ΑΤΕΙ ΚΡΗΤΗΣ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΤΙΤΛΟΣ
Σχεδιασμός Και Υλοποίηση Συστήματος Συλλογής Δεδομένων Απομακρυσμένων Αισθητηρίων Και Ελέγχου Απομακρυσμένων Ηλεκτρονόμων, Μέσω Ασύρματων Τεχνολογιών

ΓΑΡΕΦΑΛΑΚΗΣ ΕΜΜΑΝΟΥΗΛ

Αρ. Μητρώου: ΤΗ5904

ΗΡΑΚΛΕΙΟ

ΣΕΠΤΕΜΒΡΙΟΣ 2018

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ: Κορνάρος Γεώργιος

ΕΠΙΤΡΟΠΗ ΑΞΙΟΛΟΓΗΣΗΣ: Βασιλάκης Κωνσταντίνος

Παναγιωτάκης Σπυρίδων

ΗΜΕΡΟΜΗΝΙΑ ΠΑΡΟΥΣΙΑΣΗΣ: 20 Σεπτεμβρίου 2018

ΚΕΝΗ ΣΕΛΙΔΑ

Αφιέρωση

Αφιερώνεται

στη Μητέρα μου Μαρία, διότι είναι ο άνθρωπος που είναι δίπλα μου σε κάθε βήμα που κάνω

στο γιό μου Γιώργο και τη σύζυγο μου Μαρία, για το χρόνο που έχασα από κοντά τους

ΚΕΝΗ ΣΕΛΙΔΑ

Ευχαριστίες

Ευχαριστώ πολύ τον επόπτη της πτυχιακής μου εργασίας Κο Κορνάρο Γεώργιο, για την εμπιστοσύνη που έδειξε στο πρόσωπό μου, υποστηρίζοντάς με να ασχοληθώ με το θέμα αυτό και με τις χρήσιμες συμβουλές του

Η ενασχόληση μου με το αντικείμενο αυτό, διεύρυνε τους ορίζοντες μου αποκτώντας νέες γνώσεις και δεξιότητες

ΚΕΝΗ ΣΕΛΙΔΑ

Abstract

This diploma thesis deals with a new subject that has entered into the everyday life of man and seems that won't be left behind. It concerns the Internet of Things (IoT) that has entered into various areas of everyday life, such as industry, transport and distribution, on energy and applications, in the public sector and in the health sector.

The purpose of this thesis was the implementation of a system, which includes the data collection and storage of sensors in a database, and the control of power sockets, using wireless technologies. A goal, which has been achieved effectively, but also created a system that can play an important role in different levels of education. Starting from secondary education students to engineers, up to the use in the real-world applications.

The system provides the ability to acquire knowledge and develop skills in the fields of electronics design and software development in various programming languages.

The system that was implemented was a multi-level challenge. It included the design electronic circuits and the creation of printed circuit boards. Also, the development of a management system for the communication and control of heterogeneous Nodes.

The system consists of a Raspberry Pi 3, which acts as a gateway and Nodes operating remotely and communicates with the gateway via WiFi using the MQTT protocol. Raspberry Pi 3 runs administrative software, created with Node-Red and MySQL database for data storage. The Nodes are based on developmental boards of the ESP8266, such as ESP-01 and ESP-12. They execute code developed in C language in order to communicate with the gateway but also to read data from the sensors or to turn on / off relays.

For the presentation and evaluation of the system, four applications were implemented

- An Electrical panel, where it is possible to save the current and in consequence the power of two electric lines and to remotely activate / deactivate these lines.
- A Water tank level sensor that sends the information to the gateway, which is stored and represented graphically
- A Smoke sensor connected to the gateway and informing the user in case of smoke detection.
- Test Node, which connects sensors, such as temperature and a potentiometer, and two relays for test purposes.
- Two stand-alone sockets that are switched on and off remotely via the gateway

Keywords: IoT, MQTT protocol, Node-RED, Raspberry Pi 3, ESP-01, ESP-12, MySQL, PCBs

ΚΕΝΗ ΣΕΛΙΔΑ

Σύνοψη

Η παρούσα πτυχιακή εργασία, πραγματεύεται ένα νέο αντικείμενο που έχει μπει στη καθημερινότητα του ανθρώπου και απ' ότι φαίνεται, πρόκειται να μείνει. Αφορά το διαδίκτυο των πραγμάτων (ΔτΠ – Internet of Things IoT) που έχει μπει σε διάφορους τομείς της καθημερινότητας του ανθρώπου, όπως, στο έξυπνο σπίτι, στη βιομηχανία, στη μεταφορά και τη διανομή, σε θέματα ενέργεια και εφαρμογών, στο δημόσιο τομέα αλλά και στο τομέα υγείας.

Ο σκοπός της εργασίας, ήταν η υλοποίηση ενός συστήματος, που θα συμπεριλαμβάνει αφενός τη συλλογή δεδομένων από αισθητήρες και αποθήκευσή τους σε βάση δεδομένων, αφετέρου τον έλεγχο ρευματοδοτών, χρησιμοποιώντας ασύρματες τεχνολογίες. Σκοπός που επιτεύχθηκε αποτελεσματικά, επιπρόσθετα όμως δημιουργήθηκε και ένα σύστημα που μπορεί να παίξει σημαντικό ρόλο στην σε διάφορα επίπεδα εκπαίδευσης. Ξεκινώντας από μαθητές Β/θμιας εκπαίδευσης σε μηχανικούς της Γ/θμιας εκπαίδευσης, έως και τη χρήση του σε εφαρμογές σε πραγματικές συνθήκες.

Το σύστημα δίνει τη δυνατότητα απόκτησης γνώσεων και ανάπτυξης δεξιοτήτων στους τομείς σχεδιασμού ηλεκτρονικών κατασκευών, αλλά και της ανάπτυξης λογισμικού σε διάφορες γλώσσες προγραμματισμού.

Το σύστημα που υλοποιήθηκε, ήταν μια πρόκληση πολλών επιπέδων. Περιλάμβανε, το σχεδιασμό και την κατασκευή ηλεκτρονικού υλικού και τυπωμένων κυκλωμάτων. Επίσης, χρειάστηκε να γίνει προγραμματισμός διαφόρων επιπέδων, των Nodes αλλά και του διαχειριστικού συστήματος.

Το σύστημα, αποτελείται από ένα Raspberry Pi 3[1], που έχει το ρόλο της πύλης (gateway) και τα Nodes που λειτουργούν απομακρυσμένα και επικοινωνούν με την πύλη μέσω WiFi και του πρωτοκόλλου MQTT. Το Raspberry Pi 3 εκτελεί διαχειριστικό λογισμικό, που έχει δημιουργηθεί με το Node-Red και βάση δεδομένων MySQL για την αποθήκευση δεδομένων. Τα Nodes στηρίζονται σε εκδόσεις αναπτυξιακών του ολοκληρωμένου ESP8266, όπως το ESP-01 και το ESP-12. Εκτελούν κώδικα που έχει αναπτυχθεί σε γλώσσα C, προκειμένου να επικοινωνούν με την πύλη αλλά και να διαβάζουν δεδομένα από τους αισθητήρες ή να ενεργοποιούν/απενεργοποιούν ηλεκτρονόμους.

[1] Για λόγους συντομίας, στο εξής της εργασίας θα αναφερόμαστε στο Raspberry Pi 3 με το ακρωνύμιο RPI.

Για την παρουσίαση αλλά και την αξιολόγηση του συστήματος, υλοποιήθηκαν τέσσερις εφαρμογές

- Ηλεκτρικός Πίνακας, όπου δίνεται η δυνατότητα καταγραφής της έντασης του ρεύματος δύο ηλεκτρικών γραμμών καθώς και την , απομακρυσμένη ενεργοποίηση/απενεργοποίηση των γραμμών αυτών.
- Αισθητήρας μέτρησης στάθμης δεξαμενής νερού που στέλνει τις πληροφορίες στην πύλη.
- Αισθητήρας καπνού που συνδέεται με την πύλη και ενημερώνει το χρήστη σε περίπτωση ανίχνευσης καπνού.
- Δοκιμαστικό Node, στο οποίο συνδέονται αισθητήρες, όπως θερμοκρασίας και ένα ποτενσιόμετρο και δύο ηλεκτρονόμοι ώστε να πραγματοποιούνται δοκιμές.
- Δύο αυτόνομου ρευματοδότες που μέσω της πύλης ενεργοποιούνται και απενεργοποιούνται απομακρυσμένα

Λέξεις κλειδιά: Διαδίκτυο των Πραγμάτων, MQTT, Node-RED, Raspberry Pi 3, ESP-01, ESP-12, MySQL, Τυπωμένα κυκλώματα

Πίνακας περιεχομένων

Κατάλογος Εικόνων

Κατάλογος Πινάκων

Κατάλογος Γραφημάτων

Εισαγωγή

Η παρούσα εργασία, δημιουργήθηκε στα πλαίσια ολοκλήρωσης Σπουδών, του φοιτητή Γαρεφαλάκη Εμμανουήλ, με Αρ. Μητρώου ΤΗ5904 του Τμήματος Ηλεκτρολόγων Μηχανικών της Σχολής Τεχνολογικών Εφαρμογών (ΣΤΕΦ) του Ανωτάτου Τεχνολογικού Εκπαιδευτικού Ιδρύματος Κρήτης (Α.Τ.Ε.Ι. Κρήτης).

Η συγκεκριμένη πτυχιακή εργασία σχετίζεται με το Διαδίκτυο των Πραγμάτων (Internet of Things – IoT) και αφορά την υλοποίηση ενός συστήματος συλλογής και καταχώρησης δεδομένων αισθητήρων σε βάση δεδομένων, αλλά και διαχείριση ηλεκτρονόμων, που ενεργοποιούν συσκευές ισχύος ή κυκλώματα ενός οικιακού και όχι μόνο, ηλεκτρικού πίνακα.

Το σύστημα σχεδιάστηκε, για να λειτουργήσει τόσο σε πραγματικό, αλλά και σε εκπαιδευτικό περιβάλλον. Έχοντας ως προδιαγραφή την ευελιξία αυτή, το να μπορεί να χρησιμοποιηθεί και ως εκπαιδευτικό εργαλείο, το υλικό σχεδιάστηκε έτσι ώστε να είναι «σπονδυλωτό» (modular) και να δέχεται αισθητήρες απευθείας ή μέσω ενδιάμεσων ηλεκτρονικών κυκλωμάτων με τη μορφή αθρωμάτων (modules). Έτσι, το υλικό παρέχει την υποδομή σύνδεσης διαφορετικών αισθητήρων και διακοπτών, ανάλογα με το κάθε σενάριο που επιθυμείται να υλοποιηθεί. Σε αντίθετη περίπτωση, θα έπρεπε να σχεδιαστεί σε πρωτότυπη πλακέτα δοκιμής και έπειτα να υλοποιηθεί σε τυπωμένο κύκλωμα, εφόσον είναι επιθυμητό, για να υλοποιηθεί σε μόνιμη κατασκευή.

Κατά το πρώτο μέρος, θα γίνει σύνθεση αισθητηρίων που βρίσκονται στο εμπόριο ή θα σχεδιαστούν και να πραγματοποιείται ανταλλαγή δεδομένων, μεταξύ ενός μικροεπεξεργαστή (προτείνεται ένα ολοκληρωμένο σύστημα Arduino ή κάποιο εξειδικευμένο ολοκληρωμένο), μέσω ασύρματης επικοινωνίας και ένα κεντρικό υπολογιστικό σύστημα (Raspberry Pi ή κάποιο σταθμό εργασίας) στο οποίο θα εκτελείται μια εφαρμογή ως εξυπηρετητής. Στο μέρος αυτό, της πτυχιακής εργασίας, απαιτείται η ανάπτυξη λογισμικού επικοινωνίας του συστήματος του αισθητήρα με τον εξυπηρετητή.

Κατά το δεύτερο μέρος, θα γίνει η υλοποίηση του εξυπηρετητή που θα αναλαμβάνει τη συλλογή και αποθήκευση των δεδομένων σε ειδικά σχεδιασμένη βάση δεδομένων και θα ελέγχει απομακρυσμένους ηλεκτρονόμους. Επίσης θα πρέπει να αναπτυχθεί κατάλληλη διεπαφή χρήστη εισαγωγής παραμέτρων αλλά και παρουσίασης δεδομένων.

Η επέκταση της εφαρμογής είναι η δημιουργία λογισμικού που θα εκτελείται σε έξυπνες συσκευές, έτσι ώστε να γίνεται διαχείριση, επικοινωνία και έλεγχος του εξυπηρετητή μέσω έξυπνων συσκευών ή άλλο σταθμό εργασίας. Το σύστημα, χρησιμοποιώντας προγράμματα τρίτων ενσωματώνει και εκτελεί φωνητικές εντολές μέσω έξυπνων συσκευών.

Κίνητρα Διεξαγωγής Εργασίας

Τα κίνητρα διεξαγωγής της παρούσας εργασίας εντοπίζονται στα πλαίσια της ιδιότητας μου ως εκπαιδευτικού Δευτεροβάθμιας Εκπαίδευσης και του προσωπικού ενδιαφέροντος και ανάπτυξης σε θέματα που άπτονται νέων Τεχνολογιών.

Ως καθηγητής Πληροφορικής, και στα πλαίσια διαρκούς ενημέρωσης και αναζήτησης αντικειμένων που θα εντυπωσιάζουν τους μαθητές μου. Επιπροσθέτως, παρακολουθώντας τις εξελίξεις , όπως συνέδρια που σχετίζονται με την πληροφορική στην εκπαίδευση, στον Ελλαδικό χώρο, διαπιστώνεται ότι υπάρχει μια τάση αξιοποίησης στην διδασκαλία των νέων τεχνολογιών, συστημάτων όπως Arduino και Raspberry Pi [1]. Επίσης οι μαθητές, εντυπωσιάζονται, αρκετά με ότι σχετίζεται με το έξυπνο κινητό τους (Smartphone), γεγονός που μπορεί να αξιοποιηθεί κατά τη διδασκαλία.

Όσον αφορά το προσωπικό ενδιαφέρον και ανάπτυξη, σε θέματα που άπτονται νέων τεχνολογιών και πληροφορικής, το ενδιαφέρον επικεντρώνεται σε όσα συνδέουν την πληροφορική και ειδικότερα τον προγραμματισμό και τη εφαρμογή του με το εξωτερικό περιβάλλον. Γενικότερα θα λέγαμε θέματα που αφορούν Συστήματα Αυτομάτου Ελέγχου. Επίσης, επειδή διανύουμε μια περίοδο όπου το Διαδίκτυο των Πραγμάτων, βρίσκεται σε μεγάλη ανάπτυξη [2], υπήρξε η προσωπική απορία και ανάγκη υλοποίησης ενός σχετικού συστήματος, ώστε να μελετηθεί η ευρύτερη περιοχή της τεχνολογίας που σχετίζεται με το Διαδίκτυο των Πραγμάτων. Να αξιοποιηθούν οι νεώτερες εξελίξεις σχετικά με την ασύρματη επικοινωνία και την αξιοποίηση των έξυπνων κινητών ώστε να δίνονται εντολές και να λαμβάνεται πληροφορία.

Τέλος, στα κίνητρα διεξαγωγής της εργασίας, κρίνεται απαραίτητο να προστεθεί και η προσωπική μας επιθυμία, η αξιοποίηση δυνατοτήτων που παρέχονται σε ένα έξυπνο σπίτι στην οικεία του συγγραφέα και φοιτητή, ώστε να βελτιωθούν κάποιες λειτουργίες της καθημερινότητας. Επομένως, η συγκεκριμένη εργασία και συγκεκριμένα η κατασκευή γίνεται ώστε να μπορεί να αξιοποιηθεί και να λειτουργεί υπό κανονικές συνθήκες.

Σκοπός και Στόχοι Διεξαγωγής Εργασίας

Ο κύριος σκοπός της εργασία, κατά την ανάληψη της εργασίας, ήταν η κατασκευή ενός απλού συστήματος συλλογής και καταγραφής δεδομένων, αλλά και ελέγχου ηλεκτρονόμων ισχύος, στην πορεία κατά τη φάση της βιβλιογραφικής επισκόπησης μετασχηματίστηκε καθώς αποκτήθηκαν περισσότερες γνώσεις σχετικά με τα σύγχρονα ολοκληρωμένα και λογισμικά που είναι διαθέσιμα.

Ενώ στην αρχή, υπήρχε σκοπός να αξιοποιηθούν πλακέτες Arduino και μικροεπεξεργαστές της ATMEL, αποφασίστηκε πως η εργασία θα μπορούσε να πραγματοποιηθεί, αξιοποιώντας πιο σύγχρονους επεξεργαστές όπως το system on chip (SoC) ESP8266 και ειδικότερα τα αρθρώματα ESP-01 και ESP-12. Στο παράρτημα 11 υπάρχουν οι διάφοροι τύποι των ESP πλακετών σε εικόνα καθώς και πίνακας με τα χαρακτηριστικά τους [Δ.19].

Στην παρούσα εργασία χρησιμοποιήθηκαν τα ESP01 και ESP12 ενώ υπήρχε η πρόβλεψη να χρησιμοποιηθεί το ESP32, αλλά προτιμήθηκε να συμπεριληφθεί στην επέκταση της εργασίας, διότι αρκετές βιβλιοθήκες δεν είναι ακόμα συμβατές.

Επίσης, ενώ αρχικά υπήρχε η σκέψη δημιουργίας ενός λογισμικού υποστήριξης του συστήματος σε γλώσσα Python, στην πορεία μελετώντας διάφορα λογισμικά που μπορούν να αξιοποιηθούν, όπως για παράδειγμα το Node-RED άλλαξε η αρχική σκέψη.

Ολοκληρώνοντας, ακολουθώντας τι σκέψεις αυτές, οδηγηθήκαμε στον τελικό σκοπό και ευρύτερο σκοπό της εργασίας. Στο να δημιουργηθεί ένα σύστημα, που θα πληροί συγκεκριμένες προδιαγραφές, θα είναι αξιοποιήσιμο σε πραγματικά περιβάλλοντα και ευέλικτο, ώστε να είναι αξιοποιήσιμο και για εκπαιδευτικούς σκοπούς.

Επομένως, το σύστημα της παρούσας εργασίας, θα είναι σπονδυλωτό, έτσι ώστε να μπορούν οι αισθητήρες να «κουμπώνουν» με τη μορφή αρθρωμάτων (modules) και προγραμματισμός θα προσαρμόζεται αναλόγως των σεναρίων. Ο προγραμματισμός θα μπορεί να γίνεται σε διαφορετικά επίπεδα (εξυπηρετητές και nodes).

Όπως αναφέρουν οι Ramohalli & Adegbija [3], ένα σπονδυλωτό σύστημα (Modula Electronics) θα παρέχει τη δυνατότητα, εκτός από μηχανικούς και σε λιγότερο έμπειρους χρήστες των STEM (Science, Technology, Engineering and Math) πρόσβαση στο Διαδίκτυο των Πραγμάτων (IoT).

Ολοκληρώνοντας το σκοπό της παρούσας εργασίας, είναι ότι η παρούσα εργασία να αποτελέσει έναν οδηγό, αναπαραγωγής του συστήματος, το οποίο θα μπορεί να αξιοποιηθεί, εκτός από πραγματικό περιβάλλον, σε εργαστηριακά επίπεδα, ώστε οι εκπαιδευόμενοι να βελτιώσουν τις δεξιότητες τους τόσο σε επίπεδο σχεδιασμού ηλεκτρονικών κυκλωμάτων, κατασκευής τυπωμένων κυκλωμάτων, προγραμματισμό σε διάφορες γλώσσες προγραμματισμού, εγκατάσταση και παραμετροποίηση εξυπηρετητών που παίζουν ενεργό ρόλο στο παρόν σύστημα.

Δομή Εργασίας

Η δομή της εργασίας, θα είναι η ακόλουθη:

Κεφάλαιο 1ο: Θα παρουσιαστούν οι επιθυμητές προδιαγραφές του συστήματος. Τι ακριβώς θα πρέπει να ικανοποιεί το σύστημα που θα σχεδιαστεί και που θα υλοποιηθεί. Θα γίνει εκτενής παρουσίαση της Αρχιτεκτονικής του Υλικού και της Αρχιτεκτονικής του Λογισμικού που θα χρησιμοποιηθεί.

Κεφάλαιο 2ο : Στο κεφάλαιο αυτό θα παρουσιαστούν όλα όσα σχετίζονται με τη μελέτη, το σχεδιασμό και δημιουργία των τυπωμένων κυκλωμάτων.

Κεφάλαιο 3ο: Το κεφάλαιο αυτό παρουσιάζει τα προγράμματα που δημιουργήθηκαν ή που χρειάστηκαν να προσαρμοστούν. Θα γίνει επίσης αναφορά στις διαδικασίες προγραμματισμού. Το κεφάλαιο ολοκληρώνεται με την παρουσίαση των σχετικών ρυθμίσεων που πρέπει να πραγματοποιηθούν, ώστε το σύστημα να είναι προσβάσιμο από το διαδίκτυο και όχι μόνο από το τοπικό δίκτυο, αλλά και τα προγράμματα που πρέπει να εγκατασταθούν σε ένα Android smartphone, ώστε να δίνονται εντολές μέσω κινητού τηλεφώνου, εντός τοπικού δικτύου αλλά και μέσω διαδικτύου.

Κεφάλαιο 4ο: Στο κεφάλαιο αυτό, αναλύονται θέματα ασφαλείας που σχετίζονται με το πρωτόκολλο MQTT, και που κατ' επέκταση επηρεάζουν το σύστημα που σχεδιάστηκε.

Κεφάλαιο 5ο: Στο κεφάλαιο αυτό, παρουσιάζονται πρακτικές εφαρμογές που, που ενσωματώθηκαν στο σύστημα και χρησιμοποιούνται στα πλαίσια ενός έξυπνου σπιτιού, όπως πυρανίχνευση και έλεγχο στάθμης δεξαμενής νερού.

Αποτελέσματα: Στην ενότητα αυτή παρουσιάζονται τα δεδομένα που συλλέχθηκαν από τη λειτουργία του συστήματος και των διαφόρων εφαρμογών, με τη μορφή γραφικών παραστάσεων.

Μελλοντική Εργασία και Επεκτάσεις: Στην ενότητα αυτή θα παρουσιαστούν ιδέες για μελλοντικές επεκτάσεις ή βελτιώσεις θα μπορούσαν να γίνουν ώστε να εξελιχθεί το σύστημα.

Σύνοψη: Στην ενότητα αυτή, κλείνει η εργασία αυτή παρουσιάζοντας συνοπτικά το τι έχει πραγματοποιηθεί καθώς και απόψεις του συγγραφέα επί του θέματος.

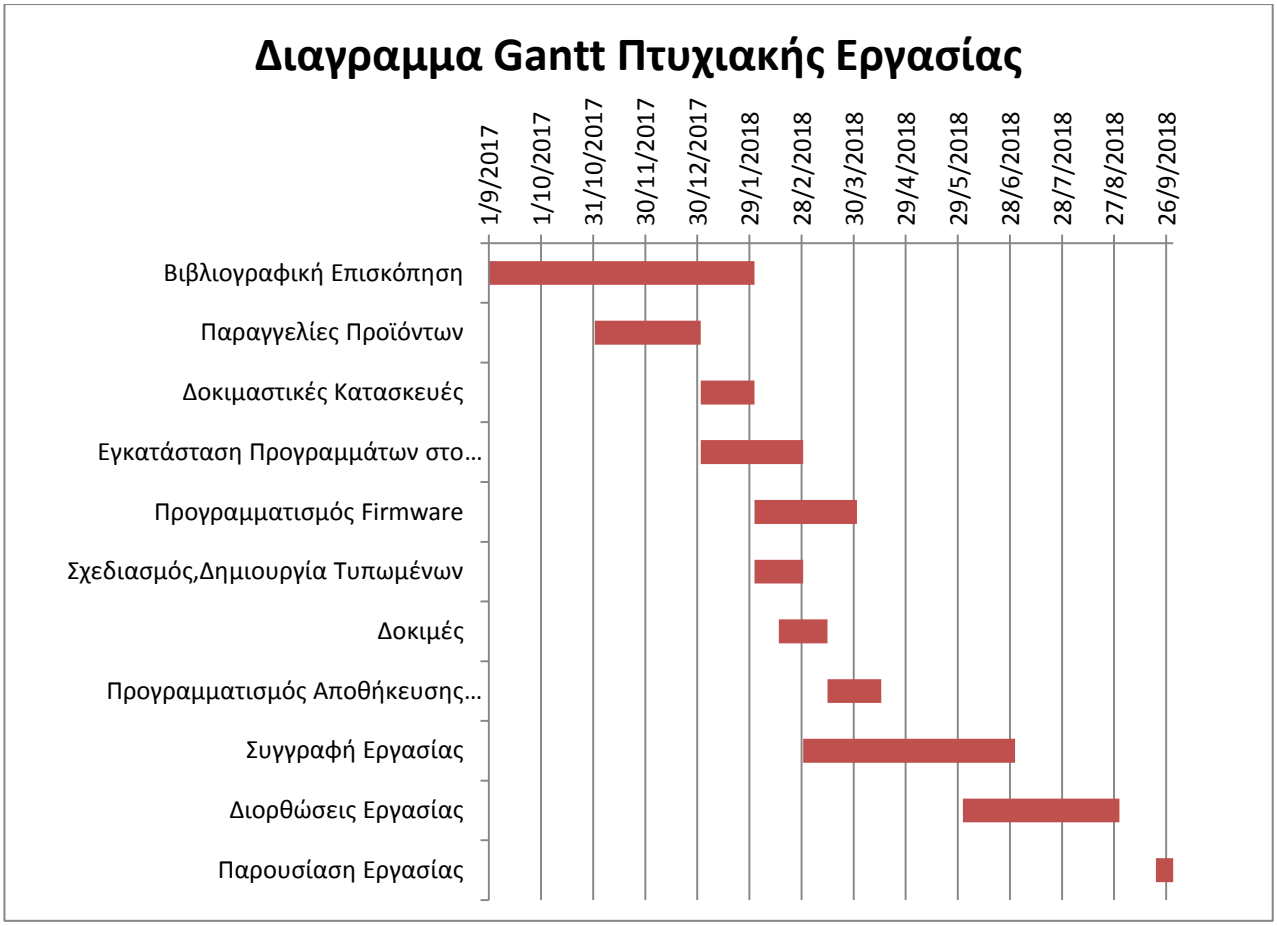
Μεθοδολογία Υλοποίησης

Στην Εικόνα 2, φαίνεται το διάγραμμα Gantt, του έργου της πτυχιακής εργασίας. Ουσιαστικά η εργασία ξεκίνησε από το Χειμερινό εξάμηνο του 2017-2018, παρά το γεγονός δεν είχε γίνει επίσημα ανάληψη του θέματος εργασίας. Η υλοποίηση της εργασίας περιλάμβανε πολλές παραγγελίες, που αφορούσαν από κουτιά κατασκευών, πλακέτες διάτρυτες, φωτοευαίσθητες και αναπτυξιακά κυκλώματα. Επειδή η παραγγελίες γινόταν είτε από το εσωτερικό είτε από το εξωτερικό υπήρχαν πολλές καθυστερήσεις, με αποτέλεσμα να υπάρχουν πολλές επικαλύψεις στην ολοκλήρωση των φάσεων του έργου.

Οι διάφορες φάσεις που ακολουθήθηκαν ήταν οι παρακάτω:

1. Βιβλιογραφική επισκόπηση
2. Παραγγελίες προϊόντων (Raspberry Pi3, Arduino, Ολοκληρωμένα κυκλώματα, αναπτυξιακά board, starter kits, κουτιά κατασκευών, τροφοδοτικά και μετατροπείς τάσεων (AC/DC, DC/DC)
3. Δημιουργία δοκιμαστικής κατασκευής
4. Εγκατάσταση Λογισμικού στο Raspberry Pi (Λειτουργικό Σύστημα Rasbian)
5. Προγραμματισμός του ESP-12 board και σύνδεση με Node-RED
6. Δοκιμαστική Αξιολόγηση
7. Δημιουργία Τυπωμένων Κυκλωμάτων
8. Έλεγχος καλής λειτουργίας και σύνδεση στο σύστημα
9. Ενημερώσεις Firmware
10. Δημιουργία κατασκευών και σύνδεση στο Σύστημα
11. Έναρξη Συγγραφής Αναφοράς
12. Εγκατάσταση Λογισμικού στο Raspberry Pi (Εγκατάσταση Access Point, MySQL)
13. Συγγραφή κώδικα για Εισαγωγή δεδομένων στη Δεδομένων
14. Συγγραφή Εφαρμογών στο Node-RED

Μια σχετική πορείας της εγκατάστασης του έργου πραγματοποίησης της εργασίας φαίνεται στο διάγραμμα Gantt στην εικόνα 2.



Εικόνα 1: Διάγραμμα Gantt Εργασίας

1. Κεφάλαιο 1ο

1.1. Πλαίσιο προδιαγραφών Συστήματος

Το πλαίσιο προδιαγραφών του συστήματος, ορίζει τις προδιαγραφές τις οποίες θα ήταν επιθυμητές, να ισχύουν για το σχεδιασμό του συστήματος, έτσι ώστε να χρησιμοποιηθούν στη λήψη αποφάσεων κατά τον σχεδιασμό και την υλοποίηση.

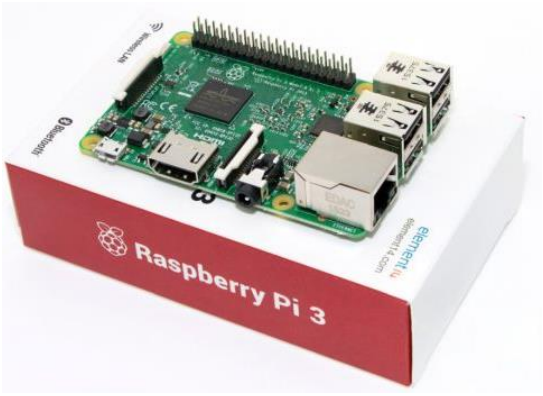
- Να μπορεί να τοποθετηθεί σε συσκευασία επαγγελματικών προδιαγραφών, ώστε να είναι εμφανίσιμο και λειτουργικό
- Το σύστημα να λειτουργεί αυτόνομα, δηλαδή να έχει τέτοια αρχιτεκτονική ώστε να είναι δυνατή η εγκατάστασή του σε χώρο απομονωνόμενο όπως για παράδειγμα σε μια εγκατάσταση στην ύπαιθρο π.χ. ένα θερμοκήπιο και να μην απαιτείται υποδομή όπως διαδίκτυο, τοπικό ενσύρματο ή ασύρματο δίκτυο
- Η απαιτούμενη τροφοδοσία να μπορεί να παρέχεται και από μπαταρίες, σε περίπτωση που δεν υπάρχει παροχή ρεύματος
- Να παρέχει τη δυνατότητα αποθήκευσης δεδομένων
- Να δίνεται η δυνατότητα αναβάθμισης του firmware των IoT απομακρυσμένα, ώστε να μην απαιτείται αποσυναρμολόγηση, για τυχόν αναβαθμίσεις ή βελτιώσεις
- Το κόστος να είναι προσιτό
- Να παρέχεται σχετική ασφάλεια
- Να υπάρχει σχετική αξιοπιστία
- Να υποστηρίζεται από ετερογενή συστήματα
- Αξιοποίηση Ελεύθερων λογισμικών και Ανοικτού Κώδικα

1.2. Αρχιτεκτονική Συστήματος

Εικόνα 2: Σύστημα Εργασίας

1.2.1. Αρχιτεκτονική Υλικού

1.2.1.1. Raspberry Pi 3



Εικόνα 3: Raspberry Pi 3

Εικόνα 4 : Raspberry Pi 3 σε κουτί

Η πλακέτα Raspberry Pi είναι ένας οικονομικός, μεγέθους πιστωτικής κάρτας ηλεκτρονικός υπολογιστής, που μπορεί να συνδεθεί με οθόνη, πληκτρολόγιο και ποντίκι. Η μικρή αυτή συσκευή, αλλά με πολλές δυνατότητες, επιτρέπει σε χρήστες όλων των ηλικιών να αποκτήσουν βασικές δεξιότητες νέων τεχνολογιών (ΤΠΕ), όπως προγραμματισμό, χρήση διαδικτύου, να βλέπουν ταινίες και να παίζουν παιχνίδια υψηλής ανάλυσης (HD) καθώς να χρησιμοποιούν προγράμματα όπως επεξεργαστές κειμένου και υπολογιστικών φύλλων.

Επιπρόσθετα στα παραπάνω το Raspberry Pi, μπορεί να επικοινωνήσει με το εξωτερικό περιβάλλον και γι' αυτό έχει χρησιμοποιηθεί σε ευρύ φάσμα ηλεκτρονικών και ψηφιακών εφαρμογών, όπως μουσικά κουτιά και ανιχνευτές παιδιών μέχρι μετεωρολογικούς σταθμούς, ενυδρεία και σπίτια πουλιών που περιλαμβάνουν υπέρυθρες κάμερες.

Βασικός σκοπός των σχεδιαστών του Raspberry Pi ήταν η παροχή τεχνολογίας σε παιδιά και ενήλικες, στην υφήλιο, ώστε να αποκτήσουν γνώσεις προγραμματισμού και να καταλάβουν τον τρόπο λειτουργίας των ηλεκτρονικών υπολογιστών.

Το Raspberry Pi μπορεί να κάνει ότιδήποτε θα περίμενε κάποιος να κάνει ένας Η/Υ, κάτι που αποτελεί μια μεγάλη λίστα εφαρμογών. Επίσης περιλαμβάνει και μια γραφική γλώσσα προγραμματισμού το Node-RED που θα μπορούσε κάποιος να μάθει πολύ εύκολα να προγραμματίσει εφαρμογές του Διαδικτύου των Πραγμάτων (IoT) [10].

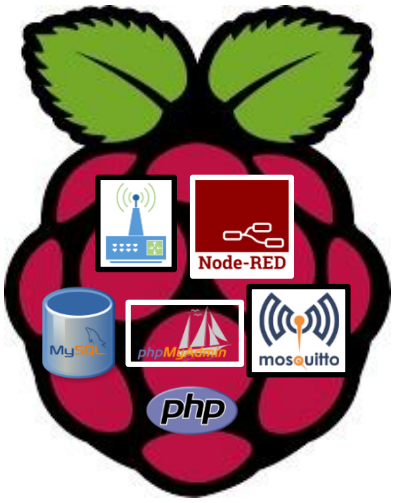
Το Raspberry Pi 3 Μοντέλο B [Δ.10], είναι το νεότερο μοντέλο της τρίτης (3ης) γενιάς Raspberry Pi και ήρθε να αντικαταστήσει Raspberry Pi 2 Model B, τον Φεβρουάριο του

2016. Υπάρχει και πιο πρόσφατο μοντέλο το Raspberry Pi 3 Model B+, που είναι το πιο πρόσφατο μοντέλο της γενιάς.

Στην παρακάτω λίστα φαίνονται τα χαρακτηριστικά του Raspberry Pi 3 Model B, που χρησιμοποιήθηκε στην εργασία μας:

- Quad Core 1.2GHz Broadcom BCM2837 64bit CPU
- 1GB RAM
- BCM43438 wireless LAN and Bluetooth Low Energy (BLE) on board
- 100 Base Ethernet
- 40-pin extended GPIO
- 4 USB 2 ports
- 4 Pole stereo output and composite video port
- Full size HDMI
- CSI camera port for connecting a Raspberry Pi camera
- DSI display port for connecting a Raspberry Pi touchscreen display
- Micro SD port for loading your operating system and storing data
- Upgraded switched Micro USB power source up to 2.5A

Το Raspberry Pi στην εργασία μας αποτελεί την καρδιά του συστήματος. Στον μικροϋπολογιστή αυτό, εγκαταστάθηκαν σημαντικές υπηρεσίες απαραίτητες στο συνολικό σύστημα και οι οποίες θα παρουσιαστούν στη συνέχεια.



Εικόνα 5:Raspberry Pi & Υπηρεσίες

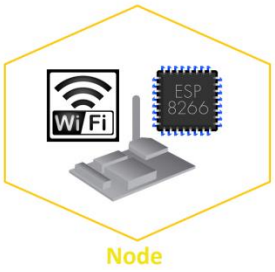
1.2.1.2. Υπηρεσίες του Raspberry στο Σύστημα

- **Access Point :** Λειτουργεί ως Access Point παρέχοντας διευθύνσεις IP στα Nodes του Συστήματος.

- **Router:** Εάν συνδεθεί σε τοπικό δίκτυο και διαδίκτυο τότε δρομολογεί τα πακέτα δεδομένων.
- Λειτουργεί ως εξυπηρετητής διαφορετικών υπηρεσιών.
- **Node-RED :** Γραφική Γλώσσα Προγραμματισμού για IoT (VPL – Visual Programming Language).
- **MySQL Server:** Παρέχει υπηρεσία βάσης δεδομένων, για την αποθήκευση των δεδομένων.
- **phpMyAdmin:** Εργαλείο διαχείρισης βάσης δεδομένων MySQL
- **PHP:** Παρέχει τη δυνατότητα να εμφανίζονται τα δεδομένα μέσω ιστοσελίδας.
- **MQTT Broker:** Παρέχει την υπηρεσία του MQTT πρωτοκόλλου, ειδική υπηρεσία για συστήματα IoT. Μέσω του συγκεκριμένου πρωτοκόλλου γίνεται ανταλλαγή δεδομένων μεταξύ των Nodes και του προγράμματος ελέγχου

1.2.1.3. Nodes Συστήματος

Με το όρο Nodes του Συστήματος ή εν συντομία Nodes, στο εξής θα αναφερόμαστε στις κατασκευές που βασίζονται στα ESP-01 και ESP-12 ή οποιαδήποτε άλλο (π.χ. ESP-32) που να ενσωματωθούν στην πορεία ή στην επέκταση της εργασίας. Ο ρόλος των Nodes είναι να επικοινωνούν ασύρματα μέσω WiFi ή οποιοδήποτε άλλο τρόπο (Bluetooth, RF κ.α.) με τον MQTT Broker του συστήματος. Θα μπορούν να λειτουργούν ως αισθητήρες ή ενεργοποιητές (Sensors or Actuators) ή συνδυασμός και θα ανταλλάσσουν εντολές και δεδομένα.

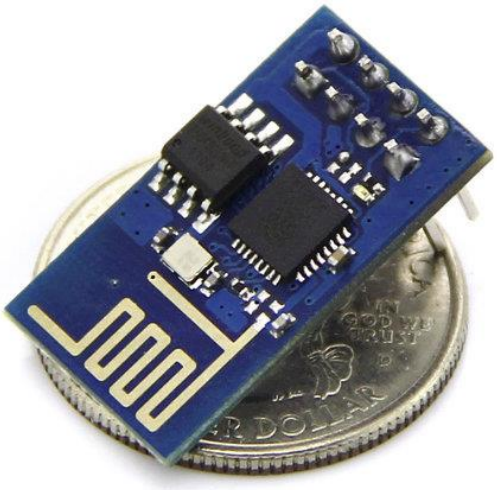


Εικόνα 6:Node

1.2.1.4. ESP8266 (ESP01)

Το ESP-01 είναι ένα απλό άρθρωμα που βασίζεται στο ολοκληρωμένο ESP8266, και παρέχει, τυπικά χωρίς παρεμβάσεις, δύο ψηφιακές θύρες εισόδου ή εξόδου (GPIO - General Purpose Input Output).

Στο σύστημα μας, χρησιμοποιήθηκε σε συνδυασμό με κατάλληλο άρθρωμα ηλεκτρονόμου ESP-01s relay για να δημιουργηθούν αυτόνομες πρίζες, ελεγχόμενες από το σύστημα.



Εικόνα 7: ESP-01

Εικόνα 8:ESP-01s Relay

1.2.1.5. ESP8266 (ESP12 – NodeMCU)

Το NodeMCU – ESP12, αναπτυξιακή πλακέτα βασισμένη στο SoC ESP8266, χρησιμοποιήθηκε για την κεντρική μας κατασκευή. Το NodeMCU έχει 8 GPIO και μια αναλογική θύρα. Προκειμένου να αναβαθμίσουμε το άρθρωμα και να αυξήσουμε τις αναλογικές θύρες χρειάστηκε να προχωρήσουμε σε κάποιο σχεδιασμό. Στο Κεφάλαιο 2 όπου θα αναφερθούμε στο σχεδιασμό των τυπωμένων κυκλωμάτων θα γίνει εκτενής αναφορά.

Εικόνα 9 : NodeMCU ESP-12 Συγκριτικό Μέγεθος

Εικόνα 10 : NodeMCU ESP-12

1.2.1.6. Αρθρώματα Αισθητήρων (Sensor Modules)

Για τις δοκιμές της κατασκευής, χρησιμοποιήθηκαν οι παρακάτω αισθητήρες:

1.2.1.6.1. Μετασχηματιστής έντασης (CT) για τη μέτρηση ένταση ρεύματος

Εικόνα 11 : Μετασχηματιστής Έντασης (CT)

Εικόνα 12 : Ενδιάμεσο Κύκλωμα CT

1.2.1.6.1.1. Βαθμονόμηση αισθητήρα

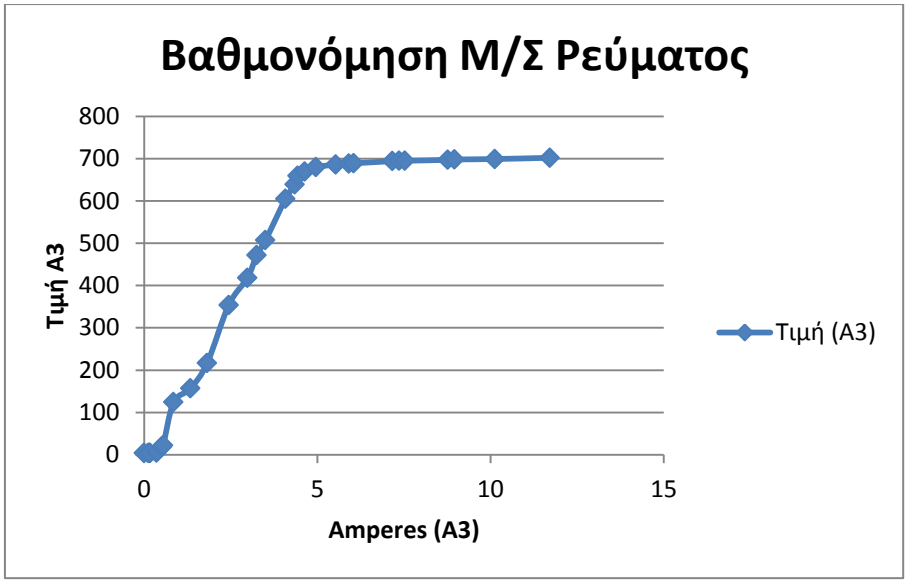
Για να γίνει δυνατή η αξιοποίηση του αισθητήρα πρέπει να ακολουθηθεί η διαδικασία βαθμονόμησης του. Από τη διαδικασία αυτή γίνεται μετατροπή των τιμών που λαμβάνονται από τον αισθητήρα και αντιστοιχούνται με την τιμή που μετράται. Για να γίνει η βαθμονόμηση χρησιμοποιήθηκε το Excel, όπου καταχωρήθηκαν σε ένα πίνακα οι τιμές που λαμβάνονται από τον αισθητήρα και αντίστοιχα οι τιμές έντασης ρεύματος που απέδιδε τις τιμές του αισθητήρα.

Τιμή (A3)	Amperes
4	0
4	0,12
4	0,14
4	0,16
4	0,17
5	0,36
8	0,38
18	0,5
20	0,52
22	0,55
124	0,85
157	1,34
217	1,82
354	2,44
418	2,98
472	3,25

Τιμή (A3)	Amperes
507	3,5
605	4,08
639	4,35
659	4,43
669	4,63
680	4,96
686	5,53
688	5,91
689	6,05
694	7,17
695	7,36
695	7,53
697	8,76
698	8,96
699	10,12
702	11,71

Πίνακας 1: Πίνακας βαθμονόμησης Αισθητήρα ρεύματος (Μ/Σ Ρεύματος)

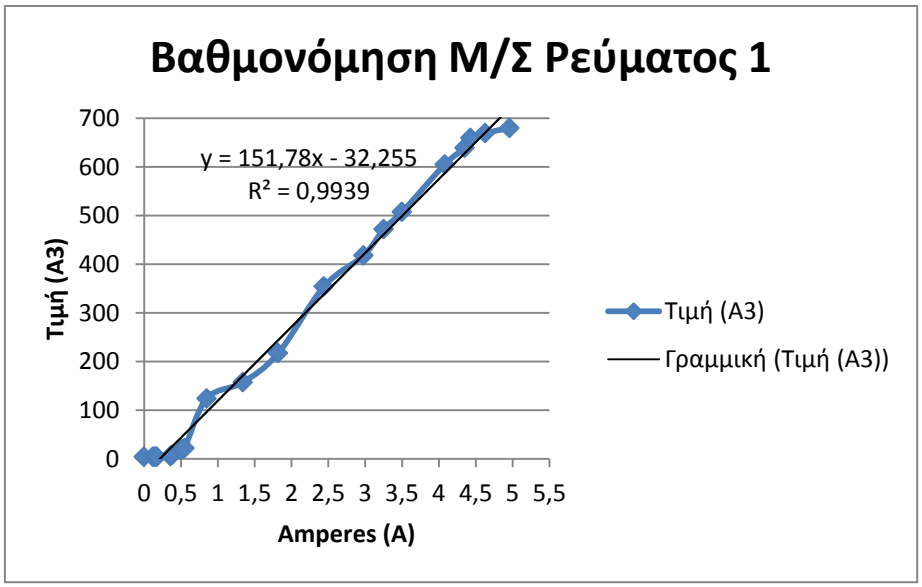
Το αποτέλεσμα της βαθμονόμησης είναι η παρακάτω καμπύλη στο Γράφημα 1.



Γράφημα 1:Καμπύλη Βαθμονόμησης Μ/Σ Ρεύματος

Από το Γράφημα 1, φαίνεται ότι ο αισθητήρας έχει δύο γραμμικές περιοχές λειτουργίας, αυτές οι περιοχές ορίζονται (0-5 Α) και (5-11 Α). Η δεύτερη περιοχή (5-11 Α), δείχνει ότι είναι σταθερή δεν γίνεται να χρησιμοποιηθεί, άρα καταλήγουμε ότι, ο συγκεκριμένος αισθητήρας μπορεί να αξιοποιηθεί και να δίνει ικανοποιητικές μετρήσεις για διάστημα (0-5Α).

Οπότε αφαιρούνται οι τιμές των μετρήσεων από 5Α και άνω και έχουμε το γράφημα 2.



Γράφημα 2:Βαθμονόμηση περιοχής μετρήσεων 0-5Α

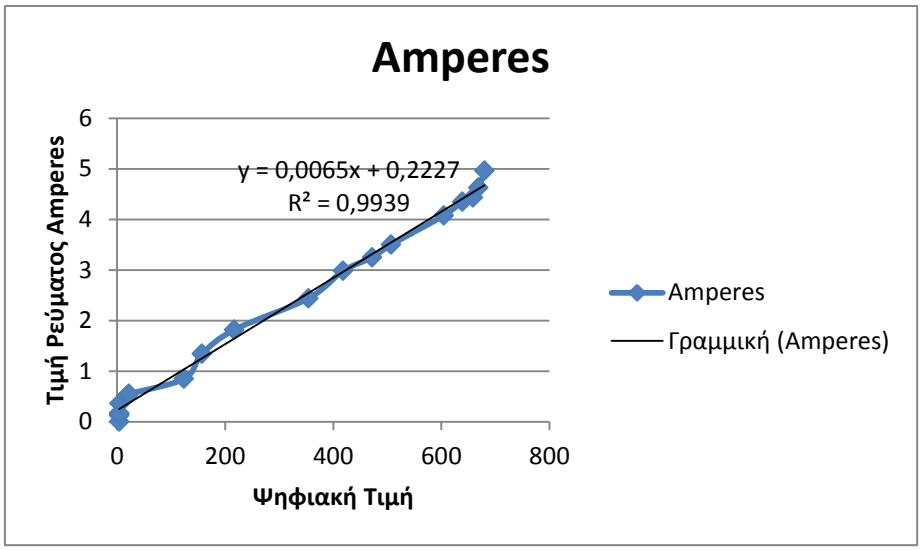
Όπως φαίνεται η γραμμή τάσης που είναι γραμμική πλέον ακολουθεί τη συνάρτηση

$$Y = 151{,}78 \times X - 32{,}255$$

με R^2=0.9939

Το γεγονός ότι το R2 είναι πολύ κοντά στη μονάδα σημαίνει πρακτικά, ότι οι τιμές της συνάρτησης είναι πολύ κοντά στις πραγματικές τιμές.

Επειδή η συνάρτηση που προέκυψε δίνει τιμές αισθητήρα ανάλογα με το ρεύμα που περνάει, εμείς χρειαζόμαστε την αντίστροφη συνάρτηση η οποία φαίνεται στο γράφημα 3.



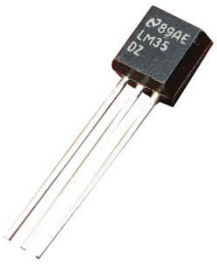
Γράφημα 3:Συμπληρωματική συνάρτηση

Επομένως η συμπληρωματική συνάρτηση είναι

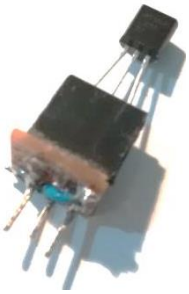
$$Y = 0{,}0065 \times X + 0{,}2227$$
$$R^2 = 0{,}9939$$

Η οποία συνάρτηση, επιστρέφει τις τιμές έντασης ρεύματος για κάθε τιμή που διαβάζεται από τον αισθητήρα.

1.2.1.6.2. Αισθητήρας Θερμοκρασίας LM35



Εικόνα 13 : LM35



Εικόνα 14 : LM35 Module

Για αισθητήρα θερμοκρασίας χρησιμοποιήθηκε το LM35DZ. Είναι γραμμικός αισθητήρας και επιστρέφει 0,10 mV αν βαθμό Κελσίου ^{o}C [M.2]. Το LM35DZ τοποθετήθηκε σε χάλκινο σωληνάριο ώστε να μπορεί να τοποθετηθεί στο κυάθιο της ηλεκτρικής αντίστασης του Ηλιακού θερμοσίφωνα (στο σημείο που μπαίνει ο θερμοστάτης), έτσι ώστε να γίνει άμεση μεταφορά θερμοκρασίας του νερού του ηλιακού. Επίσης κατασκευάστηκε σχετικό προσαρμοστικό κύκλωμα με ένα ποτενσιόμετρο ώστε να ρυθμίζεται στην κατάλληλη έξοδο.

Ενώ είναι γνωστή η γραμμικότητα του αισθητήρα, πραγματοποιήθηκε βαθμονόμηση του. Η διαδικασία είχε ως εξής, μέσα σε ποτήρι, τοποθετήθηκε νερό και παγάκια ώστε να πέσει η θερμοκρασία του νερού. Μαζί με τον αισθητήρα τοποθετήθηκε και ένα θερμόμετρο για να αντιπαραβάλουμε την πραγματική τιμή θερμοκρασίας του νερού με την μετρήσιμη τιμή τάσης εξόδου του αισθητήρα. Από τη διαδικασία βαθμονόμησης δημιουργήθηκε ο παρακάτω πίνακας τιμών.

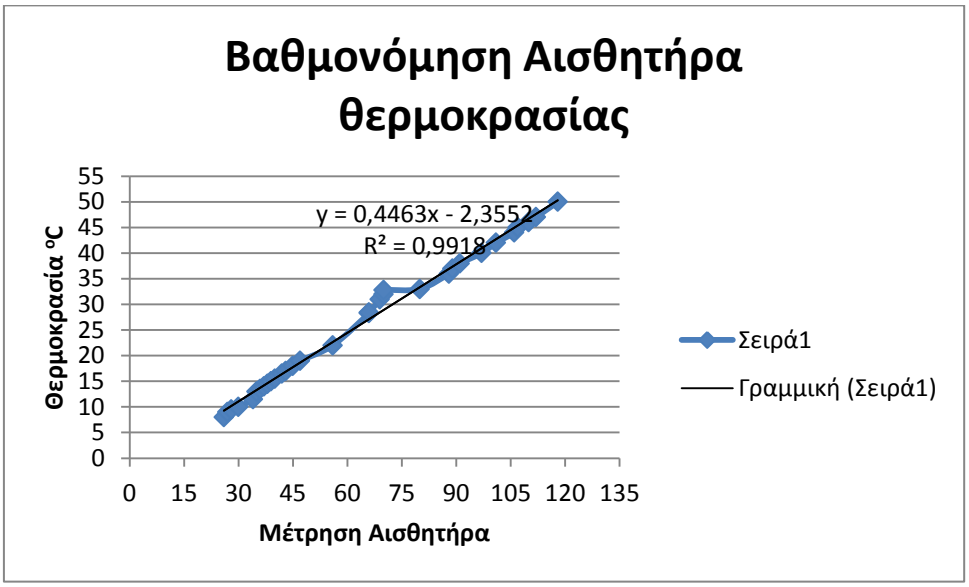
Μέτρηση	θερμοκρασία
26	8
27	9
28	9,5
30	10
34	11,5
35	13
36	13,5
37	14
38	14,5
39	15
40	15,5
42	16,5
43	17
45	18

Μέτρηση	θερμοκρασία
66	28,3
66	28,4
69	31
70	32
70	32,8
80	33
88	36
89	37
91	38
97	40
101	42
106	44
107	45
110	46

47	19
56	22

112	47
118	50

Πίνακας 2: Τιμές Βαθμονόμησης Αισθητήρα Θερμοκρασίας LM35DZ



Γράφημα 4:Βαθμονόμηση Αισθητήρα Θερμοκρασίας LM35DZ

$$Y = 0{,}4463 \times X - 2{,}3557$$
$$R^2 = 0{,}9918$$

Όπως φαίνεται από R2 =0,9918 φαίνεται ότι η συνάρτηση $Y = 0{,}4463 \times X - 2{,}3557$ είναι πάρα πολύ κοντά στην πραγματικότητα και μας επιστρέφει την τιμή θερμοκρασία για κάθε τιμή που διαβάζεται από την αναλογική θύρα.

1.2.1.6.3. Άρθρωμα Ηλεκτρονόμου (Relay Module)

Εικόνα 15: Άρθρωμα Ηλεκτρονόμου

1.2.2. Αρχιτεκτονική Λογισμικού

1.2.2.1.1. Πρωτόκολλο MQTT (MQ Telemetry Transport)

Το MQTT είναι πρωτόκολλο επικοινωνίας machine-to-machine (μηχανής προς μηχανή) (M2M) στο "Διαδίκτυο των πραγμάτων". Σχεδιάστηκε ως μια εξαιρετικά ελαφριά μεταφορά μηνυμάτων βασισμένη στην αρχιτεκτονική subscribe / publish. Είναι εξαιρετικά

χρήσιμο στην επικοινωνία απομακρυσμένων τοποθεσιών όπου απαιτείται μεταφορά μικρής πληροφορίας και το εύρος ζώνης δικτύου (bandwidth) είναι εξαιρετικά πολύτιμο. Αυτό οφείλετε στο γεγονός ότι διαθέτει κεφαλίδα πακέτου σταθερή και ίση με δύο (2) bytes. Το MQTT στηρίζεται στο TCP πρωτόκολλο και εξασφαλίζει την μεταφορά των μηνυμάτων από τον MQTT Client στον MQTT Server.

Έχει χρησιμοποιηθεί σε αισθητήρες που επικοινωνούν με έναν MQTT Server ή MQTT Broker, μέσω δορυφορικής σύνδεσης ή μέσω dial-up συνδέσεων με παρόχους υγειονομικής περίθαλψης και σε σειρά σεναρίων έξυπνου σπίτι. Είναι επίσης ιδανικό για κινητές εφαρμογές λόγω του μικρού μεγέθους, της χαμηλής κατανάλωσης ισχύος, των ελαχιστοποιημένων πακέτων δεδομένων και της αποτελεσματικής μεταφοράς πληροφοριών σε έναν ή πολλούς δέκτες.

Το MQTT σχεδιάστηκε από τον Dr Andy Stanford-Clark της IBM, and Arlen Nipper της Arcom (τώρα Eurotech), το έτος 1999. Από τον Μάρτιο του 2013, έγινε standard από τον οργανισμό OASIS (www.mqtt.org).

Οι βασικές αρχές του πρωτοκόλλου είναι οι παρακάτω[13]:

- Απλό να υλοποιηθεί
- Παρέχει επίπεδα ποιότητας της υπηρεσίας παράδοσης των μηνυμάτων (Quality of Service)
- Αντικειμενικά χαμηλές απαιτήσεις σε υπολογιστική ισχύ και δικτυακό εύρος
- Ανεξάρτητο του είδους των δεδομένων που μεταφέρονται (data agnostic)
- Continuous Session Awareness

1.2.2.1.2. Ασφάλεια Πρωτοκόλλου

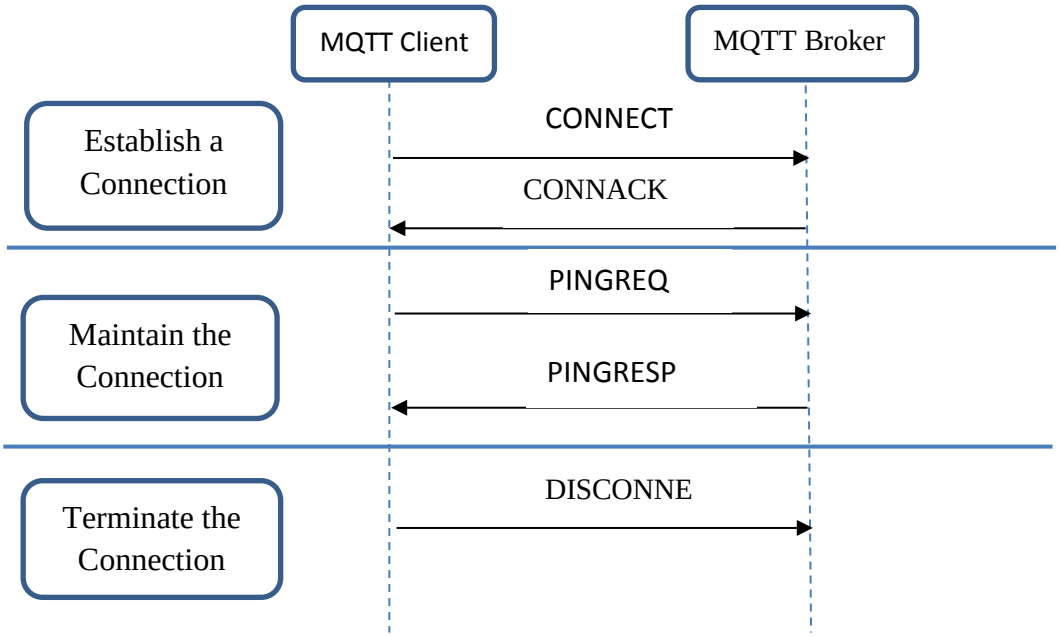
Το πρωτόκολλο MQTT V3.1, υποστηρίζει ασφάλεια, παρέχοντας όνομα χρήστη και κωδικό πρόσβαση. Επίσης υποστηρίζεται κρυπτογράφηση μέσω ασφαλούς σύνδεσης SSL, πρέπει όμως να τονιστεί ότι συνδέσεις SSL απαιτούν υπολογιστική ισχύ και επιβαρύνουν το εύρος του δικτύου λόγω αποστολής μεγάλων κεφαλίδων (header). Επιπρόσθετα θα μπορούσε να προστεθεί εφαρμογή κρυπτογράφησης και αποκρυπτογράφησης των δεδομένων που ανταλλάσσονται, είναι όμως, εκτός πρωτοκόλλου προκειμένου να μην επηρεαστεί η απλότητα του (www.mqtt.org).

Λόγω της σημαντικότητας του θέματος, θα αναπτυχθεί εκτενέστερα στο 4ο κεφάλαιο, όπως θα παρουσιαστούν οι αδυναμίες του πρωτοκόλλου και οι τρόποι αντιμετώπισης τους.

1.2.2.1.3. Λειτουργίες Πρωτοκόλλου MQTT

Μεταξύ MQTT Broker ή MQTT Server και MQTT Client πραγματοποιείται ανταλλαγή πληροφορίας. Πριν την ανταλλαγή πληροφορίας, ανταλλάσσονται πακέτα ελέγχου τα οποίο στηρίζονται στο επίπεδο ποιότητας παρεχόμενης υπηρεσίας QoS.

Κάθε πακέτο ελέγχου MQTT αποτελείται από μια σταθερού μήκους κεφαλίδας ίσης με δύο (2) bytes, μια μεταβλητή κεφαλίδα και το payload (κείμενο που μεταδίδεται). Μερικά πακέτα ελέγχου που ανταλλάσσονται είναι CONNECT, CONNACK, PUBLISH, PUBACK, PUBREC, PUBREL, SUBSCRIBE, SUBACK, κ.α.



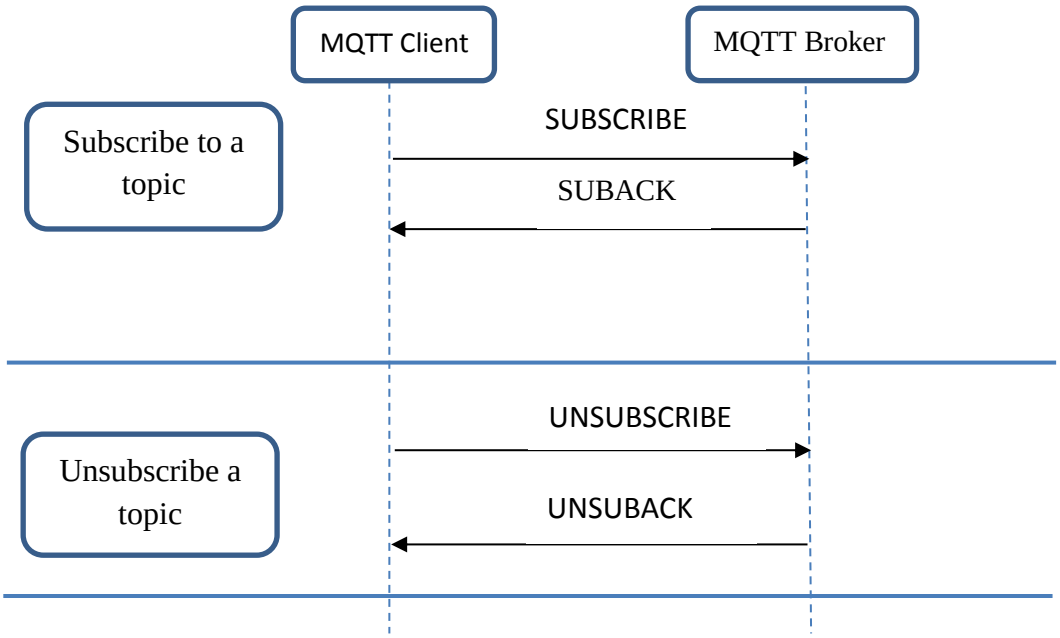
Εικόνα 16:Σύνδεση, Διατήρηση και Τερματισμός MQTT Σύνδεσης

Αποκατάσταση Σύνδεσης –Establishing Connection

Προκειμένου να υλοποιηθεί επιτυχώς μια σύνδεση μεταξύ ενός MQTT Client και του MQTT Broker ανταλλάσσονται μηνύματα. Ο MQTT Client στέλνει το μήνυμα CONNECT και ο MQTT Broker ανταποκρίνεται με ένα μήνυμα CONNACK με κωδικό μηνύματος που καθορίζει την κατάσταση σύνδεσης.

Δημοσίευση μηνυμάτων – Publishing the application messages

Όταν ένας MQTT Client θέλει να γίνει συνδρομητής σε ένα θέμα Publishing σε κάποιο θέμα, τότε στέλνει πακέτο PUBLISH στον MQTT Broker μαζί το επίπεδο ποιότητας QoS, το το θέμα και το κείμενο Payload κ.α. Ο MQTT Broker ανταποκρίνεται ανάλογα με ο QoS, όπως φαίνεται στην εικόνα 22, με ανάλογη ανταλλαγή μηνυμάτων.



Εικόνα 17: MQTT Client Εγγράφεται κασε θέμα Διαγράφεται από θέμα

Συνδρομή σε θέμα - Subscribing to a topic

Όταν ένας MQTT Client θέλει να εγγραφεί Subscribed σε κάποιο θέμα, τότε στέλνει πακέτο SUBSCRIBE στον MQTT Broker και αυτός ανταποκρίνεται με SUBACK πακέτο και επιστέφοντας τη κατάσταση σύνδεσης.

Διατήρηση σύνδεσης – Maintaining the connection alive

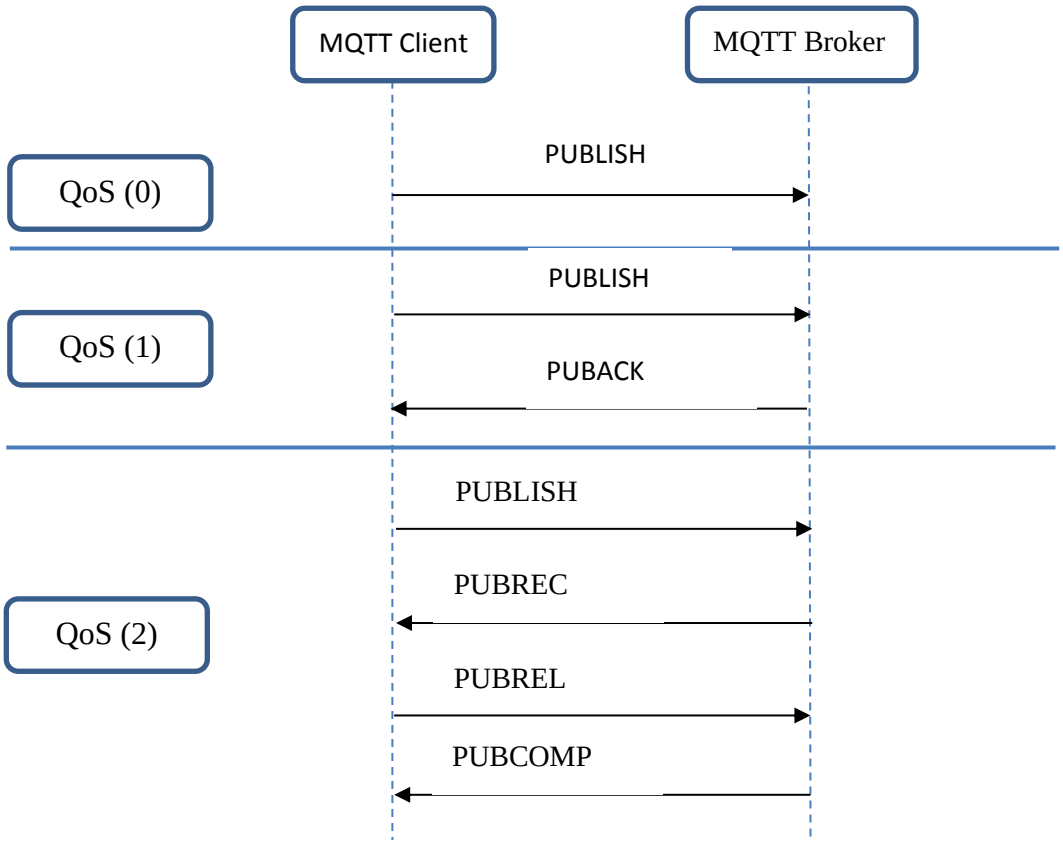
Μετά από σχετικό χρονικό διάστημα time-out, η σύνδεση μεταξύ MQTT Client και του MQTT Broker τερματίζει. Για να διατηρηθεί η σύνδεση, ο MQTT Client αναφέρει ότι είναι εκεί και λειτουργεί στέλνοντας ένα πακέτο PINGREQ και ο MQTT Broker ανταποκρίνεται με ένα πακέτο PINGRESP.

Τερματισμός Σύνδεσης – Terminating the connection

Προκειμένου να τερματιστεί μια σύνδεση, ο MQTT Client στέλνει ένα πακέτο DISCONNECT στον MQTT Broker. Ο MQTT Broker δεν στέλνει τίποτα για ανταπόκριση, όμως ό,τι απευθυνόταν στο MQTT Client σβήνεται και ο MQTT Client αποσυνδέεται από τον MQTT Broker.

Ποιότητα παρεχόμενης Υπηρεσίας (QoS – Quality of Services)

Το πρωτόκολλο παρέχει τρία είδης ποιότητας υπηρεσίας QoS(0), QoS(1) και QoS(2).



Εικόνα 18: Περιγραφή QoS του MQTT

Η περιγραφή των επιπέδων του QoS.

Επίπεδο	Ονομασία	Περιγραφή
QoS(0)	At most Once	Το μήνυμα θα παραδοθεί στον προορισμό του το πολύ μια φορά (μπορεί να μην φτάσει ποτέ)
QoS(1)	At least Once	Το μήνυμα θα παραδοθεί στον προορισμό **τουλάχιστον** μια φορά
QoS(0)	Exactly Once	Το μήνυμα θα παραδοθεί στον προορισμό **ακριβώς** μια φορά

Πίνακας 3: Τα επίπεδα QoS στο MQTT

MQTT Brokers ή MQTT Servers

Όπως αναφέρει ο Δρίβας στην πτυχιακή του εργασία [13], υπάρχει μεγάλη πληθώρα από MQTT Brokers, και οι περισσότεροι είναι ανοικτού κώδικα. Όπως για παράδειγμα οι παρακάτω:

Επωνυμία	**Δικτυακός Τόπος**
Apache Apollo	http://activemq.apache.org/apollo/
IBM Websphere MQ Telemetry	http://www-03.ibm.com/software/products/en/wmq-telemetry
IBM MessageSight	http://www-03.ibm.com/software/products/en/iot-messagesight
Mosquitto	http://mosquitto.org/

Emqttd	https://github.com/emqtt/emqttd
RabbitMQ	http://www.rabbitmq.com/
HiveMQ	http://www.hivemq.com/

Πίνακας 4: MQTT Servers

Εγκατάσταση Mosquitto MQTT Broker στο Raspberry Pi

Σύμφωνα με τον Light [8], το Mosquitto, αφορά μια εφαρμογή συμβατή με τις προδιαγραφές του Εξυπηρετητή (Server) και του πελάτη (Client) του MQTT πρωτοκόλλου μηνυμάτων (messaging protocol). Το MQTT χρησιμοποιεί το μοντέλο δημοσίευσης/εγγραφής (publish/subscribe) και έχει χαμηλό κόστος δικτύου και μπορεί να χρησιμοποιηθεί σε συσκευές χαμηλής κατανάλωσης ενέργειας, όπως μικροεπεξεργαστές που χρησιμοποιούνται σε αισθητήρες του Διαδικτύου των πραγμάτων. Γεγονός που κάνει το Mosquitto ικανό να χρησιμοποιηθεί σε περιπτώσεις που απαιτείται μικρή επικοινωνία, ειδικά σε συσκευές με περιορισμένους πόρους.

Το Mosquitto Project, αποτελείται από τρία μέρη:

1. Τον κύριο Mosquitto Server
2. Τα mosquito_pub και mosquito_sub client εργαλεία, που αποτελούν έναν τρόπο επικοινωνίας με το Mosquitto Server
3. Μια βιβλιοθήκη γραμμένη σε γλώσσα C και έναν wrapper C++

Ολοκληρώνοντας το άρθρο του ο Light [8] , αναφέρεται στην πληθώρα των δυνατοτήτων που παρέχει το Mosquitto, στην έρευνα σε σύγκριση με το Constrained Application Protocol (CoAP), ενός ανάλογου πρωτόκολλου ή την διερεύνηση εφαρμογής του OAuth μέσα στο MQTT, αλλά και την ευρεία χρήση του πρωτοκόλλου σε εφαρμογές όπως Έξυπνες Πόλεις, ανάπτυξη ενός συστήματος παρακολούθησης περιβάλλοντος κ.α.

Λαμβάνοντας υπόψη τα παραπάνω, στην εργασία μας επιλέχθηκε να χρησιμοποιηθεί το Mosquitto MQTT Broker και για τους παρακάτω λόγους. Οι λόγοι που επιλέχθηκε το Mosquitto MQTT Broker, ήταν αφενός ότι είναι ελεύθερο λογισμικό και ανοικτού κώδικα. Αυτό προσέφερε τη δυνατότητα να υπάρχει διαθέσιμος κώδικας για ανάπτυξη της εφαρμογής μας. Επίσης εγκαταστάθηκε πολύ εύκολα και παραμετροποιήθηκε με απλές εντολές. Όμως το πιο σημαντικό προτέρημα του ήταν, ότι λειτούργησε απρόσκοπτα με το Raspberry Pi 3 και από την στιγμή που εγκαταστάθηκε μέχρι και αυτή τη στιγμή που γράφεται η αναφορά της εργασίας και παράλληλα γίνονται διάφορες δοκιμές.

Εγκατάσταση του MQTT Broker στο Raspberry Pi

Στην παρούσα υποενότητα θα παρουσιάσουμε τη διαδικασία εγκατάστασης του Mosquitto MQTT Broker στο Raspeberry Pi 3. Για την εγκατάσταση, ακολουθήθηκε η διαδικασία που ακολουθεί η οποία πάρθηκε από τον δικτυακό τόπο http://www.instructables.com/id/Installing-MQTT-BrokerMosquitto-on-Raspberry-Pi/ (Τελευταία πρόσβαση 13/5/2018)

1. wget http://repo.mosquitto.org/debian/mosquitto-repo.gpg.key
2. sudo apt-key add mosquitto-repo.gpg.key
3. cd /etc/apt/sources.list.d/
4. sudo wget http://repo.mosquitto.org/debian/mosquitto-wheezy.list
5. sudo apt-get update
6. sudo apt-get install mosquitto

Εγκατάσταση του MQTT Clients στο Raspberry Pi

- apt-get install mosquitto-clients

Εντολές ελέγχου σωστής λειτουργίας του MQTT Broker

Εγγραφή στο θέμα “topic” outTopic

- mosquitto_sub -d -t outTopic

Δημοσίευση στο θέμα “topic” outTopic

- mosquitto_pub -d -t outTopic -m "Hello from Manos"

Όποιος client έχει εγγραφεί στο θέμα “topic” outTopic θα λαμβάνει κάθε δημοσίευση που γίνεται στο θέμα αυτό.

1.2.3.MySQL Server

Στα πλαίσια της εργασίας αναφερόμαστε στην αποθήκευση δεδομένων σε βάση δεδομένων, ώστε να δίνεται η δυνατότητα προβολής δεδομένων και επεξεργασίας ώστε να μπορεί να γίνεται λήψη αποφάσεων. Για να αποκτήσουμε τη δυνατότητα αυτή ακολουθήθηκε η διαδικασία εγκατάστασης στο Raspberry Pi ενός MySQL Server.

Εγκατάσταση MySQL Server

Για να γίνει η εγκατάσταση του MySQL Server εκτελέστηκαν οι παρακάτω εντολές:

1. sudo apt-get update
2. sudo apt-get upgrade
3. sudo apt-get install mysql-server --fix-missing

1.2.3.1. phpMyAdmin

Για να γίνει πιο εύκολα η διαδικασία διαχείρισης των βάσεων δεδομένων εγκαταστάθηκε ένα γραφικό περιβάλλον, το phpMyAdmin. Παρά το γεγονός ότι θα μπορούσαμε να χρησιμοποιήσουμε το περιβάλλον γραμμής εντολών του MySQL Server.

Εγκατάσταση phpMyAdmin

sudo apt-get install phpmyadmin

1.2.3.2. Node-RED

Ο Στόχος του προγραμματιστικού περιβάλλοντος Node-RED είναι να ενδυναμώσει προγραμματιστές και μηχανικούς, ώστε να κάνει τον προγραμματισμό όσο το δυνατόν ευκολότερο, με το να συνδέουν γραφικά κουτιά (blocks) που ονομάζονται nodes. Πρακτικά, με αυτό το λογισμικό ακόμα και οι αρχάριοι να μπορούν πολύ γρήγορα να σχεδιάζουν εφαρμογές για το ΔτΠ. Το ολοκληρωμένο περιβάλλον ανάπτυξης (Integrated Development Environment - IDE) του Node-RED αρχικά αναπτύχθηκε από την IBM το 2013, ως λογισμικού ανοικτού κώδικα. Την παρούσα περίοδο, το Node-RED έχει τραβήξει το ενδιαφέρον μιας κοινότητας προγραμματιστών και υποστηρικτών, οι οποίοι συνεισφέρουν με νέα προγραμματιστικά αντικείμενα (nodes). Μια αναπτυσσόμενη λίστα προγραμματιστικών αντικειμένων ανοικτού κώδικα, που επιτρέπουν τη δημιουργία πολύ προχωρημένων εφαρμογών, αρκετά εύκολα, παρέχοντας ταυτόχρονα ένα γραφικό και τονωτικό μαθησιακό περιβάλλον.

Το ερώτημα που τίθεται είναι, τι μπορεί να κάνει κάποιος με το Node-RED; Όπως αναφέρθηκε πρότινος, το προγραμματιστικό περιβάλλον του Node-RED, είναι ένα αποτελεσματικό εργαλείο για να δημιουργούνται εφαρμογές βασισμένες στο ΔτΠ. Οι εφαρμογές αυτές μπορούν να υλοποιήσουν διάφορα σενάρια μεταφοράς και συλλογής δεδομένων, παρακολούθησης και ανάλυσης. Για παράδειγμα, ομάδες ανεξάρτητων σπουδαστών, επαγγελματιών προγραμματιστών ή μηχανικών, θα μπορούσαν να αποφασίσουν να αναπτύξουν εφαρμογές που τους επιτρέπει τον έλεγχο του σπιτιού τους ή του περιβάλλοντος εργασίας τους εξ αποστάσεως. Άλλη ομάδα προγραμματιστών, μπορεί να επιθυμεί να γνωρίζει τι συμβαίνει στα κοινωνικά τους δίκτυα (π.χ Twitter ή Facebook). Το Node-RED απ’ ότι φαίνεται είναι ένα ιδανικό εργαλείο για τη δημιουργία μεγάλου εύρους εφαρμογών που κυκλοφορούν με απλό και εύκολο τρόπο. Το Node-RED, καταρχάς μπορεί να χρησιμοποιηθεί από αρχάριους προγραμματιστές, όμως όπως παρόμοια εργαλεία που δείχνουν ότι προσανατολίζονται προς αρχαρίους, γρήγορα σχετικά, απαιτούν από τους χρήστες να αναπτύξουν προχωρημένες προγραμματιστικές δεξιότητες.

Στην περίπτωσή μας, κάποιος χρήστης θα πρέπει να αποκτήσει γνώσεις προγραμματισμού σε Javascript ώστε να μπορέσει να δημιουργήσει ισχυρές εφαρμογές Node-RED.

Με το να είναι ισχυρό και γραφικό προγραμματιστικό περιβάλλον, το Node-RED έχει στόχο να δημιουργήσει ένα ευρύ φάσμα εφαρμογών του ΔτΠ που συμπεριλαμβάνουν διάφορα ετερογενή συστήματα υλικού και άλλων τύπων εφαρμογών με ευέλικτο τρόπο.

Η εργασία των Chaczko & Braun [10], παρουσιάζει σχετικά απλές εφαρμογές με ομάδα προσανατολισμού εκπαιδευόμενους σπουδαστές. Παρόλα αυτά οι χρήστες μπορούν να αναπτύξουν χρήσιμες εφαρμογές που ξεπερνούν αυτές που παρουσιάζονται.

Επιπρόσθετα, μια άλλη δυνατότητα που δείχνει πόσο δυνατό και χρήσιμο είναι το Node-RED ως εργαλείο, είναι η υποστήριξη εισαγωγής/εξαγωγής κώδικα για τη μεταφορά σε διαφορετικούς σταθμούς εργασίας.

Στην εργασία τους οι Mehari και συν. [15] προτείνουν να χρησιμοποιηθεί το Node-RED ως περιβάλλον προσομοίωσης του ΔτΠ, έτσι ώστε να μελετηθούν συγκεκριμένα προβλήματα, όπως ταχύτητα δικτύου και απόκρισης, σύγκριση απόκρισης ανάλογα με το επίπεδο ποιότητας υπηρεσίας (QoS 0,1,2) κάτι που είναι δύσκολα εφικτό σε πραγματικές συνθήκες, διότι θα πρέπει να υπάρχει το ανάλογο υλικό. Το Node RED παρέχει τη δυνατότητα αυτή, διότι εισάγοντας Nodes τα οποία αιτούνται από το σύστημα, πολύ απλά με μια διαδικασία αντιγραφής/επικόλλησης, προσομοιώνεται μια πραγματική κατάσταση.

2. Κεφάλαιο 2ο

2.1. Σχεδιασμός Τυπωμένων Κυκλωμάτων

Για τον σχεδιασμό των τυπωμένων κυκλωμάτων χρησιμοποιήθηκε το ALTIUM 16. Σε κάποιες περιπτώσεις χρειάστηκε να δημιουργηθούν εξαρτήματα διότι δεν υπήρχαν στις υπάρχουσες βιβλιοθήκες.

Στην συνέχεια παρατίθενται τα σχηματικά, το κάτω μέρος των πλακετών και το άνω μέρος του τυπωμένου κυκλώματος, ενώ και η τελική μορφή της κατασκευής.

2.1.1. Πλακέτα Ηλεκτρικού Πίνακα

Στην ενότητα αυτή θα αναλύσουμε το σχηματικό διάγραμμα της κατασκευής. Τοποθετήθηκε μια βάση για να υποδεχθεί το Nodemcu board. Αυτό το Board παρέχει όλες απαραίτητες ακίδες του ολοκληρωμένου κυκλώματος ESP8266. Επειδή το NodeMCU παρέχει μόνο μια αναλογική είσοδο/έξοδο αυτό είναι ένα μικρό μειονέκτημα. Προκειμένου

να ξεπεραστεί το πρόβλημα αυτό, στην κατασκευή μας συνδέθηκε έναν ολοκληρωμένο κύκλωμα (IC) πολυπλέκτης/αποπολυπλέκτης το CD4051 [M.1]. Το ολοκληρωμένο αυτό, μαζί με τρεις GPIO του Nodemcu επεκτείνει τις αναλογικές θύρες της κατασκευής μας σε οκτώ (8). Έτσι η τελική κατασκευή προσφέρει τέσσερις (4) GPIO θύρες εισόδου/εξόδου και οκτώ (8) αναλογικές θύρες εισόδου/εξόδου (στο firmware που έχουμε δημιουργήσει χρησιμοποιούνται μόνο ως είσοδοι διότι ο σκοπός μας είναι η συλλογή δεδομένων. Σε μελλοντική εργασία θα μπορούσαν να γίνονται και αναλογικές είσοδοι.

Επίσης στην κατασκευή τοποθετήθηκε αυτόνομο τροφοδοτικό 220V AC – 5V DC, το HLK-pm01 [M.3]. Για την παροχή τροφοδοσίας της κατασκευής. Οπότε πάνω στην πλακέτα της κατασκευής τοποθετήθηκαν οι τέσσερις (4) ψηφιακές και οκτώ (8) αναλογικές θύρες σε κατάλληλες υποδοχές μαζί με μια υποδοχή 5V και GND για να τοποθετούνται τα αρθρώματα (Δείτε Εικόνα 13).

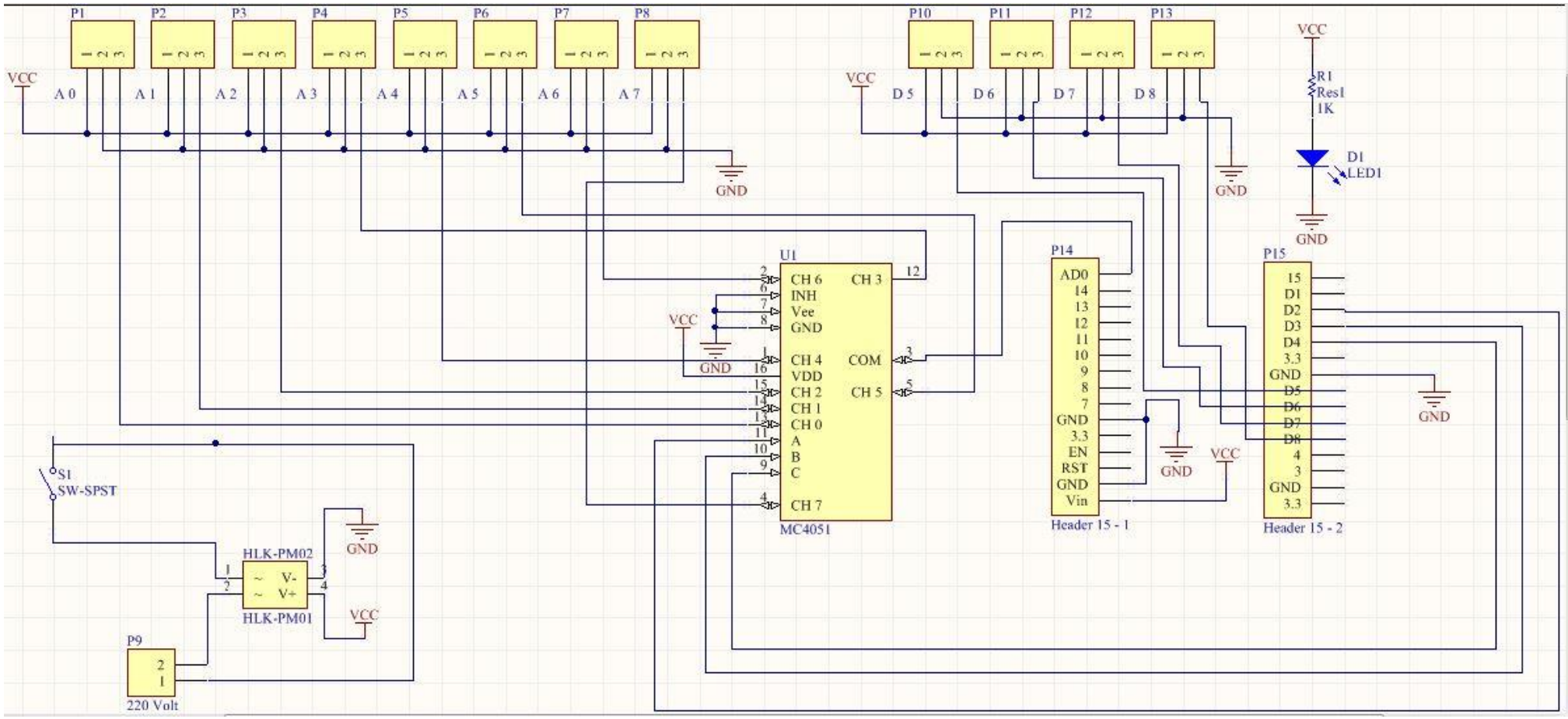
Επίσης στην κατασκευή, προστέθηκαν διακόπτης ενεργοποίησης και ενδεικτική λυχνία λειτουργίας.

Λίστα υλικών – Κατασκευής Ηλεκτρικού Πίνακα

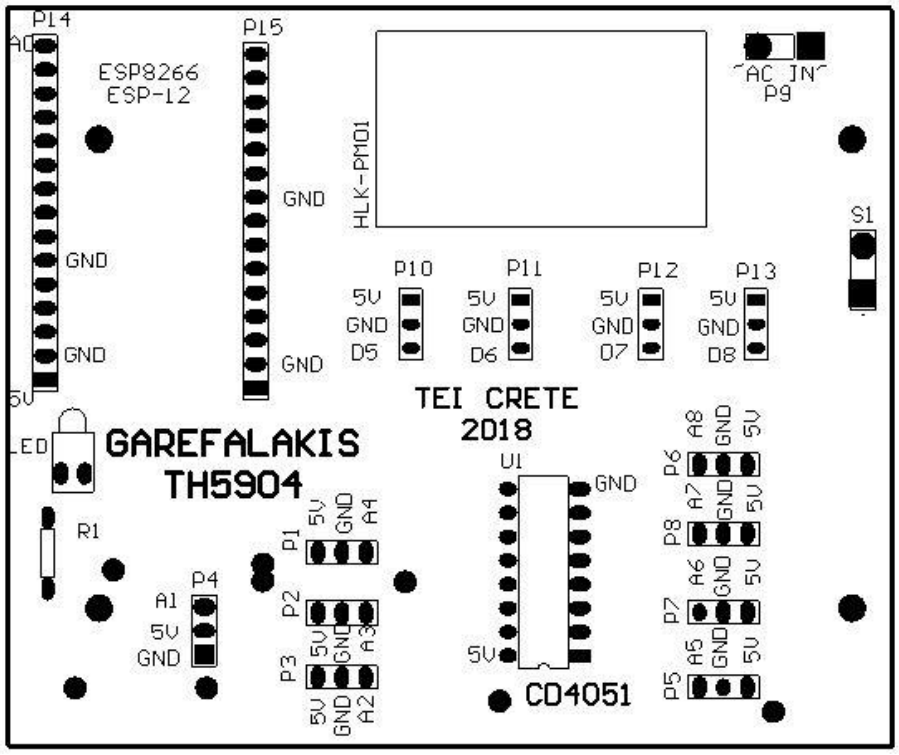
220 Volt	HDR1X2	1	P9	Header, 2-Pin
A 0	HDR1X3	1	P1	Header, 3-Pin
A 1	HDR1X3	1	P2	Header, 3-Pin
A 2	HDR1X3	1	P3	Header, 3-Pin
A 3	HDR1X3	1	P4	Header, 3-Pin
A 4	HDR1X3	1	P5	Header, 3-Pin
A 5	HDR1X3	1	P6	Header, 3-Pin
A 6	HDR1X3	1	P7	Header, 3-Pin
A 7	HDR1X3	1	P8	Header, 3-Pin
D 5	HDR1X3	1	P10	Header, 3-Pin
D 6	HDR1X3	1	P11	Header, 3-Pin
D 7	HDR1X3	1	P12	Header, 3-Pin
D 8	HDR1X3	1	P13	Header, 3-Pin
Header 15 - 1	HDR1X15	1	P14	Header, 15-Pin
Header 15 - 2	HDR1X15	1	P15	Header, 15-Pin
HLK-PM01	PCBComponent_	1	HLK-PM01	PCB Converter 220V AC/ 5V D
LED1	LED-1	1	LED	Typical RED GaAs LED
MC4051	J016D	1	U1	1-of-8 Data Selector/Multiplexe
Res1	AXIAL-0.3	1	R1	Resistor
SW-SPST	SPST-2	1	S1	Single-Pole, Single-Throw Switc

Πίνακας 5:Λίστα υλικών – Κατασκευής Ηλεκτρικού Πίνακα

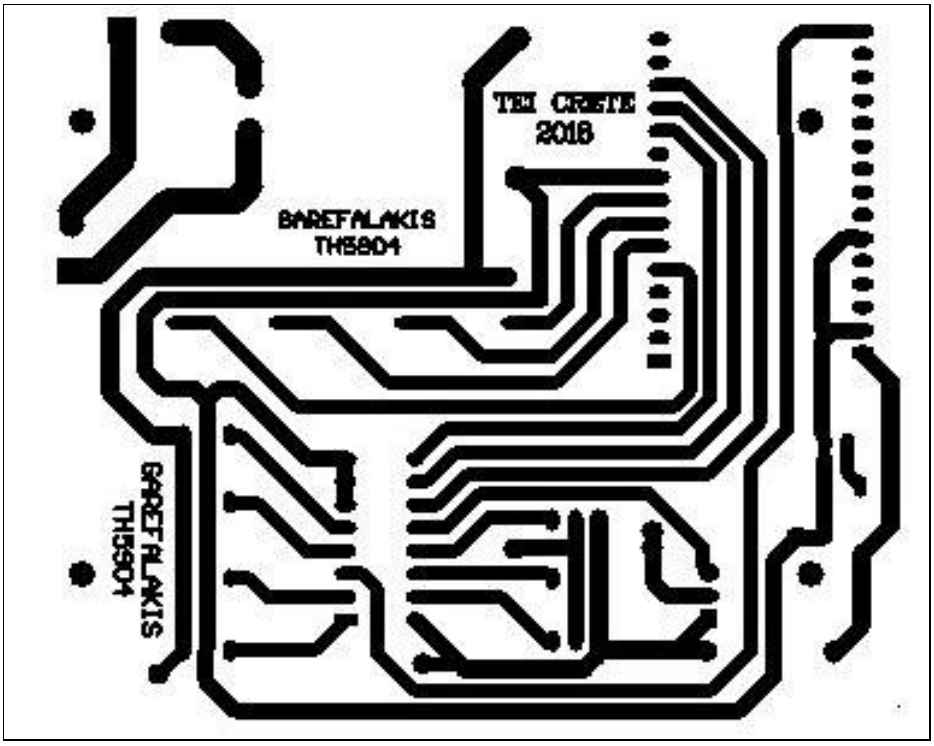
Σχηματικό Διάγραμμα



Εικόνα 19 : Σχηματικό Πλακέτας Ηλεκτρικού Πίνακα



Εικόνα 20: Πάνω μέρος Πλακέτας Ηλεκτρικού Πίνακα



Εικόνα 21: Κάτω μέρος Πλακέτας Ηλεκτρικού Πίνακα

2.1.2. Πλακέτα Αρθρώματος Ηλεκτρονόμου

Η λογική του κυκλώματος, είναι όταν θα δέχεται τάση ενεργοποίησης του ηλεκτρονόμου, να ενεργοποιείται και να κλείνει το κύκλωμα. Επειδή το NodeMCU

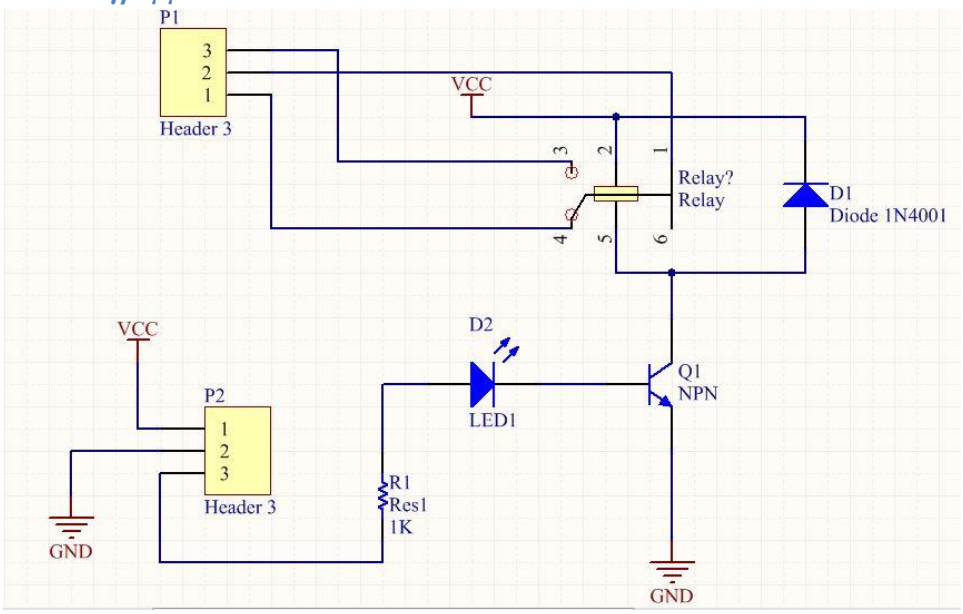
στέλνει τάση 3.3V στις ψηφιακές θύρες εισόδου και εξόδου, που δεν είναι αρκετή για να ενεργοποιήσει τον ηλεκτρονόμο SRS-05 VDC, εφαρμόζουμε το εξής τέχνασμα. Χρησιμοποιώντας ένα τρανζίστορ PNP ως διακόπτη. Εφαρμόζουμε τάση στη βάση αρκετή ώστε να ανοίξει το τρανζίστορ και κλείσει το κύκλωμα των 5V που ενεργοποιεί τον ηλεκτρονόμο. Ταυτόχρονα, για να έχουμε και οπτική αντίληψη έχει τοποθετηθεί ενδεικτική λυχνία που ανάβει όταν ενεργοποιείται ο ηλεκτρονόμος.

Λίστα υλικών - Άρθρωμα Ηλεκτρονόμου

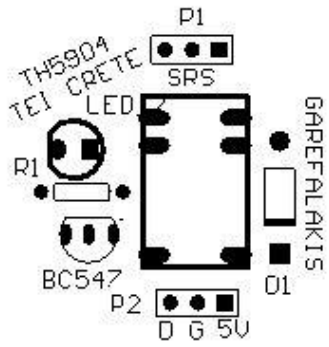
Comment	Pattern	Quantity	Components	Description
Diode 1N4001	DO-41	4	D1	1 Amp General Purpose Rectifi
Header 3	HDR1X3	2	P1, P2,	Header, 3-Pin
LED1	LED	4	LED	Typical RED GaAs LED
NPN	TO-226-AA	4	BC547	NPN Bipolar Transistor
Relay	RELAY4	4	SRS	Relay
Res1	AXIAL-0.3	4	R1	Resistor

Πίνακας 6:Λίστα υλικών - Άρθρωμα Ηλεκτρονόμου

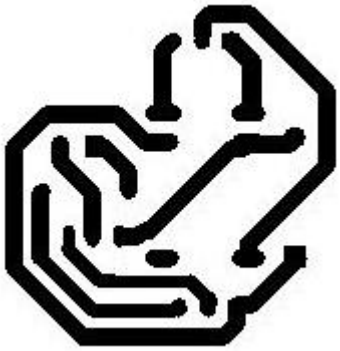
Σχηματικό Διάγραμμα



Εικόνα 22: Σχηματικό Αρθρώματος Ηλεκτρονόμου



Εικόνα 23 : Πάνω μέρος Αρθρώματος Ηλεκτρονόμου



Εικόνα 24: Κάτω μέρος Αρθρώματος Ηλεκτρονόμου

2.1.3. Πλακέτα Αρθρώματος Μετασχηματιστή Έντασης (CT)

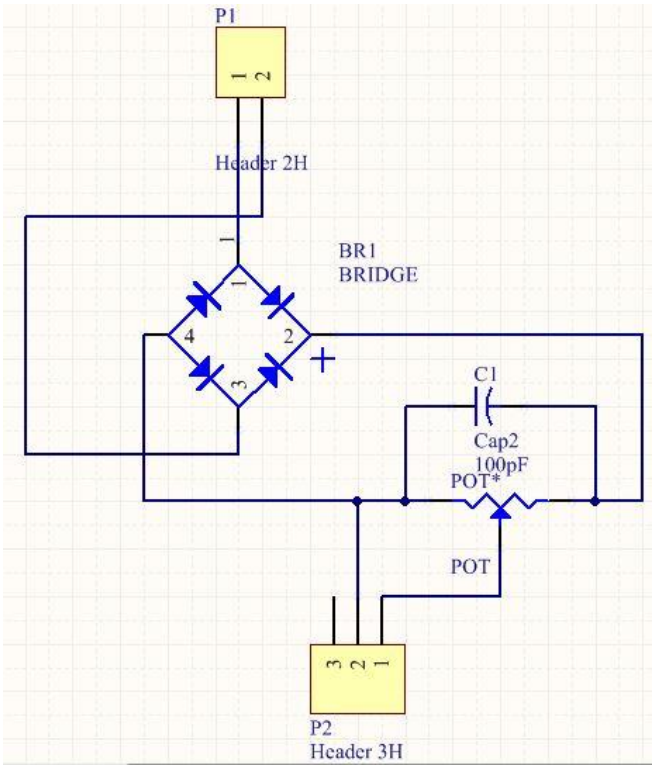
Η λογική του κυκλώματος του αρθρώματος του μετασχηματιστή έντασης έχει ως εξής: Ο μετασχηματιστής έντασης ρεύματος «αγκαλιάζει» έναν αγωγό τροφοδοσίας εναλλασσόμενου ρεύματος. Ανάλογα με ένταση του ρεύματος που διαρρέει τον αγωγό δημιουργείται ένα απαγωγικό ρεύμα στα άκρα του μετασχηματιστή. Η Τάση αυτή έχει λόγο μετασχηματισμού 2000 δηλαδή 100 A δίνουν 50 mA. Επειδή το ρεύμα που διαρρέεται είναι εναλλασσόμενο ομοίως και το ρεύμα εξόδου του Μ/Σ είναι εναλλασσόμενο. Άρα για να μπορέσουμε να πάρουμε μια σταθερή τιμή, χρησιμοποιούμε μια γέφυρα ανόρθωσης (DB104) και έναν πυκνωτή (47μF) εξομάλυνσης, ώστε να λαμβάνεται σταθερή τάση στην έξοδο του αρθρώματος. Επειδή, τάση δεν πρέπει να ξεπερνάει τα 3.3 V που μπορεί να διαβάσει η αναλογική θύρα του Nodemcu, εφαρμόσαμε έναν διαιρέτη τάση ώστε η τάση να μένει στα επιθυμητά όρια 0-3.3V.

Λίστα υλικών – Άρθρωμα Μ/Σ ρεύματος

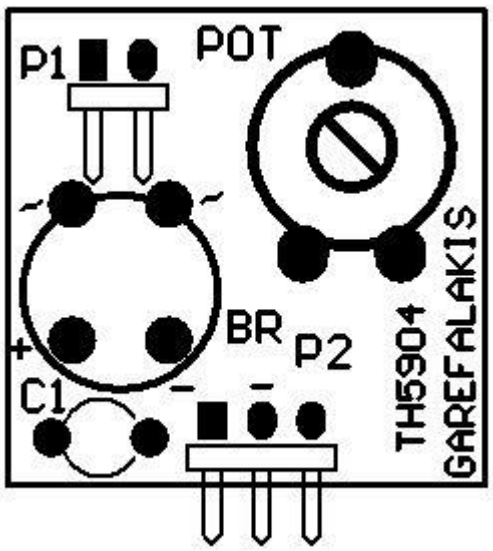
Comment	Pattern	Quantity	Components	Description
BRIDGE	BRIDGE	1	BR1	Bridge
Cap2	CAPR5-4X5	1	C1	Capacitor
Header 2H	HDR1X2H	1	P1	Header, 2-Pin, Right A
Header 3H	HDR1X3H	1	P2	Header, 3-Pin, Right A
POT	VR HOR	1	POT	Potentiometer

Πίνακας 7:Λίστα υλικών – Άρθρωμα Μ/Σ ρεύματος

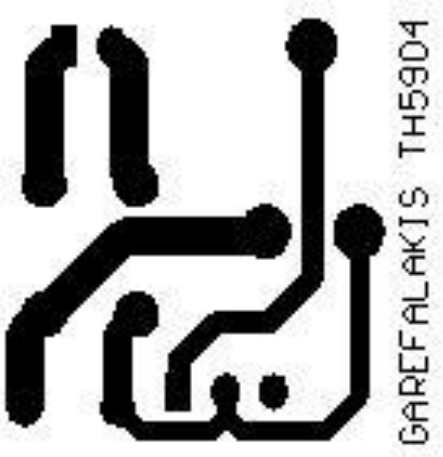
Σχηματικό Διάγραμμα



Εικόνα 25: Σχηματικό Αρθρώματος Μ/Σ Έντασης



Εικόνα 26 : Πάνω μέρος Πλακέτας



Εικόνα 27 : Κάτω μέρος Πλακέτας

2.2. Δημιουργία Τυπωμένων Κυκλωμάτων

Για τη δημιουργία των τυπωμένων κυκλωμάτων, χρησιμοποιήθηκαν φωτοευαίσθητες πλακέτες χαλκού, μονής επίστρωσης. Λόγω του γεγονότος ότι ήταν επιθυμητό στις πλακέτες να υπάρχει άνεση χώρου, δεν κρίθηκε απαραίτητο να χρησιμοποιηθούν πλακέτες διπλής επίστρωσης.

Όπως προαναφέρθηκε το πρόγραμμα δημιουργίας των τυπωμένων κυκλωμάτων ήταν το ALTIUM 16. Οι φάσεις που ακολουθήθηκαν για τη δημιουργία των τυπωμένων ήταν οι ακόλουθες:

1. Σχεδιασμός σχηματικού του ηλεκτρονικού κυκλώματος
2. Εκτύπωση της άνω όψης σε διαφάνεια διαφανοσκοπείου.
3. Εκτύπωση την κάτω όψης σε ριζόχαρτο.
4. Έκθεση της κάτω όψης στη φωτοευαίσθητη επιφάνεια της πλακέτας (Παράρτημα 1).
5. Εμφάνιση της φωτοευαίσθητης επιφάνειας, (Παράρτημα 2)
6. Αποχάλκωση πλακέτας (Παράρτημα 3)
7. Επικασσιτέρωση χαλκού (Παράρτημα 4)
8. Εκτύπωση Πάνω επιφάνεια με τη χρήση Ηλεκτρικού Σίδερου (Υψηλή θερμοκρασία)
9. Τρύπημα πλακέτας
10. Τοποθέτηση Εξαρτημάτων
11. Έλεγχος καλής λειτουργίας

3. Κεφάλαιο 3^{ο}

3.1. Ανάπτυξη Λογισμικού

Για την δημιουργία των Firmware των ESP-01 και ESP-12 χρησιμοποιήθηκε το Arduino IDE 1.8.2. διότι υπήρχε διαθέσιμος κώδικας σε γλώσσα προγραμματισμού C και βιβλιοθήκες που έδιναν σύνδεση σε MQTT Broker. Οπότε χρησιμοποιώντας το διαθέσιμο κώδικα και με τις ανάλογες βελτιώσεις δημιουργήθηκε ο κώδικας για τα ολοκληρωμένα ESP-01 και ESP-12.

Λαμβάνοντας υπόψη ότι τα δύο ολοκληρωμένα χρησιμοποιούνται από το σύστημα για διαφορετικούς σκοπούς, τα firmware είναι διαφορετικά. Όμως, για να επικοινωνούν με το κεντρικό σύστημα ενσωματώθηκαν κοινές εντολές για να

ανταποκρίνονται το ίδιο. Επίσης, επειδή ήταν επιθυμητό να αναβαθμίζονται αυτόματα, ενσωματώθηκαν εντολές αναβάθμισης firmware. Λόγω του ότι, το ESP-01 έχει μικρότερη μνήμη από το ESP-12 η αναβάθμιση γίνεται με διαφορετικό τρόπο.

3.1.1. Firmware ESP-01

Το ESP-01, όπως προαναφέραμε χρησιμοποιήθηκε για να ενεργοποιεί έναν αυτόνομο ηλεκτρονόμο που βρίσκεται ενσωματωμένος μέσα σε πρίζα. Οπότε έχει προγραμματιστεί να κάνει τις εξής λειτουργίες όταν δέχεται τις ανάλογες εντολές:

- Ενεργοποίηση πρίζας
- Απενεργοποίηση πρίζας
- Αντιστροφή κατάστασης όταν πατηθεί το κουμπί (δηλαδή να ανάψει, αν είναι σβηστό και το αντίστροφο)
- Κοινές εντολές

Εάν για κάποιο λόγο, γίνει επανεκκίνηση του ολοκληρωμένου, τότε «θυμάται» την προηγούμενη κατάστασή του. Δηλαδή, αν γίνει επανεκκίνηση και ήταν σβηστό θα γίνει επανεκκίνηση και θα παραμείνει σβηστό. Αν τώρα, γίνει επανεκκίνηση και είναι ενεργοποιημένο, τότε μετά την επανεκκίνηση θα ενεργοποιηθεί πάλι.

Για την τροφοδοσία του ESP-01 και του ηλεκτρονόμου χρησιμοποιήθηκε, έτοιμο άρθρωμα (module) μετατροπέας 220V AC σε DC 5V. Προτιμήθηκε, έτοιμο άρθρωμα, λόγω χαμηλού κόστους (~ 1,5€) αλλά κυρίως λόγω μικρού όγκου, ώστε να μπορέσουν να συμπεριληφθούν όλα μέσα στη συσκευασία.

Στο Παράρτημα 5 βρίσκονται φωτογραφίες της κατασκευής και ο κώδικας που έχει φορτωθεί.

3.1.2. Firmware ESP-12

Το ESP-12, σε αντίθεση με το ESP-01 έχει περισσότερες δυνατότητες. Αρχικά διαθέτει οκτώ (8) αναλογικές θύρες και τέσσερις (4) GPIO που έχουν τη δυνατότητα να ρυθμιστούν ως είσοδοι ή έξοδοι. Οπότε λειτουργίες που έχουν προγραμματιστεί να κάνει είναι:

- Να διαβάσει Αναλογική θύρα.
- Να διαβάσει Ψηφιακή θύρα (GPIO)
- Να ενεργοποιεί Ψηφιακή θύρα (GPIO)
- Κοινές εντολές

Στο Παράρτημα 6 βρίσκονται φωτογραφίες της κατασκευής που τοποθετείται σε Ηλεκτρικό Πίνακα και δείγμα εφαρμογής σε ηλεκτρικό πίνακα.

Κοινές Εντολές

Ο ρόλος των κοινών εντολών είναι για να ανταποκρίνονται τα Nodes ως μέρος του συστήματος και έχουν ρόλο εποπτικό και διαχειριστικό. Ο κατάλογος που υπάρχει μέχρι αυτή τη στιγμή είναι:

BROAD : Είναι εντολή Broadcast. Όταν λαμβάνεται αυτή η εντολή όλα τα Nodes απαντούν στο outTopic στέλνοντας ένα μήνυμα της μορφής **01:OK** όπου 01 είναι το ID του Node. Οπότε ο διαχειριστής γνωρίζει ότι ποια Nodes είναι συνδεδεμένα και λειτουργούν κανονικά.

IDPING: Στέλνεται η εντολή PING στο node με ID. Για παράδειγμα αν σταλεί η εντολή 01PING τότε το Node με κωδικό 01 απαντάει με μήνυμα 01:PING_OK.

IDFIRMWARE: Το Node με κωδικό ID επιστρέφει ποιο firmware έχει εγκατεστημένο.

IDUPDATE: Το Node με κωδικό ID κάνει webupdate, κατεβάζει το firmware και το εγκαθιστά στον εαυτό του.

IDHELP: Παρουσιάζει τις εντολές που μπορεί να εκτελέσει το Node με κωδικό.

3.1.3. Γραφικές Γλώσσες Προγραμματισμού στο Διαδίκτυο των Πραγμάτων

Η αλληλεπίδραση του χρήστη, είναι το κύριο μέλημα στην βιομηχανία παραγωγής λογισμικού. Μεταξύ των πολλών τεχνικών προγραμματισμού, η περισσότερο υποσχόμενη και επικρατούσα είναι οι Γραφικές Γλώσσες Προγραμματισμού (Visual Programming Languages – VPLs)2. Μια VPL, όπως κάθε γλώσσα προγραμματισμού, επιτρέπει στον χρήστη να χειρίζεται τις δομές προγραμματισμού με γραφικό τρόπο, παρά με την πληκτρολόγηση. Οι γλώσσες VPL, η αλλιώς γλώσσες ροής δεδομένων (Dataflow Languages), στηρίζονται στην ιδέα, όπου χρησιμοποιούνται βέλη και κουτιά, και τα βέλη συνδέουν τα κουτιά μεταξύ τους, δημιουργώντας έτσι μια σχέση μεταξύ των κουτιών (π.χ. οντότητες). Έχει φανεί ότι, οι γλώσσες VPL χρησιμοποιούνται σε διάφορους τομείς, όπως π.χ για εκπαιδευτικούς σκοπούς, για τη δημιουργία πολυμεσικών εφαρμογών, ηλεκτρονικά παιχνίδια, ανάπτυξη συστημάτων και προσομοιώσεων, αποθήκευση δεδομένων και ανάλυση δεδομένων κ.α.

Ο Ray [14] στην βιβλιογραφική του επισκόπηση που αφορά τις VPL, παρουσίασε μια λίστα με διάφορες γλώσσες VPL, όπως για παράδειγμα, το Scratch που

[2] Στη συνέχεια της εργασίας θα αναφερόμαστε στις Γραφικές γλώσσες προγραμματισμού με τον όρο VPL.

χρησιμοποιείται ευρέως στην εκπαίδευση,Pure Data (Pd) που χρησιμοποιείται για τον σχεδιασμό διαδραστικών πολυμεσικών εφαρμογών και μουσική μέσω H/Y, Unreal Engine 4 μαζί με το Blueprints χρησιμοποιούνται για τον σχεδιασμό παιχνιδιών, VisSim για το σχεδιασμό πολύπλοκων μαθηματικών μοντέλων και ο κατάλογος είναι πολύ μεγάλος.

Το κύριο αντικείμενο όμως του Ray [14] ήταν η σύγκριση VPLs που χρησιμοποιούνται στο σχεδιασμό εφαρμογών που σχετίζονται με Διαδίκτυο των Πραγμάτων. Από τη βιβλιογραφική του επισκόπηση προέκυψε ο παρακάτω πίνακας:

Type of VPLs **Τύπος VPLs**	**Name of VPLs** **Επωνυμία**	**Programming Environment** **Προγραμματιστικό Περιβάλλον**	**License** **Άδεια Χρήσης**	**Project repository** **Αποθετήριο Έργου**	**Platforms supported** **Υποστηριζόμενες Πλατφόρμες**
Open Source Ανοικτού Κώδικα	**Node-Red**	Web	Open Source-Apache 2.0	Github	**Raspberry Pi**, BeagleBone Black, Docker, Arduino, Android, IBM Bluemix, Amazon Web Services, Microsoft Azure under
	NETLab Toolkit	Web	GPL	Self	Arduino latest Linux embedded systems like the **Raspberry Pi**, Intel Galileo Arduino
	Ardublock	Web	GPL	Self	Arduino
	Scratch for Android (s4a)	Web	GPL2	Self	Arduino
	Modkit	Desktop	GPL2	Self	Arduino, littleBits, Particle Photon, MSP340, Tiva C
	miniBloq	Desktop	RMPL	Self	Multiplo, Arduino, RedBot, RedBoard
	NooDL	Web	NEUL	Self	Arduino, Android
Proprietary Εμπορικό	DGLux5	Desktop	DGLux Engineering License	Self	**Raspberry Pi**, BeagleBone, DGBox
	AT&T Flow Designer	Desktop	GPL3	Github	AT&T IoT SIM
	Reactive Blocks	Desktop	EPL	Self	Modbus, Raspberry Pi USB Camera
	GraspIO	Desktop	BSD	Self	Arduino, Raspberry Pi, GIO Arm, GIO TetraPod, GraspIO boards, Android
	Wyliodrin	Web	GPL3	Self	Arduino, BeagleBone Black, Raspberry Pi,

					Intel Galileo, Intel Edison, UDOO, ZedBoard Red Pitay
	Zenodys	Desktop	—	Self	**Raspberry Pi** Zenobox

Πίνακας 8: Σύγκριση Γραφικών Γλωσσών Προγραμματισμού VPLs

Από τον πίνακα 3, φαίνεται ότι η VPL Node-RED ταίριαζε στις απαιτήσεις της εργασίας μας. Αφενός ελεύθερο λογισμικό, και υποστηριζόταν από το Raspberry Pi. Πολύ σημαντικός λόγος επιλογής ήταν ότι από τις πρώτες δοκιμές, φάνηκε ότι ήταν πολύ εύχρηστη και δημιουργούσε ένα ωραίο περιβάλλον παρουσίασης των αποτελεσμάτων.

Στο σημείο αυτό, κρίνεται απαραίτητο να αναφερθούμε και σε γραφική γλώσσα προγραμματισμού του ESP8266. Στην παρούσα εργασία, χρησιμοποιήθηκε το Arduino IDE [Δ.2] ολοκληρωμένο σύστημα προγραμματισμού, σο οποίο γίνεται προγραμματισμός, στη γλώσσα C και περιλαμβάνει κατάλληλες βιβλιοθήκες που υποστηρίζουν κάθε αναπτυξιακή πλακέτα. Κατά την έρευνα μας, βρέθηκε ένα, αρκετά ενδιαφέρον, που ονομάζεται που ονομάζεται TUNIOT. Το TUNIOT, είναι ένα γραφικό σύστημα προγραμματισμού που μοιάζει πάρα πολύ στη γλώσσα γραμματισμού Scratch του MIT [Δ.11], μια αρκετά διαδεδομένη γλώσσα προγραμματισμού σε γραφικό περιβάλλον και που ο προγραμματισμός γίνεται με την μορφή γραφικών κύβων που ενθυλακώνουν μεταξύ τους.

Το TUNIOT, είναι ουσιαστικά ένας μεταφραστής που μετατρέπει κώδικα που είναι γραμμένος μορφή Scratch, σε γλώσσα C έτοιμη για να χρησιμοποιηθεί στο περιβάλλον Arduino IDE και να γίνει η ενημέρωση του Firmware των Chip. Αυτό δίνει τη δυνατότητα, ακόμα και σε νέους προγραμματιστές ή σε μικρούς μαθητές να προγραμματίζουν εύκολα και ολοκληρωμένα κυκλώματα που στηρίζονται στον επεξεργαστή ESP8266, που χρησιμοποιήθηκες στην εργασία αυτή και να γνωρίσουν τον κόσμο του προγραμματισμού του Διαδικτύου των Πραγμάτων.

Node-RED Ρυθμίσεις - Κώδικας και Στιγμιότυπα

Node-RED Ρυθμίσεις

Πριν να προχωρήσουμε στον κώδικα, πρέπει να αναφερθεί ότι χρειάστηκε να εγκατασταθούν πρόσθετα nodes στην εγκατάσταση του Node-RED, για να μπορέσουμε να δημιουργήσουμε τον κώδικα που ακολουθεί.

Το πρώτο και σημαντικό Node είναι το Dashboard, αποτελεί την επιφάνεια το γραφικό περιβάλλον, σε μορφή ιστοσελίδας όπου εισάγονται τα κουμπιά εντολών, οι μετρητές τα γραφήματα κ.α. [Δ.12].

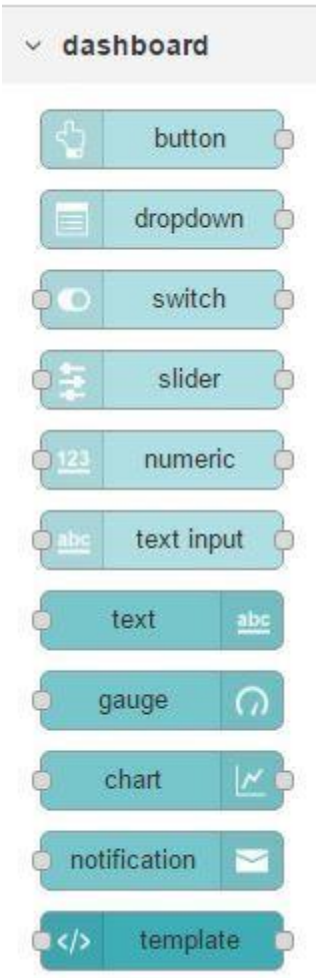
Αρχικά εγκαθιστούμε το npm πακέτο

sudo apt-get install npm
sudo npm install -g npm@2.x

Εν συνεχεία, για την εγκατάσταση του Node Dashboard εκτελούμε

sudo npm install node-red-dashboard

Μετά από επανεκκίνηση του Raspberry Pi 3 και εκτελώντας το Node-RED βλέπουμε τα νέα αντικείμενα



Εικόνα 28: Dashboard Nodes

Επίσης είναι απαραίτητο να εγκατασταθεί το Node που επιτρέπει τη σύνδεση με τη MySQL Βάση Δεδομένων, εκτελώντας την παρακάτω εντολή

npm install node-red-node-mysql

και εμφανίζεται το Node MySQL μετά από επανεκκίνηση του Node-RED.

Εικόνα 29: mysql Node

Τελευταία ρύθμιση έγινε, ήταν ο ορισμός του Node-RED, ώστε να εκτελείται ως υπηρεσία, έτσι σε κάθε επανεκκίνηση του Raspberry Pi 3, να εκτελείται αυτόματα ο Node-RED [Δ.16].

sudo systemctl enable nodered.service

Node-RED Κώδικας και Στιγμιότυπα

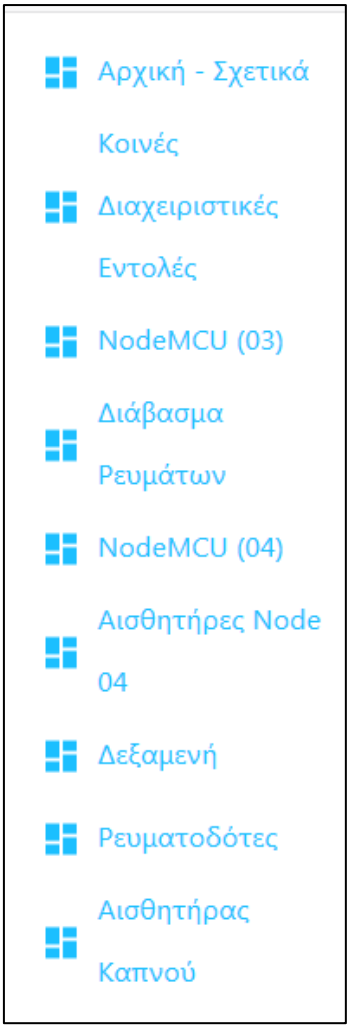
Χρησιμοποιώντας το Node-RED δημιουργήθηκε κώδικας ώστε να είναι δυνατή η ρύθμιση των Nodes της εφαρμογής, αλλά και για να παρουσιάζονται στο χρήστη τα αποτελέσματα σε γραφικό περιβάλλον.

Επειδή, τα Nodes είναι αρκετά στο Node-RED πραγματοποιήθηκε ομαδοποίηση για κάθε Node και δημιουργήθηκε αντίστοιχη καρτέλα.

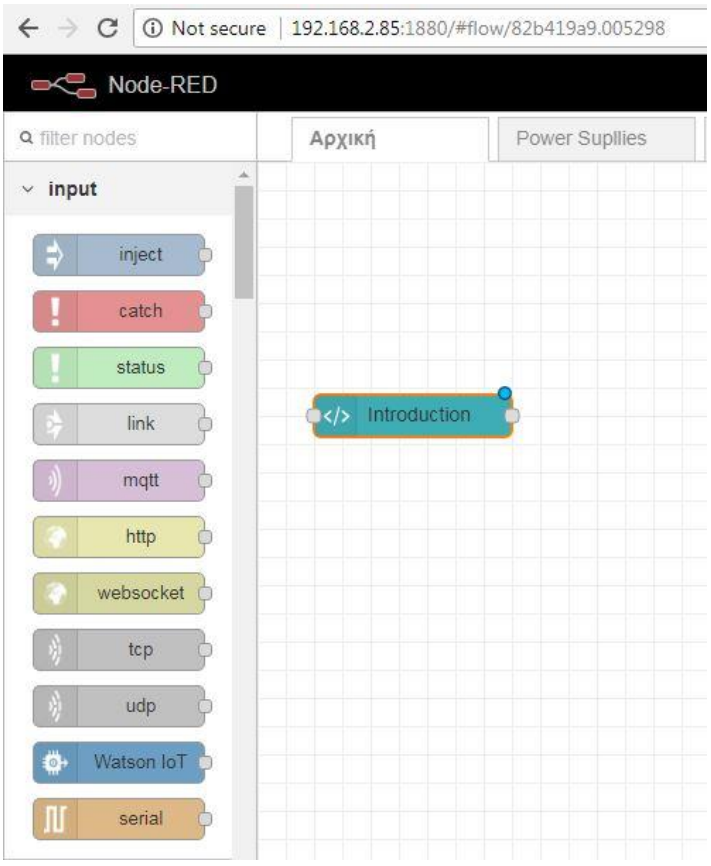
Στην πορεία της ενότητας, θα εμφανίζεται ο κώδικας σε γραφική μορφή και το αποτέλεσμα στο γραφικό περιβάλλον, που θα βλέπει ο χρήστης.

Μενού Εφαρμογής

Με τον κώδικα και τις ρυθμίσεις που έγιναν υλοποιήθηκε το παρακάτω μενού. Κάθε επιλογή του μενού θα παρουσιαστεί και θα αναλυθεί στη συνέχεια.


Αρχική - Σχετικά Κοινές
Διαχειριστικές Εντολές
NodeMCU (03)
Διάβασμα Ρευμάτων
NodeMCU (04)
Αισθητήρες Node 04
Δεξαμενή
Ρευματοδότες
Αισθητήρας Καπνού

Εισαγωγή - Σχετικά

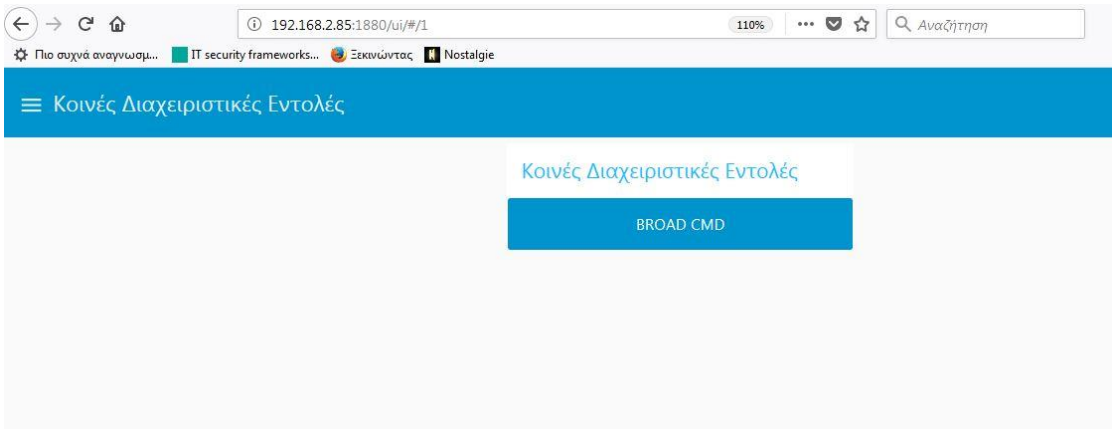


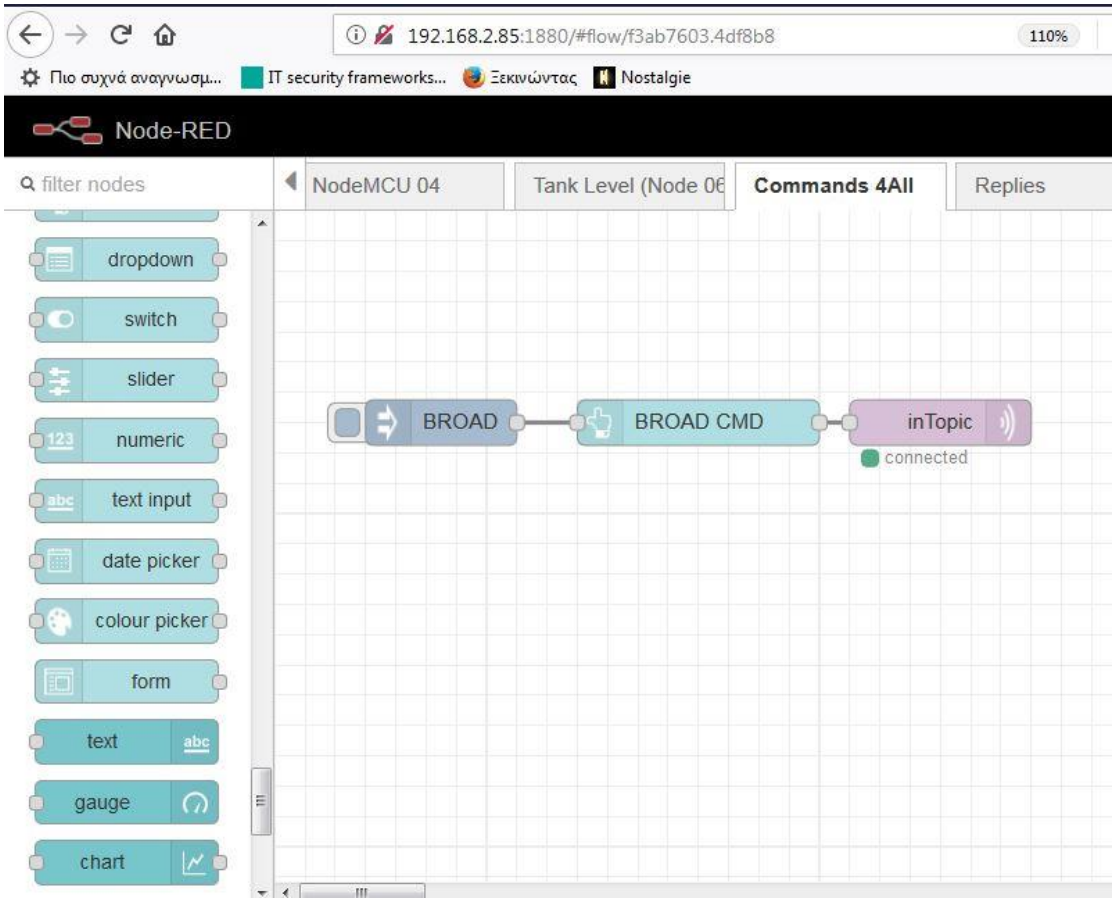
Λογική

Στο Module </>Template μπήκε ο κώδικας <img src="http://192.168.2.85/intro.jpg" alt="Thesis Introduction" style="align:center;width:800px;height:600px;">. Που δίνει εντολή στο module να φορτώσει την συγκεκριμένη εικόνα (intro.jpg).

Η λογική είναι να δημιουργηθεί μια εισαγωγική εικόνα κατά την εκκίνηση της εφαρμογής.

Κοινές Διαχειριστικές Εντολές

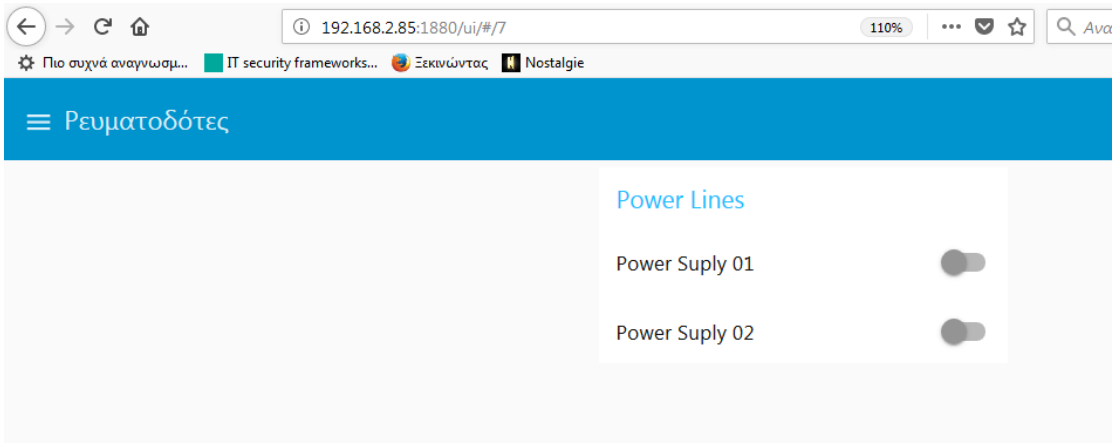


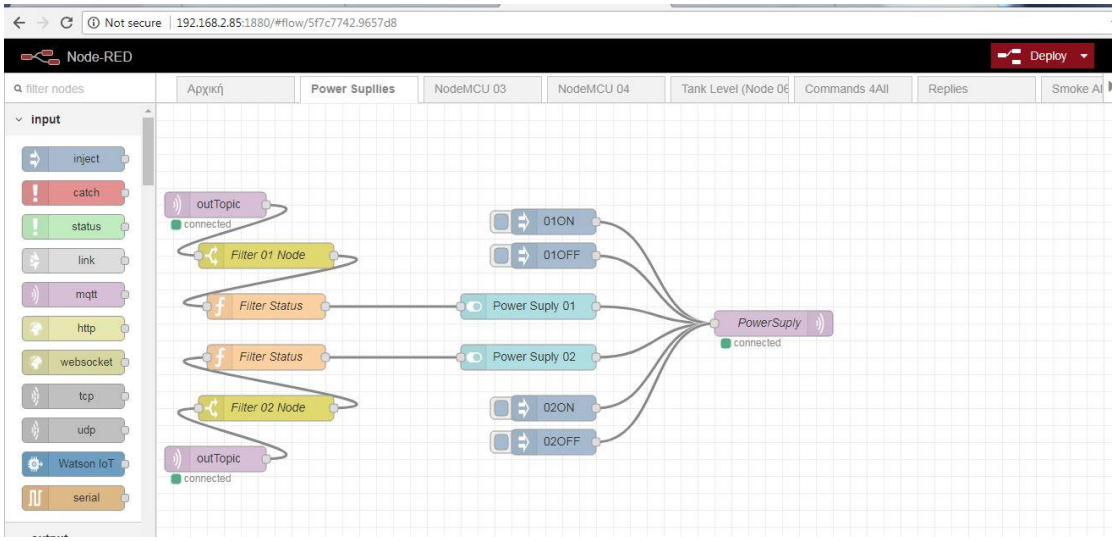


Λογική

Εδώ προωθείται η εντολή BROAD στο θέμα “inTopic”. Όλα τα Nodes ενημερώνονται για το θέμα αυτό και όταν βλέπουν την εντολή BROAD δημοσιεύουν στο θέμα “outTopic” την ύπαρξή τους. Αυτή η εντολή έχει σκοπό να ελέγχει τη σωστή λειτουργία όλων των Nodes που είναι συνδεδεμένα στο σύστημα.

Ρευματοδότες





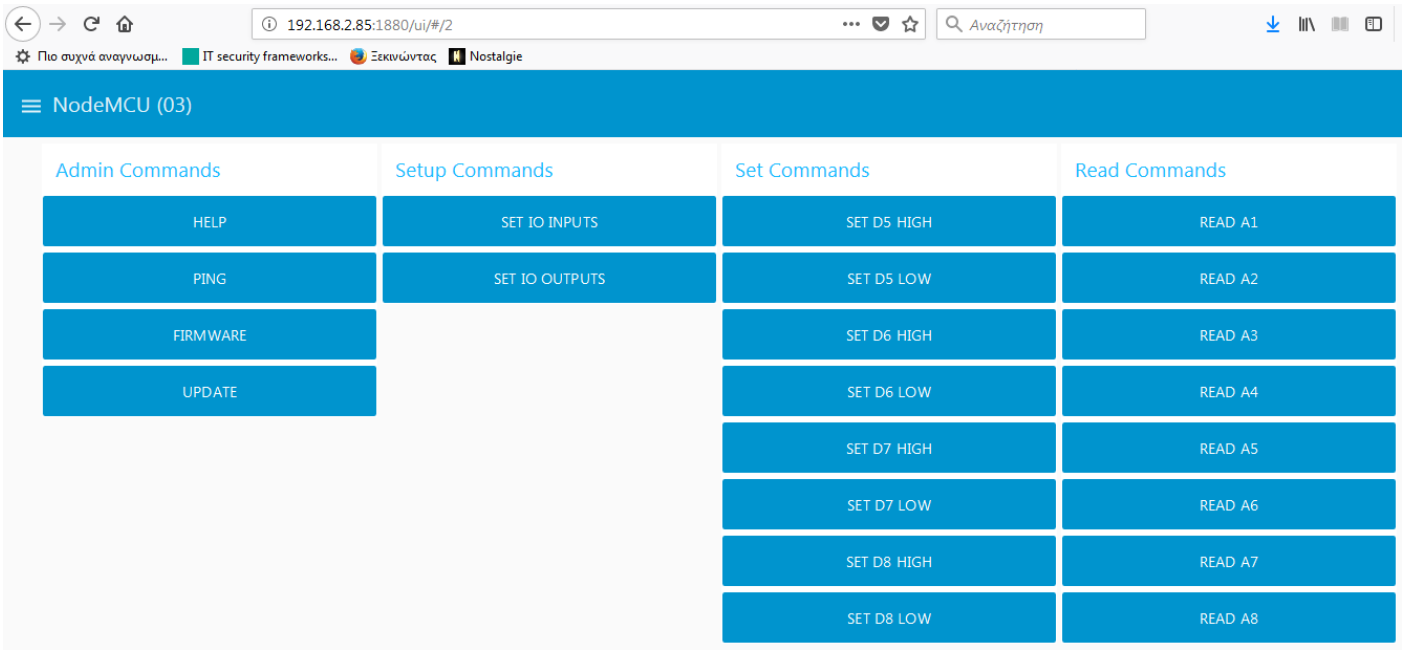
Λογική

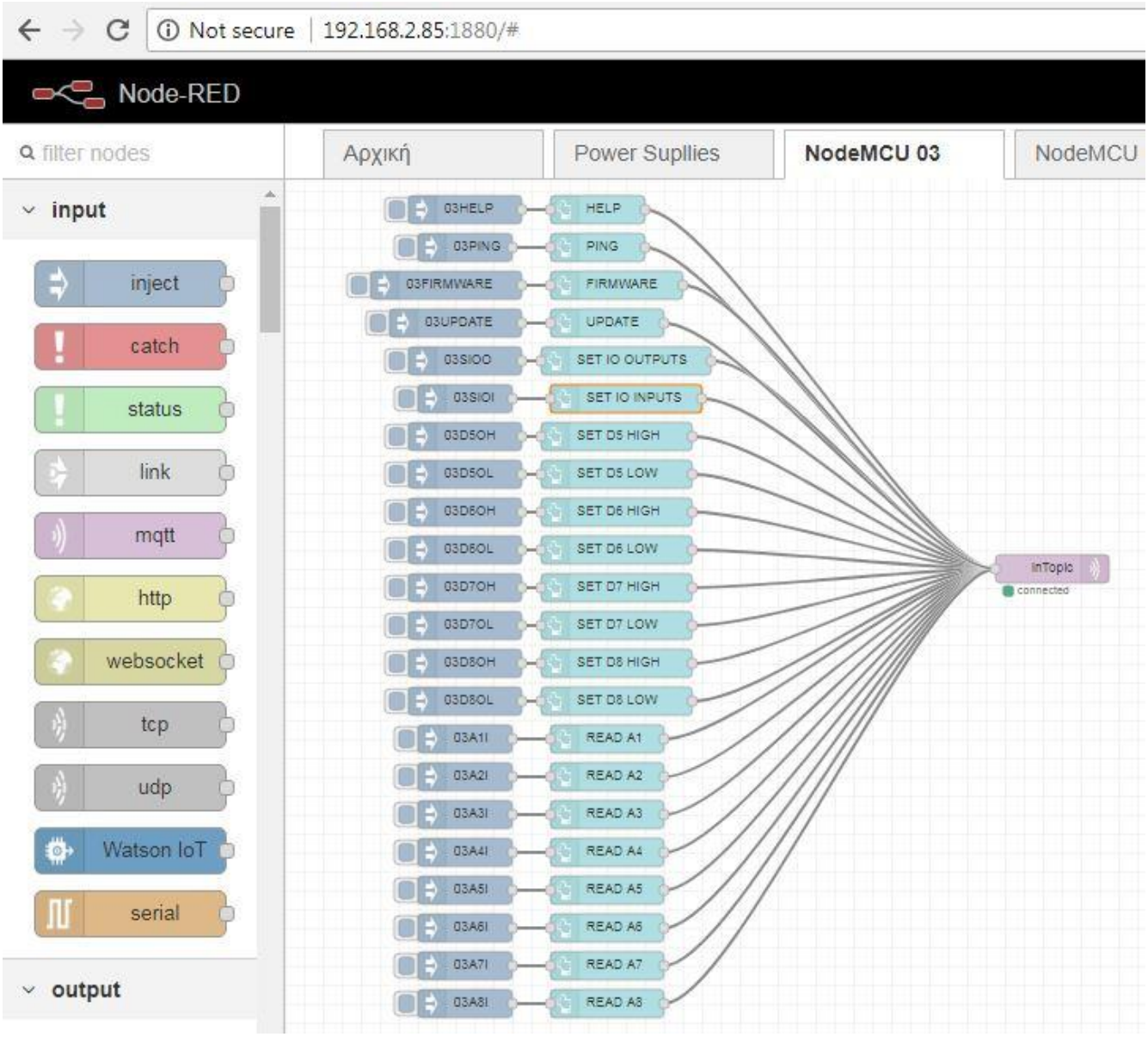
Μέσω του κώδικα αυτού δίνεται η δυνατότητα ενεργοποίησης/απενεργοποίησης των αυτόνομων ρευματοδοτών. Στη συγκεκριμένη περίπτωση, επειδή οι ρευματοδότες έχουν έναν διακόπτη που όταν πατηθεί αντιστρέφεται η λειτουργία του ρευματοδότη, δηλαδή αν ο ρευματοδότης είναι ενεργοποιημένος τότε απενεργοποιείται και το αντίστροφο. Και επειδή θέλουμε να ενημερώνεται και η κατάσταση του ρευματοδότη στο περιβάλλον της εφαρμογής μπήκε και ο ανάλογος κώδικας.

NodeMCU 03

Στη συγκεκριμένη σελίδα της εφαρμογής, δίνεται η δυνατότητα να στέλνονται εντολές στο Node 03. Οι εντολές ομαδοποιούνται σε κατηγορίες:

- Admin Commands – Διαχειριστικές εντολές
  - HELP – Ζητείται από το Node να δείξει το κατάλογο των εντολών του
  - PING – Γίνεται έλεγχος ανταπόκρισης του Node
  - FIRMWARE – Ζητείται από το Node να αναφέρει το Firmware που εκτελεί
  - UPDATE – Ζητείται από το Node να κάνει αναβάθμιση του Firmware του
- Setup Commands
  - SET IO INPUTS – Ενεργοποιεί τα pins D5 – D8 ως εισόδους
  - SET IO OUTPUTS – Ενεργοποιεί τα pins D5 – D8 ως εξόδους
- Set Commands
  - SET D5 HIGH – Στέλνει Υψηλό δυναμικό στην ακίδα D5
  - SET D5 LOW – Στέλνει Χαμηλό δυναμικό στην ακίδα D5
  - Επανάληψη για τις ακίδες D6,D7 και D8
- Read Commands
  - READ A1 – Διαβάζει την αναλογική τιμή της ακίδας A1
  - Επανάληψη για A2 έως A8





Λογική

Η λογική του κώδικα είναι να στέλνεται η κάθε εντολή στο θέμα "inTopic" το οποίο το διαβάζουν όλα τα Nodes και να ανταποκρίνεται αυτό που αναφέρεται η εντολή, δημοσιεύοντας στο θέμα "outTopic". Η λογική αυτή δίνει ετερογένεια στο σύστημα. Αφενός διότι το κάθε Node, ανάλογα την εφαρμογή του να έχει δικό του ρεπερτόριο εντολών, αφετέρου, τα Nodes μπορούν να αποτελούνται από διαφορετικό υλικό και να συνεργάζονται στο σύστημα.

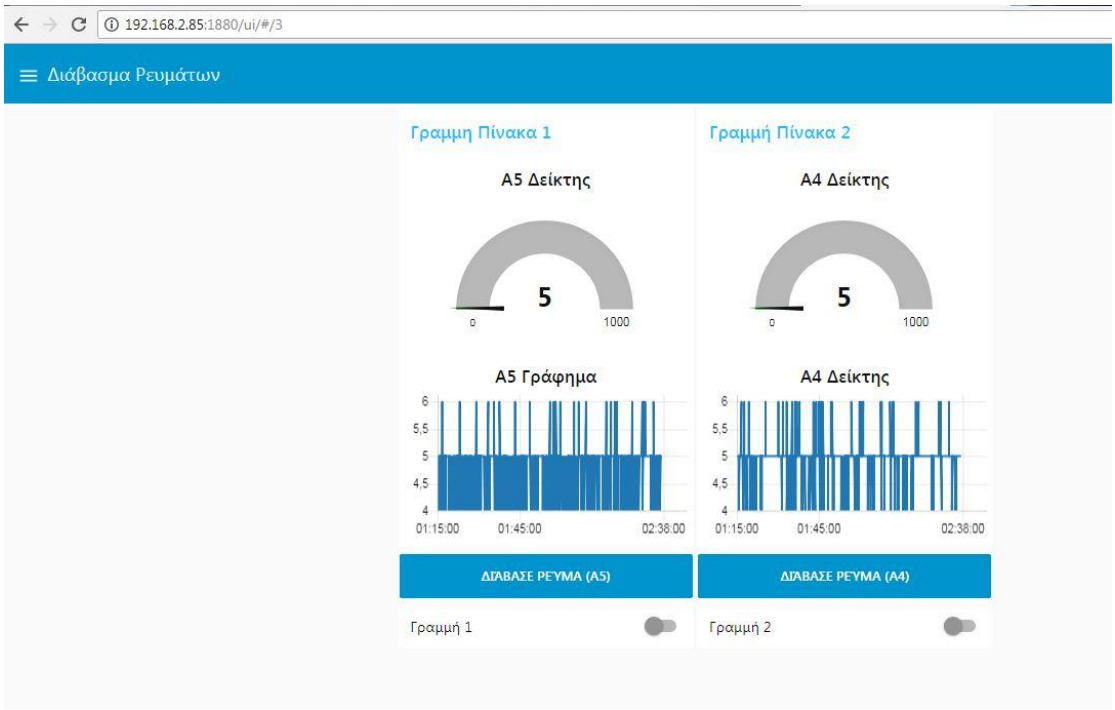
NodeMCU 04

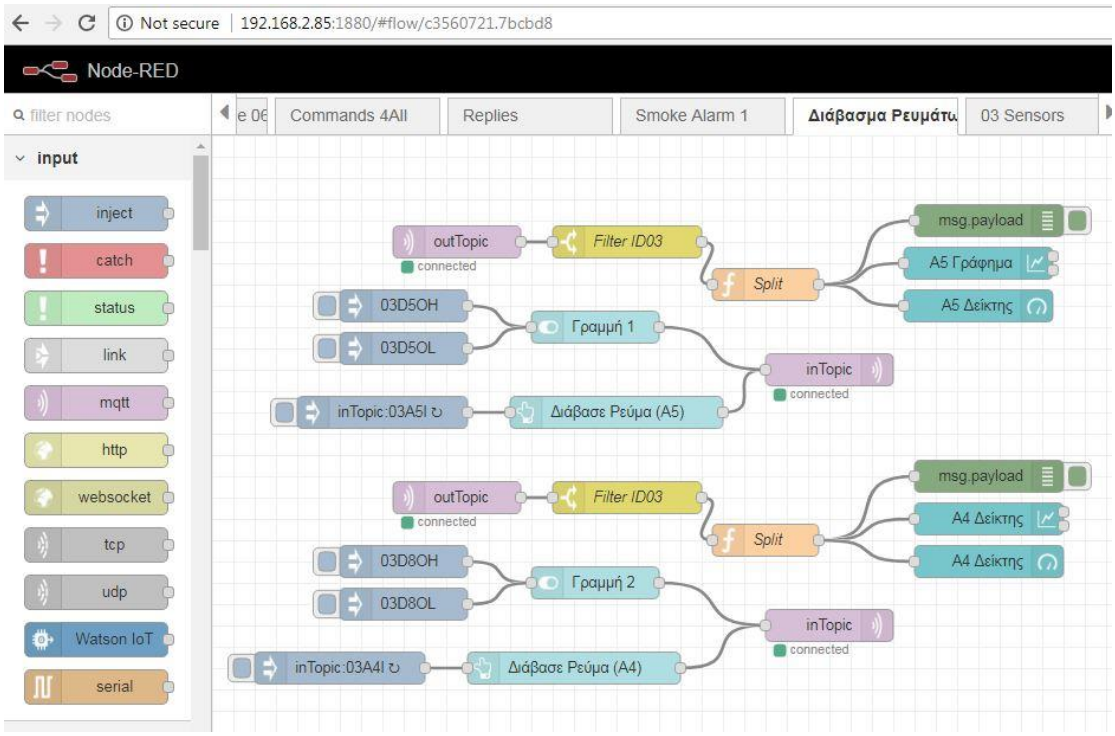
Το Node 04 είναι το ίδιο με το Node 03 οπότε ισχύουν ότι αναφέρθηκαν παραπάνω, απλά οι εντολές απευθύνονται στο Node 04, όπως θα φανεί παρακάτω χρησιμοποιήθηκε για άλλες εφαρμογές.

Διάβασμα Ρευμάτων – Ενεργοποίηση/Απενεργοποίηση Γραμμών Ηλεκτρικού πίνακα

Η εφαρμογή αυτή χρησιμοποιεί το Node 03 και έχει τοποθετηθεί σε ηλεκτρικό πίνακα (Εικόνες 57,58,59 & 60), με σκοπό να μετράται η κατανάλωση ρεύματος από κάποιες γραμμές ενός πίνακα, όπως για παράδειγμα, ηλεκτρικό μαγειρείο, πλυντήρια, θερμοσίφωνα, Air Condition ή και αντλίες θερμότητας. Διαβάζοντας ανά τακτά διαστήματα την ένταση του ρεύματος ενός κυκλώματος γίνεται να υπολογιστεί η κατανάλωση ρεύματος για κάποιο χρονικό διάστημα που επιθυμείται.

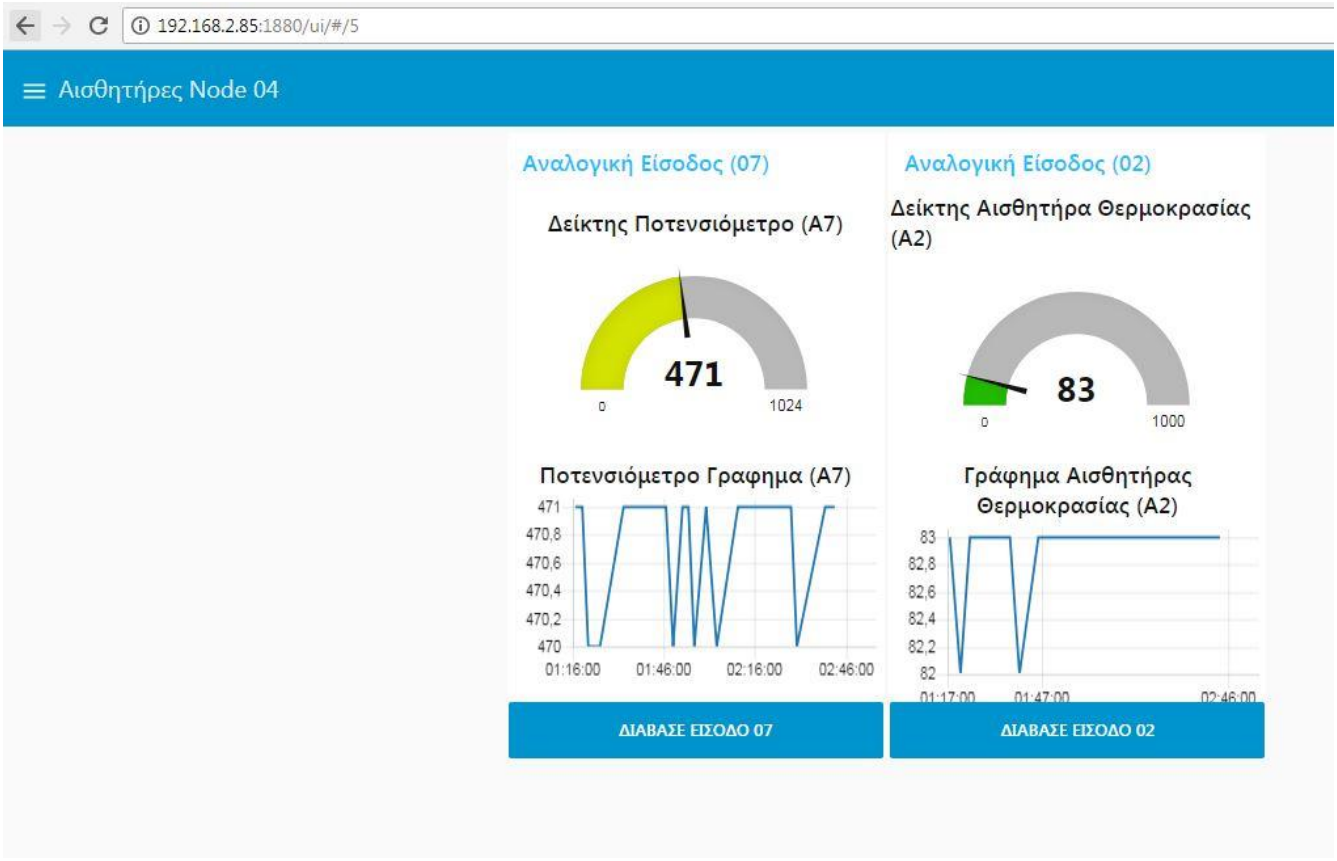
Επίσης στην συγκεκριμένη εφαρμογή, έχουν τοποθετηθεί στον πίνακα ηλεκτρονόμοι, έτσι ώστε να γίνεται απομακρυσμένα, ενεργοποίηση/απενεργοποίηση γραμμών του πίνακα, για παράδειγμα να ενεργοποιείται ο θερμοσίφωνας, να απενεργοποιείται η γραμμή μαγειρείου, να ενεργοποιείται μια γραμμή φωτισμού κ.α.

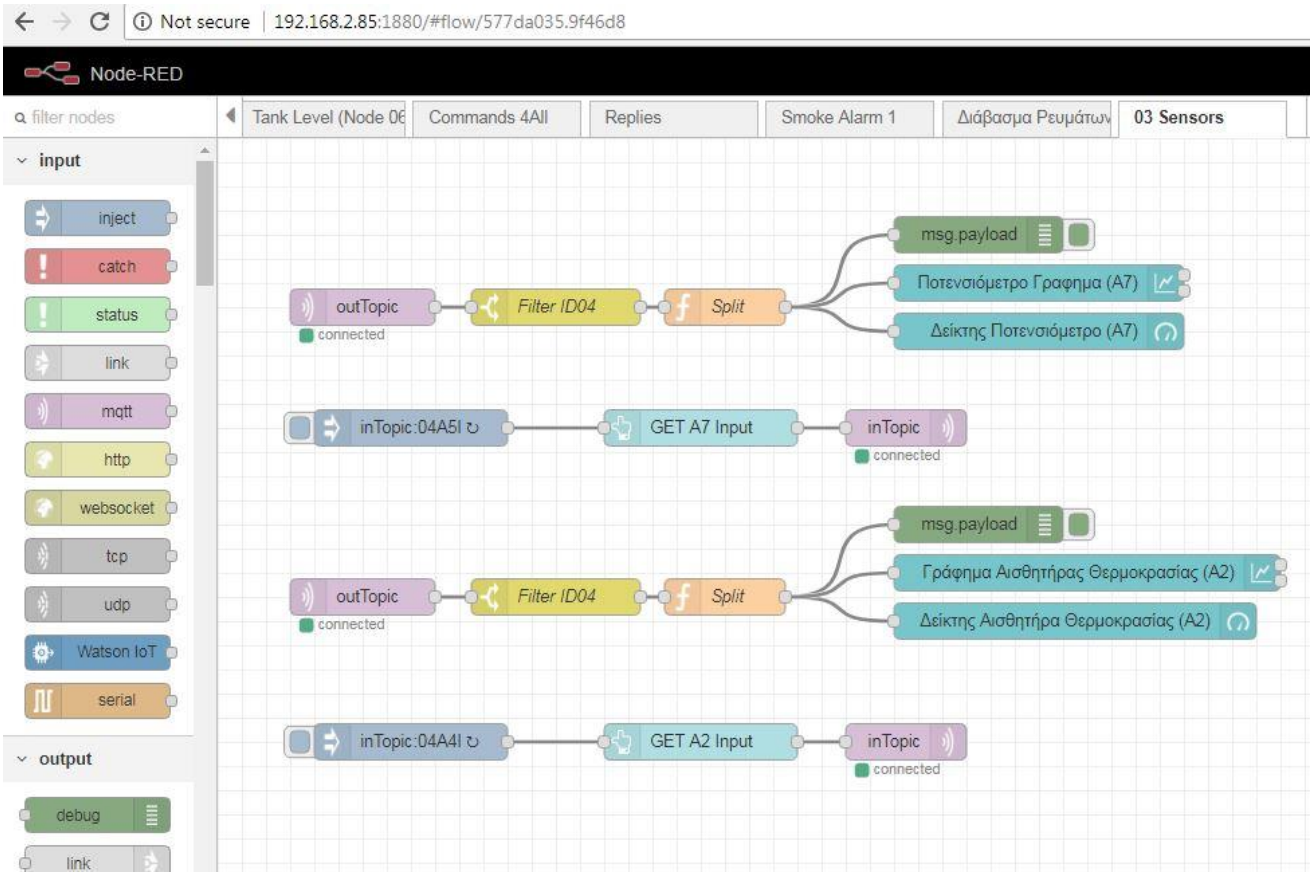




Αισθητήρες Node 04

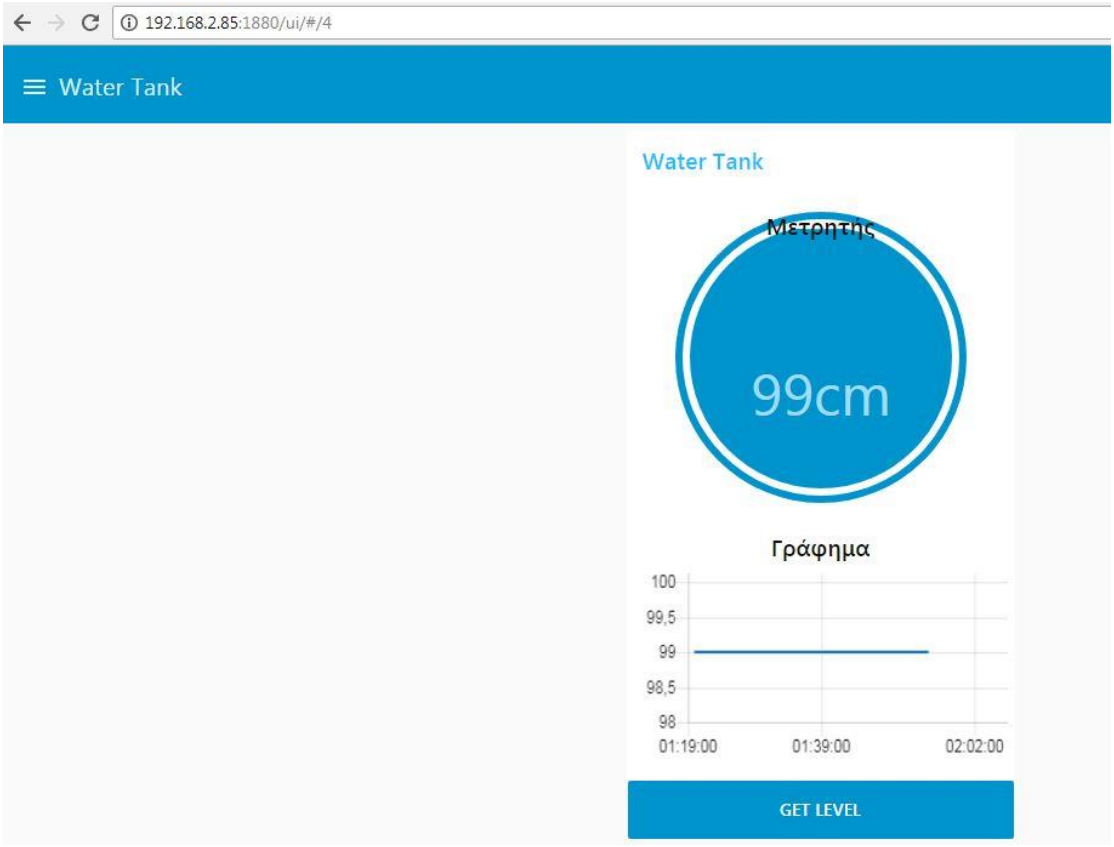
Το Node 04 είναι το ίδιο υλικό με το Node 03. Κατασκευάστηκε με το σκεπτικό να τοποθετηθεί σε έναν δεύτερο πίνακα. Αλλά για χάρη της παρουσίασης της εργασίας, τοποθετήθηκαν απλοί αισθητήρες, όπως ένα ποτενσιόμετρο και ένας αισθητήρας θερμοκρασίας LM35, στις αναλογικές θύρες του. Μπορούν να χρησιμοποιηθούν και δύο ηλεκτρονόμοι ώστε να υλοποιηθεί κάποιο σενάριο, για παράδειγμα, αν φτάσει κάποια θερμοκρασία, η κάποια τιμή το ποτενσιόμετρο να ενεργοποιηθεί μια σειρήνα ή να αναβοσβήνει μια λάμπα.

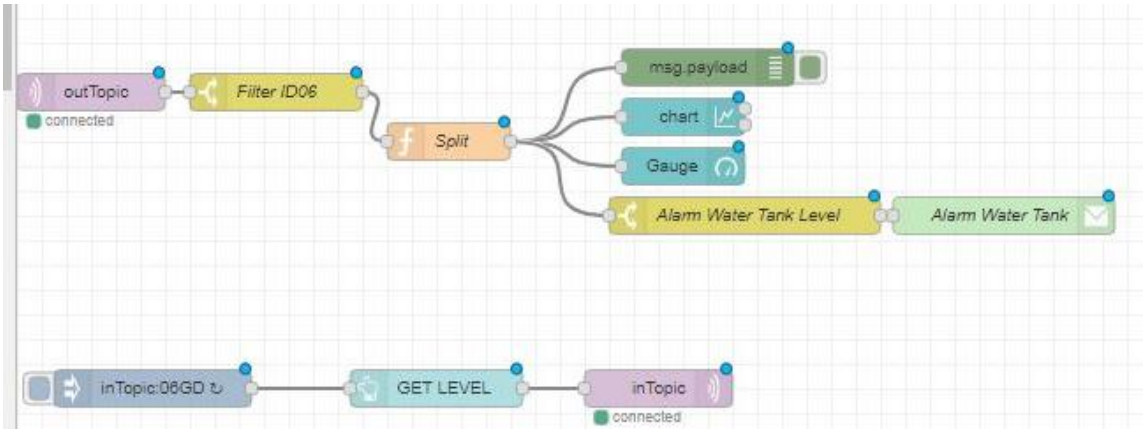

192.168.2.85:1880/ui/#/5
Αισθητήρες Node 04
Αναλογική Είσοδος (07)
Δείκτης Ποτενσιόμετρο (A7)
471
0
1024
Ποτενσιόμετρο Γραφημα (A7)
471
470,8
470,6
470,4
470,2
470
01:16:00
01:46:00
02:16:00
02:46:00
ΔΙΑΒΑΣΕ ΕΙΣΟΔΟ 07
Αναλογική Είσοδος (02)
Δείκτης Αισθητήρα Θερμοκρασίας (A2)
83
0
1000
Γράφημα Αισθητήρας Θερμοκρασίας (A2)
83
82,8
82,6
82,4
82,2
82
ΔΙΑΒΑΣΕ ΕΙΣΟΔΟ 02


Not secure | 192.168.2.85:1880/#flow/577da035.9f46d8
Node-RED
filter nodes
Tank Level (Node 06
Commands 4All
Replies
Smoke Alarm 1
Διάβασμα Ρευμάτων
03 Sensors
input
inject
catch
status
link
mqtt
http
websocket
tcp
udp
Watson IoT
serial
output
debug
link
outTopic
connected
Filter ID04
Split
msg.payload
Ποτενσιόμετρο Γραφημα (A7)
Δείκτης Ποτενσιόμετρο (A7)
inTopic:04A5I
GET A7 Input
inTopic
connected
msg.payload
Γράφημα Αισθητήρας Θερμοκρασίας (A2)
Δείκτης Αισθητήρα Θερμοκρασίας (A2)
inTopic:04A4I
GET A2 Input
inTopic
connected

Δεξαμενή

Η κατασκευή του αισθητήρα Δεξαμενής όπως αναφέρθηκε στην περιγραφή του υλικού είναι μια εφαρμογή που έχει σκοπό να ελέγχει τη στάθμη του Νερού μιας δεξαμενής Νερού. Στην ενότητα αυτή θα παρουσιαστεί η διεπαφή και ο κώδικας σε Node-RED.





Αισθητήρας Καπνού

3.1.4. Συλλογή και Αποθήκευση Δεδομένων - Data Acquisition and Storage

Στην ενότητα αυτή θα αναφερθούμε στη διαδικασία αποθήκευσης των δεδομένων σε βάση δεδομένων.

Επειδή ο MQTT Broker, και ειδικότερα ο Mosquitto, που έχουμε χρησιμοποιήσει δεν παρέχει ενσωματωμένη διαδικασία αποθήκευσης δεδομένων ήταν απαραίτητο να βρεθεί τρόπος αποθήκευσης των δεδομένων που αποστέλλονται από τους αισθητήρες να αποθηκεύονται κατά κάποιο τρόπο σε μια βάση δεδομένων. Ο λόγος για τον οποίο δεν γίνεται αυτό είναι σύμφωνα με τη γνώμη μας, ότι αφήνει την διαχείριση των δεδομένων που θα αποθηκεύονται στη βάση δεδομένων καθαρά στο χρήστη. Η εικασία αυτή γίνεται, βάση του γεγονότος ότι παρέχεται κώδικας σε γλώσσα προγραμματισμού C που επιτρέπει την αποθήκευση δεδομένων σε βάση δεδομένων. Οπότε με την κατάλληλη διαμόρφωση του κώδικα, είναι εφικτό στον χρήστη να αποθηκεύει, μόνιμα πλέον ότι πληροφορία επιθυμεί στη βάση δεδομένων.

Επίσης πρέπει να αναφερθεί, ότι το Node-RED που χρησιμοποιήθηκε ως εφαρμογή διατηρεί τα δεδομένα που δείχνει στα γραφήματά του, αλλά μετά από επανεκκίνηση η πληροφορία χάνεται, άρα κρίνεται απαραίτητο να αποθηκεύονται τα δεδομένα σε μια βάση δεδομένων, για να μπορούν να επεξεργάζονται στην πορεία. Αυτό συνεπάγεται ότι πρέπει να ληφθεί υπόψη στο σχεδιασμό της εφαρμογής και διαδικασία συντήρησης της βάσης δεδομένων, να διαγράφεται η περιττή πληροφορία και να λαμβάνεται αντίγραφο.

Στο Παράρτημα 7 βρίσκεται ο κώδικας που πάρθηκε από τα παραδείγματα του www.mosquitto.org, για την αποθήκευση δεδομένων σε βάση δεδομένων. Ο κώδικας αφού μεταγλωττιστεί, πρέπει να εκτελείται ως υπηρεσία στο Raspeberry Pi 3. Έτσι τα δεδομένα που αποστέλλονται στον MQTT Broker να αποθηκεύονται αυτόματα στη βάση δεδομένων.

Τέλος, πρέπει να ρυθμιστεί το Raspberry Pi 3, να εκτελεί το πρόγραμμα σε κάθε επανεκκίνηση. Για να το πετύχουμε αυτό χρησιμοποιήσαμε τις οδηγίες ενός οδηγού [Δ.17] που βρέθηκε στο διαδίκτυο. Ουσιαστικά τοποθετούμε το πρόγραμμα καταγραφής στο αρχείο /etc/rc.local. Και μάλιστα το ρυθμίζουμε να τρέξει στο παρασκήνιο βάζοντας το & στο τέλος της εγγραφής, έτσι ώστε να εκτελεστεί και συνεχίσει να εκτελεί τα άλλα αρχαία. Επίσης ανακατευθύνουμε την έξοδο του

αρχείου σε log αρχείο ώστε αν βγάλει κάποιο μήνυμα να καταγραφεί. Η εντολές που χρησιμοποιήθηκαν ήταν

Sudo nano /etc/rc.local

Και εισάγαμε το μονοπάτι του εκτελέσιμου αρχείου, το οποίο αποθηκεύει τα δεδομένα στη βάση.

3.2. Ρυθμίσεις Υπομονάδων

3.2.1. Setup Router Port Forwarding

Σύμφωνα με το (www.portforward.com) η προώθηση θύρας, είναι ένας τρόπος με τον οποίο κάνεις έναν οικιακό Η/Υ ή έναν σταθμό εργασίας από το επαγγελματικό δίκτυο, προσβάσιμο σε Η/Υ που είναι συνδεδεμένοι στο διαδίκτυο, παρά το γεγονός ότι βρίσκονται πίσω από έναν δρομολογητή (router). Χρησιμοποιείται ευρέως στα παιχνίδια, στις ρυθμίσεις καμερών παρακολούθησης, στην υπηρεσία voice over ip και στο κατέβασμα αρχείων. Από τη στιγμή που έχεις προωθήσει μια θύρα, τότε λέγεται πως έχεις μια ανοικτή θύρα.

Στην περίπτωση του συστήματος, κρίνεται απαραίτητο να γίνει η ρύθμιση αυτή ώστε να μπορείς να συνδεθεί κάποιος Η/Υ στον MQTT Broker, στο Node-red, ή ακόμα και στον web server που έχει εγκατασταθεί στο Raspberry Pi 3.

Για να μπορέσεις, τώρα κάποιος να συνδεθεί στις υπηρεσίες που αναφέραμε, και που τρέχουν στο Raspberry Pi, είναι απαραίτητο να, να γνωρίζει την πραγματική διεύθυνση IP, του δρομολογητή του τοπικού δικτύου μας. Δηλαδή με ποια IP μας βλέπουν οι servers του διαδικτύου.

Το να μάθεις ποια είναι η "πραγματική" του δρομολογητή σου μπορεί να γίνει με τουλάχιστον δύο τρόπους.

A. μέσω των εργαλείων διαχείρισης του δρομολογητή σου
B. χρησιμοποιώντας κάποιους servers που παρέχουν αυτήν την υπηρεσία.

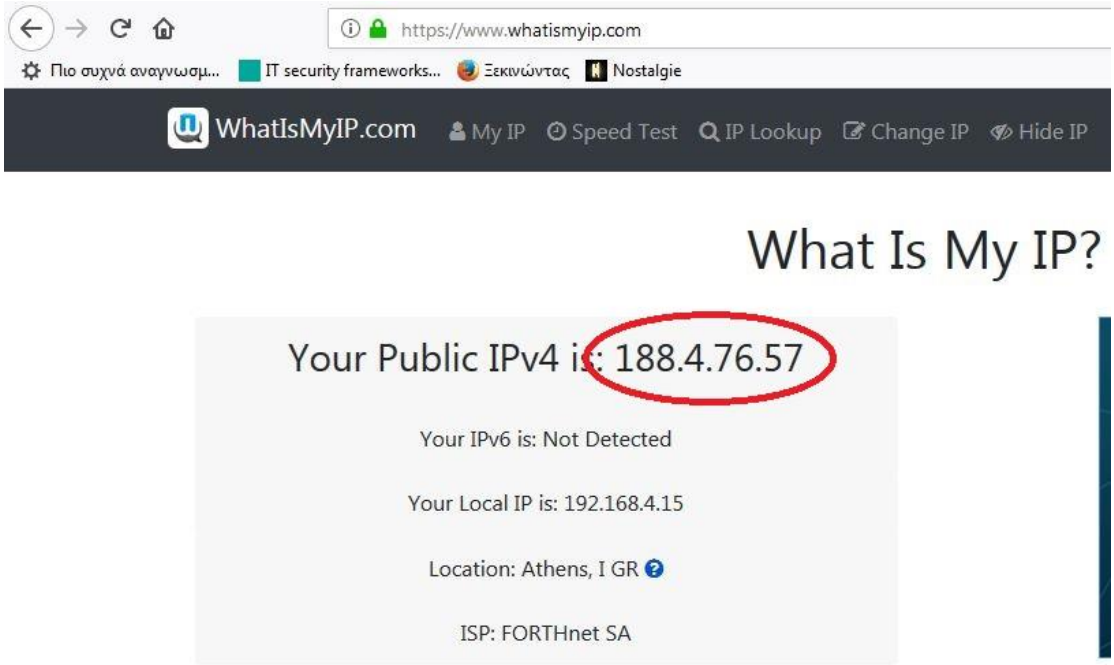
Α Τρόπος

Στην περίπτωση του οικιακού μας δικτύου, μέσα από τον δρομολογητή βλέπουμε ότι η πραγματική IP του δρομολογητή (για τη δεδομένη στιγμή) είναι **188.4.76.57** όπως φαίνεται στην εικόνα 36.

Εικόνα 30: Πραγματική IP Δρομολογητή

Β Τρόπος

Χρησιμοποιώντας για παράδειγμα τον εξυπηρετητή ***www.whatismyip.com*****.** Μόλις μπούμε στο δικτυακό τόπο παίρνουμε την εξής απάντηση. Οπότε διαπιστώνουμε ότι η πραγματική μας διεύθυνση IP είναι: **188.4.76.57**



Εικόνα 31: Πραγματική διεύθυνση IP μέσω του www.whatismyip.com

Στο σημείο αυτό αξίζει να παρατηρηθεί, το προφανές ότι η διεύθυνση IP είναι η ίδια και στις δύο περιπτώσεις, και ότι η διεύθυνση IP αφορά την ίδια χρονική στιγμή. Διότι, αν ο δρομολογητής, αποσυνδεθεί και ξανασυνδεθεί στο διαδίκτυο, τότε ο πάροχος του δίνει κάποια διαθέσιμη διεύθυνση IP που δεν θα είναι απαραίτητα η ίδια.

Αυτό οφείλεται στο είδος του συμβολαίου που έχουμε συνάψει με τον πάροχο μας. Οι απλές συνδέσεις, και πιο οικονομικές, λέγονται με δυναμική IP. Εάν επιθυμούμε να έχουμε συμβόλαιο με στατική IP τότε πρέπει να επιβαρυνθούμε τη διαφορά στην τιμή του συμβολαίου. Παρόλα αυτά, είναι εφικτό να αποφευχθεί η επιπλέον χρέωση εάν χρησιμοποιήσουμε μια υπηρεσία που ονομάζεται **Dynamic DNS** και για την οποία θα εξηγήσουμε στη συνέχεια.

3.2.2. Υπηρεσία Dynamic DNS

Συνεχίζοντας την προηγούμενη ενότητα, θα εξηγήσουμε την υπηρεσία Dynamic DNS. Όπως προαναφέρθηκε, σε μια δυναμική ευρυζωνική σύνδεση (DSL), η διεύθυνση IP του δρομολογητή μας δεν παραμένει σταθερή. Εάν γίνει αποσύνδεση και επανασύνδεση η διεύθυνση IP, πολύ πιθανά, να μεταβληθεί. Άρα από τη στιγμή που βρισκόμαστε εκτός τοπικού δικτύου δεν θα γνωρίζουμε τη πραγματική διεύθυνση IP του δρομολογητή μας, ώστε να συνδεθούμε στις υπηρεσίες που έχουμε ενεργοποιήσει.

Στο πρόβλημα αυτό έρχεται να δώσει λύση η υπηρεσία *Dynamic DNS* που την παρέχουν οι σύγχρονοι δρομολογητές ευρυζωνικών συνδέσεων (DSL). Για να χρησιμοποιηθεί η υπηρεσία αυτή δημιουργούμε έναν λογαριασμό σε συμβατούς εξυπηρετητές με την υπηρεσία αυτή (π.χ ddns.com, no-ip.com κ.α) αποκτώντας έτσι ένα όνομα τομέα, όπως για παράδειγμα **“garefalakis.no-ip.com”**. Ρυθμίζοντας τον δρομολογητή να συνδέεται στο λογαριασμό μας γίνεται ενημέρωση της αντιστοίχισης της τρέχουσας διεύθυνσης IP με το όνομα τομέα **“garefalakis.no-ip.com”**. Με τον τρόπο αυτό, δεν μας απασχολεί εάν αλλάζει η πραγματική διεύθυνση του δικτύου μας διότι, όποια και αν είναι θα αντιστοιχίζεται αυτόματα με το όνομα τομέα που έχουμε δημιουργήσει.

3.2.3. Χρήση Smartphone

Η ενότητα αυτή, ασχολείται με την περιγραφή της διαδικασίας που πρέπει να γίνει ώστε να μπορούμε να επικοινωνούμε, μέσω ενός κινητού τηλεφώνου Android Smartphone (κατά προτίμηση) με το σύστημα μας. Να λαμβάνουμε τιμές των

αισθητήρων, αλλά και να ενεργοποιούνται οι συνδεδεμένοι ενεργοποιητές ή αλλιώς ηλεκτρονόμοι.

Υπάρχουν οι εξής τρεις τρόποι:

1ος Τρόπος: Χρήση Φυλλομετρητή (Web Browser) του έξυπνου κινητού

2ος Τρόπος: Εγκατάσταση έτοιμης εφαρμογής MQTT Client που θα συνδέεται με τον MQTT Broker του συστήματος.

3ος Τρόπος: Δημιουργία δικής μας εφαρμογής που συνδέεται με τον MQTT Client που θα συνδέεται με τον MQTT Broker του συστήματος.

Κατά τον **$1^{ο}$ *Τρόπο,*** πληκτρολογούμε στον φυλλομετρητή (web browser) του έξυπνου κινητού μας της διεύθυνση IP του **Node-Red Dashboard**, στην οποία έχει σχεδιαστεί κατάλληλη διεπαφή (interface) με κουμπιά, πλαίσια κειμένου, γραφήματα και έχουμε τον πλήρη έλεγχο του συστήματός μας. Στέλνουμε εντολές ενεργοποίησης/απενεργοποίησης, διαβάζουμε τιμές αισθητηρίων μέσω γραφημάτων ή εργαλείων κ.α.

Κατά τον ***$2^{ο}$ Τρόπο,*** προχωρούμε σε εγκατάσταση ενός MQTT Client στο έξυπνο κινητό μας. Η εφαρμογή αυτή μπορεί να συνδεθεί στον MQTT Broker που έχει εγκατασταθεί στον Raspberry Pi 3. Στην περίπτωσή μας, για τις ανάγκες ελέγχου ορθής λειτουργίας του συστήματος χρησιμοποιήθηκε το **MQTT Dash**.

Εικόνα 32: MQTT Dash (IoT, Smart Home)

Η εφαρμογή δούλεψε πολύ ικανοποιητικά, ρυθμισμένη σε τοπικό δίκτυο δηλαδή ορίζοντας τη τοπική διεύθυνση IP του Raspberry Pi 3, στο οποίο έχει εγκατασταθεί η υπηρεσία του MQTT Broker, είτε ρυθμίζοντας την μέσω της πραγματικής διεύθυνσης IP, αφού είχε ενεργοποιηθεί το port forwarding στο δρομολογητή του οικιακού δικτύου.

Εικόνα 33: Ρύθμιση MQTT Dash σε Android Κινητό

Κατά τον **3^{ο} *Τρόπο,*** θα μπορούσε να αναπτυχθεί σχετική εφαρμογή στο Android, που να επικοινωνεί με το σύστημα και να γίνεται ανταλλαγή μηνυμάτων. Όμως, λόγω της μεγάλης έκτασης της εργασίας, σε διάφορους τομείς, προτιμήθηκε στη φάση συγγραφής της εργασίας να μην προχωρήσουμε στο κομμάτι ανάπτυξης του κώδικα αυτού, αλλά να παρουσιαστεί ως κομμάτι μελλοντικής εργασίας και επέκτασης της εργασίας.

Παρόλα αυτά, σε αναζήτηση που πραγματοποιήθηκε στο διαδίκτυο, υπάρχουν διαθέσιμα παραδείγματα κώδικα, ώστε να μπορεί να υλοποιηθεί μια αντίστοιχη εφαρμογή. Για παράδειγμα στον δικτυακό τόπο (https://internetofhomethings.com/homethings/?p=1317) στο άρθρο με τίτλο «*MQTT For App Inventor»* υπάρχει αναλυτικός οδηγός, για το πώς μπορεί να δημιουργηθεί εφαρμογή για Android που να συνδέεται σε MQTT Broker, χρησιμοποιώντας το App Inventor.

3.2.4. Χρήση Φωνητικών Εντολών στο σύστημα

Αυτή η ενότητα περιγράφει τη διαδικασία με την οποία, επετεύχθη η εφαρμογή φωνητικών εντολών στο σύστημα μας. Για τη διαδικασία αυτή χρησιμοποιήθηκε ένας οδηγός στο Youtube του Richard Wenner (https://www.youtube.com/watch?v=Y4Nka-jbHCY&feature=youtu.be, Τελευταία πρόσβαση 10/5/2018).

Για την υλοποίηση, χρησιμοποιήθηκαν εφαρμογές τρίτων. Οι εφαρμογές αυτές, είναι αρκετά δύσκολο να γίνουν σε επίπεδο προγραμματισμού και απαιτούν μεγάλες δεξιότητες προγραμματισμού, που ξεπερνάνε τα πλαίσια πτυχιακής εργασίας και θα μπορούσαν να είναι αυτόνομες εργασίες για μια πτυχιακή εργασία ή Διπλωματική εργασία Μεταπτυχιακού Προγράμματος, λόγω των μεγάλων απαιτήσεων. Παρόλα αυτά, εφαρμόστηκε στην εργασία αυτή, ώστε να φανεί σε ποιο επίπεδο μπορεί να φτάσει και να δοθεί κίνητρο μελλοντικής εργασίας και επέκτασης της παρούσας εργασίας.

Θα εξηγήσουμε την παρακάτω εικόνα 39, ώστε να γίνει πιο αντιληπτή η διαδικασία εφαρμογής των φωνητικών εντολών και προώθησής τους στο σύστημά μας.

Εικόνα 34: Υλοποίηση Φωνητικών Εντολών στο Σύστημα

Η ροή έχει ως εξής:

1. Ο χρήστης ενεργοποιεί τις φωνητική ακρόαση
2. Ο χρήστης δίνει την φωνητική εντολή
3. Η φωνητική εντολή συσχετίζεται με την εντολή που πρέπει να σταλεί στον MQTT Broker
4. Τροφοδοτείται η γραπτή εντολή στον MQTT Client
5. Αποστέλλεται μέσω ασύρματου δικτύου στον MQTT Broker που βρίσκεται εγκατεστημένο στο Raspberry Pi 3
6. Εκτέλεση της Εντολής
7. Ανατροφοδότηση εντολής από τον MQTT Broker
8. Εκφώνηση εντολής μέσω της φωνητικής μηχανής (Voice Engine) του Android

3.2.5. Οδηγός Υλοποίησης με εφαρμογές

Για να γίνει η υλοποίηση της ροής που αναφέραμε πρέπει να γίνει εγκατάσταση κάποιων εφαρμογών στο Android Κινητό. Οι εφαρμογές αυτές είναι Tasker, Autovoice και MQTT plugin.

Εικόνα 35: Tasker

Εικόνα 36: Autovoice

Εικόνα 37: MQTT plugin

Εγκαθιστούμε τις τρεις εφαρμογές. Η βασική εφαρμογή, είναι η Tasker και χωρίς αυτή δεν μπορεί να ολοκληρωθεί η διαδικασία. Αυτή η εφαρμογή έχει ένα μικρό κόστος (2,99€), ενώ οι άλλες εφαρμογές διαθέτουν δωρεάν εκδόσεις. Με την Tasker δίνουμε μια σειρά εντολών που εκτελούνται από τη συσκευής μας. Για παράδειγμα να ενεργοποιηθεί κάποια εφαρμογή, να σταλεί ένα μήνυμα κάποια συγκεκριμένη στιγμή ή μετά από κάποιο συμβάν που έχουμε ορίσει και πολλά άλλα.

Το Autovoice και το MQTT Plugin λειτουργούν ως πρόσθετα που συνεργάζονται με την Tasker. Αναβαθμίζουμε την εφαρμογή Google, του κινητού μας. Αυτό της δίνει μια δυνατότητα, μόλις εκφωνήσουμε “OK Google” να ενεργοποιεί την φωνητική ακρόαση, αυτόματα. Μόλις εκφωνήσουμε μια φωνητική εντολή που την αναγνωρίσει η εφαρμογή Autovoice, διακόπτεται η συνηθισμένη διαδικασία της Google εφαρμογής (για παράδειγμα να πραγματοποιηθεί αναζήτηση στη Google με την πρόταση που εκφωνήσαμε) και μετατρέπεται σε μορφή κειμένου. Το κείμενο αυτό διαβιβάζεται στο πρόσθετο MQTT και αυτό με τη σειρά του διαβιβάζει στο MQTT Broker, όπου και εκτελείται.

Άρα ακολουθώντας τη διαδικασία, αυτή μπορούν να συνδέσουμε μια φωνητική εντολή με μια σειρά εντολών που θέλουμε να εκτελούνται από το σύστημά μας (π.χ να ενεργοποιηθούν τα φώτα, να γίνει κάποιος έλεγχος κ.α).

3.3. Εγκατάσταση GSM MODEM

Για τη διεύρυνση του συστήματος, εγκαταστάθηκε και ένα GSM MODEM Huawei (k3770) σε θύρα USB του Raspberry Pi 3, διότι αυτό υπήρχε στη διάθεσή μας. Η εγκατάσταση και ρύθμιση της συγκεκριμένη συσκευής έδωσε ευελιξία στο σύστημα που υλοποιήθηκε. Αφενός έδωσε δυνατότητα, μέσω γραπτών μηνυμάτων (SMS), να στέλνονται ειδοποιήσεις σε περίπτωση ανάγκης ή σε έκτακτες περιπτώσεις, αλλά και η δυνατότητα ελέγχου ενεργοποιητών ή ρευματοδοτών. Με τη δυνατότητα αυτή μπορούν να ενεργοποιούνται συσκευές ή ηλεκτρικές γραμμές ενός ηλεκτρικού

πίνακα, απομακρυσμένα μέσω γραπτών μηνυμάτων SMS. Η δυνατότητα αυτή μπορεί να εξελιχτεί περισσότερο, ώστε να παρέχει το ίδιο το GSM MODEM πρόσβαση στο διαδίκτυο, στο RPI, και έτσι να λειτουργεί το σύστημα με πρόσβαση στο διαδίκτυο μέσω GPRS σύνδεσης.

Για να μπορέσουμε να συνδέσουμε το GSM MODEM στο RPI, χρειάστηκε να εγκατασταθεί στο RPI το πρόγραμμα **minicom** το οποίο δίνει τη δυνατότητα επικοινωνίας με τη σειριακές θύρες του RPI προσομοιώνοντας το τερματικό, ανάλογο πρόγραμμα στα window χρησιμοποιήθηκε το **putty**. Χρησιμοποιώντας το minicom έγινε και αποσφαλμάτωση κατά την επικοινωνία του Node-RED με το MODEM όπου έδειχνε τι ακριβώς χαρακτήρες ανταλλάσσονταν μεταξύ των συσκευών.

Σε πρώτη φάση, αξιοποιήθηκε η δυνατότητα του GSM MODEM να στέλνει και λαμβάνει γραπτά μηνύματα (SMS). Για να επιτευχθεί αυτό χρησιμοποιήθηκαν οι AT εντολές του MODEM [M.11].

Οι βασικές εντολές που χρησιμοποιούνται είναι οι παρακάτω:

AT+CMGF=1 : Ρυθμίζεται το GSM MODEM να στέλνει μηνύματα απλού κειμένου

Για αποστολή μηνύματος:

AT+CMGS=”+3069XXXXXXXX” (CR)

>Test sms (CR)

> ctrl + Z

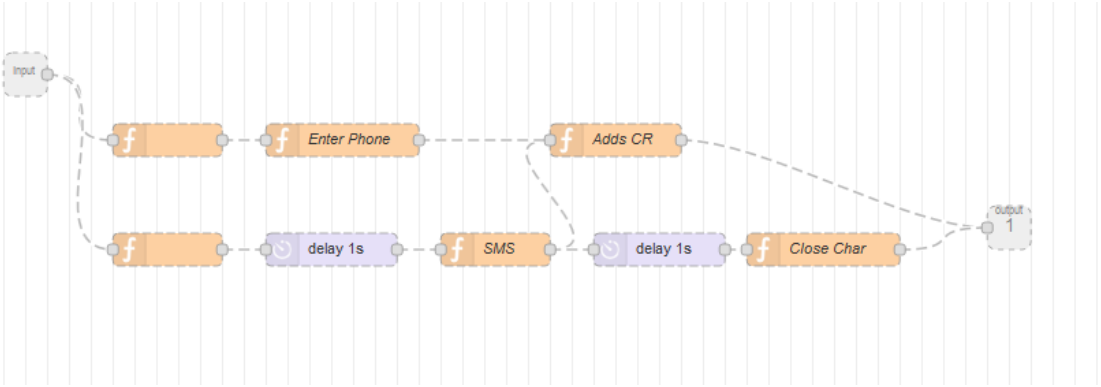
AT+CNMI=1,3,0,0,0 : Ρυθμίζει το MODEM να στέλνει ειδοποίηση όταν λαμβάνει μήνυμα

Για ανάγνωση μηνύματος

AT+CMGR=1 : Διαβάζει το μήνυμα με δείκτη 1

AT+CMGD=1 : Σβήνει το μήνυμα με δείκτη 1

Χρησιμοποιώντας τις βασικές εντολές δημιουργήθηκε στο Node-RED μια ροή ώστε να στέλνονται μηνύματα μέσα από την εφαρμογή μας και η ροή φαίνεται παρακάτω.



Ο σχεδιασμός των μηνυμάτων είναι αρκετά απλός. Ο χρήστης στέλνει όποια εντολή, ενεργοποίησης επιθυμεί προσθέτοντας το πρόθεμα CMD μπροστά στην εντολή, τότε η ροή στο Node-RED διαμορφώνεται και αποστέλλεται στο MQTT Broker. Για παράδειγμα για να ενεργοποιηθεί ο ρευματοδότης (1) πρέπει να σταλεί στο MQTT Broker η εντολή 01ON. Αν θελήσουμε να στείλουμε την εντολή μέσω SMS τότε στέλνεται το γραπτό μήνυμα CMD:01ON, και το σύστημα ενεργοποιεί τον ρευματοδότη. Όταν ενεργοποιηθεί ο ρευματοδότης τότε στον MQTT Broker αποστέλλεται από το Node η επιβεβαίωση ότι δόθηκε η εντολή ενεργοποίησης, δηλαδή 01:Relay ON.

Στο σημείο αυτό, κρίνεται σκόπιμο να επισημανθεί, ότι το Node απαντάει ότι έχει ενεργοποιηθεί ο ρευματοδότης, όμως δεν είναι η σωστή ανατροφοδότηση, διότι δεν ξέρουμε ότι όντως η γραμμή διαρρέεται από ρεύμα. Για να επιτευχθεί αυτό θα έπρεπε να ενσωματωθεί κατάλληλη διάταξη που ανιχνεύει τη ροή ρεύματος.

Τέλος, τα μηνύματα ανατροφοδότησης που στέλνονται μέσω SMS, είναι συγκεκριμένα και πρέπει να καταχωρηθούν στη ροή του Node RED. Επίσης δίνεται η δυνατότητα, να ενεργοποιείται και να απενεργοποιείται η ανατροφοδότηση μέσω SMS εάν σταλούν τα SMS 1) CMD:SMS ON και 2) CMD:SMS OFF. Ο λόγος που δίνεται η δυνατότητα αυτή είναι ότι εάν βρισκόμαστε στην εμβέλεια του ασύρματου δικτύου δεν υπάρχει λόγος, να λαμβάνουμε ανατροφοδοτήσεις μέσω SMS τα οποία έχουν σχετική χρέωση.

4. Κεφάλαιο 4ο

4.1. Η ασφάλεια στο Διαδίκτυο των πραγμάτων

Η ψηφιακή μας ασφάλεια και η παγκόσμια ασφάλεια προέχει κάθε μέρα, είτε πραγματοποιείται τραπεζική μεταφορά, είτε διαδικτυακή αγορά ή ακόμα και αν αποκτάται πρόσβαση σε προσωπικά έγγραφα μέσω του διαδικτύου. Η ιδέα του Διαδικτύου των Πραγμάτων είναι να συνδεθούν αντικείμενα μεταξύ τους, προκειμένου να προσφερθεί περισσότερη άνεση ή να βελτιωθεί η εργασία και η καθημερινότητα του ατόμου με οποιονδήποτε τρόπο. Η διασύνδεση αντικειμένων, όπως τα αυτοκίνητα, τα σπίτια, τα μηχανήματα εκθέτει επίσης πολλά ευαίσθητα δεδομένα. Για παράδειγμα, τη θέση των ανθρώπων ενός νοικοκυριού. Ίσως είναι καλό, να γνωρίζετε που βρίσκονται τα μέλη της οικογένειάς σας, αλλά δεν είναι ιδανικό να μοιραστείτε αυτές τις πληροφορίες με έναν διαρρήκτη. Υπάρχουν διαφορετικά είδη δεδομένων, τα οποία δεν προορίζονται για το κοινό και πρέπει να προστατεύονται από τους πυλώνες της ασφάλειας των πληροφοριών: *εμπιστευτικότητα, ακεραιότητα* και *διαθεσιμότητα*. Υπάρχει επίσης μια άλλη άποψη σε αυτό, όταν πραγματικές μηχανές ή πράγματα εκτίθενται, στην περίπτωση αυτή κάποιος εισβολέας μπορεί να κάνει ζημιά σε πραγματικούς ανθρώπους. Όπως το να ενεργοποιηθούν τα φρένα ενός αυτοκινήτου, απομακρυσμένα, ενώ οδηγείτε ή να σαμποταριστούν οι κινητήρες αυτοκινήτων. Ακόμη και αν δεν εμπλέκεται πραγματικό πρόσωπο, όπως όλοι γνωρίζουμε, η δημοσίευση ευαίσθητων δεδομένων μπορεί να βλάψει τη φήμη εταιρειών για μεγάλο χρονικό διάστημα. Έτσι, με όλο και περισσότερα δεδομένα να συλλέγονται κάθε μέρα και περισσότερες συσκευές να κυκλοφορούν στη ζωή μας, η ασφάλεια είναι ένα θέμα που είναι κρίσιμο και σίγουρα πιο σημαντικό από ποτέ.

Αρκετά περιστατικά στο Βιομηχανικό ΔτΠ [22] έχουν καταγραφεί, κατά τα οποία επιτεύχθηκαν με επιτυχία. Όπως για παράδειγμα το σκουλήκι Slammer, το οποίο επιτέθηκε σε σύστημα παρακολούθησης πυρηνικού σταθμού στις ΗΠΑ και το Stuxnet που προκάλεσε βλάβες σε συσκευές φυγοκέντρισης σε πυρηνικό σταθμό στο Ιράν, δείχνουν πόσο δυσμενείς συνέπειες μπορούν να υπάρξουν κατόπιν επιθέσεων σε βιομηχανικό ΔτΠ. Πρόσφατες έρευνες αναφέρουν, πως hackers έχουν χρησιμοποιήσει διακυβευμένες συσκευές ΔτΠ ώστε να υλοποιήσουν μεγάλων εκτάσεων επιθέσεις τύπου *Distributed Denial of Service DDoS όπως ο ιός Mirai [25]*. Ενώ επίσης έχει παρατηρηθεί ότι διακυβευμένες συσκευές

χρησιμοποιούνται ως Proxy μεσολαβητές προκειμένου να αποκρυφτεί η πραγματική ταυτότητα των επιτιθέμενων.

Στην περίπτωση της παρούσας εργασίας, το ζήτημα της ασφάλειας κρίνεται πάρα πολύ σημαντικό. Αφού γίνεται συζήτηση απομακρυσμένης ενεργοποίησης ρευματοδοτών και γραμμών ενός ηλεκτρικού πίνακα, γίνεται πολύ εύκολα αντιληπτό ότι δεν θα θέλαμε κάποιος άγνωστος να ενεργοποιήσει ή απενεργοποιήσει κάποιες συσκευές θέρμανσης ή ψύξης, ενός θερμοσίφωνα ενώ δεν είναι κατάλληλες συνθήκες. Επίσης επειδή γίνεται καταγραφή δεδομένων αισθητήρων σε βάση δεδομένων, να γεμίσει κάποιος τη βάση δεδομένων με σκουπίδια και άχρηστη πληροφορία.

4.2. Προκλήσεις για την ασφάλεια στο Διαδίκτυο

Δεν υπάρχει αμφιβολία για το κατά πόσο η ασφάλεια πρέπει να εφαρμοστεί ή όχι, όμως το Διαδίκτυο των πραγμάτων φέρνει νέες προκλήσεις στο τραπέζι. Παρόλο που η ασφάλεια είναι μια διαπραγμάτευση μεταξύ εξαιρετικά ασφαλούς και μεγάλης χρηστικότητας, γίνεται ακόμη πιο ενδιαφέρουσα στο Διαδίκτυο των πραγμάτων. Οι συσκευές ΔτΠ έχουν περιορισμούς συχνά σε θέματα υπολογιστικής ισχύος και χωρητικότητας μνήμης. Επομένως, είναι πρόκληση να χρησιμοποιηθούν αλγόριθμοι κρυπτογράφησης, οι οποίοι συχνά δεσμεύουν μεγάλο μέγεθος πόρων. Άλλη πρόκληση, είναι η ενημέρωση των συσκευών που είναι εγκατεστημένες στο χώρο. Συνήθως υπάρχει μια, μη αξιόπιστη σύνδεση και όταν απαιτούνται άμεσες ενημερώσεις, είναι δύσκολο να πραγματοποιηθούν ταυτόχρονα σε όλες τις συσκευές. Παρ 'όλα αυτά, η ασφάλεια θα πρέπει πάντα να υπάρχει εξ’ αρχής, ειδικά όταν σχεδιάζονται και αναπτύσσονται εφαρμογές Διαδικτύου των Πραγμάτων.

4.3. Ασφάλεια στο MQTT

Αφού παρουσιάστηκε ο ρόλος της ασφάλειας στο Διαδίκτυο των Πραγμάτων, ας γίνει μια προσπάθεια να αντιληφθούμε καταλάβουμε πώς το MQTT χειρίζεται την ασφάλεια.

Η ασφάλεια στο MQTT χωρίζεται σε πολλαπλά επίπεδα. Κάθε επίπεδο αποτρέπει διαφορετικά είδη επιθέσεων. Ο κύριος στόχος του πρωτοκόλλου, είναι να παράσχει ένα πραγματικά ελαφρύ και εύκολο στη χρήση πρωτόκολλο επικοινωνίας για το Διαδίκτυο των Πραγμάτων (ΔτΠ). Αυτός είναι ο λόγος για τον οποίο, στο ίδιο το πρωτόκολλο, υπάρχουν περιορισμένοι μηχανισμοί ασφαλείας. Συνήθως στις

εφαρμογές χρησιμοποιούνται πρότυπα ασφαλείας τελευταίας τεχνολογίας, όπως το SSL / TLS για την ασφάλεια στο επίπεδο μεταφοράς. Η γενική ιδέα είναι, ότι η ασφάλεια είναι αρκετά δύσκολη υπόθεση και αποφεύγεται η ενσωμάτωση μη τυποποιημένων μηχανισμών ασφαλείας και προτιμάται να εφαρμόζεται με αποδεκτά πρότυπα [5].

Στα Standard του MQTT Oasis [5], αφιερώνεται ολόκληρο κεφάλαιο σχετικά με την ασφάλεια. Στο κεφάλαιο αυτό παρουσιάζονται οι απειλές αλλά και οι προτάσεις αντιμετώπισης σε εφαρμογές που υλοποιούνται με το MQTT πρωτόκολλο.

Σχετικά με τις απειλές ισχύουν οι εξής πιθανές απειλές:

- Οι συσκευές ενδέχεται να διακυβευτούν
- Τα δεδομένα σε κατάσταση ηρεμίας σε πελάτες και διακομιστές ενδέχεται να είναι προσβάσιμα
- Οι συμπεριφορές πρωτοκόλλου ενδέχεται να έχουν παρενέργειες (π.χ. "επιθέσεις χρονισμού")
- Επιθέσεις άρνησης εξυπηρέτησης (DoS)
- Οι επικοινωνίες θα μπορούσαν να παρεμποδιστούν, να τροποποιηθούν, να επαναπροσανατολιστούν ή να αποκαλυφθούν
- Ένεση πλαστών πακέτων ελέγχου

Ενώ λόγω του γεγονότος ότι, οι λύσεις MQTT αναπτύσσονται συχνά σε εχθρικά περιβάλλοντα επικοινωνίας. Σε τέτοιες περιπτώσεις, οι υλοποιήσεις θα πρέπει συχνά να παρέχουν μηχανισμούς προστασίας για τα παρακάτω:

- Έλεγχο ταυτότητας χρηστών και συσκευών
- Εξουσιοδότηση πρόσβασης σε πόρους διακομιστή
- Ακεραιότητα των πακέτων ελέγχου MQTT και των δεδομένων εφαρμογής που περιέχονται σε αυτά
- Απορρήτου των πακέτων ελέγχου MQTT και των δεδομένων εφαρμογής που περιέχονται σε αυτά

Οι λύσεις που προτείνονται, είναι οι εφαρμογές να συμμορφώνονται με πρότυπα και μηχανισμούς ασφαλείας όπως:

- NIST Cyber Security Framework [NISTCSF]
- PCI-DSS [PCIDSS]
- FIPS-140-2 [FIPS1402]
- NAS SUITE [NSAB]

Και να αξιοποιείται ελαφριά κρυπτογράφηση στις περιορισμένων δυνατοτήτων συσκευών όπως

- AES Advanced Encryption Standard [AES]
- Data Encryption Standard [DES]

4.3.1. Επίπεδο δικτύου

Η χρήση ενός φυσικά ασφαλούς δικτύου ή VPN για οποιαδήποτε επικοινωνία μεταξύ Nodes και Broker είναι ένας τρόπος για την παροχή μιας ασφαλούς και αξιόπιστης σύνδεσης. Αυτό θα ήταν κατάλληλο για εφαρμογές όπου χρησιμοποιείται πύλη (gateway), όπου η πύλη (gateway Broker) συνδέεται με τις συσκευές (Nodes) μέσω VPN στην άλλη πλευρά. Η χρήση VPN ενισχύει το γεγονός ότι τα δεδομένα μεταφέρονται μόνο από πιστοποιημένους χρήστες.

4.3.2. Επίπεδο Μεταφοράς

Όταν ο κύριος στόχος είναι να παρέχεται εμπιστευτικότητα, συνήθως χρησιμοποιείται το TLS / SSL για κρυπτογράφηση των δεδομένων που μεταφέρονται. Παρέχει έναν ασφαλή και αποδεδειγμένο τρόπο για να βεβαιωθείτε ότι κανείς δεν μπορεί να διαβάσει τα δεδομένα και ακόμα και να πιστοποιήσει και τις δύο πλευρές, εφαρμόζοντας **πιστοποίηση** και **ταυτοποίηση.**

4.3.3. Επίπεδο εφαρμογής

Το MQTT πρωτόκολλο διαθέτει ενσωματωμένο σύστημα πιστοποίησης χρήση παρέχοντας όνομα χρήστη και κωδικό πρόσβασης. Ο Broker αξιολογεί τα στοιχεία που παρέχονται στο μηχανισμό πιστοποίησης και επιτρέπει ή όχι την πρόσβαση. Όταν θέτουμε σε έναν πελάτη (client) όνομα χρήστη και κωδικό πρόσβασης, αυτά αποστέλλονται στον Broker ως απλό κείμενο, αυτό κάνει το σύστημα ευάλωτο σε επίθεση τύπου **eavesdropping,** διότι το όνομα χρήστη και ο κωδικός πρόσβασης αποστέλλονται σε μορφή απλού κειμένου, από κάποιον που επιτίθεται στο σύστημα και πολύ εύκολα να αποκτήσει τα στοιχεία πιστοποίησης. Ο μόνος τρόπος που εγγυάται απόλυτα ασφαλής μετάδοση των στοιχείων πιστοποίησης είναι η χρήση κρυπτογράφησης στο επίπεδο μεταφοράς. Αυτού του είδους αδυναμίας επιτρέπουν εισβολές τύπου **Man-in-the-Middle** και **replay attacks.**

4.3.4. Προτάσεις λύσεων θεμάτων ασφαλείας του MQTT

Η παρακάτω λίστα αφορά θέματα ασφαλείας, που πρέπει να λαμβάνονται υπόψη κατά την υλοποίηση εφαρμογής με το πρωτόκολλο MQTT. Δεν είναι υποχρεωτικό να εφαρμόζονται όλα, αλλά όσα είναι δυνατόν [5].

- **Εξουσιοδότηση πελατών από τον Εξυπηρετητή**

Το θέμα εξουσιοδότησης πελάτη (nodes) από τον εξυπηρετητή (Broker) αμφότερων κατευθύνσεων είναι απαραίτητος. Ο εξυπηρετητής επιτρέπει τη χρήση πόρων του σε συγκεκριμένους πελάτες, αξιοποιώντας πληροφορίες όπως Όνομα χρήστη, Ταυτότητα πελάτη (node ID), node hostname/ node IP address η το αποτέλεσμα ενός μηχανισμού πιστοποίησης.

- **Πιστοποίηση Εξυπηρετητή από τους πελάτες**

Από την άλλη, υπάρχει η πιστοποίηση του εξυπηρετητή (Broker) από τους πελάτες (Nodes). Το MQTT πρωτόκολλο δεν παρέχει μηχανισμό από τη μεριά των πελατών (Nodes), να πιστοποιήσουν τον εξυπηρετητή (Broker).

- **Ακεραιότητα μηνυμάτων εφαρμογής και πακέτων ελέγχου**

Οι εφαρμογές, μπορούν ανεξάρτητα να συμπεριλαμβάνουν τις τιμές Hash των μηνυμάτων. Αυτό παρέχει ακεραιότητα του περιεχομένου των μηνυμάτων που δημοσιεύονται μέσω του δικτύου. Το TLS πρωτόκολλο παρέχει αυτή τη δυνατότητα όπως και ένα ιδιωτικό δίκτυο VPN.

- **Ιδιωτικότητα των Μηνυμάτων Εφαρμογών και μηνυμάτων ελέγχου**

Η εφαρμογή πρέπει μπορεί παρέχει ιδιωτικότητα στα μηνύματα που μεταφέρονται, αλλά δεν μπορεί να προσφέρει ιδιωτικότητα σε ιδιότητες μηνυμάτων ελέγχου όπως για παράδειγμα το όνομα ενός θέματος (topic Name)

- **Μη επαναλαμβανόμενη αποστολή μηνυμάτων**

Η εφαρμογή πρέπει να φροντίσει να εφαρμόζει κατάλληλη στρατηγική ώστε να μην επιτυγχάνεται επαναλαμβανόμενη αποστολή μηνυμάτων.

- **Ανίχνευση μη φυσιολογικών συμπεριφορών**

Οι εφαρμογές στον εξυπηρετητή πρέπει να παρακολουθούν την συμπεριφορά των πελάτών ώστε να εντοπίσουν μη συνηθισμένη συμπεριφορά. Όπως τα παραδείγματα που ακολουθούν:

- Επαναλαμβανόμενες προσπάθειες σύνδεσης
- Επαναλαμβανόμενες προσπάθειες πιστοποίησης
- Μη φυσιολογικοί τερματισμοί συνδέσεων
- Σάρωση θεμάτων (Topic Scanning), προσπάθειες εγγραφής σε πολλά θέματα

- Αποστολή μηνυμάτων που δεν παραδόθηκαν (στέλνονται σε περιπτώσεις που κάποιος δεν είναι συνδεδεμένος σε ένα θέμα)
- Πελάτες που συνδέονται αλλά δεν στέλνουν δεδομένα

Θα πρέπει η εφαρμογή να έχει τις παρακάτω δυνατότητες:

- Οι εξυπηρετητές πρέπει να μπορούν να αποσυνδέουν πελάτες που ξεπερνούν τους κανόνες ασφαλείας
- Οι εξυπηρετητές όταν ανιχνεύουν μη επιθυμητές συμπεριφορές θα πρέπει να μπορούν να μπλοκάρουν δυναμικά πελάτες, χρησιμοποιώντας IP διευθύνσεις ή ID πελάτη
- Η δυνατότητα περιορισμού και μπλοκαρίσματος, προτείνεται να γίνεται και σε επίπεδο δικτύου, όπου είναι εφικτό, έτσι ώστε να περιορίζονται διευθύνσεις IP.

4.4. Πρόταση υλοποίησης ασφάλειας στην παρούσα Εργασία

Όπως αναφέρθηκε στις παραπάνω ενότητες, η υλοποίηση αυστηρών πρωτοκόλλων ασφαλείας όπως το SSL είναι προβληματική λόγω των περιορισμένων πόρων που διαθέτουν το ESP8266 που χρησιμοποιήθηκε, για την υλοποίηση της εργασίας.
Στο [Δ.14] προτείνονται κάποιες λύσεις στο πρόβλημα της ασφάλειας, μια ελαφριά εναλλακτική λύση, είναι η κρυπτογράφηση μόνο του φορτίου (payload) που αποστέλλεται και να χρησιμοποιηθεί ένα sessionID και μια λειτουργία κατακερματισμού για έλεγχο ταυτότητας.
Ένας απλός τρόπος για να γίνει αυτό είναι η χρήση της Lua γλώσσα προγραμματισμού και της μονάδας κρυπτογράφησης NodeMCU, η οποία περιλαμβάνει υποστήριξη για τον αλγόριθμο AES σε λειτουργία CBC καθώς και τη συνάρτηση κατακερματισμού HMAC.
Η χρήση της κρυπτογράφησης AES απαιτεί σωστά τρία πράγματα για την παραγωγή κρυπτογράφων: ένα μήνυμα, ένα κλειδί και ένα διάνυσμα αρχικοποίησης (IV). Τα μηνύματα και τα κλειδιά είναι απλές ιδέες, αλλά το διάνυσμα αρχικοποίησης αξίζει κάποια συζήτηση.
Για παράδειγμα αν κωδικοποιηθεί ένα μήνυμα στο AES με ένα στατικό κλειδί, θα παράγει πάντα την ίδια έξοδο. Για παράδειγμα, το μήνυμα "05:D5ON" κρυπτογραφημένο με το κλειδί "1234567890ABCDEF" μπορεί να έχει ως αποτέλεσμα "6A82A32998ED180982BB5CA2 A39D5A4E" [Δ.15]. Εάν εκτελεστεί ξανά η κρυπτογράφηση με το ίδιο κλειδί και το ίδιο μήνυμα, θα δημιουργηθεί το ίδιο αποτέλεσμα. Αυτό κάνει ευάλωτο το σύστημα, σε διάφορους κοινούς τύπους επίθεσης, ειδικά σε ανάλυση προτύπων και επιθέσεις επανάληψης.

Σε μια επίθεση ανάλυσης προτύπων, χρησιμοποιείται η γνώση ότι ένα συγκεκριμένο κομμάτι δεδομένων θα παράγει πάντα το ίδιο κρυπτογράφημα για να μαντέψει ποιος είναι ο σκοπός ή το περιεχόμενο διαφορετικών μηνυμάτων χωρίς να γνωρίζει πραγματικά το μυστικό κλειδί. Για παράδειγμα, εάν το μήνυμα "E40D86C04D723AFF" αποστέλλεται πριν από όλες τις άλλες επικοινωνίες, κάποιος μπορεί να μαντέψει γρήγορα ότι πρόκειται για σύνδεση. Εν ολίγοις, εάν το σύστημα σύνδεσης είναι απλοϊκό, η αποστολή αυτού του πακέτου (μια επίθεση επανάληψης) μπορεί να είναι αρκετή για να θέσει κάποιον ως εξουσιοδοτημένο χρήστη, κάτι που μπορεί να οδηγήσει σε σύστημα σε δεινή θέση.

Τα IVs κάνουν την ανάλυση των σχεδίων πιο δύσκολη. Ένα IV είναι ένα κομμάτι δεδομένων που αποστέλλεται μαζί με το κλειδί που τροποποιεί το τελικό αποτέλεσμα κρυπτογράφων. Όπως υποδηλώνει το όνομα, αρχικοποιεί την κατάσταση του αλγορίθμου κρυπτογράφησης πριν εισέλθουν τα δεδομένα. Το IV πρέπει να είναι διαφορετικό για κάθε μήνυμα που αποστέλλεται, έτσι ώστε τα επαναλαμβανόμενα δεδομένα να κρυπτογραφούνται σε διαφορετικό κρυπτογραφικό κείμενο και ορισμένα κρυπτά (όπως το AES-CBC) απαιτούν να είναι απρόβλεπτα - ένας πρακτικός τρόπος για να επιτευχθεί αυτό είναι απλά να τυχαιοποιείται κάθε φορά. Τα IVs δεν πρέπει να κρατούνται μυστικά, αλλά είναι τυπικό να τα ανακατευτούν με κάποιο τρόπο.

Παρόλο που αυτό προστατεύει από την ανάλυση προτύπων, δεν βοηθά με τις επιθέσεις επανάληψης. Για παράδειγμα, η αναμετάδοση ενός δεδομένου συνόλου κρυπτογραφημένων δεδομένων θα εξακολουθήσει να αντιγράφει το αποτέλεσμα. Για να αποφευχθεί αυτό, πρέπει να επαληθεύεται ο αποστολέας. Μπορεί να χρησιμοποιηθεί ένα δημόσιο, ψευδοτυχαίο αναγνωριστικό περιόδου σύνδεσης για κάθε μήνυμα. Αυτό το αναγνωριστικό περιόδου σύνδεσης μπορεί να δημιουργηθεί από τη συσκευή λήψης, τοποθετώντας το σε ένα θέμα (topic) MQTT.

5. Κεφάλαιο 5ο

5.1. Κατασκευές που συνδέονται στο σύστημα

Στα πλαίσια της εργασίας και για προσωπικές ανάγκες ενσωματώθηκαν δύο κατασκευές με αισθητήρες που και αυτές συνδέθηκαν στο σύστημα.

5.1.1. Εφαρμογή Ανιχνευτής Καπνού

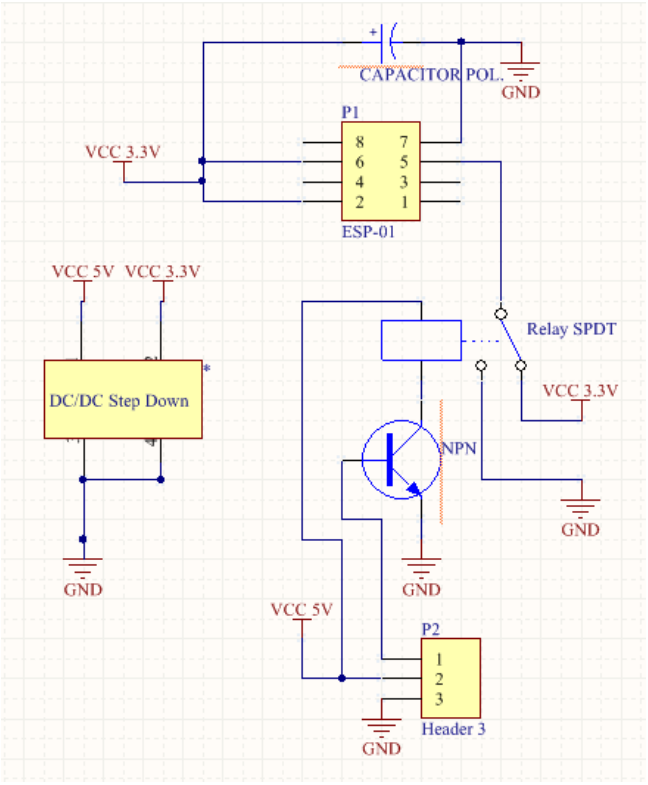
Για την συγκεκριμένη εφαρμογή, χρησιμοποιήθηκε ένας έτοιμος αισθητήρας ανίχνευσης καπνού. Συγκεκριμένα το προϊόν Konig SAS-SA100, και ο λόγος γιατί είναι ένας πολύ οικονομικός αισθητήρας γύρω στα 8€ κοστολόγιο. Κατόπιν ελέγχου της κατασκευής διαπιστώθηκε ότι χρησιμοποιεί το ολοκληρωμένο κύκλωμα BL59A10 [M.7]. Από το σχηματικό του ολοκληρωμένου κυκλώματος βρέθηκε ότι η ακίδα 7 επονομαζόμενη I/O χρησιμοποιείται είτε για να δέχεται εξωτερικά σήματα και να ενεργοποιείται ο αισθητήρας, είτε όταν ενεργοποιείται ο αισθητήρας στέλνει υψηλό δυναμικό, ώστε να δώσει σήμα ενεργοποίησης. Εκμεταλλευόμενοι τη δυνατότητα αυτή σχεδιάστηκε ένα κατάλληλο προσαρμοστικό κύκλωμα που δίνει σήμα στο σύστημά μας.

Εικόνα 38: Αισθητήρας Ανίχνευσης Καπνού

Εικόνα 39: KONIG SAS-SA100 Αισθητήρας Ανίχνευσης Καπνού

Σχηματικό διάγραμμα

Το σχηματικό διάγραμμα του προσαρμοστικού κυκλώματος το οποίο υλοποιείται με το ESP-01s ακολουθεί στην εικόνα 45.



Εικόνα 40:Σχηματικό διάγραμμα Προσαρμοστικού Κυκλώματος Αισθητήρα Καπνού

Η λογική του κυκλώματος είναι απλή, όταν δέχεται το σήμα ενεργοποίησης του αισθητήρα, το τρανζίστορ ανοίγει και συνδέει τον ηλεκτρονόμο στη γείωση του κυκλώματος. Τότε ο ηλεκτρονόμος συνδέει την κοινή επαφή που είναι συνδεδεμένο στο GPIO 02 του ESP-01s στη γείωση, ενώ στην κανονική κατάσταση Normally Open είναι συνδεδεμένο στο υψηλό δυναμικό 3.3 V. Ο κώδικας του ESP-01s μόλις ανιχνεύσει το χαμηλό δυναμικό στέλνει μήνυμα στο MQTT Server.

Ο ρόλος του ηλεκτρονόμου, είναι βοηθητικός αλλά και κρίσιμος. Το ESP-01, για να λειτουργήσει και να τρέξει το πρόγραμμα, απαιτεί τα GPIO 01 και 02 να είναι συνδεδεμένα κατά την εκκίνησε σε υψηλό δυναμικό (3.3V), αλλιώς δεν εκτελείται το πρόγραμμα. Οπότε με τον ηλεκτρονόμο, κατά την εκκίνηση έχουμε το GPIO 02 σε υψηλό δυναμικό και όταν έρθει σήμα ενεργοποίησης το γειώνουμε και το ESP-01s αντιλαμβάνεται το σήμα εισόδου διαβάζοντας χαμηλό δυναμικό.

Λίστα Υλικών Ανιχνευτής Καπνού

Comment	Pattern	Quantity	Components
DC/DC Step Down Converter		1	
ESP-01	HDR2X4	1	Header, 4-Pin, Dual row
Header 3	HDR1X3	1	Header, 3-Pin
RELAY SPDT	RELAY2	1	Relay SPDT

BC 546	NPN	1	TO-92

Πίνακας 9:Λίστα Υλικών - Ανιχνευτής Καπνού

Στην συγκεκριμένη κατασκευή δεν κατασκευάστηκε τυπωμένο κύκλωμα, λόγω περιορισμένου χρόνου, αλλά και λόγω του ότι οι συνδέσεις ήταν απλές. Χρησιμοποιήθηκε διάτρητη πλακέτα και τοποθετήθηκε σε πλαστικό κουτί κατασκευής. Στο παράρτημα 8 φαίνεται η κατασκευή αλλά και ο κώδικας που χρησιμοποιήθηκε στο ESP-01s.

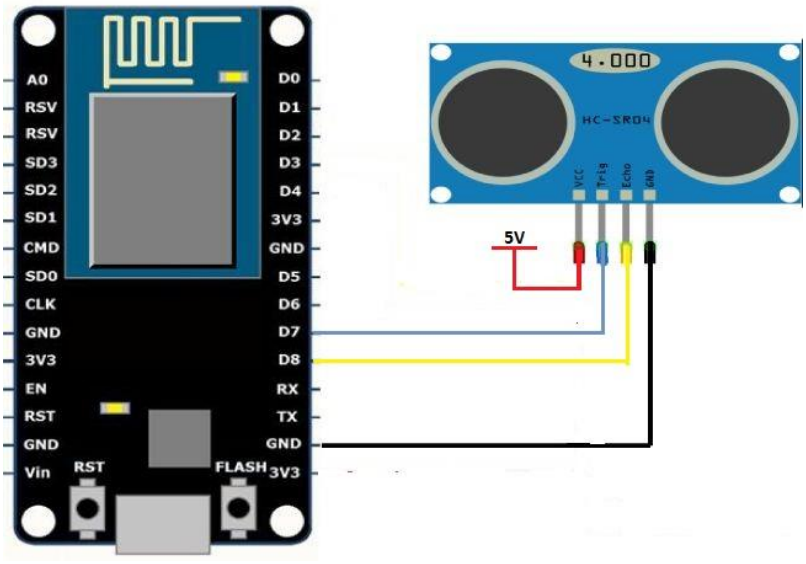
5.1.2. Αισθητήρας μέτρησης Στάθμης Δεξαμενής Νερού

Η κατασκευή αυτή, έγινε με σκοπό την αντιμετώπιση του προβλήματος λειψυδρίας της πόλης του Ηρακλείου. Λόγω διακοπών παροχής νερού, χρειάστηκε ο απομακρυσμένος έλεγχος στάθμης δεξαμενής νερού. Σχεδιάστηκε λοιπόν μια κατασκευή που στηρίζεται στο NodeMCU ESP-12, διότι αυτό είχαμε στη διάθεσή μας και υπήρχε και πλακέτα έτοιμη από προηγούμενες δοκιμές και ένας αισθητήρας υπερήχων HC-SR04 [M.8].

Η λογικής της κατασκευής είναι αρκετά απλή, συνδέσαμε τον το αισθητήρα υπερήχων HC-SR04 σύμφωνα με τη συνδεσμολογία στην εικόνα 46.

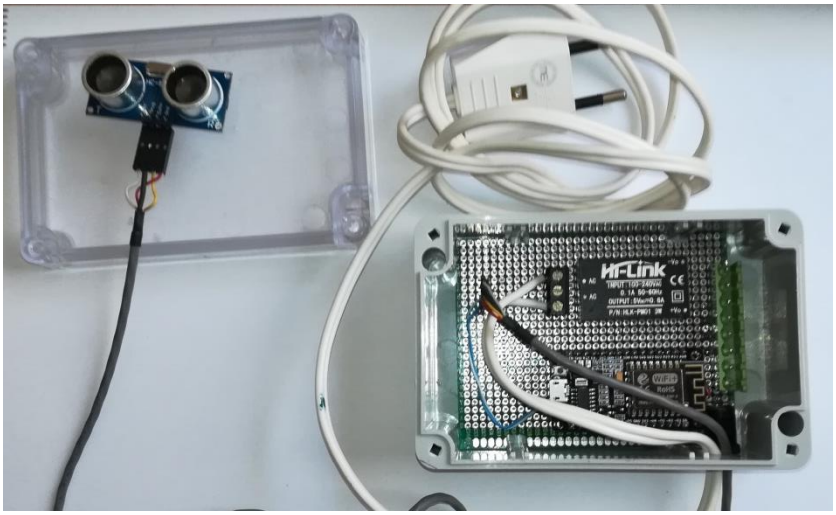
HC-SR04	NodeMCU – ESP12
Τάση 5V	Vin – 5V
Ground	Ground
Echo pin	D7
Trigger Pin	D8

Πίνακας 10: Συνδεσμολογία Αισθητήρα στάθμης



Εικόνα 41:Σύνδεση αισθητήρα Στάθμης

Η φιλοσοφία του κώδικα είναι, να στέλνει ένα σύνολο παλμών μέσω του Trigger Pin, και έπειτα να τους διαβάζει μέσω του Echo Pin. Το χρονικό διάστημα που διαρκεί να επιστρέψουν οι παλμοί, μεταφράζεται σε απόσταση από την επιφάνεια του υγρού, που στην περίπτωσή μας είναι του νερού της δεξαμενής.



Εικόνα 42: Κατασκευή Αισθητήρα Στάθμης Δεξαμενής

Στο Παράρτημα 9 βρίσκονται εικόνες της κατασκευής τοποθετημένης στη δεξαμενή και ο κώδικας που χρησιμοποιήθηκε.

6. Αποτελέσματα

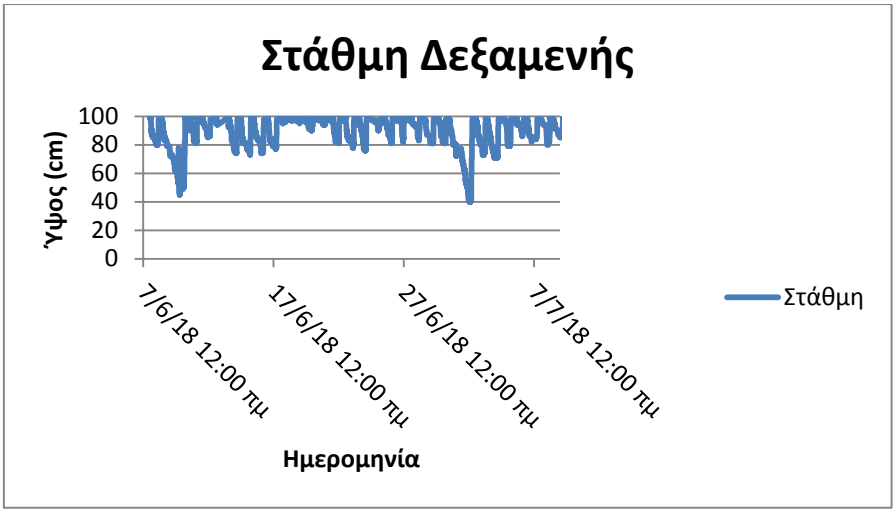
Για την μελέτη των αποτελεσμάτων του συστήματος, θα αξιοποιηθούν τα δεδομένα που έχουν αποθηκευτεί στη βάση δεδομένων. Η εφαρμογή στο Node-RED, παρά το γεγονός ότι έχει τη δυνατότητα να αντλεί και να εμφανίζει τα δεδομένα στα γραφήματα της, καθυστερεί πάρα πολύ. Λειτουργεί πολύ καλά για δεδομένα λίγων ημερών και αυτό εξαρτάται από τη δειγματοληψία. Επίσης, ένα άλλο αρνητικό είναι ότι τα γραφήματα που ενημερώνονται από Node-RED, χωρίς να αντλούνται από τη βάση δεδομένων, είναι προσωρινά και όταν γίνει επανεκκίνηση του RPI χάνονται και πρέπει να ενημερωθούν ξανά, συλλέγοντας νέα δεδομένα.

Οπότε για να μπορέσουμε να παρουσιαστούν τα δεδομένα που έχουν συλλεχθεί στη βάση δεδομένων, το διάστημα των δοκιμών, θα χρησιμοποιηθούν εξωτερικά γραφήματα. Στην περίπτωση μας θα χρησιμοποιηθεί το Excel 2010 για να αντληθούν τα δεδομένα από τη βάση MySQL και έπειτα να επεξεργαστούν ώστε να δημιουργηθεί γραφική απεικόνιση.

Κρίνεται σημαντικό να αναφερθεί ότι στη βάση δεδομένων αποθηκεύονται οι τιμές που διαβάζονται από τα αισθητήρια, οπότε για να φανεί το η τιμή του φυσικού μεγέθους θα πρέπει να εκτελεστεί κατάλληλο ερώτημα στη βάση (sql query) όπου θα ενσωματώνει τη εξίσωση βαθμονόμησης του κάθε αισθητηρίου. Επιλέχθηκε ο τρόπος αυτός, έτσι ώστε να μην αλλοιώνονται τα δεδομένα που αποθηκεύονται, αλλά και επειδή είναι πολύ εύκολο να ενσωματωθείto firmware κάθε Node ώστε να επιστρέφει τις φυσικές τιμές, αυτό αφήνεται στη κρίση του κατασκευαστή, εφόσον δημιουργηθεί ένα σταθερό σύστημα για συγκεκριμένο σκοπό.

6.1. Συλλογή Δεδομένων από Δεξαμενή Νερού

Στο γράφημα στην Εικόνα 48 φαίνεται η στάθμη του Νερού μια δεξαμενής. Όταν η στάθμη είναι 100cm τότε σημαίνει ότι η δεξαμενή είναι γεμάτη, ενώ όσο μειώνεται το ύψος η στάθμη μειώνεται. Γίνεται δειγματοληψία κάθε δέκα λεπτά και διάγραμμα δείχνει τη στάθμη του νερού χρήσης, για την περίοδο 7/6/2018 έως 7/7/2018.

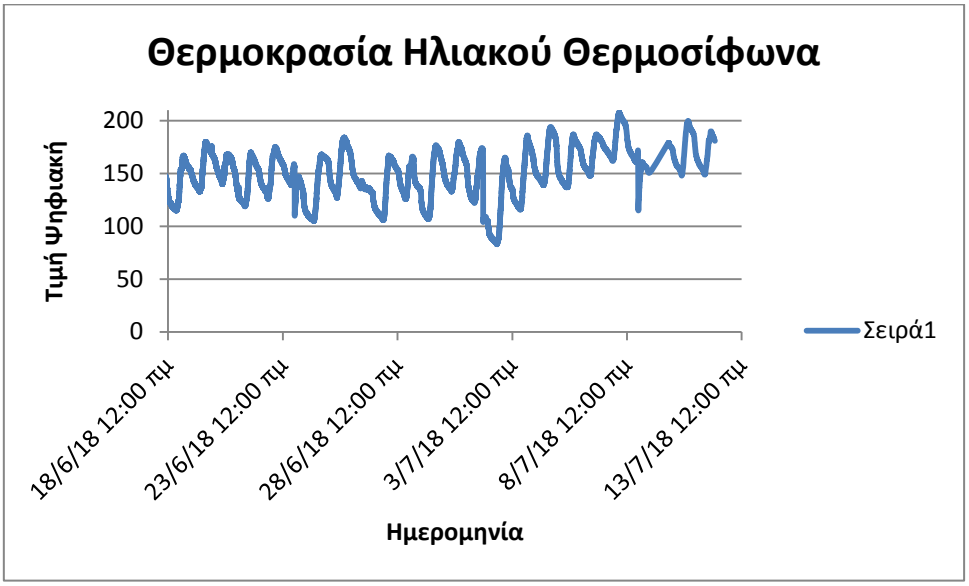


Γράφημα 5: Στάθμη Δεξαμενής Νερού

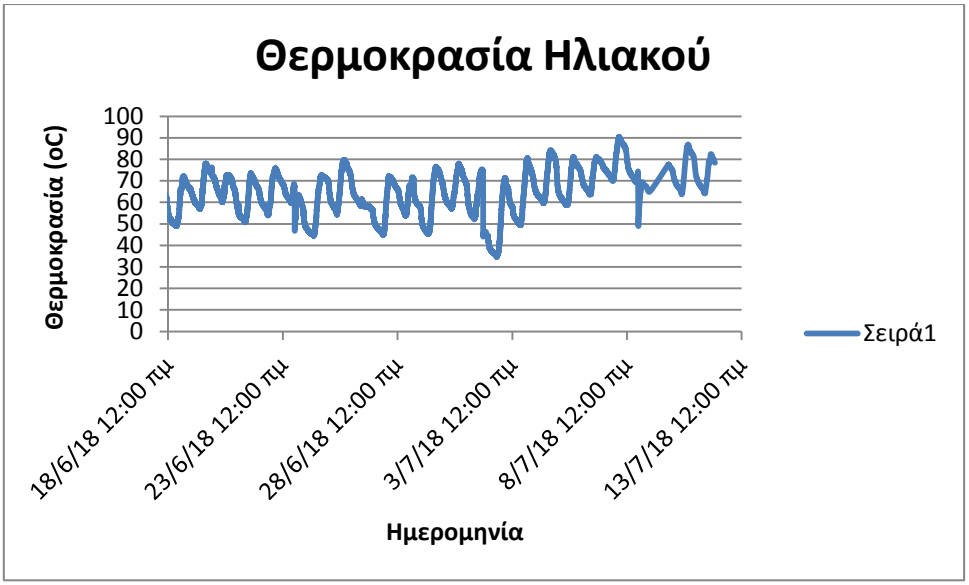
6.2. Συλλογή Δεδομένων Ηλιακού Θερμοσίφωνα

Στα Γραφήματα 2 & 3 απεικονίζονται τα δεδομένα θερμοκρασίας του ηλιακού θερμοσίφωνα. Στο Γράφημα 2 φαίνονται τα δεδομένα με την ψηφιακή τιμή που διαβάζεται από το Node του Ηλιακού, ενώ στο γράφημα 3 είναι το γράφημα της θερμοκρασίας, αφού έχει υπολογιστεί η πραγματική θερμοκρασία, βάσει της εξίσωσης βαθμονόμησης του αισθητήρα

$$Y = 0{,}4463 \times X - 2{,}3557$$



Γράφημα 6: Θερμοκρασία Ηλιακού Θερμοσίφωνα Ψηφιακές Τιμές



Γράφημα 7: Γράφημα θερμοκρασία Ηλιακού

Σαν συμπέρασμα, μπορούμε να πούμε, ότι λόγω της περιόδου συλλογής δεδομένων 18/6/2018 – 13/7/2018, καλοκαιρινός μήνας η θερμοκρασία κυμαίνεται από 50-90 ^{o}C, όπως είναι αναμενόμενο. Το ενδιαφέρον θα είναι η μελέτη του συγκεκριμένου αντικειμένου, σε συνδυασμό με ατμοσφαιρική θερμοκρασία και ηλιοφάνεια κατά την χειμερινή περίοδο.

6.3. Συλλογή Δεδομένων μέτρησης Ρεύματος Γραμμής

Αξιοποιώντας τον αισθητήρα μέτρησης ρεύματος γραμμής, έγινε καταμέτρηση του ρεύματος που καταναλώνει ένα πιεστικό ενός ίππου, για διάστημα τριών (3) ημερών. Οπότε στο γράφημα 8 φαίνεται η ψηφιακή τιμή του ρεύματος που διαρρέει την παροχή του κινητήρα.

Γράφημα 8: Κατανάλωση ρεύματος Γραμμής

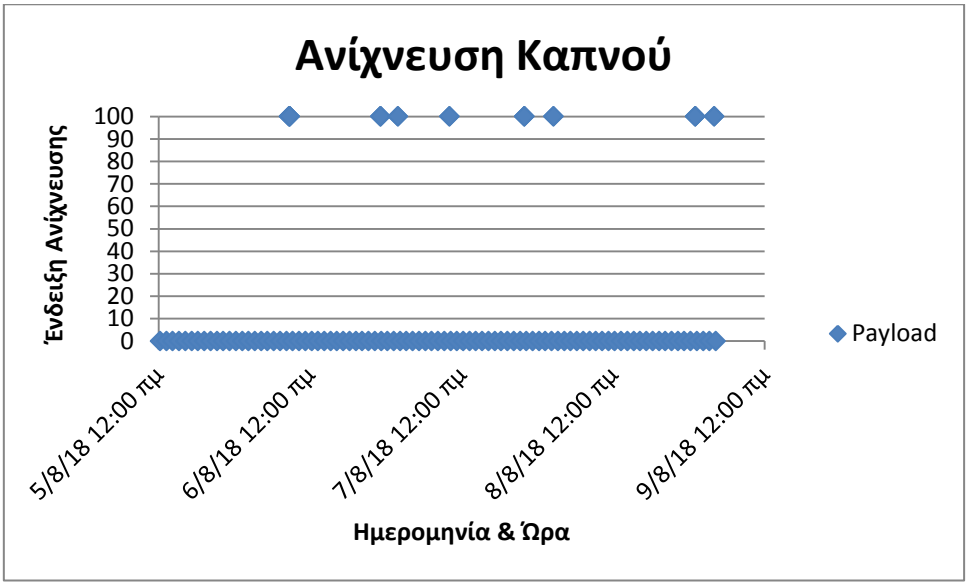
Στο γράφημα 9 απεικονίζεται η τιμή του ρεύματος αφού εφαρμοστεί η εξίσωση βαθμονόμησης του αισθητήρα

$$Y = 0{,}0065 \times X + 0{,}2227$$

Γράφημα 9:Κατανάλωση Ρεύματος Γραμμής (Amperes)

6.4. Συλλογή Δεδομένων Ανίχνευσης Καπνού

Στο γράφημα 10 φαίνονται τυχαίες στιγμές ενεργοποίησης του αισθητήρα καπνού. Στις στιγμές αυτές έχει προγραμματιστεί επίσης η αποστολή email αλλά και SMS λόγω τις κρισιμότητας της ειδοποίησης.



Γράφημα 10: Σημεία Ανίχνευσης Καπνού

7. Μελλοντική Εργασία και Επεκτάσεις

Στην ενότητα αυτή θα αναφερθούμε στις επεκτάσεις της εργασίας. Αξίζει να αναφερθεί ότι από τη φύση της η εργασία μπορεί να υποστηρίξει επεκτάσεις και βελτιώσεις σε πολλά επίπεδα, αφού σχεδιάστηκε κατ' αρχήν με αυτό το σκεπτικό.

Άρα, ξεκινάμε ότι η εργασία μπορεί να εμπλουτιστεί με διάφορα σενάρια για εκπαιδευτικούς σκοπούς. Θα μπορούσε να αξιοποιηθεί σε διάφορα εργαστήρια των σχολών Ηλεκτρολόγων Μηχανικών ή Μηχανικών Πληροφορικής και Μηχανικών Ηλεκτρονικών, έτσι ώστε οι εκπαιδευόμενοι να αποκτήσουν γνώσεις και δεξιότητες, σε τομείς ηλεκτρονικών κατασκευών και προγραμματισμού σχετικά με το Διαδίκτυο των Πραγμάτων.

Στη συνέχεια, προτείνεται η αξιοποίηση του συστήματος, προκειμένου να σχεδιαστούν αισθητήρια ή να χρησιμοποιηθούν έτοιμα για να ενσωματωθούν και να δοκιμαστούν από το σύστημα.

Ένα σημείο σημαντικό, που δεν πραγματοποιήθηκε λόγω περιορισμένου χρόνου, είναι η δημιουργία μια εφαρμογής σε κινητό Android, χρησιμοποιώντας π.χ. το Android Studio ή το App Inventor ώστε να επικοινωνεί με τον MQTT Broker και να παρουσιάζει πληροφορίες και δεδομένα, αλλά και να δίνει εντολές σε ενεργοποιητές.

Επίσης ένα κενό σημείο που, είχε ληφθεί υπόψη στον αρχικό σχεδιασμό είναι ο συνδυασμός και άλλων ασύρματων τεχνολογιών, εκτός από το WiFi. Αρχικά, υπήρχε η σκέψη να αξιοποιηθούν χρησιμοποιηθεί και επικοινωνία μέσω Bluetooth και για τον σκοπό αυτό είχε παραγγελθεί και αντίστοιχο ολοκληρωμένο το ESP-32, πάλι βασισμένο στο ESP8266, που υποστηρίζει και Bluetooth, αλλά λόγω της έκτασης της εργασίας περιορίστηκε μόνο σε WiFi.

Το Raspberry Pi 3, έχει θύρες GPIO, και θα μπορούσαν να συνδεθούν στο ίδιο το Raspberry Pi αισθητήρες και ηλεκτρονόμοι, ενσύρματα και μάλιστα επειδή το Raspberry Pi δεν διαθέτει καμία αναλογική θύρα εισόδου, είχε παραγγελθεί και το ολοκληρωμένο MCP3008 [M.10] που προσθέτει οκτώ (8) αναλογικές θύρες και επικοινωνεί με το Raspberry Pi ή οποιουδήποτε άλλου μικροελεγκτή, μέσω SPI επικοινωνίας.

Τέλος, ολοκληρώνοντας της περαιτέρω προτάσεις είναι η ενασχόληση αποκλειστικά με το μηχανισμό ασφαλείας, όπως παρουσιάστηκε στο κεφάλαιο 4. Να γίνουν δηλαδή, οι κατάλληλες διορθώσεις στο λογισμικό του συστήματος ώστε να είναι ασφαλές καθώς υπάρχει η πιθανότητα να εγκατασταθεί σε μη φιλικά περιβάλλοντα. Αφορά ένα κομμάτι που αξίζει να ασχοληθεί κάποιος, αποκλειστικά και μόνο ώστε να αποκτήσει γνώσεις γύρω από την ασφάλεια δικτύων γενικά αλλά και εξειδικευμένα στο ΔτΠ, ενώ παράλληλα έχουν υλοποιηθεί αρκετές έρευνες γύρω από το θέμα αυτό [22, 25, 26, 27].

8. Σύνοψη

Με την παρούσα ενότητα, συνοψίζεται η παρούσα πτυχιακή εργασία. Η εργασία αυτή αποτέλεσε μια διαδρομή στο κόσμο του Διαδικτύου των Πραγμάτων. Για το σύνολο της εργασίας χρειάστηκαν γνώσεις από πολλούς τομείς μαθημάτων, που αφορούν τόσο τους Ηλεκτρολόγους, Ηλεκτρονικούς αλλά και τους Μηχανικούς Πληροφορικής. Επίσης η εργασία αυτή, εμπλουτίστηκε από γνώσεις και εμπειρία αρκετών χρόνων, σε διάφορους τομείς, όπως λειτουργικών συστημάτων, Δικτύων και Τηλεπικοινωνιών, Ασφάλειας Συστημάτων αλλά και Διδακτικής της Πληροφορικής.

Το παραδοτέο της εργασίας, είναι ένας οδηγός, που μπορεί να καθοδηγήσει μαθητές, φοιτητές μηχανικούς, ενήλικες ερασιτέχνες που ενδιαφέρονται να ασχοληθούν με το Διαδίκτυο των Πραγμάτων, και μπορεί να δώσει κίνητρα και ιδέες, για ενασχόληση και επέκταση νέων γνώσεων.

Η κεντρική ιδέα της εργασίας ήταν, ο Σχεδιασμός και Υλοποίηση Συστήματος Συλλογής δεδομένων απομακρυσμένων αισθητηρίων και Ελέγχου Απομακρυσμένων Ηλεκτρονόμων, μέσω ασύρματων τεχνολογιών. Τελικά, δημιουργήθηκε ένα σπονδηλωτό σύστημα, το οποίο δίνει τη δυνατότητα ευελιξίας και προσαρμογής σε διάφορες ανάγκες, και το οποίο μπορεί να αξιοποιηθεί τόσο σε πραγματικές εφαρμογές σε κάποιο χώρο, όπως οικία, βιοτεχνία, χωράφι, θερμοκήπιο ή για εκπαιδευτικούς σκοπούς σε κάποιο εργαστήριο.

Αφού προηγήθηκε σχετική βιβλιογραφική επισκόπηση, σχετικά με το τι είναι το διαδίκτυο των πραγμάτων, αλλά και το πώς μπορεί να υλοποιηθεί το αντικείμενο της εργασίας, πήγαμε στο επόμενο βήμα που αφορούσε τον σχεδιασμό του συστήματος και έπειτα την παραγγελία υλικών για την υλοποίηση του.

Στον σχεδιασμό καταλήξαμε ότι θα χρησιμοποιούταν ένα Raspberry Pi 3 ως κεντρική πύλη (gateway) και θα παρείχε τις απαραίτητες υπηρεσίες για το σύστημα, έχοντας ως σκεπτικό ότι το RPI θα μπορούσε να αντικατασταθεί με οποιοδήποτε άλλο, υπολογιστικό σύστημα, απλά το RPI παρέχει αρκετές δυνατότητες επικοινωνίας με το εξωτερικό περιβάλλον, είναι μικρό και αρκετά οικονομικό.

Για τη δημιουργία των απομακρυσμένων συσκευών, ενώ αρχικά υπήρχε η σκέψη να χρησιμοποιηθούν Arduino, στη συνέχεια προτιμήθηκε να χρησιμοποιηθούν αναπτυξιακές πλακέτες (Boards) βασισμένα στο ολοκληρωμένο ESP8266, με κύριο

λόγω το γεγονός ότι είχαν ενσωματωμένη τη δυνατότητα ασύρματης επικοινωνίας μέσω wifi.

Από την οικογένεια των πλακετών αυτών, χρησιμοποιήθηκαν για το σκοπό μας τα ESP-01 και ESP-12. Υπήρξε η σκέψη να χρησιμοποιηθεί και το ESP-32 το τελευταίο μοντέλο και πιο εξελιγμένο της οικογενείας, όμως δεν έχουν ακόμα αναπτυχθεί όλες οι σχετικές βιβλιοθήκες, οπότε καταλήξαμε σε κάτι πιο σταθερό.

Το ESP-01 χρησιμοποιήθηκε στην κατασκευή αυτόνομων ρευματοδοτών που ενεργοποιούνται μέσω του συστήματος, ενώ το ESP-12 κατόπιν σχεδιασμού και βελτιώσεων, χρησιμοποιήθηκε ώστε να μπορεί να διαβάζει αναλογικές θύρες και να αξιοποιεί ψηφιακές εισόδους και εξόδους κατά βούληση.

Στο RPI τελικά εγκαταστάθηκε, ένας μεσολαβητής (Broker) MQTT για να επικοινωνούν τα Nodes του συστήματος, και μια βάση δεδομένων MySQL για να αποθηκεύονται τα δεδομένα που ανταλλάσσονται μεταξύ της πύλης του συστήματος (RPI) και των Nodes με τους αισθητήρες ή τους ενεργοποιητές. Επίσης ρυθμίστηκε ώστε να παρέχει υπηρεσίας Access Point, έτσι ώστε να παρέχει αυτόνομο ασύρματο δίκτυο που θα χρησιμοποιείται για την επικοινωνία των Nodes με την πύλη.

Ως εφαρμογή διαχείρισης του συστήματος, χρησιμοποιήθηκε το Node-RED που είναι μια γλώσσα προγραμματισμού για το διαδίκτυο των πραγμάτων, η οποία έρχεται προ-εγκατεστημένη στο RPI με πολλές δυνατότητες.

Τέλος για να δοθεί περισσότερη ευελιξία σε θέματα επικοινωνίας εγκαταστάθηκε στο σύστημα ένα GSM Module μέσω του οποίου το σύστημα μπορεί να στέλνει γραπτά μηνύματα (SMS) σε περίπτωση ανάγκης ή να δέχεται εντολές μέσω Δικτύου Κινητής Τηλεφωνίας.

Τέλος, παρουσιάζονται οδηγίες αξιοποίησης του κινητού, ως διεπαφή του χρήστη για να επικοινωνεί με το σύστημα, τόσο με τη χρήση κατάλληλης εφαρμογής άλλα και με φωνητικές εντολές, τόσο εντός του ασύρματου δικτύου, αλλά και εκτός εμβέλειας του ασύρματου δικτύου αξιοποιώντας το διαδίκτυο.

Η πτυχιακή εργασία, κλείνει με προτάσεις επέκτασης και μελλοντικής εργασίας, οι οποίες είναι πολλές και αφορούν διάφορους τομείς. Πιστεύεται ότι έτσι επιτυγχάνεται το όραμα της εργασίας αυτής, να είναι δηλαδή ευέλικτη, προσαρμόσιμή και να

προάγει τη δημιουργικότητα και την φαντασία, όποιου ασχοληθεί με το συγκεκριμένο αντικείμενο, με σκοπό τη μάθηση και την αξιοποίηση του συστήματος αυτού σε σενάρια της καθημερινότητάς.

9. Βιβλιογραφία

[1] Πρακτικά Συνεδρίων: Πληροφορική Στην Εκπαίδευση (2016,2017). Διαθέσιμα στον δικτυακό τόπο http://di.ionio.gr/cie (Τελευταία πρόσβαση 4/5/2018).

[2] Stolpe, M. (2016). The internet of things: Opportunities and challenges for distributed data analysis. ACM SIGKDD Explorations Newsletter, 18(1), 15-34.

[3] Ramohalli, N., & Adegbija, T. (2018, March). Modular electronics for broadening non-expert participation in STEM innovation: An IoT perspective. In Integrated STEM Education Conference (ISEC), 2018 IEEE (pp. 167-174). IEEE.

[4] Coppola, M. & Kornaros, G. (2017,June). Internet of Things for 21st-Century Learners. Άρθρο στο Computing Now.Διαθέσιμο στον δικτυακό τόπο https://www.computer.org/web/computingnow/archive/iot-for-21st-century-learners-june-2017-introduction (Τελευταία πρόσβαση 5/5/2018)

[5] MQTT Version 3.1.1. Edited by Andrew Banks and Rahul Gupta.29 October 2014. OASIS Standard.http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html.Latest version: http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html

[6] Τριανταφύλλου, Α. (2017). Διαδίκτυο των πραγμάτων. Πτυχιακή εργασία ΑΕΙ Πειραιά Τ.Τ. Διαθέσιμο στο δικτυακό τόπο http://okeanis.lib.puas.gr/xmlui/bitstream/handle/123456789/3787/%CE%99%CE%9D%CE%A4%CE%95%CE%A1%CE%9D%CE%95%CE%A4-of-THINGS.pdf?sequence=1 (Τελευταία πρόσβαση 7/5/2018).

[7] Θεοδωρακόπουλος, Λ. Α., & Καγκάνης, Α. Α. (2016). Το διαδίκτυο των πραγμάτων (internet of things–IoT) και οι εφαρμογές του. Πτυχιακή εργασία ΤΕΙ Δυτικής Ελλάδας. Διαθέσιμο στο δικτυακό τόπο http://repository.library.teimes.gr/xmlui/bitstream/handle/123456789/3175/CIED%20%CE%98%CE%95%CE%9F%CE%94%CE%A9%CE%A1%CE%91%CE%9A%CE%9F%CE%A0%CE%9F%CE%A5%CE%9B%CE%9F%CE%A3%20-%20%CE%9A%CE%91%CE%93%CE%9A%CE%91%CE%9D%CE%97%CE%A3.pdf?sequence=1&isAllowed=y (Τελευταία πρόσβαση 7/5/2018).

[8] Light, R. A. (2017). Mosquitto: server and client implementation of the MQTT protocol. Journal of Open Source Software, 2(13).

[9] Cruz-Piris, L., Rivera, D., Marsa-Maestre, I., de la Hoz, E., & Velasco, J. R. (2018). Access Control Mechanism for IoT Environments Based on Modelling Communication Procedures as Resources. Sensors, 18(3), 917. Διαθέσιμο στον δικτυακό τόπο http://www.mdpi.com/1424-8220/18/3/917/htm (Τελευταία πρόσβαση 7/5/2018).

[10] Chaczko, Z., & Braun, R. (2017, July). Learning data engineering: Creating IoT apps using the node-RED and the RPI technologies. In Information Technology Based Higher Education and Training (ITHET), 2017 16th International Conference on (pp. 1-8). IEEE.

[11] Kodali, R. K., & Soratkal, S. (2016, December). MQTT based home automation system using ESP8266. In Humanitarian Technology Conference (R10-HTC), 2016 IEEE Region 10 (pp. 1-5). IEEE.

[12] Upadhyay, Y., Borole, A., & Dileepan, D. (2016, March). MQTT based secured home automation system. In *Colossal Data Analysis and Networking (CDAN), Symposium on* (pp. 1-4). IEEE.

[13] Δρίβας, Γ. Α. (2016). Χρήση του MQTT πρωτόκολλου για απομακρυσμένη λήψη τιμών από αισθητήρες κινητού τηλεφώνου. Πτυχιακή εργασία Τ.Ε.Ι. Δυτικής Ελλάδας. Διαθέσιμο στον δικτυακό τόπο http://repository.teiwest.gr/xmlui/bitstream/handle/123456789/3169/CIED%20%CE%94%CE%A1%CE%99%CE%92%CE%91%CE%A3%20%CE%93%CE%95%CE%A9%CE%A1%CE%93%CE%99%CE%9F%CE%A3.pdf?sequence=1&isAllowed=y (Τελευταία πρόσβαση 16/5/2018).

[14] Ray, P. P. (2017). A Survey on Visual Programming Languages in Internet of Things. *Scientific Programming*, *2017*.

[15] Mehari, M. T., Shahid, A., Moerman, I., & De Poorter, E. (2017). Demo Abstract: An Intuitive Drag and Drop Framework for Wireless Network Experimentation. *Energy*, *240*, 310.

[16] Motakis, A., Kornaros, G., & Coppola, M. (2011, June). Dynamic resource management in modern multicore SoCs by exposing NoC services. In *Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC), 2011 6th International Workshop on* (pp. 1-7). IEEE.

[17] Kornaros, G., Christoforakis, I., Tomoutzoglou, O., Bakoyiannis, D., Vazakopoulou, K., Grammatikakis, M., & Papagrigoriou, A. (2015, August). Hardware support for cost-effective system-level protection in multi-core socs. In *2015 Euromicro Conference on Digital System Design (DSD)* (pp. 41-48). IEEE.

[18] Kornaros, G., Harteros, K., Christoforakis, I., & Astrinaki, M. (2014, October). I/O virtualization utilizing an efficient hardware system-level memory management unit. In *System-on-Chip (SoC), 2014 International Symposium on* (pp. 1-4). IEEE.

[19] Coppola, M., Falsafi, B., Goodacre, J., & Kornaros, G. (2013, March). From embedded multi-core socs to scale-out processors. In *Proceedings of the Conference on Design, Automation and Test in Europe* (pp. 947-951). EDA Consortium.

[20] Kornaros, G. (Ed.). (2010). Multi-core embedded systems. CRC Press.

[21] Nikam, U. V., Misalkar, H. D., & Burange, A. W. Securing MQTT protocol in IoT by payload Encryption Technique & Digital Signature.

[22] Firdous, S. N., Baig, Z., Valli, C., & Ibrahim, A. (2017, June). Modelling and Evaluation of Malicious Attacks against the IoT MQTT Protocol. In Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), 2017 IEEE International Conference on (pp. 748-755). IEEE.

[23] Heer, T., Garcia-Morchon, O., Hummen, R., Keoh, S. L., Kumar, S. S., & Wehrle, K. (2011). Security Challenges in the IP-based Internet of Things. *Wireless Personal Communications*, *61*(3), 527-542.

[24] Jeyanthi, N. (2016). Internet of Things (IoT) as Interconnection of Threats (IoT). In *Security and Privacy in Internet of Things (IoTs)* (pp. 21-39). CRC Press.

[25] Perrone, G., Vecchio, M., Pecori, R., & Giaffreda, R. (2017). The Day After Mirai: A Survey on MQTT Security Solutions After the Largest Cyber-attack Carried Out through an Army of IoT Devices. In *2nd International Conference on Internet of Things, Big Data and Security* (pp. 246-253).

[26] Niruntasukrat, A., Issariyapat, C., Pongpaibool, P., Meesublak, K., Aiumsupucgul, P., & Panya, A. (2016, May). Authorization mechanism for mqtt-based internet of things. In *Communications Workshops (ICC), 2016 IEEE International Conference on* (pp. 290-295). IEEE.

[27] Andy, S., Rahardjo, B., & Hanindhito, B. (2017, September). Attack scenarios and security analysis of MQTT communication protocol in IoT system. In *Electrical Engineering, Computer Science and Informatics (EECSI), 2017 4th International Conference on* (pp. 1-6). IEEE.

9.1. Δικτυακοί τόποι

[Δ.1] https://www.raspberrypi.org/documentation/

[Δ.2] https://www.arduino.cc/en/Main/Software

[Δ.3] http://easycoding.tn/

[Δ.4] https://nodered.org/

[Δ.5] https://mosquitto.org/

[Δ.6] http://mqtt.org/

[Δ.7] https://portforward.com/

[Δ.8] https://internetofhomethings.com

[Δ.9] http://www.instructables.com/id/Installing-MQTT-BrokerMosquitto-on-Raspberry-Pi/

[Δ.10] https://www.raspberrypi.org/products/raspberry-pi-3-model-b/

[Δ.11] https://scratch.mit.edu/

[Δ.12] http://pdacontrolen.com/installation-node-red-dashboard/

[Δ.13] https://www.hivemq.com/blog/introducing-the-mqtt-security-fundamentals

[Δ.14] https://hackaday.com/2017/06/20/practical-iot-cryptography-on-the-espressif-esp8266/

[Δ.15] http://aes.online-domain-tools.com/

[Δ.16] https://nodered.org/docs/hardware/raspberrypi

[Δ.17] https://www.dexterindustries.com/howto/run-a-program-on-your-raspberry-pi-at-startup/

[Δ.18] https://www.raspberrypi.org/documentation/configuration/wireless/access-point.md

[Δ.19] https://www.esp8266.com/wiki/doku.php?id=esp8266-module-family#esp-01

9.2. Manuals-Datasheets

[M.1] Πολυπλέκτης 4051 - http://www.ti.com/product/CD4051B (Τελευταία πρόσβαση 7/5/2018)

[M.2] Αισθητήρας θερμοκρασίας LM35 - http://www.ti.com/lit/ds/symlink/lm35.pdf (Τελευταία πρόσβαση 9/5/2018)

[M.3] Αυτόνομο τροφοδοτικό 220V AC – 5V DC HLK-PM01 http://img.filipeflop.com/files/download/Datasheet-HLK-PM01.pdf (Τελευταία πρόσβαση 7/5/2018)

[M.4] Ηλεκτρονόμος 5V - https://www.e-radionica.com/productdata/20085271543431001.pdf (Τελευταία πρόσβαση 7/5/2018)

[M.5] Μετασχηματιστής Έντασης Ρεύματος (CT) - https://www.elecrow.com/download/SCT013-000_datasheet.pdf (Τελευταία πρόσβαση 7/5/2018)

[M.6] Γέφυρα ανόρθωσης DB104 - http://www.rectron.com/data_sheets/db101-107.pdf (Τελευταία πρόσβαση 10/5/2018)

[M.7] Φωτοηλεκτρικός Αισθητήρας καπνού - http://www.chinesechip.com/chipFile/2015-07/BL5910-15551-0.pdf (Τελευταία πρόσβαση 19/5/2018)

[M.8] Ultrasonic Αισθητήρα HC-SR04 - https://elecfreaks.com/estore/download/EF03085-HC-SR04_Ultrasonic_Module_User_Guide.pdf (Τελευταία πρόσβαση 19/5/2018)

[M.9] OP Amp 741 - http://www.ti.com/lit/ds/symlink/lm741.pdf

[M.10] 8 channel ADC - https://cdn-shop.adafruit.com/datasheets/MCP3008.pdf

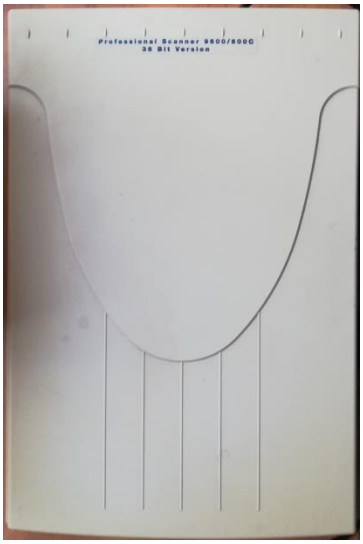
[M.11] AT Command Interface Specification - http://download-c.huawei.com/download/downloadCenter?downloadId=18427

[M.12] Quick Start Guide Vodafone Mobile Broadband USB Stick K3770 & K3771 - https://www.cleancss.com/user-manuals/QIS/K3770

10. Παραρτήματα

10.1. Παράρτημα 1 - Διαδικασία έκθεσης σε υπεριώδη ακτινοβολία

Για τις ανάγκες της εργασίας δημιουργήθηκε ιδιοκατασκευή, θάλαμος έκθεσης σε υπεριώδης ακτινοβολίας (UV). Χρησιμοποιήθηκε ένας παλιός σαρωτής και τοποθετήθηκαν μέσα δύο (2) λάμπες LED 6000K. Στο φάσμα ακτινοβολίας αυτής υπήρχε και αρκετή υπεριώδη ακτινοβολία προκειμένου να αποτυπωθεί το τυπωμένο κύκλωμα στην φωτοευαίσθητη επιφάνεια. Ο απαιτούμενος χρόνος έκθεσης ήταν δέκα (10) λεπτά.



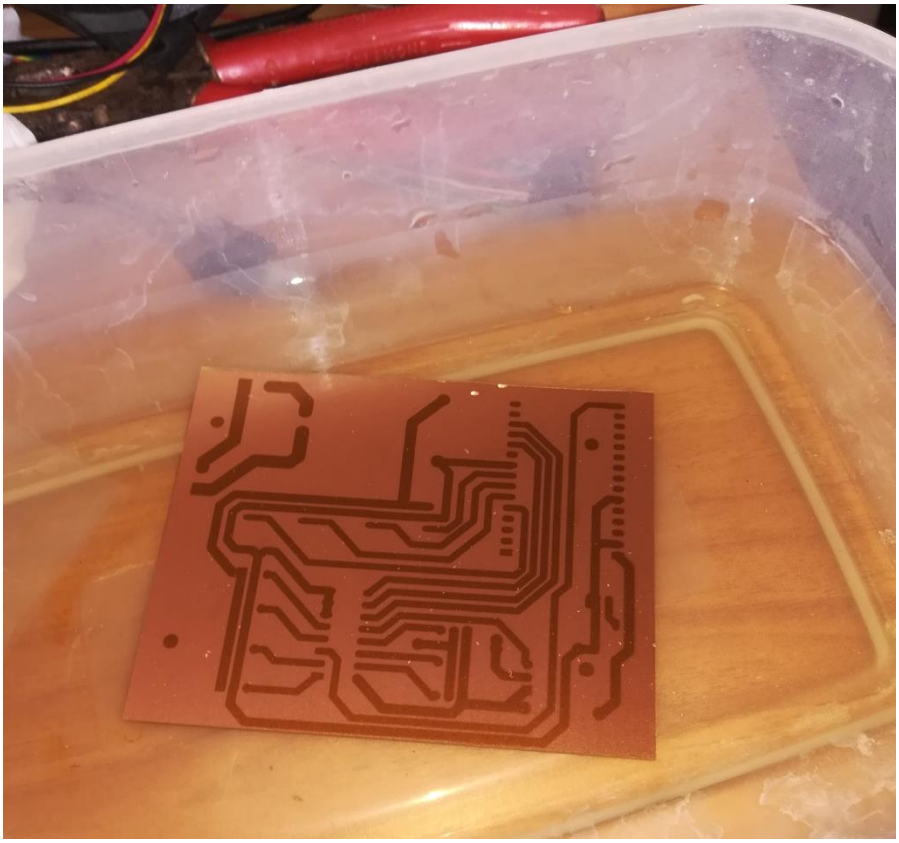
Εικόνα 43: Κουτί Παλιού Σαρωτή

Εικόνα 44 : Θάλαμος Έκθεσης UV ακτινοβολίας

10.2. Παράρτημα 2 – Εμφάνιση φωτοευαίσθητης επιφάνειας

Η εμφάνιση έγινε μέσα σε αραιό διάλυμα υδροξειδίου του νατρίου ή καυστικής σόδας (NaOH)

(Αναλογία: 1 κουταλιά γλυκού σε 500 ml νερού)



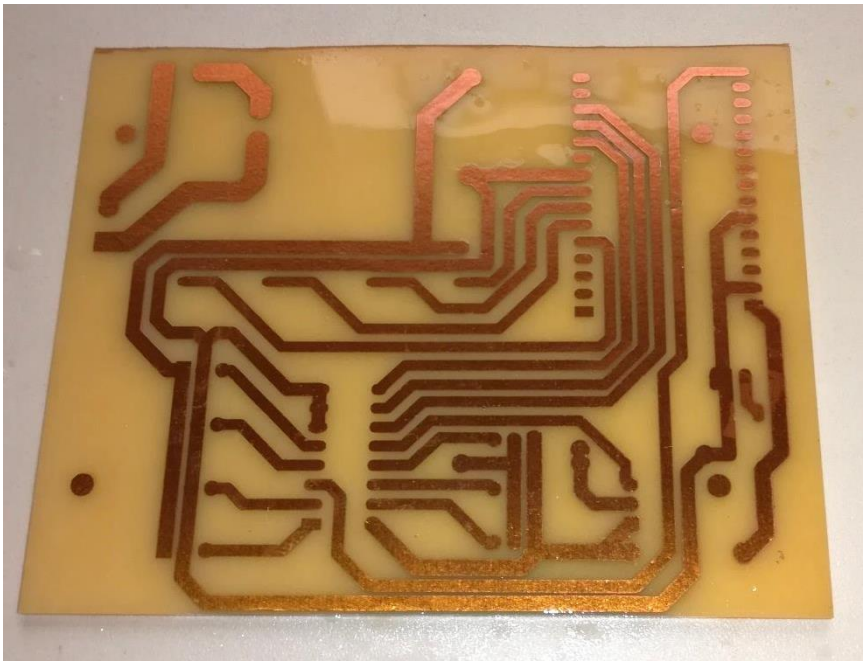
Εικόνα 45: Εμφάνιση Φωτοευαίσθητης Επιφάνειας

10.3. Παράρτημα 3 – Διαδικασία αποχάλκωσης πλακέτας

Χρησιμοποιήθηκαν δύο τρόποι:

1ος τρόπος: σε ζεστό διάλυμα τριχλωριούχου σιδήρου (FeCl3 * 6H2O)

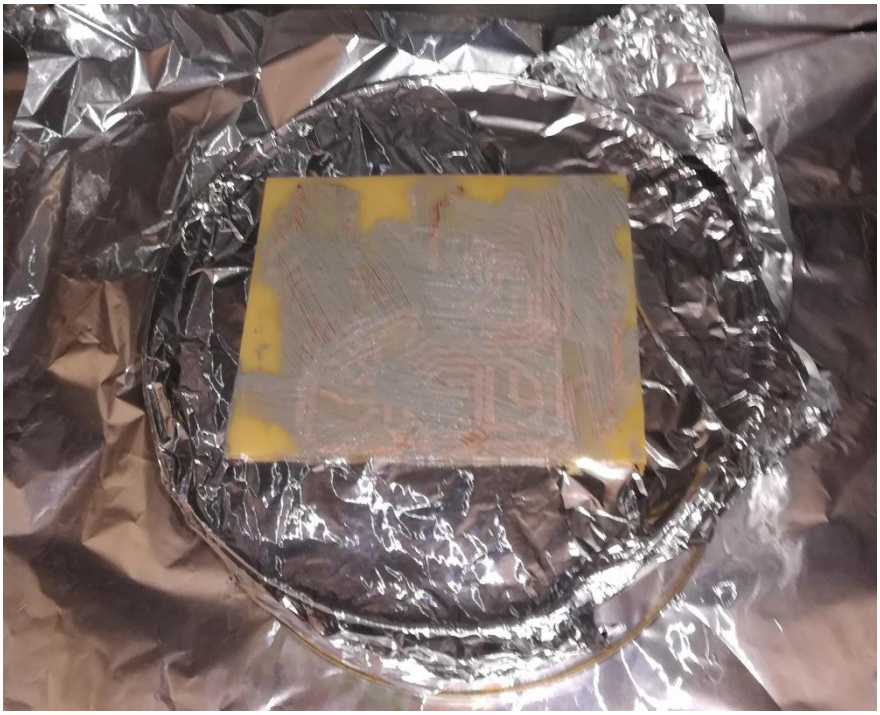
2ος τρόπος: σε διάλυμα υδροχλωρικού οξέος (HCL) και υπεροξείδιο υδρογόνου (H2O2)



Εικόνα 46: Αποχαλκωμένη Πλακέτα

10.4. Παράρτημα 4 – Επικασσιτέρωση χαλκού πλακέτας

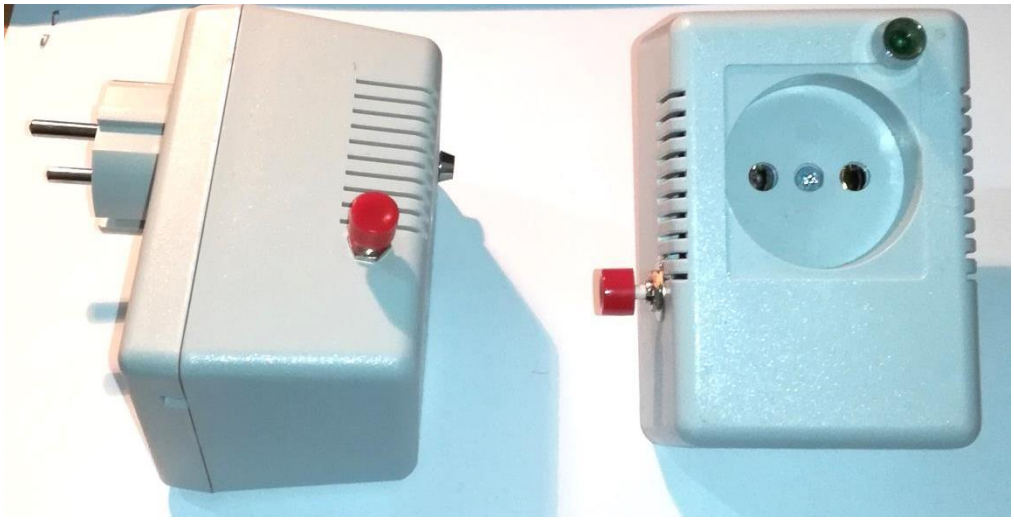
Η διαδικασία της επικασσιτέρωσης, έχει σκοπό τη βελτίωση του κυκλώματος και την προστασία του χαλκού από οξείδωση που μπορεί να προκληθεί με την πάροδο του χρόνου. Η διαδικασία έχει ως εξής, καθαρισμό του χαλκού με ισοπροπυλική αλκοόλη ή οινόπνευμα, επάλειψη με ειδική αλοιφή που περιέχει κασσίτερο και ψήσιμο για τέσσερα (4) λεπτά στους 230oC. Στην περίπτωσή μας χρησιμοποιήθηκε το ROSOL 3 της ROTHENBERGER.



Εικόνα 47: Επικασσιτέρωση με ROSOL 3 στους 230 ^{o}C

10.5. Παράρτημα 5 – Αυτόνομος Ρευματοδότης

Εικόνα 48: Αυτόνομος Ηλεκτρονόμος – Πρίζα Εσωτερικό



Εικόνα 49: Πρίζα Πλάγια Όψη & Πρόσοψη

10.5.1. Κώδικας Firmware

```
//Βιβλιοθήκη διασύνδεσης wifi
#include <ESP8266WiFi.h>
//Βιβλιοθήκη Δημοσίευσης & Εγγραφής σε MQTT Broker
#include "PubSubClient.h"
//Βιβλιοθήκη σύνδεσης πελάτη HTTP
#include <ESP8266HTTPClient.h>
//Βιβλιοθήκη ενημέρωσης Firmware μέσω HTTP
#include <ESP8266httpUpdate.h>
//Βιβλιοθήκη χρήσης EEPROM
#include <EEPROM.h>
//Βιβλιοθήκη Debounce
#include <Bounce2.h>

// Ορίσμός στοιχείων σύνδεσης στο Ασύρματο Δίκτυο
const char* ssid = "manos";
const char* password = "coucoutsi";
const char* mqtt_server = "192.168.2.85";


WiFiClient espClient;
PubSubClient client(espClient);

long lastMsg = 0;
char msg[75];

//Ορισμός Ταυτότητας του Node
const char* ID="01";

//Instantiate a Bounce object :
Bounce debouncer = Bounce();
int relay_pin = 2;
int button_pin = 0;
bool relayState = LOW;

//Το κείμενο που στέλνει το Node όταν σταλεί στο Node η Εντολή HELP
char Help1[100]="\nBROAD\nIDHELP\nIDPING\nIDFIRMWARE\nIDUPDATE\nIDON\nIDOFF";
char Help2[100]="\nCommand Help use Command? e.q BROAD?";

//Ορισμός της τρέχουσας έκδοσης Firmware
```

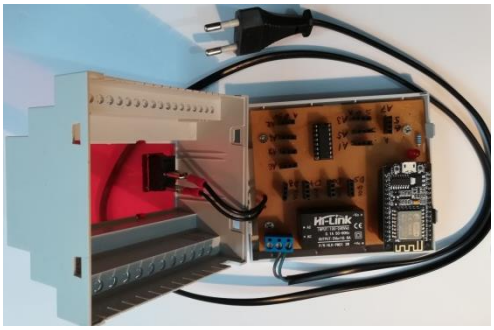
```
//Στέλνεται από το Node όταν δεχτεί την εντολή FIRMWARE
char firmware[20]="1.3";

//Σύνδεση στο ασύρματο Δίκτυο
void setup_wifi() {
  delay(10);
//  Serial.print("Connecting to ");
//  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
//    Serial.print(".");
  }
  randomSeed(micros());
//  Serial.println("");
//  Serial.println("WiFi connected");
//  Serial.println("IP address: ");
//  Serial.println(WiFi.localIP());
}

//Η ρουτίνα callback διαβάσει το inTopic ώστε να λάβει εντολή το Node.
//Ανάλογα με την εντολή που διαβάζει πράττει αναλόγως
void callback(char* topic, byte* payload, unsigned int length)
{
//BROADCAST
//Εάν διαβάσει ένα Node BROAD στέλνει την ταυτότητα του και ενα OK στο outTopic
//Αποτέλεσμα της εντολής να ανταποκριθούν όλα τα Nodes που είναι συνδεδεμένα στο σύστημα
// να ανταποκριθούν και ο χρήστης διαπιστώνει ποια Nodes λειτουργούν.
//Αυτή είναι διαχειριστική εντολή και ανταποκρίνονται όλα τα Nodes.
if (payload[0] == 'B' && payload[1] == 'R' && payload[2] == 'O' && payload[3] == 'A' &&
payload[4] == 'D'  )
   {
         snprintf (msg, 75, "%c%c:OK",ID[0],ID[1]);
//          Serial.print("Publish message: ");
//          Serial.println(msg);
        client.publish("outTopic", msg);
   }
//     Serial.print("Message arrived [");
//     Serial.print(topic);
//     Serial.print("] ");
//     for (int i = 0; i < length; i++) {
//     Serial.print((char)payload[i]);
//     }
//   Serial.println();

//Εάν τα δύο πρώτα ψηφία της εντολής που διαβάστηκε έιναι ίσα με την ταυτότητα του Node
//τότε ανταποκρίνεται μόνο το συγκεκριμένο Node.
//Διαβάζεται σε ποιό Node απευθύνεται η εντολή
if (payload[0] == ID[0] && payload[1] == ID[1])
   {
//Εάν διαβαστεί η εντολή 01UPDATE σημαίνει ότι το Node 01 πρέπει να κάνει HTTP update από διεύθυνση που του ορίζουμε
//Επειδή το ESP-01 δεν έχει αρκετά μεγάλη μνήμη ώστε να κάνει HTTP Update χρησιμοποιείται ένα trick
//Κάνει UPDATE το LOADER Firmware, το οποίο δεν έχει καθόλου κώδικα.
//Ο LOADER έχει τον κώδικα με το αρχείο με το νεο firmware που πρέπει να αναβαθμιστεί.
//Έτσι δεν χρειάζεται να βγει το ESP από το κουτί που έχει τοποθετηθεί.
if (payload[2] == 'U' && payload[3] == 'P' && payload[4] == 'D' && payload[5] == 'A' &&
payload[6] == 'T' && payload[7] == 'E' )
     {
```

```
      //snprintf (msg, 75,"http://garefalakis.eu/esp01-01-Loader.bin", ID[0],ID[1] );
      snprintf (msg, 75,"http://192.168.2.85/esp12-%c%c-Loader.bin", ID[0],ID[1] );
      t_httpUpdate_return ret = ESPhttpUpdate.update(msg);

      //t_httpUpdate_return  ret = ESPhttpUpdate.update("https://server/file.bin");
      switch(ret) {
        case HTTP_UPDATE_FAILED:
//            Serial.printf("HTTP_UPDATE_FAILD Error (%d): %s", ESPhttpUpdate.getLastError(),
ESPhttpUpdate.getLastErrorString().c_str());
            snprintf (msg, 75, "%c%c:HTTP_UPDATE_FAILD Error (%d): %s",ID[0],ID[1],
ESPhttpUpdate.getLastError(), ESPhttpUpdate.getLastErrorString().c_str());
            client.publish("outTopic", msg);
            break;
        case HTTP_UPDATE_NO_UPDATES:
//            Serial.println("HTTP_UPDATE_NO_UPDATES");
            snprintf (msg, 75, "%c%c:HTTP_UPDATE_NO_UPDATES",ID[0],ID[1]);
            client.publish("outTopic", msg);
            break;
        case HTTP_UPDATE_OK:
//            Serial.println("HTTP_UPDATE_OK");
            snprintf (msg, 75, "%c%c:HTTP_UPDATE_OK",ID[0],ID[1]);
            client.publish("outTopic", msg);
            break;
      }
    }
//FIRMWARE
//Εάν διαβαστεί η εντολή 01FIRMWARE σημαίνει ότι το Node 01 πρέπει να στείλει το firmware
του.
if (payload[2] == 'F' && payload[3] == 'I' && payload[4] == 'R' && payload[5] == 'M' && payload[6]
== 'W' && payload[7] == 'A' && payload[8] == 'R' && payload[9] == 'E')
      {
       snprintf (msg, 75,"%c%c:%s",ID[0],ID[1],firmware);
       client.publish("outTopic", msg);
      }
//HELP
//Εάν διαβαστεί η εντολή 01HELP σημαίνει ότι το Node 01 πρέπει να στείλει το κείμενο βοήθειας
του  Help1[100] και char Help2[100]
if (payload[2] == 'H' && payload[3] == 'E' && payload[4] == 'L' && payload[5] == 'P')
      {
       snprintf (msg, 110,"%c%c:%s",ID[0],ID[1],Help1);
       client.publish("outTopic", msg);
       snprintf (msg, 110,"%c%c:%s",ID[0],ID[1],Help2);
       client.publish("outTopic", msg);
    }
//PING
//Εάν διαβαστεί η εντολή 01PING σημαίνει ότι το Node 01 πρέπει να απανατήσει με ένα
01:PING_OK
//ώστε ο χρήστης να διαπιστώσει ότι το Node λειτουργεί
if (payload[2] == 'P' && payload[3] == 'I' && payload[4] == 'N' && payload[5] == 'G' )
      {
            snprintf (msg, 75, "%c%c:PING_OK",ID[0],ID[1]);
            client.publish("outTopic", msg);
      }
//ON
//Εάν διαβαστεί η εντολή 01ON σημαίνει ότι το Node 01 πρέπει να ενεργοποιήσει τον
ηλεκτρονόμο του.
if (payload[2] == 'O' && payload[3] == 'N' )
       {
        digitalWrite(2, 1);   // Turn the LED on (Note that LOW is the voltage level
        snprintf (msg, 75, "%c%c:Relay ON:",ID[0],ID[1]);
```

```
//       Serial.println(msg);
         client.publish("outTopic", msg);
         EEPROM.write(0, 1);    // Write state to EEPROM
         EEPROM.commit();
       }
//OFF
//Εάν διαβαστεί η εντολή 01ON σημαίνει ότι το Node 01 πρέπει να απενεργοποιήσει τον ηλεκτρονόμο του.
if   (payload[2] == 'O' && payload[3] == 'F' && payload[4] == 'F' )
       {
         digitalWrite(2, 0);  // Turn the LED off by making the voltage HIGH
         snprintf (msg, 75, "%c%c:Relay OFF:",ID[0],ID[1]);
//         Serial.println(msg);
         client.publish("outTopic", msg);
         EEPROM.write(0, 0);    // Write state to EEPROM
         EEPROM.commit();
       }
   }
}
//Η ρουτίνα αυτή αντιστρέφει τη κατάσταση του ηλεκτρονόμου μόλις πατηθεί το μπουτόν στο ρευματοδότη
//Επίσης αποθηκεύει την κατάσταση του ρευματοδότη στην EEPROM ώστε αν διακοπή η τάση
// ο ρευματοδότης να επανέλθει στην κατάσταση που βρισκόταν κατά τη διακοπή τροφοδοσίας
void extButton() {
  debouncer.update();
   // Call code if Bounce fell (transition from HIGH to LOW) :
   if ( debouncer.fell() ) {
//     Serial.println("Debouncer fell");
     // Toggle relay state :
     relayState = !relayState;
     digitalWrite(2,relayState);
     EEPROM.write(0, relayState);    // Write state to EEPROM
     if (relayState == 1){
         snprintf (msg, 75, "%c%c:Relay ON",ID[0],ID[1]);
//         Serial.println(msg);
         client.publish("outTopic", msg);
     }
     else if (relayState == 0){
         snprintf (msg, 75, "%c%c:Relay OFF",ID[0],ID[1]);
//         Serial.println(msg);
         client.publish("outTopic", msg);
     }
   }
}

//Ρουτίνα επανασύνδεσης του Node στον MQTT Broker/Server σε περίπτωση αποσύνδεσης
void reconnect()
{
  // Loop until we're reconnected
  while (!client.connected())
    {
//    Serial.print("Attempting MQTT connection...");
    // Create a random client ID
    String clientId = "ESP8266Client-";
    clientId += String(random(0xffff), HEX);
    // Attempt to connect
    if (client.connect(clientId.c_str()))
    {
//      Serial.println("connected");
      // Once connected, publish an announcement...
```

```
      client.publish("outTopic", "hello from ESP-01");
      // ... and resubscribe
      client.subscribe("inTopic");
      snprintf (msg, 75, "%c%c:Relay Connected", ID[0],ID[1]);
//      Serial.print("Publish message: ");
//      Serial.println(msg);
      client.publish("outTopic", msg);
     }
    else
     {
//      Serial.print("failed, rc=");
//      Serial.print(client.state());
//      Serial.println(" try again in 5 seconds");
      // Wait 5 seconds before retrying
      delay(5000);
     }
    }
 }

//Setup/Αρχικοποίηση του Node
void setup()
{
  Serial.begin(115200);
//  Serial.println("Booting");
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);
 // pinMode(BUILTIN_LED, OUTPUT);      // Initialize the BUILTIN_LED pin as an output
  EEPROM.begin(300);                  // Begin eeprom to store on/off state
  pinMode(2, OUTPUT);               // Initialize the relay pin as an output
  pinMode(0, INPUT);                  // Initialize the relay pin as an output
  relayState = EEPROM.read(0);
  digitalWrite(2,relayState);
  debouncer.attach(button_pin);       // Use the bounce2 library to debounce the built in button
  debouncer.interval(50);             // Input must be low for 50 ms
  setup_wifi();
  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);
}
//Ρουτίνα επανάληψης του Node
void loop()
{
  if (!client.connected())
  {
    reconnect();
  }
  client.loop();
  extButton();
}
```

10.6. Παράρτημα 6 – Κατασκευή για Ηλεκτρικό Πίνακα



Εικόνα 50 : ESP-12 σε Κουτί για Ηλ. Πίνακα

Εικόνα 51: ESP-12 σε Κουτί για Ηλ. Πίνακα

Εικόνα 52: ESP-12 σε Ηλ. Πίνακα

Εικόνα 53:ESP-12 σε Ηλ. Πίνακα

10.6.1. Κώδικας Firmware

```
//Βιβλιοθήκη διασύνδεσης wifi
#include <ESP8266WiFi.h>
//Βιβλιοθήκη Δημοσίευσης & Εγγραφής σε MQTT Broker
#include "PubSubClient.h"
//Βιβλιοθήκη σύνδεσης πελάτη HTTP
#include <ESP8266HTTPClient.h>
//Βιβλιοθήκη ενημέρωσης Firmware μέσω HTTP
#include <ESP8266httpUpdate.h>
//Βιβλιοθήκη χρήσης EEPROM
#include <EEPROM.h>

//Ενεργοποίηση DEBUG Mode.
#define DEBUG
//Ενεργοποίηση DEBUGMQTT Mode.
//#define DEBUGMQTT
//Ενεργοποίηση Ρύθμισης εισόδων και εξόδου μέσω κώδικα HARDCODED.
#define HARD_CODED_SETUP

// Τα PIN που επιλέγεται η Αναλογική θύρα που διαβάζεται
// από τον τον Πολυπλέκτη MC4051
#define MUX_A D4
#define MUX_B D3
#define MUX_C D2
#define IO5 D5
#define IO6 D6
#define IO7 D7
#define IO8 D8
//Ορισμός Αναλογικής θύρα που διαβάζεται
#define ANALOG_INPUT A0
#define BUILTIN_LED D1

// Update these with values suitable for your network.
// Ορίσμός στοιχείων σύνδεσης στο Ασύρματο Δίκτυο
const char* ssid = "manos";
const char* password = "coucoutsi";

//Ορισμός Διεύθυνσης του MQTT Server/Broker
const char* mqtt_server = "192.168.2.85";

WiFiClient espClient;
PubSubClient client(espClient);
long lastMsg = 0;
char msg[300];
int value[9];
//Το Node ρυθμίζεται σε Polling Mode
int FunctionMode=1;
//Ορισμός Ταυτότητας του Node
const char* ID="04";

const char* ModuleType="1";
//Το κείμενο που στέλνει το Node όταν σταλεί στο Node η Εντολή HELP
char Help1[100]="\nHELP\nBROAD\nIDPING\nIDSIOO\nIDSIOI\nIDFIRMWARE\nIDDXOH (X:5-
8)\nIDDXI(X:5-8)\nIDAX (A:1-8)\n";
char Help2[100]="\nCommand Help use Command? e.q BROAD?";

int DigiVal[9];
int DigiSetup[4];
int i=0;
//Ορισμός της τρέχουσας έκδοσης Firmware
```

```
//Στέλνεται από το Node όταν δεχτεί την εντολή FIRMWARE
char firmware[20]="1.3";

void setup_wifi() {
  //Σύνδεση στο ασύρματο Δίκτυο
  delay(10);
  // We start by connecting to a WiFi network
  #ifdef DEBUG
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
  #endif

  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    #ifdef DEBUG
    Serial.print(".");
    #endif
  }
  randomSeed(micros());
  #ifdef DEBUG
  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
  #endif
}
//Ρουτίνα που διαβάσει όλες τις Αναλογικές θύρες (8) και στέλνει τις
//τιμές τους στο outTopic
void GetAllAnalogs()
{
  changeMux(LOW, LOW, LOW);
      value[0] = analogRead(ANALOG_INPUT); //Value of the sensor connected Option 0 pin of Mux
      //snprintf (msg, 300, "Sensor 0:%ld", value);
      //Serial.print("Publish message: ");
      //Serial.println(msg);
      //client.publish("outTopic-01", msg);
      delay(1000);
      changeMux(LOW, LOW, HIGH);
      value[1] = analogRead(ANALOG_INPUT); //Value of the sensor connected Option 1 pin of Mux
      //snprintf (msg, 75, "Sensor 1:%ld", value);
      //Serial.print("Publish message: ");
      //  Serial.println(msg);
      //  client.publish("outTopic", msg);
      delay(1000);
      changeMux(LOW, HIGH, LOW);
      value[2] = analogRead(ANALOG_INPUT); //Value of the sensor connected Option 2 pin of Mux
      //snprintf (msg, 75, "Sensor 2:%ld", value);
      //Serial.print("Publish message: ");
      //  Serial.println(msg);
      //  client.publish("outTopic", msg);
      delay(1000);
      changeMux(LOW, HIGH, HIGH);
      value[3] = analogRead(ANALOG_INPUT); //Value of the sensor connected Option 3 pin of Mux
      //snprintf (msg, 75, "Sensor 3:%ld", value);
      //Serial.print("Publish message: ");
      //  Serial.println(msg);
      //  client.publish("outTopic", msg);
      delay(1000);
```

```
      changeMux(HIGH, LOW, LOW);
      value[4] = analogRead(ANALOG_INPUT); //Value of the sensor connected Option 4 pin of Mux
      //snprintf (msg, 75, "Sensor 4:%ld", value);
      //Serial.print("Publish message: ");
      //  Serial.println(msg);
      //  client.publish("outTopic", msg);
      delay(1000);
      changeMux(HIGH, LOW, HIGH);
      value[5] = analogRead(ANALOG_INPUT); //Value of the sensor connected Option 5 pin of Mux
      //snprintf (msg, 75, "Sensor 5:%ld", value);
      //Serial.print("Publish message: ");
      //  Serial.println(msg);
      //  client.publish("outTopic", msg);
      delay(1000);
      changeMux(HIGH, HIGH, LOW);
      value[6] = analogRead(ANALOG_INPUT); //Value of the sensor connected Option 6 pin of Mux
      //snprintf (msg, 75, "Sensor 6:%ld", value);
      //Serial.print("Publish message: ");
      //  Serial.println(msg);
      //  client.publish("outTopic", msg);
      delay(1000);
      changeMux(HIGH, HIGH, HIGH);
      value[7] = analogRead(ANALOG_INPUT); //Value of the sensor connected Option 7 pin of Mux
      //snprintf (msg, 75, "Sensor 7:%ld", value);
      snprintf (msg, 75,
"ID:%c%c:%ld:%ld:%ld:%ld:%ld:%ld:%ld:%ld",ID[0],ID[1],value[0],value[1],value[2],value[3],value[4],value[5],
value[6],value[7]);
      Serial.print("Publish message: ");
      Serial.println(msg);
      client.publish("outTopic", msg);
      delay(1000);
}
//Ρουτίνα που ρυθμίζει τα pins IO5 έως IO8 ως εξόδους
void Set_IO_pins_output()
{
  pinMode(IO5, OUTPUT);
  EEPROM.write(5,1);
  pinMode(IO6, OUTPUT);
  EEPROM.write(6,1);
  pinMode(IO7, OUTPUT);
  EEPROM.write(7,1);
  pinMode(IO8, OUTPUT);
  EEPROM.write(7,1);
  snprintf (msg, 75, "%c%c:IO5-IO8 Set OUTPUT",ID[0],ID[1]);
  client.publish("outTopic",msg);
  EEPROM.commit();
}
//Ρουτίνα που ρυθμίζει τα pins IO5 έως IO8 ως εισόδους
void Set_IO_pins_input()
{
  pinMode(IO5, INPUT);
  EEPROM.write(5,0);
  pinMode(IO6, INPUT);
  EEPROM.write(6,0);
  pinMode(IO7, INPUT);
  EEPROM.write(7,0);
  pinMode(IO8, INPUT);
  EEPROM.write(7,0);
  snprintf (msg, 75, "%c%c:IO5-IO8 Set INPUT",ID[0],ID[1]);
  client.publish("outTopic",msg);
```

```
  EEPROM.commit();
}

void Setup_IO_pins()
{
  //Η μεταβλητή HARD_CODED_SETUP καθορίζει αν θα οριστούν τα PINS IO5-IO8
 // μέσω της EEPROM ή μέσω Κώδικα HARD CODED
    #ifdef HARD_CODED_SETUP
         pinMode(BUILTIN_LED, OUTPUT);     // Initialize the BUILTIN_LED pin as an output
         pinMode(IO5, OUTPUT);
         pinMode(IO6, INPUT);
         pinMode(IO7, INPUT);
         pinMode(IO8, OUTPUT);
    #else
        if (EEPROM.read(5)==1)
        {
        pinMode(IO5, OUTPUT);
        #ifdef DEBUGMQTT
        snprintf (msg, 75, "%c%c:IO5 Set OUTPUT:%s",ID[0],ID[1],EEPROM.read(5));
        client.publish("outTopic",msg);
        #endif
        }
        else
        {
        pinMode(IO5, INPUT);
        #ifdef DEBUGMQTT
        snprintf (msg, 75, "%c%c:IO5 Set INPUT:%s",ID[0],ID[1],EEPROM.read(5));
        client.publish("outTopic",msg);
        #endif
        }
        if (EEPROM.read(6)==1)
        {
        pinMode(IO6, OUTPUT);
        #ifdef DEBUGMQTT
        snprintf (msg, 75, "%c%c:IO6 Set OUTPUT:%s",ID[0],ID[1],EEPROM.read(6));
        client.publish("outTopic",msg);
        #endif
        }
        else
        {
        pinMode(IO6, INPUT);
        #ifdef DEBUGMQTT
        snprintf (msg, 75, "%c%c:IO6 Set INPUT:%s",ID[0],ID[1],EEPROM.read(6));
        client.publish("outTopic",msg);
        #endif
        }
        if (EEPROM.read(7)==1)
        {
        pinMode(IO7, OUTPUT);
        #ifdef DEBUGMQTT
        snprintf (msg, 75, "%c%c:IO7 Set OUTPUT:%s",ID[0],ID[1],EEPROM.read(7));
        client.publish("outTopic",msg);
        #endif
        }
        else
        {
        pinMode(IO7, INPUT);
        #ifdef DEBUGMQTT
        snprintf (msg, 75, "%c%c:IO7 Set INPUT:%s",ID[0],ID[1],EEPROM.read(7));
        client.publish("outTopic",msg);
```

```
      #endif
      }
      if (EEPROM.read(8)==1)
      {
      pinMode(IO8, OUTPUT);
      #ifdef DEBUGMQTT
      snprintf (msg, 75, "%c%c:IO8 Set OUTPUT:%s",ID[0],ID[1],EEPROM.read(8));
      client.publish("outTopic",msg);
      #endif
      }
      else
      {
      pinMode(IO8, INPUT);
      #ifdef DEBUGMQTT
      snprintf (msg, 75, "%c%c:IO8 Set INPUT:%s",ID[0],ID[1],EEPROM.read(8));
      client.publish("outTopic",msg);
      #endif
      }
   #endif

}
// Επιλογή αναλογικής εισόδου A0-A7
void changeMux(int c, int b, int a) {
  digitalWrite(MUX_A, a);
  digitalWrite(MUX_B, b);
  digitalWrite(MUX_C, c);
}
//Η ρουτίνα callback διαβάσει το inTopic ώστε να λάβει εντολή το Node.
//Ανάλογα με την εντολή που διαβάζει πράτει αναλόγως
void callback(char* topic, byte* payload, unsigned int length) {
  #ifdef DEBUG
    Serial.print("Message arrived [");
    Serial.print(topic);
    Serial.print("] ");
    for (int i = 0; i < length; i++)
      {
     Serial.print((char)payload[i]);
      }
 #endif
    //Broadcast Request
    //Εάν διαβάσει ένα Node BROAD στέλνει την ταυτότητα του και ενα OK στο outTopic
    //Αποτέλεσμα της εντολής να ανταποκριθούν όλα τα Nodes που είναι συνδεδεμένα στο σύστημα
    // να ανταποκριθούν και ο χρήστης διαπιστώνει ποια Nodes λειτουργούν.
    //Αυτή είναι διαχειριστική εντολή και ανταποκρίνονται όλα τα Nodes.
    if (payload[0] == 'B' && payload[1] == 'R' && payload[2] == 'O' && payload[3] == 'A' && payload[4] == 'D'  )
    {
          snprintf (msg, 75, "%c%c:OK",ID[0],ID[1]);
  #ifdef DEBUG
         Serial.print("Publish message: ");
         Serial.println(msg);
  #endif
          client.publish("outTopic", msg);
    }
//Εάν τα δύο πρώτα ψηφία της εντολής που διαβάστηκε έιναι ίσα με την ταυτότητα του Node
//τότε ανταποκρίνεται μόνο το συγκεκριμένο Node
if (payload[0] == ID[0]&& payload[1] == ID[1] )
{
//Εάν διαβαστεί η εντολή 01SIOO σημαίνει ότι το Node 01 πρέπει να θέσει (S) όλα τα IO pins (IO) ως εξόδους (O).
if (payload[2] == 'S' && payload[3] == 'I' && payload[4] == 'O' && payload[5] == 'O')
```

```
{
   Set_IO_pins_output();
}
```
//Εάν διαβαστεί η εντολή 01SIOI σημαίνει ότι το Node 01 πρέπει να θέσει (S) όλα τα IO pins (IO) ως εισόδους (I).
```
if (payload[2] == 'S' && payload[3] == 'I' && payload[4] == 'O' && payload[5] == 'I')
{
   Set_IO_pins_input();
}
```
//Εάν διαβαστεί η εντολή 01HELP σημαίνει ότι το Node 01 πρέπει να στείλει το κείμενο βοήθειας του Help1[100] και char Help2[100]
```
if (payload[2] == 'H' && payload[3] == 'E' && payload[4] == 'L' && payload[5] == 'P')
      {
       snprintf (msg, 110,"%c%c:%s",ID[0],ID[1],Help1);
       client.publish("outTopic", msg);
       snprintf (msg, 110,"%c%c:%s",ID[0],ID[1],Help2);
       client.publish("outTopic", msg);
    }
```
//Εάν διαβαστεί η εντολή 01FIRMWARE σημαίνει ότι το Node 01 πρέπει να στείλει το firmware του.
```
if (payload[2] == 'F' && payload[3] == 'I' && payload[4] == 'R' && payload[5] == 'M' && payload[6] == 'W' &&
payload[7] == 'A' && payload[8] == 'R' && payload[9] == 'E')
      {
       snprintf (msg, 75,"%c%c:%s",ID[0],ID[1],firmware);
       client.publish("outTopic", msg);
    }
```
//Εάν διαβαστεί η εντολή 01UPDATE σημαίνει ότι το Node 01 πρέπει να κάνει HTTP update από διεύθυνση που του ορίζουμε
//Στην συγκεκριμένη περίπτωση κάνει firmware update το ίδιο το RASPBERRY
```
if (payload[2] == 'U' && payload[3] == 'P' && payload[4] == 'D' && payload[5] == 'A' && payload[6] == 'T' &&
payload[7] == 'E' )
      {
       snprintf (msg, 75,"%c%c:UPDATING", ID[0],ID[1] );
       client.publish("outTopic",msg );
       snprintf (msg, 75,"http://192.168.2.85/esp12-%c%c.bin", ID[0],ID[1] );
       //snprintf (msg, 75,"http://garefalakis.eu/esp12-04.bin");
       t_httpUpdate_return ret = ESPhttpUpdate.update(msg);
       //t_httpUpdate_return  ret = ESPhttpUpdate.update("https://server/file.bin");
       switch(ret) {
          case HTTP_UPDATE_FAILED:
//              Serial.printf("HTTP_UPDATE_FAILD Error (%d): %s", ESPhttpUpdate.getLastError(),
ESPhttpUpdate.getLastErrorString().c_str());
             snprintf (msg, 75, "%c%c:HTTP_UPDATE_FAILD Error (%d): %s",ID[0],ID[1],
ESPhttpUpdate.getLastError(), ESPhttpUpdate.getLastErrorString().c_str());
             client.publish("outTopic", msg);
             break;
//             Serial.println("HTTP_UPDATE_NO_UPDATES");
             snprintf (msg, 75, "%c%c:HTTP_UPDATE_NO_UPDATES",ID[0],ID[1]);
             client.publish("outTopic", msg);
             break;
          case HTTP_UPDATE_OK:
//             Serial.println("HTTP_UPDATE_OK");
             snprintf (msg, 75, "%c%c:HTTP_UPDATE_OK",ID[0],ID[1]);
             client.publish("outTopic", msg);
             break;
       }
    }
```
//Εάν διαβαστεί η εντολή 01PING σημαίνει ότι το Node 01 πρέπει να απανατήσει με ένα 01:PING_OK
//ώστε ο χρήστης να διαπιστώσει ότι το Node λειτουργεί
```
if (payload[2] == 'P' && payload[3] == 'I' && payload[4] == 'N' && payload[5] == 'G' )
      {
```

```
            snprintf (msg, 75, "%c%c:PING_OK",ID[0],ID[1]);
            client.publish("outTopic", msg);
        }
// M0 Continuous Data Sending
//Εάν διαβαστεί η εντολή 01MO σημαίνει ότι το Node 01 πρέπει να γυρίσει για Continuous Mode
//Διαβάζει τις εισόδους και στέλνει όλες τις τιμές σε σταθερό χρόνο
    if (payload[2] == 'M' && payload[3] == '0')
    {
       FunctionMode=0;
       snprintf (msg, 75, "%c%c:FunctionMode:Continuous",ID[0],ID[1]);
       client.publish("outTopic", msg);
       #ifdef DEBUG
       Serial.print("Publish message: ");
       Serial.println(msg);
       #endif
    }
// MODE Selection
//Εάν διαβαστεί η εντολή 01M1 σημαίνει ότι το Node 01 πρέπει να γυρίσει για Polling Mode
//Διαβάζει συγκεκριμένη είσοδο όταν ζητηθεί
// M1 Polling Data
    if (payload[2] == 'M' && payload[3] == '1')
    {
       FunctionMode=1;
       snprintf (msg, 75, "%c%c:FunctionMode:Polling",ID[0],ID[1]);
       client.publish("outTopic", msg);
       #ifdef DEBUG
       Serial.print("Publish message: ");
       Serial.println(msg);
       #endif
    }
//Εάν διαβαστεί η εντολή 01S5O σημαίνει ότι το Node 01 πρέπει να θέσει το IO5 ως έξοδο
  if (payload[2] == 'S' && payload[3] == '5' && payload[4] == 'O')
  {
       pinMode(IO5, OUTPUT);
       EEPROM.write(0,1);
       EEPROM.commit();
       #ifdef DEBUG
       Serial.print("\nD5 Set Output");
       #endif
  }
//Εάν διαβαστεί η εντολή 01S5I σημαίνει ότι το Node 01 πρέπει να θέσει το IO5 ως είσοδο
  if (payload[2] == 'S' && payload[3] == '5' && payload[4] == 'I')
  {
       pinMode(IO5, INPUT);
       EEPROM.write(0,0);
       EEPROM.commit();
       #ifdef DEBUG
       Serial.print("\nD5 Set Input");
       #endif
  }
//Εάν διαβαστεί η εντολή 01S6O σημαίνει ότι το Node 01 πρέπει να θέσει το IO6 ως έξοδο
  if (payload[2] == 'S' && payload[3] == '6' && payload[4] == 'O')
    {
       pinMode(IO6, OUTPUT);
       EEPROM.write(1,1);
       EEPROM.commit();
       #ifdef DEBUG
       Serial.print("\nD6 Set Output");
       #endif
    }
```

```
//Εάν διαβαστεί η εντολή 01S6I σημαίνει ότι το Node 01 πρέπει να θέσει το IO6 ως είσοδο
  if (payload[2] == 'S' && payload[3] == '6' && payload[4] == 'I')
     {
      pinMode(IO6, INPUT);
      EEPROM.write(1,0);
      EEPROM.commit();
      #ifdef DEBUG
      Serial.print("\nD6 Set Input");
      #endif
     }
//Εάν διαβαστεί η εντολή 01S7O σημαίνει ότι το Node 01 πρέπει να θέσει το IO7 ως έξοδο
    if (payload[2] == 'S' && payload[3] == '7' && payload[4] == 'O')
    {
      pinMode(IO7, OUTPUT);
      EEPROM.write(2,1);
      EEPROM.commit();
      #ifdef DEBUG
      Serial.print("\nD7 Set Output");
      #endif
    }
//Εάν διαβαστεί η εντολή 01S7I σημαίνει ότι το Node 01 πρέπει να θέσει το IO7 ως είσοδο
     if (payload[2] == 'S' && payload[3] == '7' && payload[4] == 'I')
      {
      pinMode(IO7, INPUT);
      EEPROM.write(2,0);
      EEPROM.commit();
      #ifdef DEBUG
      Serial.print("\nD7 Set Input");
      #endif
     }
//Εάν διαβαστεί η εντολή 01S8O σημαίνει ότι το Node 01 πρέπει να θέσει το IO8 ως έξοδο
    if (payload[2] == 'S' && payload[3] == '8' && payload[4] == 'O')
    {
      pinMode(IO8, OUTPUT);
      EEPROM.write(3,1);
      EEPROM.commit();
      #ifdef DEBUG
      Serial.print("\nD8 Set Output");
      #endif
    }
//Εάν διαβαστεί η εντολή 01S8I σημαίνει ότι το Node 01 πρέπει να θέσει το IO8 ως είσοδο
    if (payload[2] == 'S' && payload[3] == '8' && payload[4] == 'I')
     {
      pinMode(IO8, INPUT);
      EEPROM.write(3,0);
      EEPROM.commit();
      #ifdef DEBUG
      Serial.print("\nD8 Set Input");
      #endif
     }
}
//Εάν διαβαστεί η εντολή 01A1 σημαίνει ότι το Node 01 πρέπει να διαβάσει την είσοδο A1
if (payload[0] == ID[0] && payload[1] == ID[1])
{
      if (payload[2] == 'A' && payload[3] == '1')
       {
       changeMux(LOW, LOW, LOW);
       value[1] = analogRead(ANALOG_INPUT);
       snprintf (msg, 75, "%c%c:A1:%ld",ID[0],ID[1], value[1]);
       client.publish("outTopic", msg);
```

```
      #ifdef DEBUG
      Serial.println("\nA1 Request Received");
      Serial.print("\nPublish message: ");
      Serial.println(msg);
      #endif
      }
//Εάν διαβαστεί η εντολή 01A2 σημαίνει ότι το Node 01 πρέπει να διαβάσει την είσοδο A2
      if (payload[2] == 'A' && payload[3] == '2')
      {
      changeMux(LOW, LOW, HIGH);
      value[2] = analogRead(ANALOG_INPUT);
      snprintf (msg, 75, "%c%c:A2:%ld",ID[0],ID[1], value[2]);
      client.publish("outTopic", msg);
      #ifdef DEBUG
      Serial.println("\nA2 Request Received");
      Serial.print("\nPublish message: ");
      Serial.println(msg);
      #endif
      }
//Εάν διαβαστεί η εντολή 01A3 σημαίνει ότι το Node 01 πρέπει να διαβάσει την είσοδο A3
      if (payload[2] == 'A' && payload[3] == '3')
      {
      changeMux(LOW, HIGH, LOW);
      value[3] = analogRead(ANALOG_INPUT);
      snprintf (msg, 75, "%c%c:A3:%ld",ID[0],ID[1], value[3]);
      client.publish("outTopic", msg);
      #ifdef DEBUG
      Serial.println("\nA3 Request Received");
      Serial.print("\nPublish message: ");
      Serial.println(msg);
      #endif
      }
//Εάν διαβαστεί η εντολή 01A4 σημαίνει ότι το Node 01 πρέπει να διαβάσει την είσοδο A4
      if (payload[2] == 'A' && payload[3] == '4')
      {
        changeMux(LOW, HIGH, HIGH);
        value[4] = analogRead(ANALOG_INPUT);
        snprintf (msg, 75, "%c%c:A4:%ld",ID[0],ID[1], value[4]);
        client.publish("outTopic", msg);
        #ifdef DEBUG
        Serial.println("\nA4 Request Received");
        Serial.print("\nPublish message: ");
        Serial.println(msg);
        #endif
      }
//Εάν διαβαστεί η εντολή 01A5 σημαίνει ότι το Node 01 πρέπει να διαβάσει την είσοδο A5
      if (payload[2] == 'A' && payload[3] == '5')
      {
        changeMux(HIGH, LOW, LOW);
        value[5] = analogRead(ANALOG_INPUT);
        snprintf (msg, 75, "%c%c:A5:%ld",ID[0],ID[1], value[5]);
        client.publish("outTopic", msg);
        #ifdef DEBUG
        Serial.println("\nA5 Request Received");
        Serial.print("\nPublish message: ");
        Serial.println(msg);
        #endif
      }
//Εάν διαβαστεί η εντολή 01A6 σημαίνει ότι το Node 01 πρέπει να διαβάσει την είσοδο A6
      if (payload[2] == 'A' && payload[3] == '6')
```

```
    {
     changeMux(HIGH, LOW, HIGH);
     value[6] = analogRead(ANALOG_INPUT);
     snprintf (msg, 75, "%c%c:A6:%ld",ID[0],ID[1], value[6]);
     client.publish("outTopic", msg);
     #ifdef DEBUG
     Serial.println("\nA6 Request Received");
     Serial.print("\nPublish message: ");
     Serial.println(msg);
     #endif
    }
//Εάν διαβαστεί η εντολή 01A7 σημαίνει ότι το Node 01 πρέπει να διαβάσει την είσοδο A7
    if (payload[2] == 'A' && payload[3] == '7')
    {
     changeMux(HIGH, HIGH, LOW);
     value[7] = analogRead(ANALOG_INPUT);
     snprintf (msg, 75, "%c%c:A7:%ld",ID[0],ID[1], value[7]);
     client.publish("outTopic", msg);
     #ifdef DEBUG
     Serial.println("\nA7 Request Received");
     Serial.print("\nPublish message: ");
     Serial.println(msg);
     #endif
    }
//Εάν διαβαστεί η εντολή 01A8 σημαίνει ότι το Node 01 πρέπει να διαβάσει την είσοδο A8
    if (payload[2] == 'A' && payload[3] == '8')
    {
     changeMux(HIGH, HIGH, HIGH);
     value[8] = analogRead(ANALOG_INPUT);
     snprintf (msg, 75, "%c%c:A8:%ld",ID[0],ID[1], value[8]);
     client.publish("outTopic", msg);
     #ifdef DEBUG
     Serial.println("\nA8 Request Received");
     Serial.print("\nPublish message: ");
     Serial.println(msg);
     #endif
    }
//Εάν διαβαστεί η εντολή 01D5OH σημαίνει ότι το Node 01 πρέπει να θέσει το pin IO5 σε υψηλό δυναμικό
HIGH
    if (payload[2] ==  'D' && payload[3] ==  '5' && payload[4] == 'O' && payload[5] == 'H')
    {
     digitalWrite(IO5, HIGH);
     #ifdef DEBUG
     Serial.println("\nD5 Set High");
     #endif
    }
//Εάν διαβαστεί η εντολή 01D5OL σημαίνει ότι το Node 01 πρέπει να θέσει το pin IO5 σε χαμηλό δυναμικό
LOW
    if (payload[2] ==  'D' && payload[3] ==  '5' && payload[4] == 'O' && payload[5] == 'L')
    {
     digitalWrite(IO5, LOW);
     #ifdef DEBUG
     Serial.println("\nD5 Set Low");
     #endif
    }
//Εάν διαβαστεί η εντολή 01D5I σημαίνει ότι το Node 01 πρέπει να διαβάσει το δυναμικό του pin IO5
    if (payload[2] ==  'D' && payload[3] ==  '5' && payload[4] == 'I')
    {
     DigiVal[5]= digitalRead(IO5);
     snprintf (msg, 75, "%c%c:D5:%ld",ID[0],ID[1], DigiVal[5]);
```

```
  client.publish("outTopic", msg);
  #ifdef DEBUG
  Serial.print("\nPublish message: ");
  Serial.println(msg);
  #endif
 }
```

//Εάν διαβαστεί η εντολή 01D6OH σημαίνει ότι το Node 01 πρέπει να θέσει το pin IO6 σε υψηλό δυναμικό HIGH

```
 if (payload[2] ==  'D' && payload[3] ==  '6' && payload[4] == 'O' && payload[5] == 'H')
 {
  digitalWrite(IO6, HIGH);
  #ifdef DEBUG
  Serial.println("\nD6 Set High");
  #endif
 }
```

//Εάν διαβαστεί η εντολή 01D6OL σημαίνει ότι το Node 01 πρέπει να θέσει το pin IO6 σε χαμηλό δυναμικό LOW

```
 if (payload[2] ==  'D' && payload[3] ==  '6' && payload[4] == 'O' && payload[5] == 'L')
 {
  digitalWrite(IO6, LOW);
  #ifdef DEBUG
  Serial.println("\nD6 Set Low");
  #endif
 }
```

//Εάν διαβαστεί η εντολή 01D6I σημαίνει ότι το Node 01 πρέπει να διαβάσει το δυναμικό του pin IO6

```
 if (payload[2] == 'D' && payload[3] == '6' && payload[4] == 'I')
 {
   DigiVal[6]= digitalRead(IO6);
   snprintf (msg, 75, "%c%c:D6:%ld",ID[0],ID[1], DigiVal[6]);
   client.publish("outTopic", msg);
   #ifdef DEBUG
   Serial.print("\nPublish message: ");
   Serial.println(msg);
   #endif
 }
```

//Εάν διαβαστεί η εντολή 01D7OH σημαίνει ότι το Node 01 πρέπει να θέσει το pin IO7 σε υψηλό δυναμικό HIGH

```
 if (payload[2] == 'D' && payload[3] == '7' && payload[4] == 'O' && payload[5] == 'H')
 {
  digitalWrite(IO7, HIGH);
  #ifdef DEBUG
  Serial.println("\nD7 Set High");
  #endif
 }
```

//Εάν διαβαστεί η εντολή 01D7OL σημαίνει ότι το Node 01 πρέπει να θέσει το pin IO7 σε χαμηλό δυναμικό LOW

```
 if (payload[2] == 'D' && payload[3] == '7' && payload[4] == 'O' && payload[5] == 'L')
 {
  digitalWrite(IO7, LOW);
  #ifdef DEBUG
  Serial.println("\nD7 Set Low");
  #endif
 }
```

//Εάν διαβαστεί η εντολή 01D7I σημαίνει ότι το Node 01 πρέπει να διαβάσει το δυναμικό του pin IO7

```
 if (payload[2] == 'D' && payload[3] == '7' && payload[4] == 'I')
 {
  DigiVal[7]= digitalRead(IO7);
  snprintf (msg, 75, "%c%c:D7:%ld",ID[0],ID[1], DigiVal[7]);
  client.publish("outTopic", msg);
  #ifdef DEBUG
```

```
        Serial.print("\nPublish message: ");
        Serial.println(msg);
        #endif
       }
//Εάν διαβαστεί η εντολή 01D8OH σημαίνει ότι το Node 01 πρέπει να θέσει το pin IO8 σε υψηλό δυναμικό HIGH
      if (payload[2] == 'D' && payload[3] == '8' && payload[4] == 'O' && payload[5] == 'H')
      {
       digitalWrite(IO8, HIGH);
       #ifdef DEBUG
       Serial.println("\nD8 Set High");
       #endif
      }
//Εάν διαβαστεί η εντολή 01D8OL σημαίνει ότι το Node 01 πρέπει να θέσει το pin IO8 σε χαμηλό δυναμικό LOW
      if (payload[2] == 'D' && payload[3] == '8' && payload[4] == 'O' && payload[5] == 'L')
      {
       digitalWrite(IO8, LOW);
       #ifdef DEBUG
       Serial.println("\nD8 Set Low");
       #endif
      }
//Εάν διαβαστεί η εντολή 01D8I σημαίνει ότι το Node 01 πρέπει να διαβάσει το δυναμικό του pin IO8
      if (payload[2] == 'D' && payload[3] == '8' && payload[4] == 'I')
      {
       DigiVal[8]= digitalRead(IO8);
       snprintf (msg, 75, "%c%c:D8:%ld",ID[0],ID[1], DigiVal[8]);
       client.publish("outTopic", msg);
       #ifdef DEBUG
       Serial.print("\nPublish message: ");
       Serial.println(msg);
       #endif
      }
     }
}
//Ρουτίνα επανασύνδεσης του Node στον MQTT Broker/Server σε περίπτωση αποσύνδεσης
void reconnect() {
 // Loop until we're reconnected
 while (!client.connected()) {
  #ifdef DEBUG
  Serial.print("Attempting MQTT connection...");
  #endif
  // Create a random client ID
  String clientId = "ESP8266Client-";
  clientId += String(random(0xffff), HEX);
  // Attempt to connect
  if (client.connect(clientId.c_str())) {
   #ifdef DEBUG
   Serial.println("connected");
   #endif
   // Once connected, publish an announcement...
   snprintf (msg, 75, "%c%c:hello",ID[0],ID[1]);
   client.publish("outTopic", msg);
   digitalWrite(BUILTIN_LED,LOW);
   // ... and resubscribe
   client.subscribe("inTopic");
  } else {
   #ifdef DEBUG
   Serial.print("failed, rc=");
   Serial.print(client.state());
```

```
    Serial.println(" try again in 5 seconds");
    #endif
    // Wait 5 seconds before retrying
    digitalWrite(BUILTIN_LED,HIGH);
    delay(5000);
   }
  }
}

//Setup του Node
void setup() {
  pinMode(MUX_A, OUTPUT);
  pinMode(MUX_B, OUTPUT);
  pinMode(MUX_C, OUTPUT);
  EEPROM.begin(20);
  Setup_IO_pins();

  Serial.begin(115200);
  WiFi.mode(WIFI_STA);
  setup_wifi();
  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);
}
//Ρουτίνα που επαναλαμβάνεται συνέχεια
void loop() {
  if (!client.connected()) {
    reconnect();
  }
  client.loop();
  long now = millis();
  if (now - lastMsg > 2000)
  {
    lastMsg = now;
     if (FunctionMode == 0)
            {
                  GetAllAnalogs();
            }
  }
}
```

10.7. Παράρτημα 7 – Κώδικας Αποθήκευσης Δεδομένων MQTT Broker σε Βάση Δεδομένων

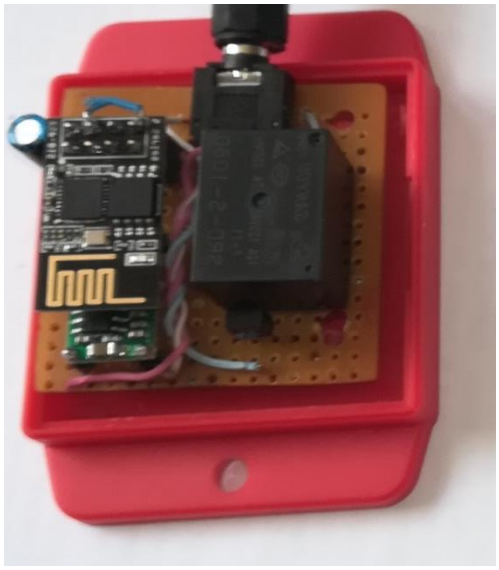
Ο κώδικας αυτός τρέχει σε μορφή υπηρεσία στην εφαρμογή. Όποτε στέλνεται ένα σήμα εντολή στον MQTT Broker αυτό καταγράφεται στη βάση δεδομένων που έχει δημιουργηθεί, και στον πίνακα που έχει δημιουργηθεί. Ο πίνακας, αποτελείται από τρία πεδία ,Topic, Payload και Timestamp. Το Timestamp, ενημερώνεται αυτόματα σε κάθε εγγραφή με την ημερομηνία και ώρα που έγινε η εγγραφή. Αυτό σημαίνει πως καταγράφονται τόσο οι εντολές που στέλνονται στα Nodes αλλά και οι απαντήσεις που στέλνουν. Για λόγους συντήρησης του πίνακα αποθήκευσης δεδομένων διαγράφονται οι εντολές που στέλνονται στα Nodes, όσες εγγραφές έχουν Topic=’inTopic’. Επίσης για να μην μεγαλώνει ο πίνακας μεταφέρονται παλιότερα δεδομένα σε έναν πίνακα mqtt_log_backup, που έχει δημιουργηθεί στη βάση δεδομένων.

(Τελευταία έκδοση κώδικα 12/5/2018)

```
#include <signal.h>
#include <stdio.h>
#include <string.h>
#ifndef WIN32
#  include <unistd.h>
#else
#  include <process.h>
#  define snprintf sprintf_s
#endif
#include "mosquitto.h"
#include "mysql.h"
#define db_host "localhost" //Η βάση είναι τοπικά στο Raspberry θα μπορούσε να βρίσκεται σε άλλο server
#define db_username "mqtt_log"          //Ο Χρήστης που συνδέεται στη βάση
#define db_password "password"          //Ο κωδικός πρόσβασης του χρήστη
#define db_database "mqtt_log"          // Η Βάση που αποθηκεύονται τα δεδομένα
#define db_port 3306                    // Η θύρα σύνδεσης στη βάση
#define db_query "INSERT INTO mqtt_log (topic, payload) VALUES (?,?)" //SQL εντολή αποθήκευσης εγγραφής
#define mqtt_host "localhost"           //Διεύθυνση του MQTT Broker
#define mqtt_port 1883                  //Θύρα του MQTT Broker
static int run = 1;
static MYSQL_STMT *stmt = NULL;
void handle_signal(int s)
{
	run = 0;
}
void connect_callback(struct mosquitto *mosq, void *obj, int result)
{
}
void message_callback(struct mosquitto *mosq, void *obj, const struct mosquitto_message *message)
{
	MYSQL_BIND bind[2];
	memset(bind, 0, sizeof(bind));
	bind[0].buffer_type = MYSQL_TYPE_STRING;
	bind[0].buffer = message->topic;
	bind[0].buffer_length = strlen(message->topic);
	// Note: payload is normally a binary blob and could contains
	// NULL byte. This sample does not handle it and assume payload is a  string.
	bind[1].buffer_type = MYSQL_TYPE_STRING;
	bind[1].buffer = message->payload;
	bind[1].buffer_length = message->payloadlen;
	mysql_stmt_bind_param(stmt, bind);
	mysql_stmt_execute(stmt);
}
int main(int argc, char *argv[])
{
	MYSQL *connection;
	my_bool reconnect = true;
	char clientid[24];
	struct mosquitto *mosq;
	int rc = 0;
	signal(SIGINT, handle_signal);
	signal(SIGTERM, handle_signal);
	mysql_library_init(0, NULL, NULL);
	mosquitto_lib_init();
	connection = mysql_init(NULL);
	if(connection){
	mysql_options(connection, MYSQL_OPT_RECONNECT, &reconnect);
```

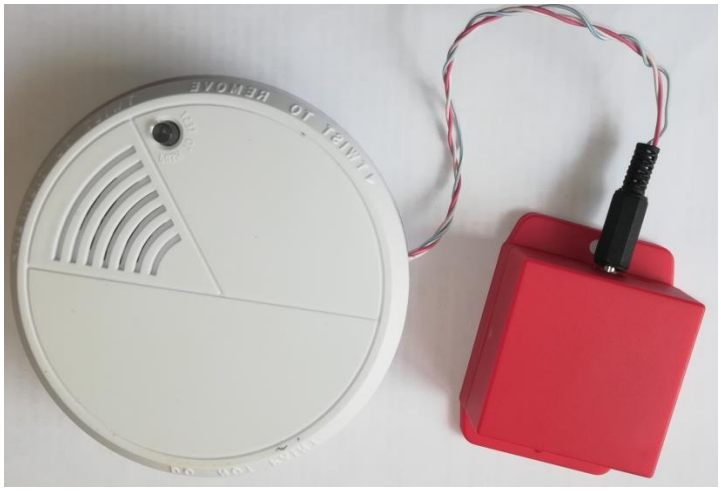
```
connection = mysql_real_connect(connection, db_host, db_username, db_password, db_database, db_port, NULL,
0);
		if(connection){
			stmt = mysql_stmt_init(connection);
			mysql_stmt_prepare(stmt, db_query, strlen(db_query));
			memset(clientid, 0, 24);
			snprintf(clientid, 23, "mysql_log_%d", getpid());
			mosq = mosquitto_new(clientid, true, connection);
			if(mosq){
				mosquitto_connect_callback_set(mosq, connect_callback);
				mosquitto_message_callback_set(mosq, message_callback);
rc = mosquitto_connect(mosq, mqtt_host, mqtt_port, 60);
				mosquitto_subscribe(mosq, NULL, "#", 0);
				while(run){
					rc = mosquitto_loop(mosq, -1, 1);
					if(run && rc){
						sleep(20);
						mosquitto_reconnect(mosq);
					}
				}
				mosquitto_destroy(mosq);
			}
			mysql_stmt_close(stmt);
			mysql_close(connection);
		}else{
			fprintf(stderr, "Error: Unable to connect to database.\n");
			printf("%s\n", mysql_error(connection));
			rc = 1;
		}
	}else{
		fprintf(stderr, "Error: Unable to start mysql.\n");
		rc = 1;
	}
	mysql_library_end();
	mosquitto_lib_cleanup();
	return rc;
}
```

10.8. Παράρτημα 8 – Εφαρμογή Ανιχνευτής Καπνού



Εικόνα 54: Προσαρμοστικό Κύκλωμα

Εικόνα 55: Αισθητήρας Καπνού & Προσαρμοστικό Κύκλωμα I



Εικόνα 56: Αισθητήρας Καπνού & Προσαρμοστικό Κύκλωμα II

10.8.1. Κώδικας Firmware

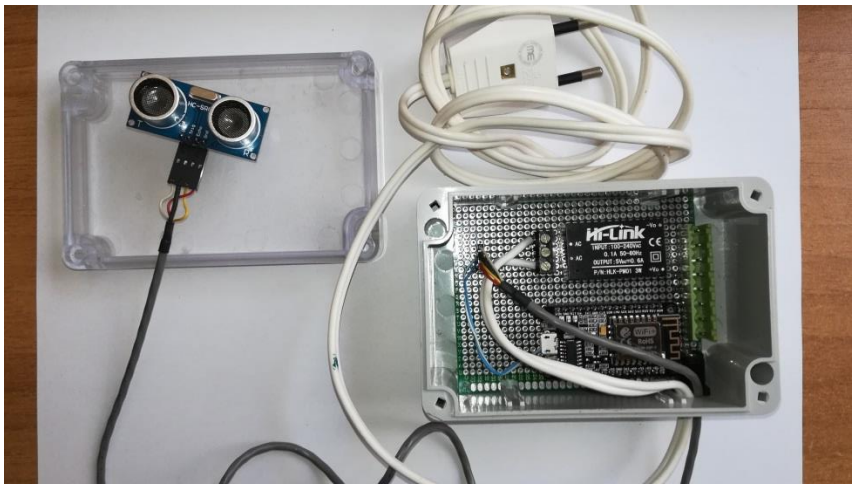
```
//Βιβλιοθήκη διασύνδεσης wifi
#include <ESP8266WiFi.h>
//Βιβλιοθήκη Δημοσίευσης & Εγγραφής σε MQTT Broker
#include "PubSubClient.h"
//Βιβλιοθήκη σύνδεσης πελάτη HTTP
#include <ESP8266HTTPClient.h>
//Βιβλιοθήκη ενημέρωσης Firmware μέσω HTTP
#include <ESP8266httpUpdate.h>

// Ορίσμός στοιχείων σύνδεσης στο Ασύρματο Δίκτυο
const char* ssid = "manos";
const char* password = "coucoutsi";
//Ορισμός Διεύθυνσης του MQTT Server/Broker
const char* mqtt_server = "192.168.2.85";

WiFiClient espClient;
PubSubClient client(espClient);

long lastMsg = 0;
char msg[75];
//Ορισμός Ταυτότητας του Node
const char* ID="05";

//Το κείμενο που στέλνει το Node όταν σταλεί στο Node η Εντολή HELP
char Help1[100]="\nHELP\nBROAD\nIDPING\nIDFIRMWARE\nIDUPDATE\nIDSTATUS";
char Help2[100]="\nCommand Help use Command? e.q BROAD?";
int i=0;

//Ορισμός της τρέχουσας έκδοσης Firmware
//Στέλνεται από το Node όταν δεχτεί την εντολή FIRMWARE
char firmware[20]="1.4";

void setup_wifi() {

  delay(10);
//  Serial.print("Connecting to ");
//  Serial.println(ssid);
//Σύνδεση στο ασύρματο Δίκτυο
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
//    Serial.print(".");
  }
  randomSeed(micros());
//  Serial.println("");
//  Serial.println("WiFi connected");
//  Serial.println("IP address: ");
//  Serial.println(WiFi.localIP());
}

//Η ρουτίνα callback διαβάσει το inTopic ώστε να λάβει εντολή το Node.
//Ανάλογα με την εντολή που διαβάζει πράτει αναλόγως
void callback(char* topic, byte* payload, unsigned int length)
{
   //Broadcast Request
   //Εάν διαβάσει ένα Node BROAD στέλνει την ταυτότητα του και ενα OK στο outTopic
   //Αποτέλεσμα της εντολής να ανταποκριθούν όλα τα Nodes που είναι συνδεδεμένα στο σύστημα
   // να ανταποκριθούν και ο χρήστης διαπιστώνει ποια Nodes λειτουργούν.
```

```
   //Αυτή είναι διαχειριστική εντολή και ανταποκρίνονται όλα τα Nodes.
   if (payload[0] == 'B' && payload[1] == 'R' && payload[2] == 'O' && payload[3] == 'A' &&
payload[4] == 'D'  )
   {
         snprintf (msg, 75, "%c%c:OK",ID[0],ID[1]);
//          Serial.print("Publish message: ");
//          Serial.println(msg);
        client.publish("outTopic", msg);
   }
//     Serial.print("Message arrived [");
//     Serial.print(topic);
//     Serial.print("] ");
//     for (int i = 0; i < length; i++) {
//       Serial.print((char)payload[i]);
//     }
//   Serial.println();
//Εάν τα δύο πρώτα ψηφία της εντολής που διαβάστηκε έιναι ίσα με την ταυτότητα του Node
//τότε ανταποκρίνεται μόνο το συγκεκριμένο Node
//Διαβάζεται σε ποιό Node απευθύνεται η εντολή

if (payload[0] == ID[0] && payload[1] == ID[1])
   {
//FIRMWARE
//Εάν διαβαστεί η εντολή 01FIRMWARE σημαίνει ότι το Node 01 πρέπει να στείλει το firmware
του.
   if (payload[2] == 'F' && payload[3] == 'I' && payload[4] == 'R' && payload[5] == 'M' &&
payload[6] == 'W' && payload[7] == 'A' && payload[8] == 'R' && payload[9] == 'E')
      {
       snprintf (msg, 75,"%c%c:%s",ID[0],ID[1],firmware);
       client.publish("outTopic", msg);
      }
//HELP
//Εάν διαβαστεί η εντολή 01HELP σημαίνει ότι το Node 01 πρέπει να στείλει το κείμενο βοήθειας
του  Help1[100] και char Help2[100]
if (payload[2] == 'H' && payload[3] == 'E' && payload[4] == 'L' && payload[5] == 'P')
      {
       snprintf (msg, 110,"%c%c:%s",ID[0],ID[1],Help1);
       client.publish("outTopic", msg);
       snprintf (msg, 110,"%c%c:%s",ID[0],ID[1],Help2);
       client.publish("outTopic", msg);
    }
//Εάν διαβαστεί η εντολή 01UPDATE σημαίνει ότι το Node 01 πρέπει να κάνει HTTP update από
διεύθυνση που του ορίζουμε
//Επειδή το ESP-01 δεν έχει αρκετά μεγάλη μνήμη ώστε να κάνει HTTP Update χρησιμοποιείται
ένα trick
//Κάνει UPDATE το LOADER Firmware, το οποίο δεν έχει καθόλου κώδικα.
//Ο LOADER έχει τον κώδικα με το αρχείο με το νεο firmware που πρέπει να αναβαθμιστεί.
//Ετσι δεν χρειάζεται να βγει το ESP από το κουτί που έχει τοποθετηθεί.
if (payload[2] == 'U' && payload[3] == 'P' && payload[4] == 'D' && payload[5] == 'A' &&
payload[6] == 'T' && payload[7] == 'E' )
      {
       snprintf (msg, 75, "%c%c:UPDATING",ID[0],ID[1]);
       client.publish("outTopic", msg);
       //snprintf (msg, 75,"http://garefalakis.eu/esp01-01-Loader.bin", ID[0],ID[1] );
       snprintf (msg, 75, "http://192.168.2.85/esp01-%c%c-Loader.bin", ID[0],ID[1]);
       t_httpUpdate_return ret = ESPhttpUpdate.update(msg);
       //t_httpUpdate_return  ret = ESPhttpUpdate.update("https://server/file.bin");

       switch(ret) {
          case HTTP_UPDATE_FAILED:
```

```
//              Serial.printf("HTTP_UPDATE_FAILD Error (%d): %s", ESPhttpUpdate.getLastError(),
ESPhttpUpdate.getLastErrorString().c_str());
            snprintf (msg, 75, "%c%c:HTTP_UPDATE_FAILD Error (%d): %s",ID[0],ID[1],
ESPhttpUpdate.getLastError(), ESPhttpUpdate.getLastErrorString().c_str());
            client.publish("outTopic", msg);
            break;
          case HTTP_UPDATE_NO_UPDATES:
//              Serial.println("HTTP_UPDATE_NO_UPDATES");
            snprintf (msg, 75, "%c%c:HTTP_UPDATE_NO_UPDATES",ID[0],ID[1]);
            client.publish("outTopic", msg);
            break;
          case HTTP_UPDATE_OK:
//              Serial.println("HTTP_UPDATE_OK");
            snprintf (msg, 75, "%c%c:HTTP_UPDATE_OK",ID[0],ID[1]);
            client.publish("outTopic", msg);
            break;
       }
     }
//PING
//Εάν διαβαστεί η εντολή 01PING σημαίνει ότι το Node 01 πρέπει να απανατήσει με ένα
01:PING_OK
//ώστε ο χρήστης να διαπιστώσει ότι το Node λειτουργεί
if (payload[2] == 'P' && payload[3] == 'I' && payload[4] == 'N' && payload[5] == 'G' )
      {
            snprintf (msg, 75, "%c%c:PING_OK",ID[0],ID[1]);
            client.publish("outTopic", msg);
      }
//Εάν διαβαστεί η εντολή 01STATUS σημαίνει ότι το Node 01 πρέπει να απανατήσει με ένα 01:0
//Αυτή η εντολή ενσωματώθηκε ώστε να δημιουργείται γραφική παράσταση για τον αισθητήρα
αυτό
//Κάθε φορά στέλνει STATUS=0 και όταν ενεργοποιηθεί ο συναγερμός στέλνει STATUS=100 και
η τιμή καταγράφεται ως σημείο στη γραφική παράσταση.
if (payload[2] == 'S' && payload[3] == 'T' && payload[4] == 'A' && payload[5] == 'T' && payload[6]
== 'U' && payload[7] == 'S' )
      {
            snprintf (msg, 75, "%c%c:0",ID[0],ID[1]);
            client.publish("outTopic", msg);
      }

   }
}
//Η ρουτίνα αυτή καλείται όταν ενερεγοποιθεί συναγερμός καπνού
//Στέλνει μήνυμα Smoke Alarm On στον Broker και στη συνεχεια ενεργοποιούνται μέσω του
//Node-RED triggers για να σταλούν e-mail και SMS
void extAlarm() {
  if ( digitalRead(2)==LOW )
  {
    Serial.println("Alarm ON");
    snprintf (msg, 75, "%c%c:Smoke Alarm ON",ID[0],ID[1]);
    client.publish("outTopic", msg);
    snprintf (msg, 75, "%c%c:100",ID[0],ID[1]);
    client.publish("outTopic", msg);
    delay(5000);
  }
}

void reconnect()
{
 // Loop until we're reconnected
 while (!client.connected())
```

```
    {
//    Serial.print("Attempting MQTT connection...");
    // Create a random client ID
    String clientId = "ESP8266Client-";
    clientId += String(random(0xffff), HEX);
    // Attempt to connect
    if (client.connect(clientId.c_str()))
    {
//      Serial.println("connected");
      // Once connected, publish an announcement...
      client.publish("outTopic", "hello from ESP-01");
      // ... and resubscribe
      client.subscribe("inTopic");
      snprintf (msg, 75, "%c%c:Relay Connected", ID[0],ID[1]);
//      Serial.print("Publish message: ");
//      Serial.println(msg);
      client.publish("outTopic", msg);
     }
    else
     {
//      Serial.print("failed, rc=");
//      Serial.print(client.state());
//      Serial.println(" try again in 5 seconds");
      // Wait 5 seconds before retrying
      delay(5000);
     }
    }
 }
//Setup του Node
void setup()
{
  Serial.begin(115200);
//  Serial.println("Booting");
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);
 // pinMode(BUILTIN_LED, OUTPUT);     // Initialize the BUILTIN_LED pin as an output
  pinMode(2, INPUT);     // Initialize the relay pin as an output
  pinMode(0, INPUT);     // Initialize the relay pin as an output
  setup_wifi();
  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);
}

//Ρουτίνα που επαναλαμβάνεται συνέχεια
void loop()
{
  if (!client.connected())
  {
    reconnect();
  }
  client.loop();
  extAlarm();
}
```

10.9. Παράρτημα 9 – Αισθητήρας μέτρησης Στάθμης Δεξαμενής Νερού



Εικόνα 57: Ultrasonic Αισθητήρας μέτρησης Στάθμης Νερού

Εικόνα 58: Ultrasonic Αισθητήρας μέτρησης Στάθμης Νερού Εγκατάσταση

Εικόνα 59:Ultrasonic Αισθητήρας μέτρησης Στάθμης Νερού Τοποθετημένος

10.9.1. Κώδικας Firmware

```
//Βιβλιοθήκη διασύνδεσης wifi
#include <ESP8266WiFi.h>
//Βιβλιοθήκη Δημοσίευσης & Εγγραφής σε MQTT Broker
#include "PubSubClient.h"
//Βιβλιοθήκη σύνδεσης πελάτη HTTP
#include <ESP8266HTTPClient.h>
//Βιβλιοθήκη ενημέρωσης Firmware μέσω HTTP
#include <ESP8266httpUpdate.h>

// Update these with values suitable for your network.
// Ορίσμός στοιχείων σύνδεσης στο Ασύρματο Δίκτυο
const char* ssid = "manos";
const char* password = "coucoutsi";
//Ορισμός Διεύθυνσης του MQTT Server/Broker
const char* mqtt_server = "192.168.2.85";

// defines variables
long duration;
int distance;

WiFiClient espClient;
PubSubClient client(espClient);
long lastMsg = 0;
char msg[75];
int value[8];
// FunctionMode=0 then polling
// FunctionMode=1 then Collecting Data
//To Node ρυθμίζεται σε Polling Mode
int FunctionMode=1;

//Ορισμός Ταυτότητας του Node
const char* ID="06";

//Το κείμενο που στέλνει το Node όταν σταλεί στο Node η Εντολή HELP
char Help1[100]="\nHELP\nBROAD\nIDPING\nIDFIRMWARE\nIDUPDATE\IDGD";
char Help2[100]="\nCommand Help use Command? e.q BROAD?";
int i=0;

//Ορισμός της τρέχουσας έκδοσης Firmware
//Στέλνεται από το Node όταν δεχτεί την εντολή FIRMWARE
char firmware[20]="1.3";

//Ρουτίνα που επιστρέφει την απόσταση της στάθμης

void GetDistance()
{
  // Clears the trigPin
  digitalWrite(D7, LOW);
  delayMicroseconds(2);
  // Sets the trigPin on HIGH state for 10 micro seconds
  digitalWrite(D7, HIGH);
  delayMicroseconds(10);
  digitalWrite(D7, LOW);
  // Reads the echoPin, returns the sound wave travel time in microseconds
```

```
   duration = pulseIn(D8, HIGH);
   // Calculating the distance
   distance= 128-(duration*0.034/2);
   // Prints the distance on the Serial Monitor
   Serial.print("Distance: ");
   Serial.println(distance);
   snprintf (msg, 75, "ID:%c%c:%ld",ID[0],ID[1],distance);
   Serial.print("Publish message: ");
   Serial.println(msg);
   client.publish("outTopic", msg);
}

//Αρχικοποίηση του WiFi
void setup_wifi() {
  delay(10);
  // We start by connecting to a WiFi network
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  randomSeed(micros());
  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}
//Η ρουτίνα callback διαβάσει το inTopic ώστε να λάβει εντολή το Node.
//Ανάλογα με την εντολή που διαβάζει πράτει αναλόγως
void callback(char* topic, byte* payload, unsigned int length) {
  Serial.print("Message arrived [");
  Serial.print(topic);
  Serial.print("] ");
  for (int i = 0; i < length; i++)
    {
    Serial.print((char)payload[i]);
    }
//Εάν τα δύο πρώτα ψηφία της εντολής που διαβάστηκε έιναι ίσα με την ταυτότητα του
Node
//τότε ανταποκρίνεται μόνο το συγκεκριμένο Node
//Διαβάζεται σε ποιό Node απευθύνεται η εντολή
if (payload[0] == ID[0]&& payload[1] == ID[1] )
{
//FIRMWARE
//Εάν διαβαστεί η εντολή 01FIRMWARE σημαίνει ότι το Node 01 πρέπει να στείλει το
firmware του.
   if (payload[2] == 'F' && payload[3] == 'I' && payload[4] == 'R' && payload[5] == 'M'
&& payload[6] == 'W' && payload[7] == 'A' && payload[8] == 'R' && payload[9] == 'E')
      {
       snprintf (msg, 75,"%c%c:%s",ID[0],ID[1],firmware);
       client.publish("outTopic", msg);
      }
```

```
//HELP
//Εάν διαβαστεί η εντολή 01HELP σημαίνει ότι το Node 01 πρέπει να στείλει το κείμενο βοήθειας του  Help1[100] και char Help2[100]
   if (payload[2] == 'H' && payload[3] == 'E' && payload[4] == 'L' && payload[5] == 'P')
      {
       snprintf (msg, 110,"%c%c:%s",ID[0],ID[1],Help1);
       client.publish("outTopic", msg);
       snprintf (msg, 110,"%c%c:%s",ID[0],ID[1],Help2);
       client.publish("outTopic", msg);
    }
//UPDATE
//Εάν διαβαστεί η εντολή 01UPDATE σημαίνει ότι το Node 01 πρέπει να κάνει HTTP update από διεύθυνση που του ορίζουμε
  if (payload[2] == 'U' && payload[3] == 'P' && payload[4] == 'D' && payload[5] == 'A'
&& payload[6] == 'T' && payload[7] == 'E' )
      {
       snprintf (msg, 75, "%c%c:UPDATING",ID[0],ID[1]);
       client.publish("outTopic", msg);
       snprintf (msg, 75,"http://192.168.2.85/esp12-%c%c.bin", ID[0],ID[1] );
       //snprintf (msg, 75,"http://garefalakis.eu/esp12-04.bin");
       t_httpUpdate_return ret = ESPhttpUpdate.update(msg);
       //t_httpUpdate_return  ret = ESPhttpUpdate.update("https://server/file.bin");
       switch(ret)
       {
          case HTTP_UPDATE_FAILED:
//              Serial.printf("HTTP_UPDATE_FAILD Error (%d): %s",
ESPhttpUpdate.getLastError(), ESPhttpUpdate.getLastErrorString().c_str());
             snprintf (msg, 75, "%c%c:HTTP_UPDATE_FAILD Error (%d): %s",ID[0],ID[1],
ESPhttpUpdate.getLastError(), ESPhttpUpdate.getLastErrorString().c_str());
             client.publish("outTopic", msg);
             break;
          case HTTP_UPDATE_NO_UPDATES:
//              Serial.println("HTTP_UPDATE_NO_UPDATES");
             snprintf (msg, 75, "%c%c:HTTP_UPDATE_NO_UPDATES",ID[0],ID[1]);
             client.publish("outTopic", msg);
             break;
          case HTTP_UPDATE_OK:
//              Serial.println("HTTP_UPDATE_OK");
             snprintf (msg, 75, "%c%c:HTTP_UPDATE_OK",ID[0],ID[1]);
             client.publish("outTopic", msg);
             break;
       }
    }
//PING
//Εάν διαβαστεί η εντολή 01PING σημαίνει ότι το Node 01 πρέπει να απανατήσει με ένα 01:PING_OK
if (payload[2] == 'P' && payload[3] == 'I' && payload[4] == 'N' && payload[5] == 'G' )
      {
             snprintf (msg, 75, "%c%c:PING_OK",ID[0],ID[1]);
             client.publish("outTopic", msg);
      }
//MODE
// M0 Continuous Data Sending
//Εάν διαβαστεί η εντολή 01M0 σημαίνει ότι το Node 01 πρέπει να γυρίσει για Continuous Mode
```

```
//Εάν διαβαστεί η εντολή 01M1 σημαίνει ότι το Node 01 πρέπει να γυρίσει Polling Mode
if (payload[2] == 'M')
  {
    if (payload[3] == '0')
    {
     FunctionMode=0;
    }
    else if (payload[3] == '1')
    {
     FunctionMode=1;
    }
  }
//GET DISTANCE
//Εάν διαβαστεί η εντολή 01GD σημαίνει ότι το Node 01 πρέπει να εκτελέσει την ρουτίνα  GetDistance()
if (payload[2] == 'G' && payload[3] == 'D')
  {
     GetDistance();
  }
 }
//Broadcast Request
//Εάν διαβάσει ένα Node BROAD στέλνει την ταυτότητα του και ενα OK στο outTopic
//Αποτέλεσμα της εντολής να ανταποκριθούν όλα τα Nodes που είναι συνδεδεμένα στο σύστημα
// να ανταποκριθούν και ο χρήστης διαπιστώνει ποια Nodes λειτουργούν.
//Αυτή είναι διαχειριστική εντολή και ανταποκρίνονται όλα τα Nodes.
if (payload[0] == 'B' && payload[1] == 'R' && payload[2] == 'O' && payload[3] == 'A' &&
payload[4] == 'D'  )
  {
        snprintf (msg, 75, "%c%c:OK:",ID[0],ID[1],payload);
        Serial.print("Publish message: ");
        Serial.println(msg);
        client.publish("outTopic", msg);
  }
}
void reconnect()
{
 // Loop until we're reconnected
 while (!client.connected())
 {
   Serial.print("Attempting MQTT connection...");
   // Create a random client ID
   String clientId = "ESP8266Client-";
   clientId += String(random(0xffff), HEX);
   // Attempt to connect
   if (client.connect(clientId.c_str()))
   {
     Serial.println("connected");
     // Once connected, publish an announcement...
     snprintf (msg, 75, "%c%c:hello",ID[0],ID[1]);
     client.publish("outTopic", msg);
     // ... and resubscribe
     client.subscribe("inTopic");
     digitalWrite(D1, HIGH);
   }
```

```
    else
    {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      // Wait 5 seconds before retrying
      digitalWrite(D1, LOW);
      delay(5000);
    }
  }
}
//Setup του Node
void setup() {
  pinMode(D1,OUTPUT);
  pinMode(D7, OUTPUT);                  // Sets the trigPin as an Output
  pinMode(D8, INPUT);                           // Sets the echoPin as an Input
  pinMode(BUILTIN_LED, OUTPUT);         // Initialize the BUILTIN_LED pin as an
output
  Serial.begin(115200);
  WiFi.mode(WIFI_STA);
  setup_wifi();
  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);
}
//Ρουτίνα που επαναλαμβάνεται συνέχεια
void loop()
{
  if (!client.connected())
  {
    reconnect();
  }
  client.loop();
  long now = millis();
  if (now - lastMsg > 2000)
  {
    lastMsg = now;
     if (FunctionMode == 0)
                {
        GetDistance();
        delay(1000);
                }
  }
}
```

10.10. Παράρτημα 10 – Κώδικας Εφαρμογής Node-RED

Ο κώδικας του Node –RED δεν είναι ευκολονόητος για να διαβαστεί. Πρέπει να εισαχθεί μέσω του προχείρου στο Node-RED. Και για το λόγο αυτό καταχωρείται στο παράρτημα αυτό με σκοπό να αντιγραφεί από την ηλεκτρονική μορφή του εγγράφου και να εισαχθεί σε νέα πλατφόρμα με εγκατεστημένο το Node-RED.

10.11. Παράρτημα 11 – Οικογένεια ESP8266

Εικόνα 60: ESP8266 Τύποι

Board ID	#Pins	Pitch	Form factor	LEDs	Antenna	Ant.Socket	Shielded	Dimensions mm	Flash Size in Bytes and (bits)
ESP-01	8	0.1“	2×4 DIL	Yes	Etched-on PCB	No	No	14.3 x 24.8	512KB (4Mb) ××
ESP-02	8	0.1”	2×4 notch	No?	None	Yes	No	14.2 x 14.2	512KB (4Mb) ×
ESP-03	14	2mm	2×7 notch	No	Ceramic	No	No	17.3 x 12.1	512KB (4Mb) ×
ESP-04	14	2mm	2×4 notch	No?	None	No	No	14.7 x 12.1	512KB (4Mb) ×
ESP-05	5	0.1“	1×5 SIL	No	None	Yes	No	14.2 x 14.2	512KB (4Mb) ×
ESP-06	12+GND	misc	4×3 dice	No	None	No	Yes	16.3 x 13.1	512KB (4Mb) ×
ESP-07	16	2mm	2×8 pinhole	Yes	Ceramic	Yes	Yes	21.2 x 16.0	1MB (8Mb) ××
ESP-07S	16	2mm	2×8 pinhole	No	None	Yes	Yes	17.0 x 16.0	4MB (32Mb)
ESP-08	14	2mm	2×7 notch	No	None	No	Yes	17.0 x 16.0	?? *(please fill if you know)*

Board ID	#Pins	Pitch	Form factor	LEDs	Antenna	Ant.Socket	Shielded	Dimensions mm	Flash Size in Bytes and (bits)
ESP-08 New	16	2mm	2×8 notch	No	None	No	Yes	18.0 x 16.0	?? *(please fill if you know)*
ESP-09	12+GND	misc	4×3 dice	No	None	No	No	10.0 x 10.0	1MB (8Mb)
ESP-10	5	2mm ??	1×5 notch	No	None	No	No	14.2 x 10.0	512KB (4Mb) *
ESP-11	8	1.27mm	1×8 pinhole	No?	Ceramic	No	No	17.3 x 12.1	512KB (4Mb) *
ESP-12	16	2mm	2×8 notch	Yes	Etched-on PCB	No	Yes	24.0 x 16.0	4MB (32Mb) *??*
ESP-12F	22	2mm	2×8 notch	Yes	Etched-on PCB	No	Yes	24.0 x 16.0	4MB (32Mb)
ESP-12E	22	2mm	2×8 notch	Yes	Etched-on PCB	No	Yes	24.0 x 16.0	4MB (32Mb)
ESP-12S	16	2mm	2×8 notch	Yes	Etched-on PCB	No	Yes	24.0 x 16.0	4MB (32Mb)
ESP-13	18	1.5mm	2×9	?	Etched-on PCB	No	Yes	20.0 x 19.9	4MB (32Mb)
ESP-14	22	2mm	2×8 + 6	1	Etched-on PCB	No	Yes	24.3 x 16.2	?? *(please fill if you know)*
ESP-201	22+4	0.1”	2×11 + 4	2	Etched-on PCB ×××	Yes	No	33.5 x 25.5	512KB (4Mb)
WROOM-02	18	1.5mm	2×9	No	Etched on PCB	No	Yes	20.0 x 18.0	?? *(please fill if you know)*
WT8266-S1	18	1.5mm	3×6	1	Etched on PCB	No	Yes	15.0 x 18.6	4MB (32Mb)

Πίνακας 11:Χαρακτηριστικά των τύπων ESP