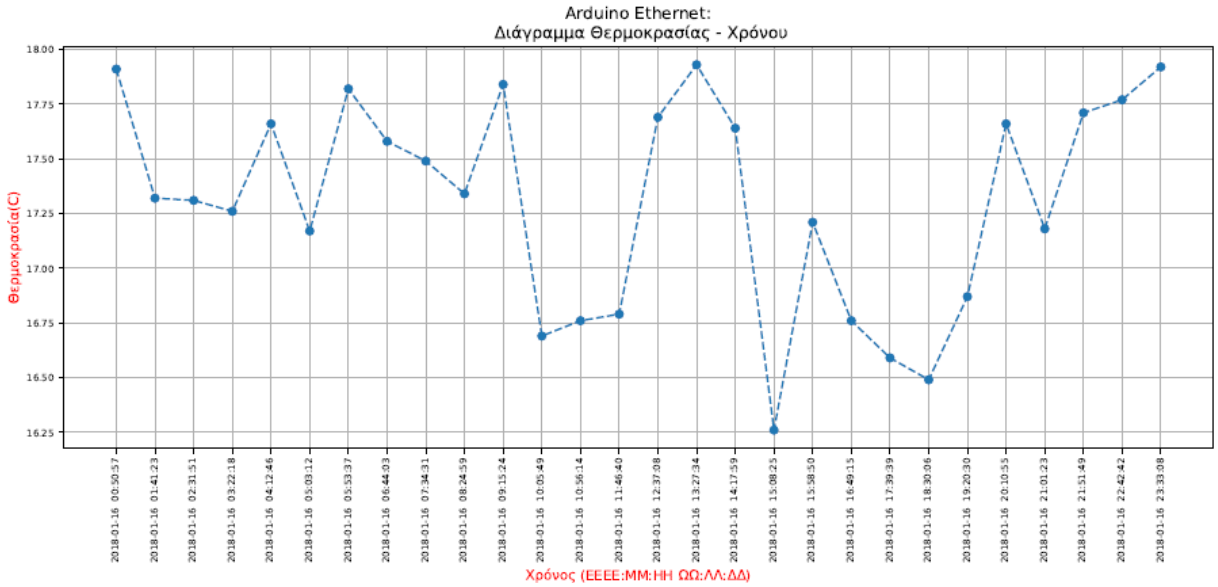
Εικόνα 20 - Διάγραμμα πλατφόρμας Grafana

Matplotlib Python library

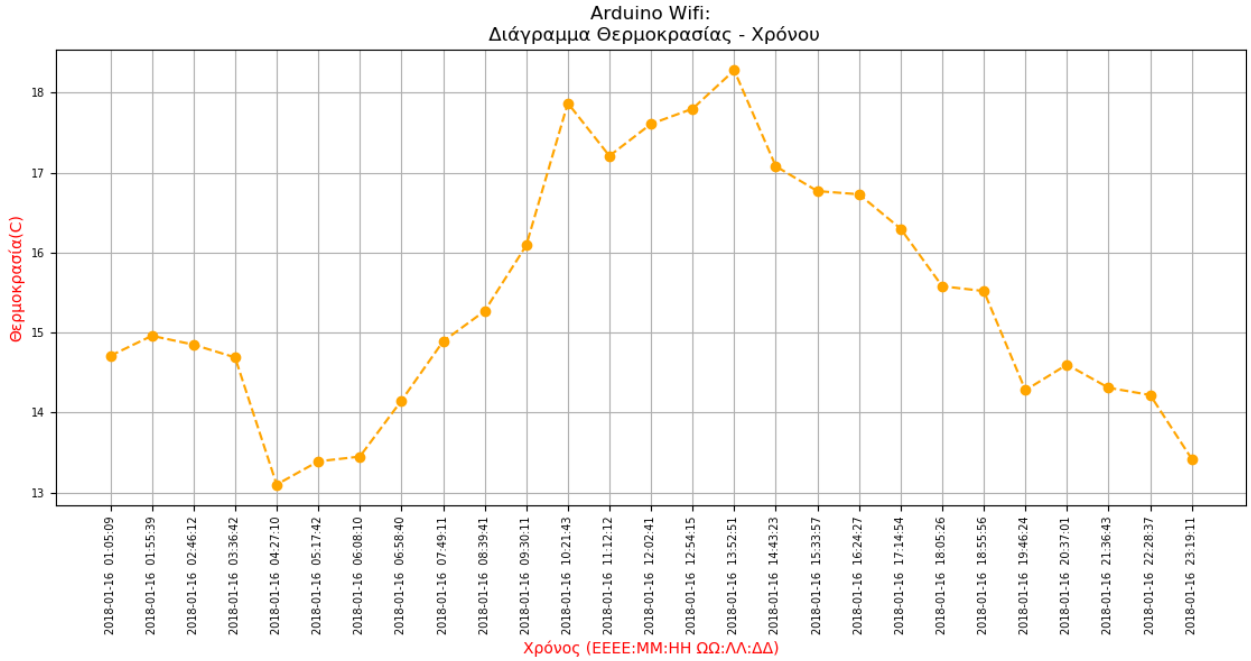
Άλλο ένα εργαλείο που χρησιμοποιήθηκε για την οπτικοποίηση των δεδομένων είναι η βιβλιοθήκη Matplotlib της γλώσσας προγραμματισμού Python. Η matplotlib [28] είναι μια βιβλιοθήκη γραφικών της Python η οποία μπορεί να παράξει ένα μεγάλο είδος από γραφικές παραστάσεις μέσω προγραμμάτων Python. Με μερικές μόνο γραμμές κώδικα μπορούν να δημιουργήθούν διαγράμματα, ιστογράμματα, φάσματα ισχύος, γραφήματα στηλών, διάγραμματα διασποράς κλπ.

Η βασική εντολή της Matplotlib ειναι η plot() η οποία χρειάζεται ως μοναδικά ορίσματα δύο λίστες με τις xx', αντίστοιχα, yy' συντεταγμένες των σημείων που μας ενδιαφέρουν. Η προκαθορισμένη συμπεριφορά της συνάρτησης plot() είναι να ενώσει αυτά τα σημεία με μια συνεχόμενη γραμμή.

Στη συνέχεια παρουσιάζονται δύο γραφήματα των μετρήσεων απο τις δύο πλακέτες Arduino. Οι μετρήσεις αρχικά συλλέγονται απο την βάση και στη συνέχεια μέσω της Matplotlib οπτικοποιούνται σε διαγράμματα. Τα συγκεκριμένα διαγράμματα απεικονίζουν τις τιμές τις θερμοκρασίας στις 16/01/18 ανα ώρα.



Εικόνα 21 - Διάγραμμα Θερμοκρασίας - Χρόνου της πλακέτας Arduino στις 16-01-18



Εικόνα 22 - Διάγραμμα Θερμοκρασίας - Χρόνου της πλακέτας Wemos στις 16-01-18

Ο κώδικας με τον οποίο δημιουργούνται τα διαγράμματα, παρουσιάζεται στο **παράρτημα Α.**

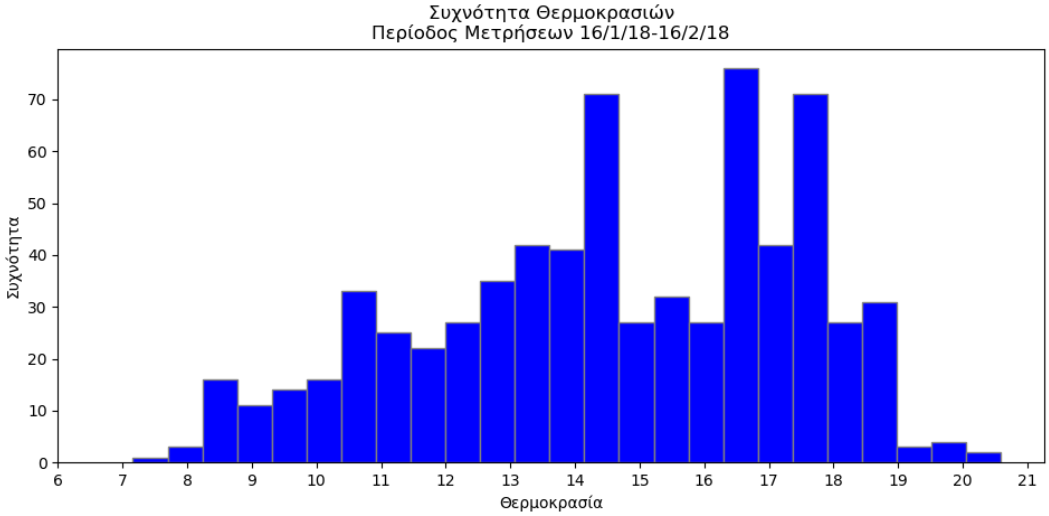
6.3 Επεξεργασία Δεδομένων

Η τεχνολογία επιτρέπει την αποθήκευση εξαιρετικά μεγάλου όγκου δεδομένων, ωστόσο τέτοιες συλλογές είναι άχρηστες αν δεν μπορούμε να εξάγουμε συγκεκριμένες πληροφορίες από τα δεδομένα αυτά.

Ιστόγραμμα Συχνοτήτων

Μια μέθοδός για να κατανοήσουμε τα δεδομένα, ώστε να παράγουμε σωστά συμπεράσματα για αυτά είναι η δημιουργία ιστογραμμάτων. Παρακάτω έχει δημιουργηθεί ένα ιστόγραμμα συχνοτήτων για τις μετρήσεις θερμοκρασίας που έγιναν από την πλακέτα Arduino W στο διάστημα 16/1/18 – 16/2/18 ανά μία ώρα. Από το παρακάτω ιστόγραμμα μπορούμε να εξάγουμε συμπεράσματα όπως ποια θερμοκρασία ήταν πιο συχνή ή πιο σπάνια, ποια ήταν η μικρότερη ή μεγαλύτερη θερμοκρασία κατά το διάστημα που έγιναν οι μετρήσεις.

Ο κώδικας με τον οποίο δημιουργήθηκε τα ιστόγραμμα συχνοτήτων παρουσιάζεται στο **παράρτημα Α.**



Εικόνα 23 - Ιστόγραμμα

Regression

Άλλη μια τεχνική για την εξαγωγή χρήσιμων συμπερασμάτων για τα δεδομένα που έχουν ήδη αποθηκευτεί, αλλά κυρίως για την πρόβλεψη μελλοντικών δεδομένων είναι η μέθοδος της Παλινδρόμησης (Data Regression).

Η μέθοδος της Παλινδρόμησης χρησιμοποιείτε στη στατιστική για τη δημιουργία μαθηματικών μοντέλων που εξετάζουν σχέσεις μεταξύ δύο ή περισσότερων μεταβλητών. Η ανάλυση παλινδρόμησης (regression analysis) είναι ο κλάδος της Στατιστικής που εξετάζει τη σχέση μεταξύ δύο ή περισσότερων μεταβλητών με σκοπό την πρόβλεψη μιας από αυτές μέσω των άλλων.

Για να βρεθεί το μοντέλο παλινδρόμησης, μια τεχνική που χρησιμοποιείται είναι αυτή του διαχωρισμού των δεδομένων που υπάρχουν ήδη, σε 2 ομάδες, τα δεδομένα εκτίμησης (training data) και τα δεδομένα πρόβλεψης (prediction data). Η πρώτη ομάδα χρησιμοποιείται για την εκτίμηση του μοντέλου παλινδρόμησης, ενώ η δεύτερη, χρησιμοποιώντας το μοντέλο που θα βρεθεί, χρησιμοποιείται για την πρόβλεψη των δεδομένων.

Στην παρούσα εργασία, για την εφαρμογή αυτών των τεχνικών έχουν χρησιμοποιηθεί οι μετρήσεις θερμοκρασίας που έγιναν από την πλακέτα Arduino W στο διάστημα 16/1/18–16/2/18 ανά μία ώρα. Χρησιμοποιήθηκαν τα ζεύγη χρόνος-θερμοκρασία, τα οποία χωρίστηκαν σε training και testing data σε αναλογία 80%-20%.

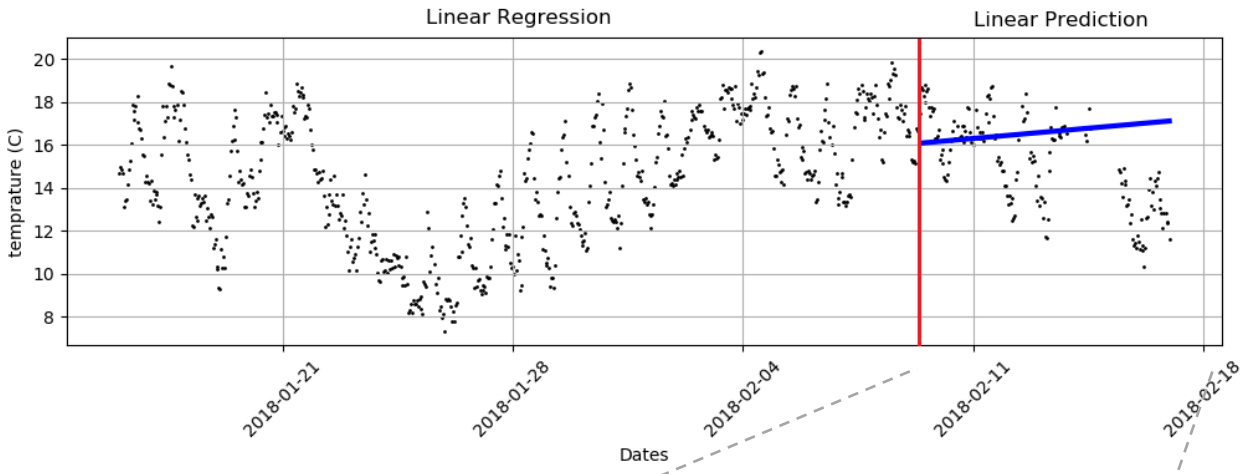
Σε επίπεδο προγραμματισμού, χρησιμοποιήθηκε η βιβλιοθήκη Scikit η οποία ειναι ένα machine learning API για προγραμματισμό σε γλώσσα Python.

Linear Regression - Scikit

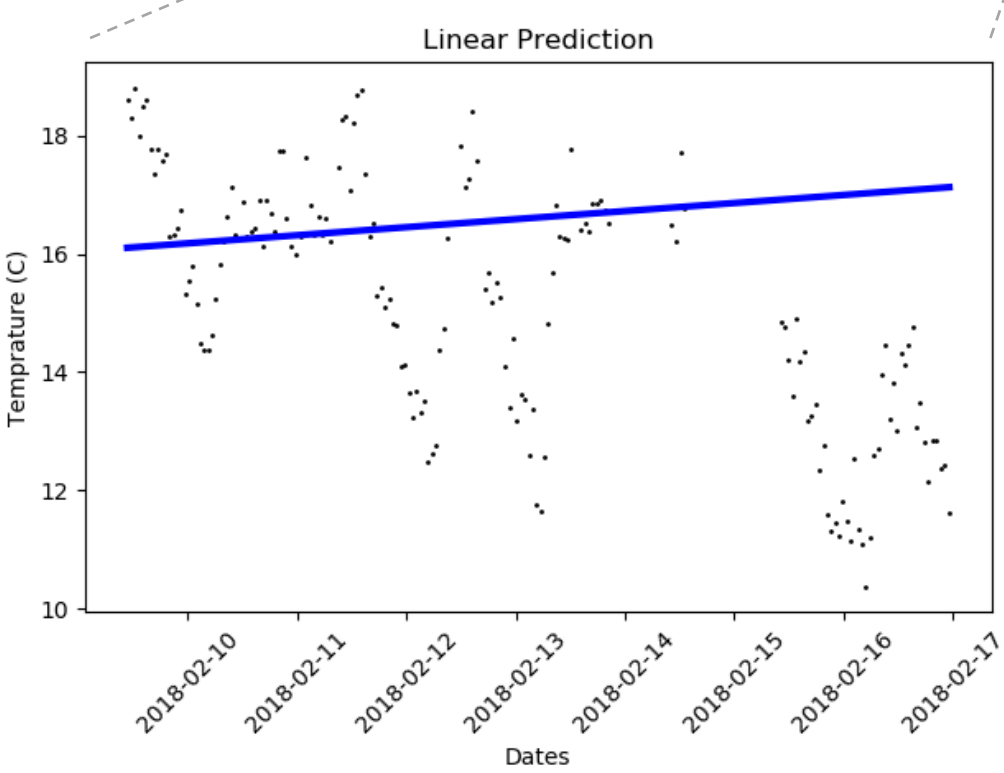
Η απλούστερη περίπτωση παλινδρόμησης είναι η γραμμική παλινδρόμηση (linear regression), κατά την οποία υπάρχει μόνο μια ανεξάρτητη μεταβλητή X και η εξαρτημένη μεταβλητή Y, η οποία μπορεί να προσεγγιστεί ικανοποιητικά από μία γραμμική συνάρτηση του X (παλινδρόμηση του Y πάνω στο X). Για την υλοποίηση χρησιμοποιήθηκε το Linear Regression της βιβλιοθήκης Scikit.

Από τις 16/01/18 μέχρι και τις 9/02/18 χρησιμοποιήθηκαν τα δεδομένα εκτίμησης (training data) ώστε να βρεθεί το μοντέλο παλινδρόμησης. Στη συνέχεια χρησιμοποιώντας το συγκεκριμένο μοντέλο προβλέπονται οι μελλοντικές θερμοκρασίες από τις 10/02/18 μέχρι και τις 17/02/18.

Στο γράφημα που ακολουθεί παρουσιάζονται σε διασπορά (μαύρες κουκίδες) τα δεδομένα που συλλέξαμε από την πλακέτα Arduino W, ενώ σε μπλε γραμμή οι θερμοκρασίες που προβλέπονται σύμφωνα με το μοντέλο Linear Regression.



Εικόνα 24 - Linear Regression & Prediction



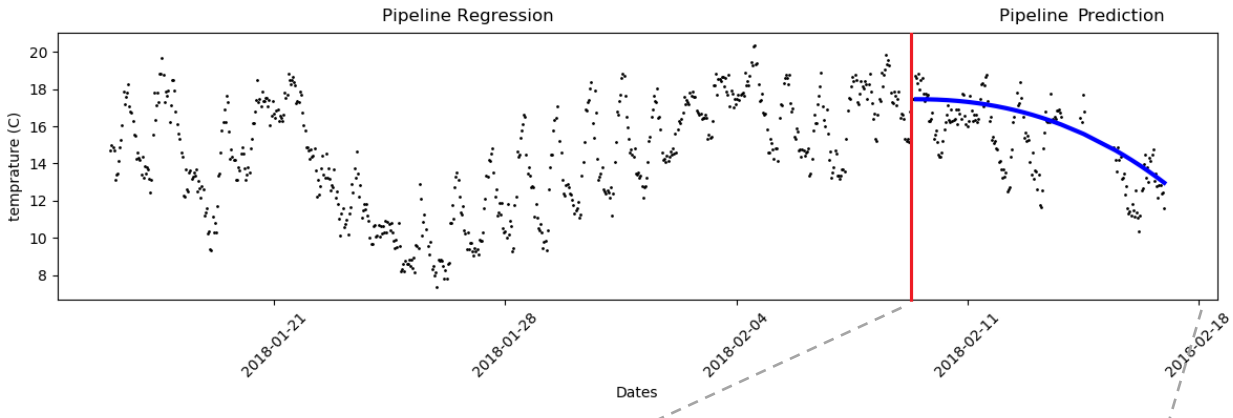
Εικόνα 25 – Linear Prediction

Όπως παρατηρούμε από το γράφημα το Linear regression, δημιουργείται μία γραμμική συνάρτηση, η οποία οδηγεί σε λανθασμένες προβλέψεις.

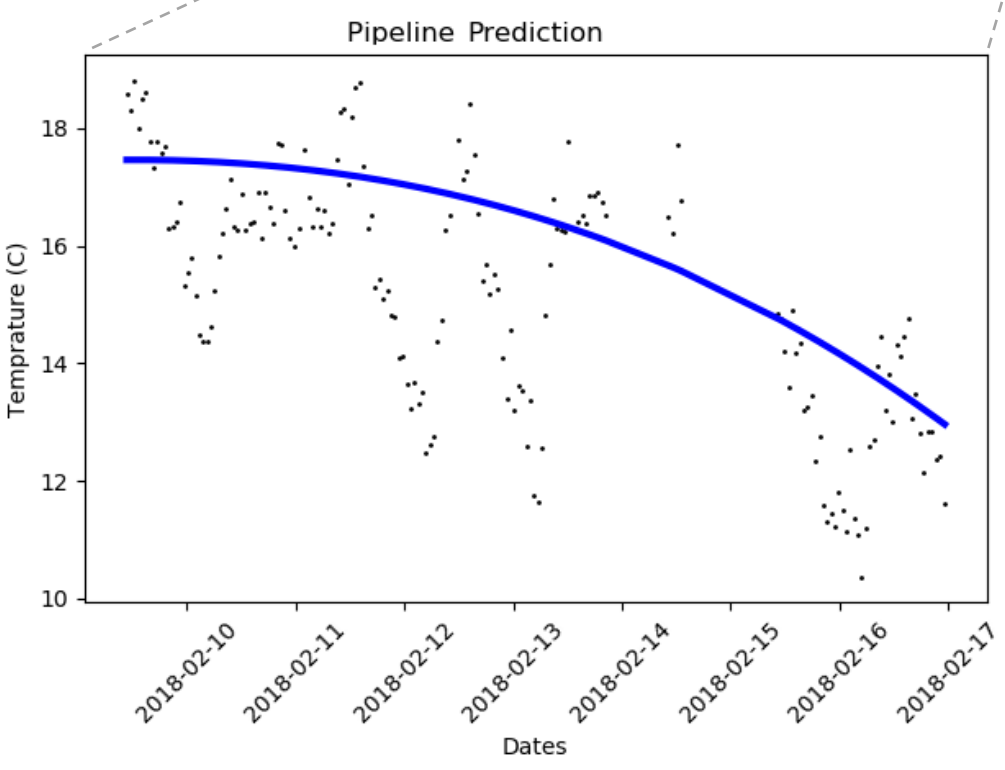
Pipeline Regression

Άλλη μια μέθοδος παλινδρόμησης που χρησιμοποιήθηκε στην εργασία είναι η Pipeline Regression. Είναι μια μη γραμμική παλινδρόμηση, η οποία δημιουργεί ένα πολυωνυμικό μοντέλο.

Στο γράφημα που ακολουθεί παρουσιάζονται σε διασπορά (μαύρες κουκίδες) τα δεδομένα που συλλέξαμε από την πλακέτα Arduino W, ενώ σε μπλε γραμμή οι θερμοκρασίες που προβλέπονται σύμφωνα με το μοντέλο Pipeline Regression.



Εικόνα 26 - Pipeline Regression - Prediction



Εικόνα 27 - Pipeline Regression - Prediction

Όπως παρατηρούμε από το γράφημα το Pipeline regression δημιουργεί μία πολυωνυμική συνάρτηση, η οποία οδηγεί σε προβλέψεις πολύ κοντά στις πραγματικές.

Ο κώδικας με τον οποίο έγινε χρήση των δύο μεθόδων παλινδρόμησης παρουσιάζεται στο **παράρτημα Α.**

Κεφάλαιο 7

7.1 Συμπεράσματα

Στη παρούσα εργασία δημιουργήθηκε ένα μικρό σύστημα καταγραφής μετρήσεων με χρήση αισθητήρων οι οποίοι είναι εγκατεστημένοι σε πλακέτες Arduino. Εκτός όμως από την παρουσίαση και τον τρόπο λειτουργίας του, μέσω της παρούσας εργασίας μπορούν να εξαχθούν ορισμένα χρήσιμα συμπεράσματα για τις τεχνικές που χρησιμοποιήθηκαν.

Αρχικά θα πρέπει να συνειδητοποιήσουμε ότι ένα IoT σύστημα βελτιώνει την καθημερινότητα του ανθρώπου. Απομακρυσμένος έλεγχος, εύκολη πρόσβαση στη πληροφόρηση και αυτοματοποιημένες λειτουργίες είναι μόνο μερικά από τα οφέλη που έχει ο άνθρωπος, συνεπώς τα IoT συστήματα είναι πλέον αλληλένδετα με την καθημερινότητα μας.

Από τεχνικής άποψης, ένα IoT σύστημα είναι εύκολα παραμετροποιήσιμο, είναι ελαφρύ, γρήγορο, χωρίς να απαιτεί πολλούς πόρους. Παράλληλα, η υλοποίηση του είναι οικονομική και παρέχει τη δυνατότητα κλιμάκωσης.

Από την άλλη μεριά, ένα IoT σύστημα δεν προσφέρει μεγάλη ασφάλεια αφού η πληροφορία διακινείται μέσα στο δίκτυο χωρίς μεγάλα επίπεδα ασφαλείας. Επίσης, η υλοποίηση του αρκετές φορές είναι πολύπλοκη αφού απαιτεί την χρήση και τον προγραμματισμό πολλών και διαφορετικών οντοτήτων.

7.2 Μελλοντική Εργασία και Επεκτάσεις

Μια μελλοντική επέκταση του IoT συστήματος θα μπορούσε να είναι η χρήση περισσότερων αισθητήρων. Υπάρχει μια μεγάλη γκάμα αισθητήρων, έτσι στο IoT σύστημα που κατασκευάστηκε στη παρούσα εργασία θα μπορούσε να εγκατασταθούν πάνω στις υπάρχουσες πλακέτες Arduino και Wemos, αισθητήρες υγρασίας, επιπέδου φωτός, επιπέδου θορύβου κλπ.

Παράλληλα, η δημιουργία ενός πιο ασφαλούς περιβάλλοντος για την μεταφορά δεδομένων από τους Publisher στον MQTT Broker θα μπορούσε να είναι μια ακόμη επέκταση της εργασίας. Για να επιτευχθεί κάτι τέτοιο, μία λύση είναι η κωδικοποίηση της πληροφορίας που μετακινείται από τις πλακέτες προς τον MQTT Broker και αντίστροφα.

Επίσης, θα ήταν αρκετά λειτουργικό η δημιουργία μιας web πλατφόρμας η οποία θα δίνει την δυνατότητα άμεσης οπτικοποίησης των δεδομένων που παράγονται και ελέγχου των ρυθμίσεων των πλακετών Arduino.

Τέλος για την εξαγωγή πιο χρήσιμων συμπερασμάτων για τα δεδομένα αλλά κυρίως για την καλύτερη πρόβλεψη νέων δεδομένων θα μπορούσε να χρησιμοποιηθεί η αρχιτεκτονική των νευρωνικών δικτύων.

Αναφορές

[1] https://en.wikipedia.org/wiki/Internet_of_things

[2] http://internetofthingsagenda.techtarget.com/definition/Internet-of-Things-IoT

[3] https://www.webopedia.com/TERM/I/internet_of_things.html

[4] https://iot-analytics.com/internet-of-things-definition

[5] http://lelantos.in/category/smart-world-2/page/5/

[6] https://www.cisco.com/c/en/us/solutions/collateral/service-provider/global-cloud-index-gci/white-paper-c11-738085.html

[7] https://spectrum.ieee.org/tech-talk/telecom/internet/popular-internet-of-things-forecast-of-50-billion-devices-by-2020-is-outdated

[8] http://www.ijste.org/articles/IJSTEV3I11049.pdf

[9] https://www.rfc-editor.org/rfc/rfc7452.txt

[10] https://www.kernelsphere.com/four-internet-things-communications-models/

[11] https://www.kernelsphere.com/four-internet-things-communications-models/

[12] https://spectrum.ieee.org/tech-talk/telecom/internet/popular-internet-of-things-forecast-of-50-billion-devices-by-2020-is-outdated

[13] https://www.emc.com/collateral/analyst-reports/idc-the-digital-universe-in-2020.pdf

[14] https://www.ghdonline.org/uploads/big-data-in-healthcare_B_Kaplan_2012.pdf

[15] https://en.wikipedia.org/wiki/Data_analysis

[16] https://el.wikipedia.org/wiki/Εξόρυξη_δεδομένων

[17] https://www.sas.com/el_gr/insights/big-data/data-visualization.html

[18] http://opensourceforu.com/2017/07/sitewhere-open-platform-connected-devices/

[19] http://documentation.sitewhere.io

[20] http://MQTT.org

[21] http://docs.oasis-open.org/MQTT/MQTT/v3.1.1/os/MQTT-v3.1.1-os.html

[22] https://www.hivemq.com/blog/introducing-the-MQTT-security-fundamentals

[23] www.arduino.cc

[24] https://el.wikipedia.org/wiki/Arduino

[25] https://www.tutorialspoint.com/arduino/arduino_board_description.htm

[26] https://www.wemos.cc/

[27] https://www.arduino.cc/en/Guide/Environment

[28] https://matplotlib.org/

Παράρτημα Α

Στο συγκεκριμένο παράρτημα έχουν τοποθετηθεί όλα τα τμήματα κώδικα που δημιουργήθηκαν για την υλοποίηση της παρούσας εργασίας. Παρουσιάζεται ο τρόπος προγραμματισμού των πλακετών Arduino Uno (ως publisher) και Wemos (ως publisher & subscriber). Παράλληλα βρίσκονται όλα τα scripts (python) που δημιουργήθηκαν για την ανάκτηση, οπτικοποίηση και επεξεργασία των δεδομένων.

Κώδικας 1: Προγραμματισμός πλακέτας Arduino Uno ως Publisher

```
#include <DHT.h>
#include <Ethernet.h>
#include <Dns.h>
#include <PubSubClient.h>
#include <TimeLib.h>
#include <ArduinoJson.h>

// Variables
unsigned int localPort = 8888;                              // Local port to listen  UDP packets
const char*  myTopic= "DataManagement/input/arduino_vc";    // Topic
int i;

byte mac[]    = {  0xDE, 0xED, 0xBE, 0xFE, 0xFE, 0xED };
IPAddress ip(192, 168, 1, 156);                             // Arduino IP
IPAddress MQTTserver(195, 134, 79, 240);                    // MQTT Server

//Constants
#define DHTPIN 2        // The pin which the DHT Sensor is connected
#define DHTTYPE DHT22   // DHT 22

// NTP Servers
char ntpServerName[] = "us.pool.ntp.org";  // Ntp Server Name
const int timeZone = 2;                   // Greece TimeZone

const int NTP_PACKET_SIZE = 48; // NTP time stamp is in the first 48 bytes of the message
byte packetBuffer[ NTP_PACKET_SIZE]; //buffer to hold incoming and outgoing packets

EthernetUDP Udp;
DNSClient dnClient;
EthernetClient ethClient;
DHT dht(DHTPIN, DHTTYPE); // Initialize DHT sensor

//Function headers
void callback(char* topic, byte* payload, unsigned int length);
void sendNTPpacket(IPAddress &address);
String ConvertToXX(int digit);
```

```
37.
38.
39.void callback(char* topic, byte* payload, unsigned int length) {
40.  }
41.
42.PubSubClient MQTTClient(MQTTserver, 1883, callback, ethClient);
43.
44.void setup()
45.{
46.  Ethernet.begin(mac, ip);  // Configure Ethernet using DHCP
47.  Serial.begin(9600);       // Open Serial View
48.  dht.begin();
49.
50.  //For ntp take back ip
51.  dnClient.begin(Ethernet.dnsServerIP());
52.
53.  //Synchronizing to NTP Server
54.  Serial.println("1. ~~~~~~~~~~~~~~~~~~~~Starting UDP~~~~~~~~~~~~~~~~~~~~~~~~");
55.  Udp.begin(localPort);
56.  Serial.println("Waiting for sync to NTP Server");
57.  setSyncProvider(getNtpTime);
58.  while (timeStatus() == timeNotSet){
59.    Serial.print(".");
60.    setSyncProvider(getNtpTime);
61.  }
62.  setSyncInterval(300);
63.
64.  Serial.println("2. ~~~~~~~~~~~~~~~~~~~~Starting MQTT~~~~~~~~~~~~~~~~~~~~~~~~");
65.  i=0;
66.  MQTTClient.setServer(MQTTserver, 1883);
67.  if (MQTTClient.connect("", "", "")){
68.    Serial.println("Connected to MQTT");
69.    MQTTClient.setCallback(callback);
70.  }
71.  else
72.    Serial.println("Unable to connect to MQTT");
73.  dht.begin();
74.}
75.
76.void loop()
77.{
78.  float DTH22temp;
79.  char JSONmessageBuffer[100];
80.  StaticJsonBuffer<150> JSONbuffer;
81.  JsonObject& JSONencoder = JSONbuffer.createObject();
82.
83.  if (MQTTClient.connect("", "", "")) {
```

```
84.
85.     //---------- GET TEMPERATURE FROM SENSOR -------------------
86.     DTH22temp= dht.readTemperature();
87.     while (isnan(DTH22temp)){
88.       DTH22temp=  dht.readTemperature();
89.     }
90.
91.     //----------------CREATE JSON MESSAGE--------------------
92.     JSONencoder["id"] = "arduinoE";
93.     JSONencoder["value"] = String(DTH22temp);
94.     JSONencoder["Timestamp"] = String(year())+"-"+ ConvertToXX(month())+"-
   "+ConvertToXX(day())+" "+ConvertToXX(hour()) +":"+ConvertToXX(minute()) +":"+ Convert
   ToXX(second()) +" "+ "+0200";
95.     JSONencoder.printTo(JSONmessageBuffer, sizeof(JSONmessageBuffer));
96.
97.     //----------------PRINT JSON MESSAGE to SERIAL VIEW--------------------
98.     Serial.print("Json Message: ");
99.     Serial.println(JSONmessageBuffer);
100.
101.      //----------------PUBLISH JSON MESSAGE--------------------
102.      MQTTClient.publish(myTopic, JSONmessageBuffer);
103.
104.      Ethernet.maintain();
105.      delay(60000); //Wait 1min/60sec
106.   }
107. }
108.
109. // Send NTP request to the time server
110. void sendNTPpacket(IPAddress &address) {
111.   // set all bytes in the buffer to 0
112.   memset(packetBuffer, 0, NTP_PACKET_SIZE);
113.   // Initialize values needed to form NTP request
114.   // (see URL above for details on the packets)
115.   packetBuffer[0] = 0b11100011;   // LI, Version, Mode
116.   packetBuffer[1] = 0;     // Stratum, or type of clock
117.   packetBuffer[2] = 6;     // Polling Interval
118.   packetBuffer[3] = 0xEC;  // Peer Clock Precision
119.   // 8 bytes of zero for Root Delay & Root Dispersion
120.   packetBuffer[12]  = 49;
121.   packetBuffer[13]  = 0x4E;
122.   packetBuffer[14]  = 49;
123.   packetBuffer[15]  = 52;
124.
125.   // all NTP fields have been given values, now
126.   // you can send a packet requesting a timestamp:
127.   Udp.beginPacket(address, 123); //NTP requests are to port 123
128.   Udp.write(packetBuffer, NTP_PACKET_SIZE);
```

```
  Udp.endPacket();
}


time_t getNtpTime()
{

  IPAddress ntpServerIP; // NTP server's ip address
  dnClient.getHostByName(ntpServerName,ntpServerIP);
  Serial.print("ntp Server IP: ");
  Serial.println(ntpServerIP);
  sendNTPpacket(ntpServerIP);
  uint32_t beginWait = millis();
  while (millis() - beginWait < 1500) {
    int size = Udp.parsePacket();
    if (size >= NTP_PACKET_SIZE) {
      Serial.println("Receive NTP Response");
      Udp.read(packetBuffer, NTP_PACKET_SIZE);  // read packet into the buffer
      unsigned long secsSince1900;
      // convert four bytes starting at location 40 to a long integer
      secsSince1900 =  (unsigned long)packetBuffer[40] << 24;
      secsSince1900 |= (unsigned long)packetBuffer[41] << 16;
      secsSince1900 |= (unsigned long)packetBuffer[42] << 8;
      secsSince1900 |= (unsigned long)packetBuffer[43];
      return secsSince1900 - 2208988800UL + timeZone * SECS_PER_HOUR;
    }
  }
  Serial.println("No NTP Response :-(");
  return 0; // return 0 if unable to get the time
}


//Convert Integer to String
String ConvertToXX(int digit)
{
    String XX = String(digit);
    if (digit<10)
      XX = "0"+ String(digit);
    return XX;
}
```

Κώδικας 2: Προγραμματισμός πλακέτας Wemos ως Publisher

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
```

```
#include <ArduinoJson.h>
#include <TimeLib.h>
#include <WiFiUdp.h>
#include <DHT.h>

#define DHTPIN D2
#define DHTTYPE DHT22
DHT dht (DHTPIN, DHTTYPE);

const char* ssid     = "piroskaki_";
const char* password = "6942422657";
const char* myTopic = "DataManagement/input/arduino_vc";

// Variables
float DTH22temp;
int value=0;
int i=0;

IPAddress ip(192, 168, 1, 205);
IPAddress MQTTserver(195, 134, 79, 240);  // MQTT Server


// NTP Servers:
static const char ntpServerName[] = "us.pool.ntp.org";
const int timeZone = 2;         // Greece TimeZone
unsigned int localPort = 8888;  // local port to listen for UDP packets

time_t getNtpTime();
void sendNTPpacket(IPAddress &address);

WiFiUDP Udp;
WiFiClient ethClient;
PubSubClient MQTTClient(ethClient);


void setup() {
  Serial.begin(115200);
  Serial.println("1. ~~~~~~~~~~~~~~~ DHT Version ~~~~~~~~~~~~~~~~~~~~~~~");
  Serial.println("Starting DHT");
  dht.begin();

  //Connecting to a WiFi network
  Serial.println("2. ~~~~~~~~~~~~~~~ Connecting to WiFi ~~~~~~~~~~~~~~~~");
  Connect_WIFI();

  //Sync to NTP Server
  Serial.println("3. ~~~~~~~~~~~~~~ Starting UDP~~~~~~~~~~~~~~~~~~~~~~~");
  Udp.begin(localPort);
  Serial.print("Local port: ");
  Serial.println(Udp.localPort());
  Serial.println("waiting for sync");
  setSyncProvider(getNtpTime);
  setSyncInterval(300);
  Serial.println("");

  //Connecting to MQTT
  Serial.println("4. ~~~~~~~~~~~~~ Connecting to MQTT ~~~~~~~~~~~~~~~~~~");
  MQTTClient.setServer(MQTTserver, 1883);
  if (MQTTClient.connect("arduinoClient")) {
    Serial.println("MQTT connected");
  }else{
```

```
64.     Serial.println("Failed");
65.   }
66. }
67.
68. void loop() {
69.
70.   char JSONmessageBuffer[100];
71.   StaticJsonBuffer<300> JSONbuffer;
72.   JsonObject& JSONencoder = JSONbuffer.createObject();
73.   value++;
74.   if (WiFi.status() != WL_CONNECTED) {
75.     Connect_WIFI();
76.   }
77.
78.   if (MQTTClient.connect("arduinoClient")) {
79.
80.     //--- Get Temperature from DTH22 Sensor ---//
81.     DTH22temp=  dht.readTemperature();
82.     while (isnan(DTH22temp)){
83.       DTH22temp=  dht.readTemperature();
84.       }
85.
86.     //--- Create JSON message ---//
87.     JSONencoder["id"] = "arduinoW";
88.     JSONencoder["value"] = String(DTH22temp);
89.     JSONencoder["Timestamp"] = String(year()) +"-"+ConvertToXX(month()) +"-
   "+ConvertToXX(day()) +" "+ConvertToXX(hour()) +":"+ConvertToXX(minute()) +":"+ Conver
   tToXX(second()) +" "+ GettimeZone(timeZone);//"+0200";
90.     JSONencoder.printTo(JSONmessageBuffer, sizeof(JSONmessageBuffer));
91.
92.     //--- Print JSON message to Serial View  ---//
93.     Serial.print("Json Message: ");
94.     Serial.println(JSONmessageBuffer);
95.
96.     //--- Publish JSON message ---//
97.     MQTTClient.publish(myTopic, JSONmessageBuffer);
98.
99.     delay(60000);
100.   }
101. }
102.
103. //--- Function Connect board to Wifi ---//
104. void Connect_WIFI(){
105.     Serial.print("Connecting to WEP network, SSID: ");
106.     Serial.println(ssid);
107.     while ( WiFi.begin(ssid, password) != WL_CONNECTED) {
108.       Serial.print(".");
109.       delay(100);
110.     }
111.     Serial.println("\nWiFi connected");
112. }
113.
114. //--- Function convert digits to String ---//
115. String ConvertToXX(int digit){
116.     String XX = String(digit);
117.     if (digit<10)
118.       XX = "0"+ String(digit);
119.     return XX;
120. }
121.
122.
```

```
String GettimeZone(int zone){
  String myZone;
  if (zone>=0 && zone<10)
      myZone = "+0" + String(zone) + "00";
  else if (zone>=10)
      myZone = "+" + String(zone) + "00";
  else if (zone<0 && zone>-10)
      myZone = "-0" + String(zone) + "00";
  else if (zone<=-10)
      myZone = "-" + String(zone) + "00";

  return myZone;
}


/*-------- NTP code ----------*/
const int NTP_PACKET_SIZE = 48; // NTP time is in the first 48 bytes of message
byte packetBuffer[NTP_PACKET_SIZE]; //buffer to hold incoming & outgoing packets
time_t getNtpTime()
{
  IPAddress ntpServerIP; // NTP server's ip address
  WiFi.hostByName(ntpServerName, ntpServerIP);
  Serial.print(ntpServerName);
  Serial.print(": ");
  Serial.println(ntpServerIP);
  sendNTPpacket(ntpServerIP);
  uint32_t beginWait = millis();
  while (millis() - beginWait < 1500) {
    int size = Udp.parsePacket();
    if (size >= NTP_PACKET_SIZE) {
      Serial.println("Receive NTP Response");
      Udp.read(packetBuffer, NTP_PACKET_SIZE);  // read packet into the buffer
      unsigned long secsSince1900;
      // convert four bytes starting at location 40 to a long integer
      secsSince1900 =  (unsigned long)packetBuffer[40] << 24;
      secsSince1900 |= (unsigned long)packetBuffer[41] << 16;
      secsSince1900 |= (unsigned long)packetBuffer[42] << 8;
      secsSince1900 |= (unsigned long)packetBuffer[43];
      return secsSince1900 - 2208988800UL + timeZone * SECS_PER_HOUR;
    }
  }
  Serial.println("No NTP Response :-(");
  return 0; // return 0 if unable to get the time
}

// send an NTP request to the time server at the given address
void sendNTPpacket(IPAddress &address)
{
  // set all bytes in the buffer to 0
  memset(packetBuffer, 0, NTP_PACKET_SIZE);
  // Initialize values needed to form NTP request
  // (see URL above for details on the packets)
  packetBuffer[0] = 0b11100011;   // LI, Version, Mode
  packetBuffer[1] = 0;     // Stratum, or type of clock
  packetBuffer[2] = 6;     // Polling Interval
  packetBuffer[3] = 0xEC;  // Peer Clock Precision
  // 8 bytes of zero for Root Delay & Root Dispersion
  packetBuffer[12] = 49;
  packetBuffer[13] = 0x4E;
```

```
  packetBuffer[14] = 49;
  packetBuffer[15] = 52;
  // all NTP fields have been given values, now
  // you can send a packet requesting a timestamp:
  Udp.beginPacket(address, 123); //NTP requests are to port 123
  Udp.write(packetBuffer, NTP_PACKET_SIZE);
  Udp.endPacket();
}
```

Κώδικας 3: Προγραμματισμός πλακέτας Wemos ως Subscriber

```
#include <PubSubClient.h>
#include <ESP8266WiFi.h>
#include <ArduinoJson.h>

// initialize pins
int LED_pin = D5;         // The pin which Alarm LED  is connected

int Speaker_pin = D4;     // The pin which Speaker Alarm  is connected
int LED_wifi_pin = D3;    // The pin which the Wifi Alarm is connected

const char* ssid = "piroskaki_";
const char* password = "6942422657";
char* topic = "DataManagement/input/arduino_vc";    // Topic
char* server = "195.134.79.240";                    // MQTT server ip
char message_buff[100];                             // storage buffer
int status = WL_IDLE_STATUS;                        // the Wifi status
int  i;

WiFiClient wifiClient;

void callback(char* topic, byte* payload, unsigned int length) {
  i = 0;

  // Create character buffer - Ending with \0
  for(i=0; i<length; i++) {
    message_buff[i] = payload[i];
  }
  message_buff[i] = '\0';
  StaticJsonBuffer<300> JSONBuffer;                     // Memory
  JsonObject& parsed = JSONBuffer.parseObject(payload); // Parse message

  // Check for errors in parsing
  if (!parsed.success()) {
    Serial.println("Parsing failed");
    delay(5000);
    return;
  }
  //Get values from buffer
  const char * sensorType = parsed["id"]; // Get sensor type value
  float value = parsed["value"];          // Get value of sensor measurement

  //Printing JSON to SERIAL VIEW
  String msgString = String(message_buff);
  Serial.println("Payload: " + msgString);

  //Check the values and create ALERTS
```

```
  if ((value>18) or (value<10)){
      Alarm_LED(LED_pin);
      Alarm_SPEAKER(Speaker_pin);
  }
}

PubSubClient MQTTClient(server, 1883, callback, wifiClient);

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

  pinMode(LED_wifi_pin,OUTPUT);
  digitalWrite(LED_wifi_pin, LOW);

  // -------- Connect to Wifi -------- //
  Serial.println("\n1. ~~~~~~~~~~~~ Connect to Wifi ~~~~~~~~~~~~~~~~~~~~~~~");
  Connect_WIFI();

  // -------- Connect & Subscribe to MQTT Broker  -------- //
  Serial.println("\n2. ~~~~~~~~~~~~ Connect to MQTT ~~~~~~~~~~~~~~~~~~~~~~~");
  Connect_MQTT();

  Serial.println("\n3. ~~~~~~~~~~~~ Waiting for receiving JSON messages ~~~");
}

void loop() {

  if (WiFi.status() == WL_CONNECTED)
  {
    MQTTClient.loop();
    digitalWrite(LED_wifi_pin, HIGH);
  }
  else{
      digitalWrite(LED_wifi_pin, LOW);
      WiFi.setOutputPower(0);
      Connect_WIFI();
      Connect_MQTT();
  }
}

//--- Function Enable LED Alarm ---//
void Alarm_LED(int pin){
   for(i=0; i<10; i++) {
      pinMode(pin,OUTPUT);
      digitalWrite(pin, HIGH);
      delay(1000);
      digitalWrite(pin, LOW);
      delay(700);
   }
}

//--- Function Enable Speaker Alarm ---//
void Alarm_SPEAKER(int pin){
      for(i=0; i<10; i++) {
         pinMode(pin,OUTPUT);
         delay(1000);
         digitalWrite(pin, HIGH);
         delay(700);
         digitalWrite(pin, LOW);
      }
   }
```

```
108.
109. //--- Function Connect board to Wifi ---//
110. void Connect_WIFI(){
111.     Serial.print("Connecting to WEP network, SSID: ");
112.     Serial.println(ssid);
113.     while ( WiFi.begin(ssid, password) != WL_CONNECTED) {
114.       Serial.print(".");
115.       delay(100);
116.     }
117.     Serial.println("\nWiFi connected");
118.
119. }
120.
121. //--- Function Connect board to MQTT Broker ---//
122. void Connect_MQTT(){
123.   while (!MQTTClient.connect("arduinoClient"))  {}
124.   MQTTClient.subscribe(topic);
125.   Serial.println("Connected to MQTT and Subscribed");
126. }
```

Κώδικας 4: Decoder

```
1.  import com.fasterxml.jackson.databind.*;
2.  import com.sitewhere.rest.model.device.communication.*;
3.  import com.sitewhere.rest.model.device.event.request.*;
4.  import com.sitewhere.spi.device.event.request.*;
5.  // import com.java.text.*;
6.  // import com.java.util.*;
7.
8.  // Create String from payload and parse as Jackson JsonNode.
9.
10. def message = new String(payload);
11. def mapper = new ObjectMapper()
12. def json = mapper.readTree(message)
13.
14. // Build a request for adding a new measurements event.
15. def decoded = new DecodedDeviceRequest<IDeviceMeasurementsCreateRequest>()
16.
17. // Add measurements to event create request if they are present.
18. def arduinoUno = new DeviceMeasurementsCreateRequest()
19.
20. // Parse JSON.
21. def jsonId = json.path("id")
22. if (jsonId == null) {
23.     throw new RuntimeException("No id value passed.")
24. }
25. decoded.setHardwareId(jsonId.asText());
26.
27. def jsonValue = json.path("value")
28. if (jsonValue == null) {
29.     throw new RuntimeException("No measurement value passed.")
30. }
31. arduinoUno.addOrReplaceMeasurement("temp_"+jsonId.asText(), jsonValue.asDouble());
32.
33. def jsonDATE = json.path("Timestamp");
34.
35. String pattern = "yyyy-MM-dd HH:mm:ss Z";
36. def date = new Date().parse(pattern, jsonDATE.asText());
37.
```

```
38. arduinoUno.setEventDate(date);
39. decoded.setRequest(arduinoUno)
40.
41. events.add(decoded);
```

Κώδικας 5: Python Scripts – Συλλογή δεδομένων απο την βάση και αποθήκευση σε αρχείο Measurements.csv (select_all.py)

```
1.  import argparse
2.  from influxdb import InfluxDBClient
3.  import json
4.  import csv
5.  from pprint import pprint
6.  import pandas as pd
7.  import datetime
8.  import pylab
9.  import numpy as np
10.
11. host='195.134.79.240'
12. port=8086
13. user = 'Vangelis'
14. password = 'arduino2017'
15. dbname = 'IoT_test'
16.
17. query = 'select  time as T, "mx:temp_arduinoE" as E,"mx:temp_arduinoW" as W from ev
    ents order by time'
18. client = InfluxDBClient(host, port, user, password, dbname, verify_ssl=False)
19. result = client.query(query)
20.
21. #---------------------Remove extra characher from the Query Result----------------
    -------
22. my = repr(result)
23. my = my.replace("ResultSet({'('events', None)': ", "")
24. my = my.replace("})", "")
25. my = my.replace('', '')
26. my = my.replace("'", "\"")
27. my = my.replace("None", "\"None\"")
28. #-------------------------------------------------------------------------------
    -------
29. mydict = json.loads(my)
30. tempE = []
31. tempW = []
32. timeE = []
33. timeW = []
34. #----------------Create CSV File + Create Data Forma for Diagram (Matplot)---------
    --------
35. outputFile = open('d:\Measurements.csv', 'w')
36. f = csv.writer(outputFile)
37. f.writerow(['\"Time,ArduinoE,ArduinoW\"'])
38. for myrow in mydict:
39.     f.writerow([myrow['T']]+[myrow['E']]+[myrow['W']])
40.     if (repr(myrow['W']) != "\'None\'"):
41.         t = myrow['T']
42.         t = t.replace("Z", "")
43.         t = t.replace("T", "  ")
44.         timeW.append(t)
45.         tempW.append(myrow['W'])
46.     if (repr(myrow['E']) != "\'None\'"):
47.         t = myrow['T']
```

```
        t = t.replace("Z", "")
        t = t.replace("T", "  ")
        timeE.append(t)
        tempE.append(myrow['E'])
outputFile.close()
```

Κώδικας 6: Matplotlib Python library

Ανάκτηση Δεδομένων του Arduino Ethernet και Δημιουργία διαγραμμάτος

```
import argparse
import random
from influxdb import InfluxDBClient
import json
import csv

from pprint import pprint
import pandas as pd

import datetime

import pylab
import matplotlib.pyplot as plt
import matplotlib.pyplot as pltE
import matplotlib.pyplot as pltT
import matplotlib.pyplot as pltW
import numpy as np


#folder = "d:/"
metric = "server_data.cpu_idle"
host='195.134.79.240'
port=8086
user = 'Vangelis'
password = 'arduino2017'
dbname = 'IoT_test'

query = 'select  time as T, "mx:temp_arduinoE" as E,"mx:temp_arduinoW" as W from ev
   ents   WHERE "mx:temp_arduinoE" >-
   100 and time >= 1516060801000ms and time <= 1516147201000ms order by time'
client = InfluxDBClient(host, port, user, password, dbname, verify_ssl=False)
result = client.query(query)

#---------------------Remove extra characher from the Query Result-----------------

my = repr(result)
my = my.replace("ResultSet({'('events', None)': ", "")
my = my.replace("})", "")
my = my.replace('', '')
my = my.replace("'", "\"")
my = my.replace("None", "\"None\"")
#print(my)
#--------------------------------------------------------------------------------------


#Skip n rows in csv file
n = 50

mydict = json.loads(my)
tempE = []
```

```
timeE = []

#---------------- Create Data Forma for Diagram (Matplot)-------------------------
i=0
for myrow in mydict:
    i=i+1
    if (i % n == 0):
        if (repr(myrow['E']) != "\'None\'"):
            t = myrow['T']
            t = t.replace("Z", "")
            t = t.replace("T", "  ")
            timeE.append(t)
            tempE.append(myrow['E'])

#----------------Create Ethernet Plot + Pdf file with the Diagram ----------------
fig = pltT.figure(figsize=(15,7))
ax = fig.add_subplot(111)
locs, labels = pltT.xticks()
pltT.setp(labels, rotation=90)
pltT.xlabel('Χρόνος (EEEE:MM:HH ΩΩ:ΛΛ:ΔΔ)', color="red")
pltT.ylabel('Θερμοκρασία(C)', color="red")
pltT.grid(True)
pltT.title('Arduino Ethernet:\n Διάγραμμα Θερμοκρασίας - Χρόνου')
pltT.tick_params(axis='both', which='major', labelsize=7)
pltT.gcf().subplots_adjust(bottom=0.3)

ax.plot(timeE, tempE, marker='o', linestyle='--')
from matplotlib.backends.backend_pdf import PdfPages
pp = PdfPages('d:\Arduino_Ethernet.pdf')
plt.savefig(pp, format='pdf')
pp.close()
plt.show()


#-------------Create TXT File and Fill with data---------------------------------


#-------------Replace charachers to create lines---------------------------------

l= repr(result)
l= l.replace("}, {", "}\n{")
l= l.replace(": [{", ": [{")
l= l.replace("}]})", "}\n]})")
l= l.replace("None", "None ")
l= l.replace("ResultSet({'('events', None )': [", "")
l= l.replace("]})", "")
now = datetime.datetime.now()

#-------------Input Data to File-------------------------------------------------
try:
    file = open('d:\pythondataE.txt', 'w')
    file.write(str(now))
    file.write("\n")
    file.write(l)
    file.close()
 except IOError:
     print("Error: File does not appear to exist.")
```

Κώδικας 7: Matplotlib Python library

Ανάκτηση Δεδομένων της πλακέτας Wemos και Δημιουργία διαγραμμάτος

```
1. import argparse
2. import random
3. from influxdb import InfluxDBClient
4. import json
5. import csv
6.
7. from pprint import pprint
8. import pandas as pd
9.
10. import datetime
11.
12. import pylab
13. import matplotlib.pyplot as plt
14. import matplotlib.pyplot as pltE
15. import matplotlib.pyplot as pltT
16. import matplotlib.pyplot as pltW
17. import numpy as np
18.
19.
20. host='195.134.79.240'
21. port=8086
22. user = 'Vangelis'
23. password = 'arduino2017'
24. dbname = 'IoT_test'
25.
26. query = 'select  time as T, "mx:temp_arduinoE" as E,"mx:temp_arduinoW" as W from ev
    ents   WHERE "mx:temp_arduinoW" >-
    100 and time >= 1516060801000ms and time <= 1516147201000ms order by time'
27. client = InfluxDBClient(host, port, user, password, dbname, verify_ssl=False)
28. result = client.query(query)
29.
30. #---------------------Remove extra characher from the Query Result-----------------

31. my = repr(result)
32. my = my.replace("ResultSet({'('events', None)': ", "")
33. my = my.replace("})", "")
34. my = my.replace('', '')
35. my = my.replace("'", "\"")
36. my = my.replace("None", "\"None\"")
37. #-------------------------------------------------------------------------------

38.
39. #Skip n rows in csv file
40. n = 50
41. mydict = json.loads(my)
42. tempW = []
43. timeW = []
44.
45. #----------------Create CSV File + Create Data Forma for Diagram (Matplot)---------
    ---------------------
46. i=0
47. for myrow in mydict:
48.     i=i+1
49.     if (i % n == 0):
50.         if (repr(myrow['W']) != "\'None\'"):
```

```
            t = myrow['T']
            t = t.replace("Z", "")
            t = t.replace("T", "  ")
            timeW.append(t)
            tempW.append(myrow['W'])

#----------------Create Wifi Plot ----------------------------------------
fig = pltT.figure(figsize=(15,7))
ax = fig.add_subplot(111)
locs, labels = pltT.xticks()
pltT.setp(labels, rotation=90)
pltT.xlabel('Χρόνος (EEEE:MM:HH ΩΩ:ΛΛ:ΔΔ)', color="red")
pltT.ylabel('Θερμοκρασία(C)', color="red")
pltT.grid(True)
pltT.title('Arduino Wifi:\n Διάγραμμα Θερμοκρασίας - Χρόνου')
pltT.tick_params(axis='both', which='major', labelsize=7)
pltT.gcf().subplots_adjust(bottom=0.3)
#Data
ax.plot(timeW, tempW, marker='o', linestyle='--', color="orange")
from matplotlib.backends.backend_pdf import PdfPages
pp = PdfPages('d:\Arduino_Wifi.pdf')
plt.savefig(pp, format='pdf')
pp.close()
plt.show()

#-------------Create TXT File and Fill with data----------------------------------


#-------------Replace charachers to create lines----------------------------------

l= repr(result)
l= l.replace("}, {", "}\n{")
l= l.replace(": [{", ": [{")
l= l.replace("}]})", "}\n]})")
l= l.replace("None", "None ")
l= l.replace("ResultSet({'('events', None )': [", "")
l= l.replace("]})", "")
now = datetime.datetime.now()
#-------------Input Data to File-------------------------------------------------
-
try:
    file = open('d:\pythondataW.txt', 'w')
    file.write(str(now))
    file.write("\n")
    file.write(l)
    file.close()
except IOError:
    print("Error: File does not appear to exist.")
```

Κώδικας 8: Ιστόγραμμα

```
import datetime
import numpy as np
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn import datasets, linear_model
from pandas import DataFrame, Series
from sklearn.linear_model import Ridge
```

```
from sklearn.preprocessing import PolynomialFeatures
from sklearn.pipeline import make_pipeline

#Skip n rows in csv file
n = 120
num_lines = sum(1 for l in open("D:\Measurements.csv"))
print("num lines:")
print(num_lines)
skip_idx = [x for x in range(1, num_lines) if x % n != 0]

#Open CVS file
Data = pd.read_csv("D:\Measurements.csv", parse_dates=[0], skiprows=skip_idx )
Date_1970=datetime.datetime(1970,1,1,0,0,0)

#Convert Timestampt to SEC (from 1/1/1970)
Data['Time_Sec']= (pd.to_datetime(Data['Time'])-Date_1970) /np.timedelta64(1,'s')

#Take Data (Date & Temperature) from pandas
Date = Data.ix[:,3]
Temperature = Data.ix[:,2]

# Split the Data into training/testing sets 80% / 20%
Date_train = Date[1:len(Date)*80//100]
Temperature_train = Temperature[1:len(Temperature)*80//100]
Date_test = Date[len(Date)*80//100:]
Temperature_test= Temperature[len(Temperature)*80//100:]

#Create 2D Array
Date = np.array(Date).reshape((len(Date), 1))
Temperature= np.array(Temperature).reshape((len(Temperature), 1))
Date_train = np.array(Date_train).reshape((len(Date_train), 1))
Temperature_train = np.array(Temperature_train).reshape((len(Temperature_train), 1)
)
Date_test = np.array(Date_test).reshape((len(Date_test), 1))
Temperature_test = np.array(Temperature_test).reshape((len(Temperature_test), 1))

Official_date = Data['Time'].tolist()
Official_date_train = Official_date[1:len(Official_date)*80//100]
Official_date_test = Official_date[len(Official_date)*80//100:]

fig, ax = plt.subplots(nrows=1, ncols=1)
ax.hist(Temperature, facecolor='blue', edgecolor='gray', bins=25, rwidth=1.10, alig
n='mid')
bins=[6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21]
ax.set_xticks(bins)
ax.set_ylabel('Συχνότητα')
ax.set_xlabel('Θερμοκρασία')
ax.set_title('Συχνότητα Θερμοκρασιών\nΠερίοδος Μετρήσεων 16/1/18-16/2/18')
plt.show()
```

Κώδικας 8: Linear Regression – Prediction

```
from sklearn import datasets, linear_model
from pandas import DataFrame, Series
import datetime
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
```

```
8.
9.  from sklearn.tree import DecisionTreeRegressor
10. from sklearn.ensemble import AdaBoostRegressor
11.
12. #Skip n rows in csv file
13. n = 100
14. num_lines = sum(1 for l in open("D:\Measurements.csv"))
15. print("num lines:")
16. print(num_lines)
17. skip_idx = [x for x in range(1, num_lines) if x % n != 0]
18.
19. #Open CVS file
20. Data = pd.read_csv("D:\Measurements.csv", parse_dates=[0], skiprows=skip_idx )
21. Date_1970=datetime.datetime(1970,1,1,0,0,0)
22. #Convert Timestampt to SEC (from 1/1/1970)
23. Data['Time_Sec']= (pd.to_datetime(Data['Time'])-Date_1970) /np.timedelta64(1,'s')
24. #Take Data (Date & Temperature) from pandas
25. Date = Data.ix[:,3]
26.
27. Temperature = Data.ix[:,2]
28. # Split the Data into training/testing sets 90% / 10%
29. Date_train = Date[1:len(Date)*80//100]
30. Temperature_train = Temperature[1:len(Temperature)*80//100]
31. Date_test = Date[len(Date)*80//100:]
32. Temperature_test= Temperature[len(Temperature)*80//100:]
33. #Create 2D Array
34. Date = np.array(Date).reshape((len(Date), 1))
35.
36. Temperature= np.array(Temperature).reshape((len(Temperature), 1))
37. Date_train = np.array(Date_train).reshape((len(Date_train), 1))
38. Temperature_train = np.array(Temperature_train).reshape((len(Temperature_train), 1)
    )
39. Date_test = np.array(Date_test).reshape((len(Date_test), 1))
40. Temperature_test = np.array(Temperature_test).reshape((len(Temperature_test), 1))
41.
42. # Create linear regression object
43. Regr = linear_model.LinearRegression()
44. # Train the model using the training sets
45. Regr.fit(Date_train,Temperature_train)
46. # Make predictions using the testing set
47. Temprature_pred= Regr.predict(Date_test)
48.
49. Official_date = Data['Time'].tolist()
50. Official_date_train = Official_date[1:len(Official_date)*80//100]
51. Official_date_test = Official_date[len(Official_date)*80//100:]
52.
53. plt.figure(figsize=(12,4))
54. plt.plot(Official_date_test, Temprature_pred, color='blue', linewidth=3)
55. plt.scatter(Official_date, Temperature,  color='black', s=1)
56.
57. #plt.axvline(x="2018-02-09", color="red")
58. locs, labels = plt.xticks()
59. plt.setp(labels, rotation=45)
60. plt.xlabel('Dates')
61. plt.ylabel('temprature (C)')
62. plt.title('Linear Regression')
63.
64. plt.grid(True)
65. Temprature_pred= Regr.predict(Date_test)
66. plt.figure(2)
67. plt.plot(Official_date_test, Temprature_pred, color='blue', linewidth=3)
```

```
plt.scatter(Official_date_test, Temperature_test,  color='black', s=1)
locs, labels = plt.xticks()
plt.setp(labels, rotation=45)
plt.xlabel('Dates')
plt.ylabel('Temprature (C)')
plt.title('Linear Prediction')
plt.tight_layout()
plt.show()
```

Κώδικας 9: Pipeline Regression – Prediction

```
import datetime
import numpy as np
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn import datasets, linear_model
from pandas import DataFrame, Series
from sklearn.linear_model import Ridge
from sklearn.preprocessing import PolynomialFeatures
from sklearn.pipeline import make_pipeline

#Skip n rows in csv file
n = 100
num_lines = sum(1 for l in open("D:\Measurements.csv"))
print("num lines:")
print(num_lines)
skip_idx = [x for x in range(1, num_lines) if x % n != 0]

#Open CVS file
Data = pd.read_csv("D:\Measurements.csv", parse_dates=[0], skiprows=skip_idx )
Date_1970=datetime.datetime(1970,1,1,0,0,0)

#Convert Timestampt to SEC (from 1/1/1970)
Data['Time_Sec']= (pd.to_datetime(Data['Time'])-Date_1970) /np.timedelta64(1,'s')

#Take Data (Date & Temperature) from pandas
Date = Data.ix[:,3]
Temperature = Data.ix[:,2]

# Split the Data into training/testing sets 80% / 20%
Date_train = Date[1:len(Date)*80//100]
Temperature_train = Temperature[1:len(Temperature)*80//100]
Date_test = Date[len(Date)*80//100:]
Temperature_test= Temperature[len(Temperature)*80//100:]

#Create 2D Arrays
Date = np.array(Date).reshape((len(Date), 1))
Temperature= np.array(Temperature).reshape((len(Temperature), 1))
Date_train = np.array(Date_train).reshape((len(Date_train), 1))
Temperature_train = np.array(Temperature_train).reshape((len(Temperature_train), 1)
)
Date_test = np.array(Date_test).reshape((len(Date_test), 1))
Temperature_test = np.array(Temperature_test).reshape((len(Temperature_test), 1))

Official_date = Data['Time'].tolist()
Official_date_train = Official_date[1:len(Official_date)*80//100]
Official_date_test = Official_date[len(Official_date)*80//100:]

```

```
#Pipeline Polynomial Prediction
model = make_pipeline(PolynomialFeatures(4), Ridge())
model.fit(Date_train,Temperature_train)
Temprature_pred = model.predict(Date_test)

plt.figure(figsize=(12,4))
plt.plot(Official_date_test, Temprature_pred, color='blue', linewidth=3)
plt.scatter(Official_date, Temperature,  color='black', s=1)
#plt.axvline(x="2018-02-10", color="red")
plt.xlabel('Dates')
plt.ylabel('temprature (C)')
plt.title('Pipeline Regression')
locs, labels = plt.xticks()
plt.setp(labels, rotation=45)
plt.tight_layout()

Temprature_pred = model.predict(Date_test)
plt.figure(2)
plt.plot(Official_date_test, Temprature_pred, color='blue', linewidth=3)
plt.scatter(Official_date_test, Temperature_test,  color='black', s=1)
locs, labels = plt.xticks()
plt.setp(labels, rotation=45)
plt.xlabel('Dates')
plt.ylabel('Temprature (C)')
plt.title('Prediction')
plt.tight_layout()
plt.show()
```