ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΝΑΥΠΗΓΩΝ ΜΗΧΑΝΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ

Τομέας Μελέτης Πλοίων και Θαλάσσιων Μεταφορών

Διπλωματική Εργασία

«Μελέτη, Κατασκευή και Εγκατάσταση Συστήματος Αισθητήρων σε Σκάφος Αναψυχής»

Μποζίκης Διονύσης

Επιβλέπων: Γκίνης Αλέξανδρος

Αναπληρωτής Καθηγητής Ε.Μ.Π.

Αθήνα, Φλεβάρης 2021

Ευχαριστίες

Με την ολοκλήρωση της παρούσας διπλωματικής εργασίας θα ήθελα να ευχαριστήσω θερμά τον καθηγητή μου, κ. Αλέξανδρο Γκίνη, για την ανάθεση του θέματος και την καθοδήγηση που μου παρείχε κατά τη διάρκεια εκπόνησης της παρούσας εργασίας, καθώς και για την διάθεσή του να με βοηθήσει σε ένα τέτοιο θέμα.

Επιπλέον, θα ήθελα να ευχαριστήσω τον Στέλιο Σκαρίμπα που μου έδωσε την δυνατότητα να κάνω τις δοκιμές του συστήματος στο σκάφος του, που μου παρείχε υλική και θεωρητική βοήθεια επί διαφόρων κομματιών της διπλωματικής, για την ψυχολογική στήριξη και για όλες τις συμβουλές του επί του ζητήματος. Ακόμα να ευχαριστήσω τον φίλο και πρώην συγκάτοικο Φλας, για την καθοδήγησή του και συνολικά για την αμέριστη βοήθεια του όσον αφορά τη βελτίωση του κώδικά μου και για την επίλυση τεχνικών ζητημάτων που προέκυψαν. Ευχαριστώ τον φίλο Γιάννη που μου έδωσε την δυνατότητα να χρησιμοποιήσω τον τρισδιάστατο εκτυπωτή του για του σκοπούς της παρούσας διπλωματικής εργασίας, καθώς και την φίλη μου Βασιλική που με βοήθησε στο στήσιμο του παρόντος.

Τέλος, θα ήθελα να ευχαριστήσω ιδιαίτερα τη μητέρα μου, Βιβή, και τον πατέρα μου, Νίκο, που μου έδωσαν την ευκαιρία να σπουδάσω. Δεν ξεχνώ όλο τον υπόλοιπο κοντινό μου κόσμο που ανά τα χρόνια στάθηκε δίπλα μου στην διάρκεια των σπουδών μου, που χωρίς την συνεχή στήριξή τους δεν θα μπορούσα να φέρω εις πέρας.

Περιεχόμενα

Κεφάλαιο 1 - Εισαγωγή

1.1 Περιγραφή Προβλήματος

Στην σύγχρονη ναυτιλία η ύπαρξη συστημάτων συνεχούς παρατήρησης (monitoring) των διαφόρων μεταβλητών των συστημάτων του σκάφους (πιέσεις λειτουργικών και ψυκτικών υγρών, στοιχεία περιβάλλοντος και χώρων κλπ) είναι απαραίτητη για την ασφαλή και ομαλή λειτουργία τους. Τα συστήματα αυτά προκύπτουν ύστερα από μελέτες των κατασκευαστών είναι πλέον προαπαιτούμενα χαρακτηριστικά για την κατασκευή ενός σκάφους εμπορικής κλίμακας.

Στα σκάφη αναψυχής και ειδικά σε αυτά που είναι παλιότερης τεχνολογίας υπάρχει μεγάλο έλλειμα σε αυτόν τον τομέα. Ένα τέτοιο σκάφος συνήθως έχει εγκατεστημένο ένα απλό σύστημα επιτήρησης των βασικών ενδείξεων της μηχανής και ύστερα υπάρχει η δυνατότητα επιπρόσθετης εγκατάστασης σχετικών συστημάτων για την παρατήρηση των λειτουργιών του σκάφους καθώς και των μετεωρολογικών στοιχείων, με αρκετά αυξημένο κόστος.

Κρίνεται λοιπόν απαραίτητη η ύπαρξη μίας οικονομικής λύσης, που βασίζεται σε σύγχρονα τεχνολογικά πρότυπα, η οποία θα μπορεί να καλύψει τις ανάγκες της εκάστοτε ιδιοκτήτριας τέτοιας κλίμακας σκάφους αλλά και το κενό που υπάρχει στην αγορά σε αυτόν τον τομέα. Η λύση σε αυτό το πρόβλημα αποτελεί πρόκληση, καθώς είναι απαραίτητο να εξασφαλίζει την ομαλή και σταθερή λειτουργία του συστήματος, την εύκολη εγκατάσταση του συστήματος από την οποιαδήποτε (plug and play) αλλά και την αξιοπιστία του συστήματος όσο αφορά τόσο την ποιότητα των μετρήσεων όσο και την εύκολη διασυνδεσιμότητα της πληροφορίας με την χρήστρια.

1.2 Σκοπός Διπλωματικής

Ο σκοπός της παρούσας διπλωματικής εργασίας είναι να επεξεργαστεί το πρόβλημα που αναφέρθηκε παραπάνω με μία λύση Internet of Things (IoT), δηλαδή ο σχεδιασμός, η δημιουργία και η κατασκευή ενός ολοκληρωμένου συστήματος παρακολούθησης διαφόρων στοιχείων που μας ενδιαφέρουν, με χρήση διαφόρων αισθητήρων που είναι εύκολα διαθέσιμοι στην καθεμία, με πολύ χαμηλό κόστος αγοράς, όπως επίσης και με χρήση μικροεπεξεργαστών για την διαχείριση της πληροφορίας. Αναλυτικότερα το ενλόγω σύστημα θα είναι υπεύθυνο για τις παρακάτω λειτουργίες:

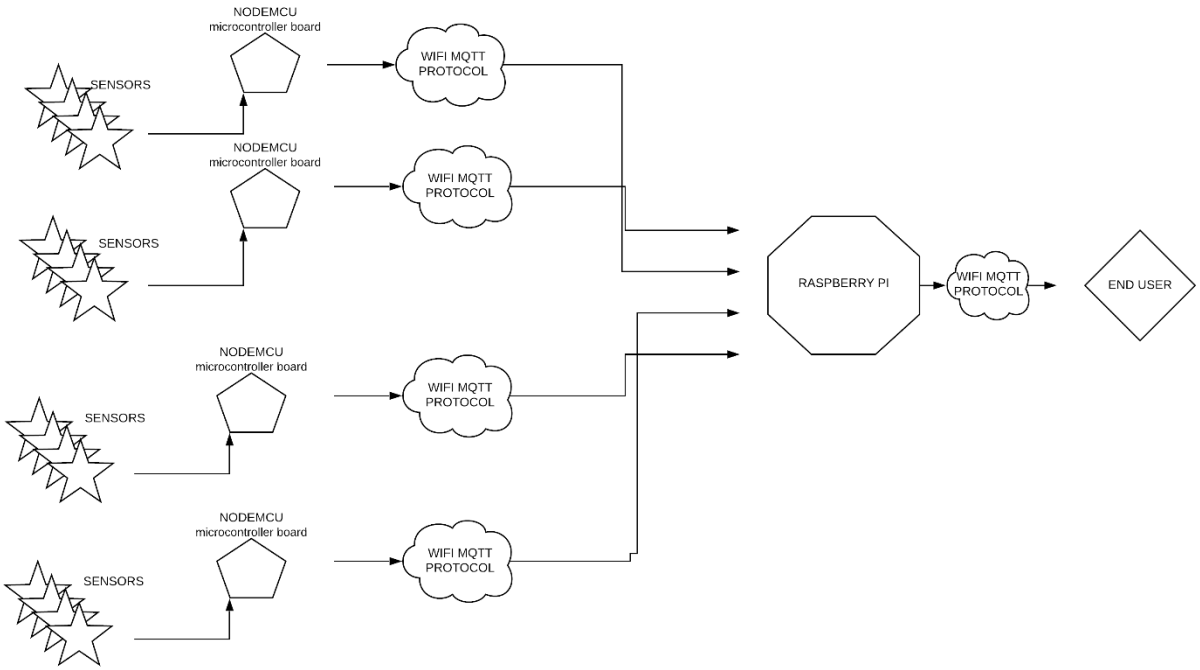
- Μέτρηση στοιχείων περιβάλλοντος σε διάφορους χώρους του σκάφους αλλά και της ατμόσφαιρας με σχετικούς αισθητήρες.
- Μέτρηση στοιχείων λειτουργίας μπαταρίας του σκάφους.
- Ανίχνευση κατάκλισης σε διάφορους χώρους του σκάφους με σχετικούς αισθητήρες.
- Μέτρηση λειτουργικών στοιχείων της μηχανής του σκάφους.
- Αποστολή των στοιχείων και των μετρήσεων αυτών μέσω δικτύου και παρουσίασή τους σε εφαρμογή σε smartphone, είτε tablet του καπετάνιου του σκάφους (end user)

Το πρωτόκολλο επικοινωνίας που θα χρησιμοποιηθεί είναι το MQTT μέσω τοπικού ασύρματου δικτύου. Το ενλόγω πρωτόκολλο μας δίνει την δυνατότητα να έχουμε πρακτικά άπειρα μέρη στην επικοινωνία μας και ως εκ τούτου να μπορούμε να προσαρμόσουμε όσους αισθητήρες επιθυμούμε, χωρίς περιορισμό.

Οι κύριες προκλήσεις που πρέπει να αντιμετωπίσουμε είναι η διατήρηση του χαμηλού κόστους απόκτησης και κατασκευής του συστήματος, η εύκολη προσαρμογή του κάθε χρήστη στην χρήση του γραφικού περιβάλλοντος και η εύκολη εγκατάσταση του

συστήματος από τον καθέναν και την καθεμία. Τέλος θα πρέπει το σύστημα να ειναι στιβαρό και να αντέχει σε περιβάλλον με κραδασμούς, καθώς και θα πρέπει να τροφοδοτείται από την παροχή ρεύματος του σκάφους ώστε να μην υπάρχει ζήτημα επάρκειας ρευματοδότησης όσο αφορά την λειτουργία του.

Μία απλή σχεδιαγραμματική περιγραφή του συστήματος αυτού παρουσιάζεται στο παρακάτω σχήμα



Σχήμα 1.1 Σχεδιαγραμματική Περιγραφή Συστήματος

1.3 Περιγραφή Λύσης

Η γενικότερη ιδέα είναι να κατασκευαστεί ένα σύστημα το οποίο κατα την εκκίνηση της μηχανής του σκάφους, θα ξεκινά την λειτουργία του αυτόματα, θα θέτει σε λειτουργία τους επιμέρους αισθητήρες του, θα δημιουργεί ένα τοπικό ασύρματο δίκτυο WiFi, στο οποίο θα συνδέονται η χρήστρια ή όποια άλλη end user επιθυμεί να παρακολουθήσει τις λειτουργίες του σκάφους, και μέσω μιας ήδη υπάρχουσας

εφαρμογής για android συσκευές, θα μπορεί να λάβει τα αποτέλεσματα των μετρήσεων που δίνει το σύστημα.

Τα απαραίτητα δομικά στοιχεία του συστήματος είναι τα εξής :

- **Ακραία στοιχεία.**

  Πρόκειται ουσιαστικά για ομάδες αισθητήρων οι οποίες επικοινωνούν η κάθε ομάδα με τον δικό της μικροεπεξεργαστή, τοποθετημένες σε διαφορετικά σημεία του σκάφους, με τέτοιο τρόπο ώστε να μπορούν να εκτελούν τις επιθυμητές μετρήσεις. Οι καλωδιώσεις πηγαίνουν παράλληλα με τις ήδη υπάρχουσες του σκάφους, ώστε να υπάρχει πρόσβαση σε διαφορετικούς χώρους, χωρίς την παρουσία του μικροεπεξεργαστή σε κάθε έναν χώρο ξεχωριστά.

- **Κεντρικός διανομέας δικτύου.**

  Είναι η μονάδα διαμοιρασμού της πληροφορίας μέσα στο σκάφος, συνδέοντας τα δύο άκρα. Το ένα άκρο ειναι το πλήθος των μικροεπεξεργαστών που κατέχουν την πληροφορία που μας ενδιαφέρει και το άλλο άκρο είναι η end user που στην περίπτωσή μας ειναι ο δέκτης της πληροφορίας. Ακόμα μία πολύ σημαντική λειτουργία της μονάδας αυτής είναι η υλοποίηση ενός τοπικού ασύρματου δικτύου, στο οποίο συνδέονται όλα τα άκρα του συστήματος.

- **Γραφικό περιβάλλον οπτικοποίησης και παρουσίασης των δεδομένων.**

  Στο νέο τεχνολογικό ρεύμα του Internet of Things είναι απαραίτητη η διασύνδεση των συστημάτων με την χρήστρια. Είναι ανάγκη η πληροφορία να παρουσιάζεται στον άνθρωπο με τρόπο κατανοητό ώστε να μπορεί να την διαχειριστεί και να την αξιοποιήσει. Εδώ λοιπόν έχει χρησιμοποιηθεί μια έτοιμη πλατφόρμα για κινητά τηλέφωνα και tablets που οπτικοποιεί με απλό και ευανάγνωστο τρόπο την πληροφορία για να διευκολύνει την χρήστρια.

- **Θήκες για τους μικροεπεξεργαστές.**

Καλούμενοι να απαντήσουμε στο πρόβλημα προσαρμωγής του συστήματος στο σκάφος, επιλέξαμε την χρήση κουτιών τα οποία εκτυπώθηκαν σε τρισδιάστατο εκτυπωτή και ύστερα προσαρμώστηκαν και στερεώθηκαν στα διάφορα σημεία που χρειαζόταν.

1.4 Απαιτήσεις

Εξαιτίας της ιδιαιτερότητας του περιβάλλοντος που εγκαταστάθηκε το σύστημα, έγινε σύγκριση μεταξύ διαφόρων λύσεων όσο αφορά την συνολική υλοποίηση του προβλήματος.

Ο κύριος γνώμονας ήταν ότι πρέπει να κατασκευαστεί ένα σύστημα το οποίο θα είχε την δυνατότητα να φέρνει σε πέρας όλες τις λειτουργίες του, χωρίς καμία ανάγκη για εξωτερική επίβλεψη από άνθρωπο και χωρίς να υπάρχει η ανάγκη για ανθρώπινη παρέμβαση σε περίπτωση σφάλματος του συστήματος. Συνεπώς σε επίπεδο δημιουργίας λογισμικού, ρευματοδότησης αλλά και επιλογής hardware έπρεπε να επιλεγεί μία λύση η οποία θα μας καλύπτει στους παραπάνω τομείς.

Ακόμη υπάρχει η ανάγκη για άμεση διασύνδεση της πληροφορίας, καθώς σε κάποιες περιπτώσεις, όπως για παράδειγμα στο ενδεχόμενο κατάκλησης, δεν υπάρχει δυνατότητα καθυστέρησης. Άρα είναι σαφές ότι έχουμε απαίτηση για ένα πρωτόκολλο επικοινωνίας το οποίο θα είναι ελαφρύ, γρήγορο, αξιόπιστο και αρκετά απλό στην υλοποίησή του, με κατα το δυνατόν λιγότερα βήματα σύνδεσης των επιμέρους συσκευών, ώστε να πετύχουμε ένα ποσοστό επιτυχίας ζεύξης της επικοινωνίας όσο το δυνατόν ποιο κοντά στο 100%.

Επιπρόσθετα έχουμε κάποιες πρόσθετες λειτουργικές απαιτήσεις, όπως για παράδειγμα την στιβαρότητα της κατασκευής και την απουσία οποιουδήποτε κινούμενου μέρους στο σύστημα. Είναι απαραίτητη η ύπαρξη μιας υποτυπώδους προστασίας απο χτυπήματα , αλλά και προστασία από νερό και υγρασία. Ακόμα τα μέρη στα οποία θα

προσαρτηθούν οι επιμέρους μονάδες επεξεργασίας (οι μικροεπεξεργαστές), θα πρέπει να είναι τέτοια ώστε να μην ενοχλούν και να μην εμποδίζουν την διέλευση των ανθρώπων, ειδικά σε ένα περιβάλλον με στενούς χώρους, όπως τα μικρά σκάφη αναψυχής

Τέλος πρέπει κάποια απο τα στοιχεία τα οποία de facto θα έρχονται σε επαφή με νερό ή με καιρικά φαινόμενα να είναι κατάλληλα για αυτή την χρήση.

1.5 Οργάνωση Τόμου

Η παρούσα διπλωματική εργασία απαρτίζεται από 7 Κεφάλαια. Το παρόν πρώτο κεφάλαιο περιγράφει το πρόβλημα και παραθέτει μια IoT λύση. Στο 2^{ο} κεφάλαιο γίνεται μία εισαγωγή στην τεχνολογία του Internet of Things. Στο 3^{ο} κεφάλαιο περιγράφεται η αρχιτεκτονική του συστήματος, ο σχεδιασμός των κουτιών προστασίας των πλακετών, καθώς και η διαδικτυακή εφαρμογή που επιλέχθηκε για την παρουσίαση των αποτελεσμάτων. Στο 4^{ο} κεφάλαιο παρουσιάζονται με λεπτομέρεια τα πειράματα, οι δοκιμές, η εγκατάσταση σε πραγματικό σκάφος, η λειτουργία σε πραγματικές συνθήκες και στο 5^{ο} κεφάλαιο σημειώνονται συμπεράσματα που προέκυψαν. Το 6^{ο} κεφάλαιο αποτελείται απο τα διάφορα προβλήματα που υπήρξαν καθ' όλα τα στάδια της υλοποίησης του συστήματος, απο την αρχή μέχρι και την χρήση του στην πράξη. Στο 7^{ο} κεφάλαιο παρατείθενται τεχνικά κομμάτια όπως κώδικες, και οδηγίες χρήσης της εφαρμογής. Τέλος παρατίθεται και η βιβλιογραφία.

Κεφάλαιο 2 - Internet of Things

2.1 Ο κόσμος του Internet of Things

2.1.1 Ορισμός της έννοιας

Η γενικότερη ιδέα του Internet of Things είναι η πρόσδοση της διευρυμένης δυνατότητας διασύνδεσης μεταξύ αντικειμένων, αλλά και εξαγωγής, εισαγωγής και διαμοιρασμού των πληροφοριών σε ένα σύστημα μέσω ενός δικτύου, με απότερο σκοπό την εξαγωγή συμπερασμάτων και την εκτέλεση λειτουργιών, έχοντας πάντα ως γνώμονα την βελτίωση της ποιότητας της καθημερινότητας του ανθρώπου. Σε αυτό το νέο κύμα στον τομέα της τεχνολογίας, μεγάλο πλήθος συσκευών και αντικειμένων διαθέτουν τα απαραίτητα ηλεκτρονικά μέσα (όπως για παράδειγμα λογισμικό, αισθητήρες και άλλα) ώστε να επιτύχουν τους σκοπούς που αναφέρθηκαν παραπάνω.

Η έννοια αντικείμενα, ή «πράγματα» , όπως μεταφράζεται το Things στο "Internet of Things" δεν αναφέρεται με συγκεκριμένο τρόπο σε μία κατηγορία αντικειμένων. Χωρίς να λαμβάνει υπόψιν κάποιον περιορισμό στην χρήση τους, στο μέγεθός τους, στο πώς δομούνται, περιλαμβάνει ένα εύρος «πραγμάτων» όπως έξυπνα ψυγεία, πλυντήρια και οικιακές συσκευές, κινητά τηλέφωνα και tablets, ηλεκτρολογικά στοιχεία και λοιπά κομμάτια του ηλεκτρολογικού πίνακα του σπιτιού, πλήθος αισθητήρων, διακόπτες και φώτα, ακόμα και έξυπνα τζάκια και αυτοκίνητα. Το κοινό στοιχείο όλων αυτών είναι η δυνατότητά τους να συνδέονται σε ένα κοινό δίκτυο (το διαδίκτυο) και να συμμετέχουν στο διαμοιρασμό της πληροφορίας. Εξού και το όνομα, «Internet of Things».

Παρά την αυξανόμενη χρήση των διαφόρων μερών του Internet Of Things από μεγάλο μέρος του παγκόσμιου πλυθησμού, ακόμα υπάρχουν πολλοί διαφορετικοί ορισμοί που χρησιμοποιούνται από διάφορες ομάδες για να περιγράψουν ή και να προωθήσουν μία συγκεκριμένη εκδοχή του τι σημαίνει Internet of Things. Ακολουθούν μερικοί από τους πιο διαδεδομένους ορισμούς του Internet of Things:

- IEEE Communication Magazine [1]

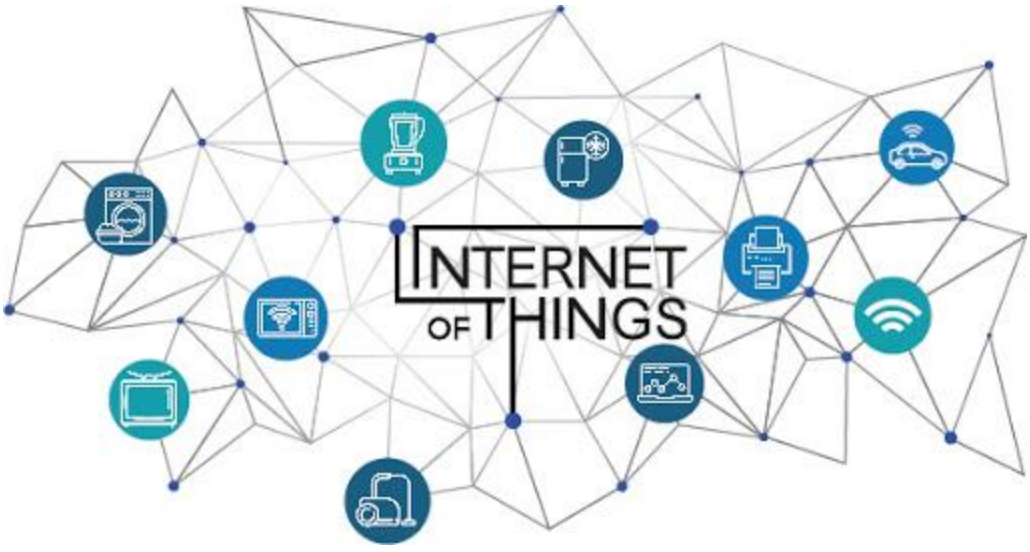
  Το διαδίκτυο των πραγμάτων αποτελεί ένα πλαίσιο στο οποίο όλα τα αντικείμενα έχουν μια ταυτότητα και μία αντιπροσώπευση στον κόσμο του διαδικτύου. Πιο συγκεκριμένα, το Internet of Things έχει ως στόχο να προσφέρει νέες εφαρμογές και υπηρεσίες γεφυρώνοντας τον φυσικό με τον εικονικό-ψηφιακό κόσμο.

- Wikipedia [2]

  Το διαδίκτυο των πραγμάτων (Internet of Things – IoT) είναι το δίκτυο των φυσικών αντικειμένων ή «πραγμάτων», ενσωματωμένο με ηλεκτρονικά, λογισμικό, αισθητήρες και συνδεσιμότητα έτσι ώστε να μπορέσει να επιτευχθεί η ανταλλαγή δεδομένων μεταξύ του κατασκευαστή, του χειριστή ή/και άλλων συνδεδεμένων συσκευών.

- International Telecommunication Union (ITU) [3]

  Διαδίκτυο των πραγμάτων: Μία παγκόσμια υποδομή για την κοινωνία της πληροφορίας που επιτρέπει προηγμένες υπηρεσίες μέσω της διασύνδεσης φυσικών και εικονικών πραγμάτων με βάση την υφιστάμενη και την σε εξέλιξη διαλειτουργικότητα των τεχνολογιών της πληροφορίας και της επικοινωνίας.



Σχήμα 2.1 : Σχηματική απεικόνιση διαφόρων συσκευών του Internet of Things.

2.1.2 Ιστορία του Internet of Things

« Η μεταφορά μικρών πακέτων δεδομένων σε ένα μεγάλο σύνολο κόμβων, προκειμένου να ενοιποιηθούν και να αυτοματοποιηθούν τα πάντα, από μικρές οικιακές συσκευές μέχρι ολόκληρωθεί. »

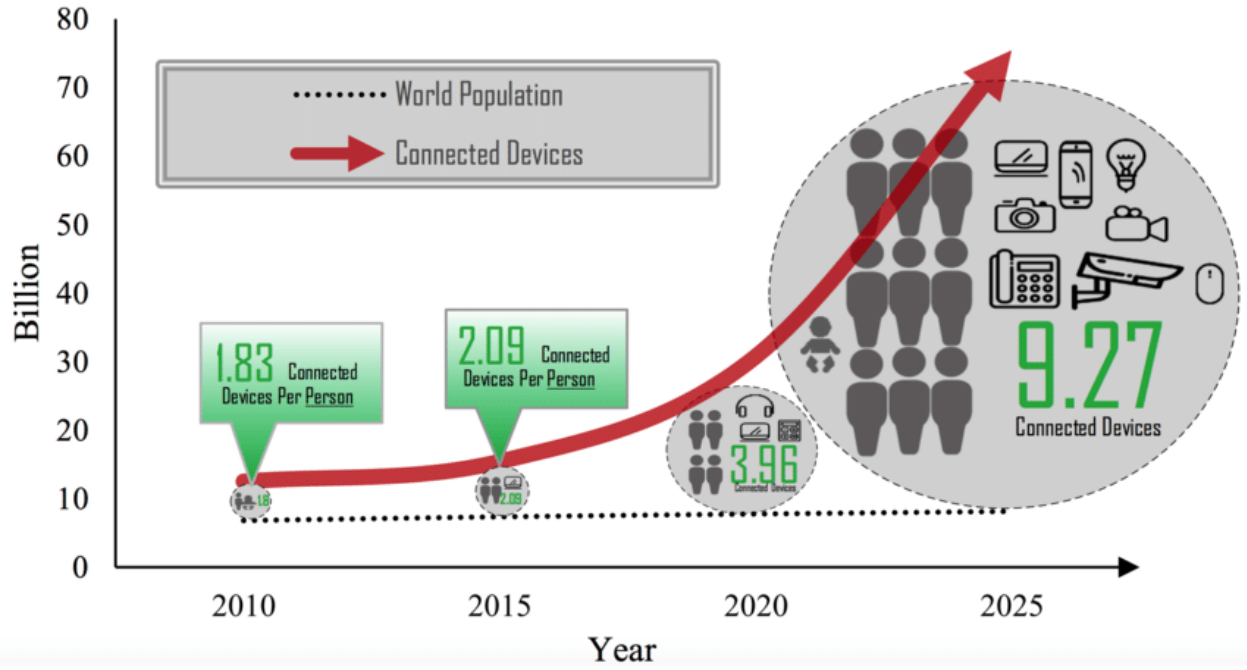
Έτσι περιέγραψε πρώτη φορά το Reza Raji το 1994 την ιδέα, τότε, του Internet of things. Προγενέστερα το 1982 είχε ξεκινήσει μία αφαιρετική υλοποίηση διασύνδεσης μικρών έξυπνων συσκευών, στο Carnegie Mellon University στο οποίο και δημιουργήθηκε και η πρώτη συσκευή που συνδεόταν στο διαδίκτυο. Ήταν ένας αυτόματος πωλητής αναψυκτικών που έλεγχε και έκανε post ένα status που αφορούσε την θερμοκρασία των αναψυκτικών που τοποθετούνταν. Ακολούθησε μία σειρά από τον Mark Weiser μια σειρά από δημοσιεύσεις το 1991, η οποίες αφορούσαν την εξέλιξη των ηλεκτρονικών υπολογιστών και έθεσαν τις βάσεις για την ανάπτυξη του Internet of Things.

Ύστερα από την διατύπωση του Reza Raji, ακολούθησαν διάφορες προτάσεις για το ζήτημα κατά την επόμενη δεκαετία, με σημείο καμπής το 1999 όπου κατά το Παγκόσμιο Οικονομικό Φόρουμ προτάθηκε η επικοινωνία συσκευής – συσκευής και μέσω του Auto-ID Center του MIT έγινε πλέον ευρέως γνωστή με τον όρο Internet of Things. Θεωρήθηκε απαραίτητο το να βρεθεί ένας τρόπος να υπάρξει ένας χαρακτηρισμός όλων των αντικειμένων – μία ταυτοτητοποίηση δηλαδή (ID) ώστε να μπορούν οι χρήστες-υπολογιστές του δικτύου να μπορούν να τα αναγνωρίζουν. Έτσι αρχικά κατευθύνθηκαν προς τη χρήση του RFID (Radio Frequency Identification – Ταυτοποίηση με χρήσης ραδιοσυχνοτήτων) η οποία και θεωρήθηκε ως προυπόθεση για την λειτουργία και την εξέλιξη του Internet of Things. Πληθώρα άλλων τεχνολογιών όπως επικοινωνία κοντινού πεδίου (NFC , Near Field Communication), QR codes, Barcodes είτε ψηφιακό υδατογράφημα.

Η αυστηρή χρήση του ορου καθιερώθηκε μετά από μία παρουσίαση του Kevin Ashton, founder του Auto ID center το 1999 και από εκεί και ύστερα τα τελευταία 20

χρόνια περιγράφει έναν ολόκληρο τεχνολογικό τομέα, με ένα φάσμα εφαρμογών που περιλαμβάνει από νοσοκομειακές εγκαταστάσεις και βιομηχανικές εφαρμογές μέχρι οικιακή χρήση, hobbies και tech-enthusiasts-applications μέχρι ολόκληρους τομείς στο πεδίο των δημόσιων υπηρεσιών, αυτοδιοίκησης και κοινωνικών μεταφορών. Η έκρηξη που παρατηρήθηκε κατά τα τελευταία χρόνια της πρώτης δεκαετίας του 2000 στο πεδίο χρήσης του Intert of Things με εφαρμογή του σε μεγάλο πλήθος συσκευών, έθεσε και επι χάρτου το γεγονός ότι πλέον ο όρος "THINGS" έχει μεταβηλθεί μεν, αλλά ο στόχος παρέμενε ο ίδιος. Η διευκόλυνση της ανθρώπινης ζωής μέσω της χρήσης του διαδικτύου.

Όλα αυτά επιβεβαιώνονται από την διαρκώς αυξανόμενη αναλογία συνδεδεμένων συσκευών από το 2003 με πλυθησμό 6.3 δισεκατομμύρια ανθρώπων με 500 εκατομμύρια συσκευών, στο 2020 με 30 δισεκατομμύρια συνδεδεμένες συσκευές και μόλις 7.3 δισεκατομμύρια ανθρώπων. (σχήμα 2)



Σχήμα 2.2

2.1.3 Χρησιμότητα του Τεχνολογικού ρεύματος Internet of Things.

Υπό το πρίσμα της διαρκούς αναζήτησης λύσεων σε ανθρώπινα προβλήματα της βελτίωσης της ποιότητας της ζωής μας, αλλά και την προστασία του περιβάλλοντος βρίσκουμε πολλές εφαρμογές κατά τις οποίες το Internet of Things θα μπορούσε να δώσει απαντήσεις και πρακτικές λύσεις.

Για παράδειγμα στον τομέα της αυτοκίνησης, εναν τομέα που είναι βαθιά ριζομένος στην καθημερινότητά μας, ήδη το IoT έχει περιλάβει σαν αντικείμενο κάτω από την ομπρέλα του το μεγαλύτερο μέρος του, ειδικά τα τελευταία χρόνια. Αυτό γίνεται σαφές αν σκεφτούμε πως πλέον ένα πολύ μεγάλο κομμάτι των δεδομένων που καταγράφονται από τα επιμέρους λογισμικά των σύγχρονων αυτοκινήτων αλλά και διάφορων άλλων πληροφοριών που έρχονται ως output από τα περιφεριακά συστήματα ελέγχου και λοιπά αισθητήρια, διαμοιράζονται στα γειτονικά συστήματα, είτε και σε κάποια κεντρικοποιημένη υπηρεσία επιτήρησης (monitoring) του συστήματος. Πέρα από το κομμάτι της εσωτερικού διαμοιρασμού της πληροφορίας (data interchange), φυσικά έχουμε και παραδείγματα εκτός συστήματος (data exchange) όπως για παράδειγμα λειτουργικά πλοήγησης και πρόβληψης της κίνησης στους δρόμους, η ενημέρωση σε πραγματικό χρόνο για τυχόν ατυχήματα, προς αξιοποίησή της πληροφορίας από ασθενοφόρα για παράδειγμα και άλλα πολλά.

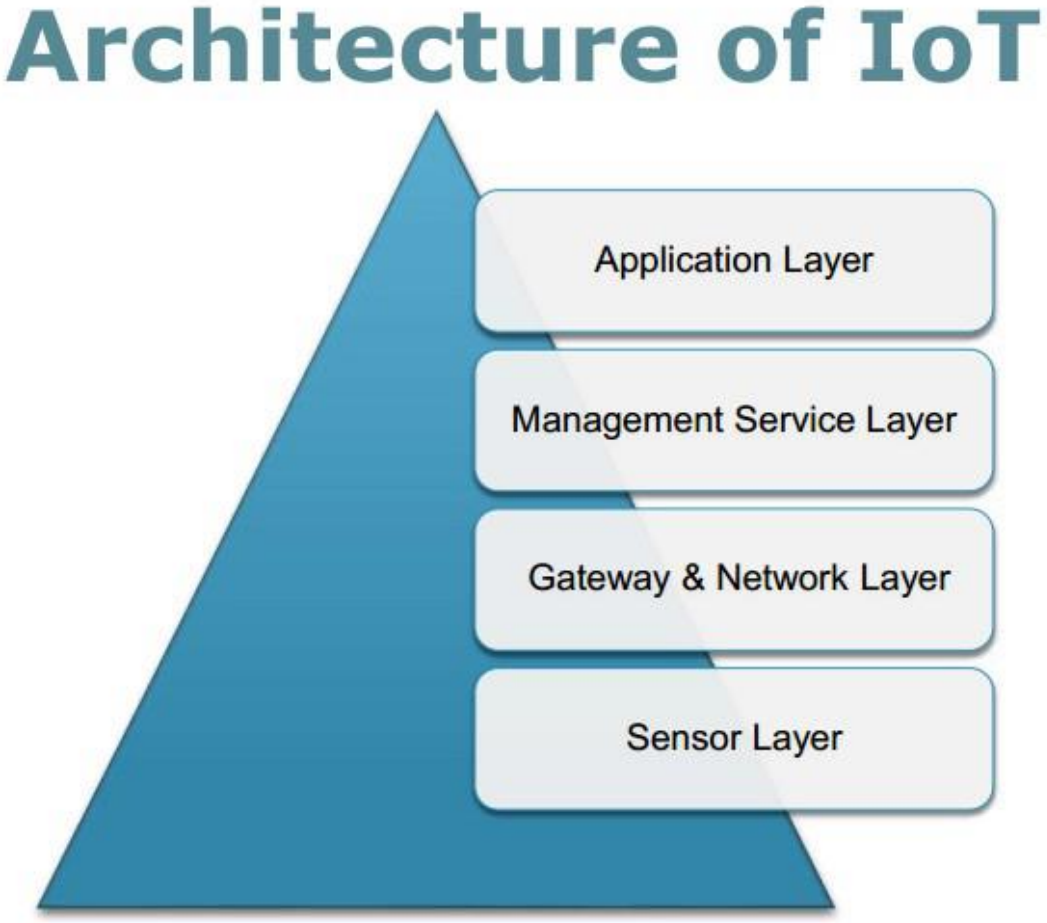
Στον τομέα επίσης της βελτίωσης της καθημερινότητας των ανθρώπων, έχουμε εμπλοκή του IoT στην κτηνοτροφία και την γεωπονία ή και το σύστημα υγείας το ίδιο. Χρόνια τώρα κατασκευάζονται έξυπνες φάρμες, με πλήθος αισθητήρων που παρέχουν δεδομένα όπως υγρασία στο έδαφος, μετεωρολογικά στοιχεία, συστήματα γεωεντοπισμού του κοπαδιού και άλλες πολλές εφαρμογές που ουσιαστικά απλουστεύουν την ανθρώπινη επέμβαση στην πράξη, καθώς ψηφιοποιούν πολλές διαδικασίες που υπό κανονικές συνθήκες θα ήταν δύσκολες και απαιτητικές σωματικά και ψυχικά. Αντίστοιχα στο κομμάτι της υγείας, πάλι με το ίδιο μοτίβο συλλογής και διαμοιρασμού της πληροφορίας, ουσιαστικά

έχουμε κεντρικοποίηση της επιμέρους διαμοιρασμένης πληροφορίας από τα διάφορα μηχανήματα που παρακολουθούν έναν ασθενή, και ως εκ τούτου, μας δίνεται με πολύ πιο εύκολο τρόπο η δυνατότητα για συνολική αξιοποίησή τους, σε πολύ λιγότερο χρόνο και πολύ πιο αποτελεσματικά. Είναι σημαντικό σημείο εδώ για να σταθούμε, καθώς γίνεται σαφές πως το IoT και η επένδυση σε αυτό, μπορεί να σώσει ζωές.

Άλλος πολύ διαδεδομένος τομέας τα τελευταία χρόνια στον οποίο όχι μονο κυριαρχεί το Internet of Things, αλλά ουσιαστικά βασίζεται σε αυτό, είναι η διαδικασία μετατροπής του σπιτιού μας σε «έξυπνο» ("Smart Home"). Υπάρχει πληθώρα εφαρμογών, η οποία εκτείνεται από την απομακρυσμένη διαχείριση συσκευών (πληντύρια, κλιματιστικά, θερμοσίφωνες και λοιπες συσκευες), την παρακολούθηση της ενεργειακής κατανάλωσης του σπιτιού και των συσκευών ξεχωριστά, καθώς και διάφορες «έξυπνες» τροποιποιήσεις με χρονορύθμιση φώτων, πατζουριών και λοιπά, ανάλογα με την ώρα της ημέρας. Αυτό μπορεί να μειώσει αρκετά τον χρόνο και τις έννοιες που έχει κανείς ως προς την λειτουργικότητα του σπιτιού του, καθώς όλα είτε μπορούν να αυτοματοποιηθούν, είτε πολύ εύκολα να τα χειριστεί κανείς από το τηλέφωνό του.

Ένα άλλο κεφάλαιο στο οποίο βρίσκουμε έντονη χρήση του Internet of Things, είναι η βιομηχανία, βαριά ή όχι, αλλά και γενικότερα οι επιχειρήσεις. Η μετάβαση σε paperless office καθώς και η διαρκής παρακολούθηση όλων των συστημάτων της εταιρίας, είναι ένα πολύ μικρό κομμάτι των εφαρμογών. Δίνεται ουσιαστικά η δυνατότητα για πολύ αποδοτικότερη διαχείριση των στοιχείων που έτσι και αλλιώς θα χρειάζονταν μελέτη και αξιολόγηση, τώρα όμως έχοντας την δυνατότητα και την ευχαίρια να γίνονται σε πραγματικό χρόνο, και χωρίς ουσιαστικό κόπο. Επίσης καθίσταται άχρηστη η υπερβολικά χρονοβόρα διαδικασία του data entry, πράγμα που έχει πολλαπλά οφέλει, κυρίως οικονομικά.

2.1.4 Αρχιτεκτονική των IoT Συστημάτων



Σχήμα 3.1

Ο σχεδιασμός της αρχιτεκτονικής του Internet of Things εμπεριέχει το δίκτυο, τους αισθητήρες, τις υπηρεσίες και την διαχείριση των δεδομένων τους καθώς και το κομμάτι της ασφάλειας. Είναι απαράιτητο τα στοιχεία αυτά να αλληλοσυνδέονται και εκεί ουσιαστικά παίρνει σάρκα και ουσία ο όρος «αρχιτεκτονική του διαδικτύου των πραγμάτων».

Απαραίτητη προϋπόθεση σαφώς είναι και η ύπαρξη εύκολης λειτουργικότητας μεταξύ συσκευών διαφορετικής φύσης. Έτσι οι εφαρμογές πρέπει να είναι βασισμένες σε υπηρεσίες service oriented architecture, ώστε η διαχείριση πολύπλοκων συστημάτων να

γίνεται με απλό και τμημένο τρόπο, αντιμετωπίζοντας τα επιμέρους στοιχεία του κάθε συστήματος ως ξεχωριστές «οντότητες» και επαναχρησιμοποιώντας τα σε διαφορετικά συστήματα Internet Of Things.

Τα 4 επίπεδα που ουσιαστικά απαρτίζουν με τεχνικό τρόπο τις εφαρμογές Internet of Things (Internet of Things layers) είναι τα εξής:

- **Sensor Layer**

  Το πρώτο επίπεδο που αποτελεί ουσιαστικά την βάση των συστημάτων Internet of Things, εμπεριέχει ηλεκτρονικές συσκευές, μικρού μεγέθους, μικρής κατάναλωσης και με παροχή ρεύματος από μπαταρίες, ώστε να μπορούν να είναι αυτόνομες και ανεξάρτητες ως προς το σύστημα. Έτσι δομούν όλες μαζί το πλήθος της πληροφορίας στα πρωτόλεια στάδια του «ταξιδιού» της που στο τέλος θα βοηθήσουν στην κατανόηση, παρατήρηση και συστηματική καταγραφή ενός φαινομένου.

  Τα βασικά στοιχεία που απαιτούνται να έχει μία συσκευή για να είναι μέρος του Internet of Things είναι :

  - Να έχει μία ελάχιστη υπολογιστική ισχύ ώστε να εκτελεί βασικούς υπολογισμούς.
  - Να είναι σε θέση να καταγράφει δεδομένα και να τα αντιλαμβάνεται στο περιβάλλον της
  - Να έχει την δυνατότητα συννδεσης στο διαδίκτυο ώστε να μπορεί να στείλει αυτά τα δεδομένα

- **Gateway & Network Layer**

  Το επίπεδο του δικτύου είναι το απαραίτητο συστατικό για την δημιουργία μίας εφυούς διασύνδεσης των συσκευών της βάσης. Συγκεντρώνει τα δεδομένα και τα μεταφέρει με ασφαλή τρόπο στην κεντρικοποιημένη υπηρεσία που είναι υπεύθυνη για την μετέπειτα διαχείρισή τους. Συνήθως οι

πύλες επικοινωνίας Internet of Things για λόγους χρηστικότητας και προσαρμοστικότητας, έχουν πολλές δυνατότητες για επιλογή πρωτοκώλλου ασύρματης επικοινωνίας, με κύριο γνώμονα την χαμηλή ενεργειακή κατανάλωση. Για παράδειγμα Bluetooth Low Energy, Zigbee, LoRa WAN, WPAN και άλλα.

- **Management-Service Layer**

Το σύνολο των δεδομένων που μεταφέρονται από τους αισθητήρες, μέσω των πυλών IoT, εντέλει αποθηκεύονται σε διακομιστές που υπάρχουν στο cloud, είτε σε κάποια βιομηχανική-εργαστηριακή φυσική εγκατάσταση. Έτσι μέσω αυτών των διακομιστών, τα δεδομένα επεξεργάζονται και αναλύονται περαιτέρω με σκοπό την δημιουργία live πινάκων, κατανομών και διαγραμμάτων αποτύπωσης των εν λόγω δεδομένων, ώστε να γίνεται ευκολότερη και αποτελεσματικότερη η λήψη αποφάσεων και η εξαγωγή εκτιμήσεων για διάφορα ζητήματα που αφορούν το φαινόμενο που μελετάμε κάθε φορά.

- **Application Layer**

Εδώ έχουμε το επίπεδο που ο μέσος χρήστης γνωρίζει και είναι εξοικιωμένος περισσότερο από κάθε άλλο, το επίπεδο της διεπαφής ανθρώπου – συστήματος (μηχανής). Σύμφωνα με τις ανάγκες του τελικού χρήστη (end user) , μπορούμε να έχουμε πληθώρα εφαρμογών που να τις καλύπτουν, όπως για παράδειγμα εφαρμογή στο κινητό που να δίνει δυνατότητα ελέγχου λειτουργιών του σπιτιού μας, σύνδεση σε ιστοσελίδα για monitoring μεταβλητών σε βιομηχανικές εγκαταστάσεις και άλλα πολλά.

Στο σχήμα 3 στην αρχή αυτής της υποενότητας παρουσιάζονται συγκεντρωτικά τα επίπεδα του Internet of Things.

2.2 Οι προκλήσεις του Internet of Things.

Η φυσική συνέχεια της εισόδου μια νέας τεχνολογίας με τόσες μεγάλες δυνατότητες στο κόσμο της τεχνολογίας, είναι η αναπόφευχτη δημιουργία νέων προκλήσεων όσο αφορά την υλοποίησή της. Η συνδεσιμότητα, η πολυπλοκότητα και η δόμηση των δικτύων που μεταφέρουν έναν μεγάλο όγκο πληροφορίας, η διαχείριση αυτού του όγκου, καθώς και η επίλυση του ζητήματος της πολλαπλής συνδεσιμότητας χρηστών / συσκευών, είναι μόνο μερικά από τα ζητήματα που πρέπει να επιλυθούν από τους εμπλεκόμενους φορείς και ανθρώπους , υπό την δυσκολία που προσφέρει η νεανικότητα της συγκεκριμένης τεχνολογία που περιορίζει εκ των πραγμάτων τις δυνατότητες επίλυσης των διαφόρων ζητημάτων.

Αναλυτικότερα:

- *Διαχείριση και επεξεργασία του μεγάλου όγκου δεδομένων*
  Γίνεται εύκολα αντιληπτό το ζήτημα το οποίο προκύπτει από την χρήση μιας τεχνολογίας από μεγάλο αριθμό χρηστών, και δεν είναι άλλο από τον τεράστιο όγκο πληροφορίας που χρειάζονται για την αποθήκευσή του, αλλά και της απαιτούμενης υπολογιστικής ισχύος η οποία θα είναι ικανή να επεγεργαστεί τα ενλόγω δεδομένα.
- *Ασφάλεια και ακεραιότητα*
- *Τα προσωπικά δεδομένα και ασφαλής μεταχείρισή τους*
- *Πρωτόκολλο Δικτύου*
- *Ενεργειακές Απαιτήσεις*

Κεφάλαιο 3 - Αρχιτεκτονική του Συστήματος

3.1Εισαγωγή

Στο κεφάλαιο αυτό λαμβάνοντας υπόψιν τις απαιτήσεις του προβλήματος, ουσιαστικά θα περιγραφούν αναλυτικά οι επιλογές των υλικών και μη στοιχείων που επιλέξαμε για ικανοποιήσουμε τις εν λόγω απαιτήσεις. Στα πλαίσια αυτής της διπλωματικής εργασίας δημιουργήθηκε ένα ολοκληρωμένο σύστημα από αισθητήρες όπως ανεμόμετρο, θερμόμετρο, υγρασιόμετρο, βαρόμετρο, βολτόμετρο, αμπερόμετρο, πιεσόμετρο και θερμόμετρο λαδιού, καθώς και χρήση μετατροπέων τάσης αλλά και μικροεπεξεργαστών και μιας φορητής υπολογιστικής μονάδας. Στο επόμενο στάδιο χρησιμοποιήθηκε ένας εξυπηρητητής MQTT που ανέλαβε τον ρόλο του διακομιστή της πληροφορίας από τους μικροεπεξεργαστές που λαμβάνουν τα δεδομένα των αισθητήρων προς την τελική χρήστρια/ χρήστη. Τέλος μια εφαρμογή σε περιβάλλον android χρησιμοποιήθηκε για να οπτικοποιηθούν τα αποτελέσματα, με τέτοιο τρόπο ώστε να είναι ευανάγνωστα και άμεσα προσπελάσιμα από την ή τον καπετάνιο του σκάφους.

3.2 Επιλογή Πλατφόρμας Internet of Things

Λαμβάνοντας υπόψιν τους παρακάτω συγκεκριμένους παράγοντες, θα διαλέξουμε μια συσκευή που να μας δίνει την δυνατότητα να υλοποιήσουμε στην πράξη την λύση του αρχικού προβλήματός μας.

- Ενσωματωμένη κεραία και δυνατότητα χρήσης πρωτοκόλλου WiFi.
- Πολλαπλές είσοδοι αναλογικού και ψηφιακού σήματος ώστε να υπάρχει συνδεσιμότητα με διαφόρων ειδών αισθητήρες.
- Χαμηλή κατανάλωση ενέργειας για να μην έχουμε αυξάνουμε τον φόρτο του δικτύου του σκάφους, το οποίο εκ φύσεως έχει χαμηλές δυνατότητες.

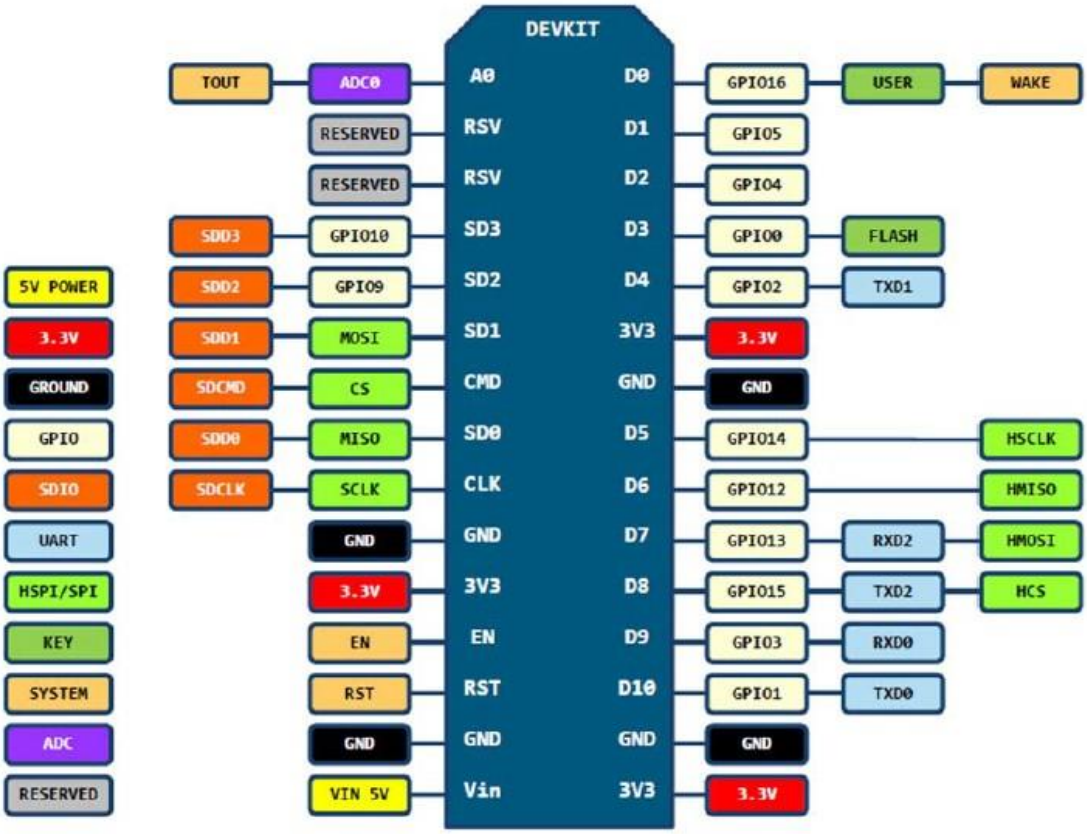
- Μικρό μέγεθος ώστε να μπορεί να προσαρμοστεί σε όσο δυνατόν μικρότερη πλακέτα.
- Δυνατότητα χρήσης γλώσσας προγραμματισμού υψηλού επιπέδου, ώστε να διευκολύνεται η υλοποίησή του project.
- Δυνατότητα χρήσης βιβλιοθηκών που να μπορούν να υποστηρίξουν την χρήση των αισθητήρων καθώς και την μεταφορά της πληροφορίας μέσω του κατάλληλου πρωτοκόλλου
- Το πολύ χαμηλό κόστος σε συνάρτηση με την αξιοπιστία της κατασκευής καθώς πρώτων είναι απαραίτητη η προμήθεια αρκετών τέτοιων μονάδων και στα πλαίσια της παρούσας διπλωματικής οι πόροι είναι περιορισμένοι και δεύτερον σε ένα πιθανό μελλοντικό εμπορικό επίπεδο πρέπει να πετύχουμε ένα όσο το δυνατόν χαμηλό κόστος παραγωγής.

Με βάση τα παραπάνω θα κινηθούμε σε μία συσκευή που κάνει χρήση του μικροεπεξεργαστή ESP8266. Είναι κατασκευασμένος από την Espressif Systems και έχει WiFi module. Χρησιμοποιείται ευρέως για Internet Of Things εφαρμογές και συστήματα αυτοματισμού σε οικιακά περιβάλλοντα (Smart Home & Home automation).

Η πλέον ενδεδειγμένη επιλογή συσκευής είναι το NodeMCU , καθώς είναι η πιο διαδεδομένη συσκευή αυτού του είδους. Στο διαδίκτυο υπάρχει πληθώρα πληροφορίας, forum που μπορεί κανείς να ανατρέξει για λύσεις σε τυχόν προβλήματα που μπορεί να παρουσιαστούν, καθώς και έτοιμες λύσεις για να μπορέσει ο οποιοσδήποτε με στοιχειώδεις γνώσεις χρήσης ηλεκτρονικού υπολογιστή να ξεκινήσει ένα δικό του project.

3.2.1 Πλεονεκτήματα του NodeMCU

- Ευκολία προγραμματισμού και υλοποίησης του project με χρήση του ευρέως διαδεδομένου προγράμματος Arduino IDE, που δημιουργήθηκε για αντίστοιχες πλατφόρμες τύπου Arduino
- Ύπαρξη ιδιαίτερα μεγάλης κοινότητας ατόμων που χρησιμοποιούν ευρέως τις διάφορες συσκευές της Arduino, με αποτέλεσμα να υπάρχει διαθέσιμος ένας μεγάλος όγκος πληροφορίας στο διαδίκτυο για οριακά οποιοδήποτε ζήτημα χρειαστεί να επιλυθεί, και αυτό με όρους συλλογικής και ανιδιοτελούς διάθεσης της πληροφορίας, χωρίς λογικές κέρδους.
- Αντίστοιχα μεγάλη γκάμα βιβλιοθηκών που δίνουν την δυνατότητα επικοινωνίας με κάθε λογής αισθητήρες, είτε αναλογικού σήματος είτε ψηφιακού.
- Γλώσσα προγραμματισμού αντίστοιχη με αυτή που χρησιμοποιείτα στα προϊόντα της Arduino, που είναι υψηλού επιπέδου και είναι βασισμένη στην C.
- Χαμηλό κόστος κτήσης, κάτω από 2-3 ευρώ ανά μονάδα. Έτσι υπάρχει η δυνατότητα αρκετά εύκολα να μπορεί να πειραματιστεί κανείς με άνεση, χωρίς φόβο καταστροφής κάποιας μονάδας, αλλά και με άνεση να διευρύνει το project του ανά πάσα στιγμή χωρίς ιδιαίτερη αύξηση του κόστους κατασκευής.
- Ονομαστική τάση λειτουργίας είναι τα 3.3V και έχει 11 ψηφιακές εισόδους και μία αναλογική.



3.1 Node MCU pin layout

3.3 Αισθητήρες

Σε αυτό το κεφάλαιο να περιγραφούν αναλυτικά όλοι οι αισθητήρες που χρησιμοποιήθηκαν.

- Θερμοϋγρασιόμετρο

Επιλέχθηκε εδώ ένας πολύ διαδεδομένος αισθητήρας που έχει την δυνατότητα να μετράει υγρασία και θερμοκρασία περιβάλλοντος, σε ένα ικανοποιητικό για τις ανάγκες μας εύρος τιμών. Είναι ο AM2302 κινέζικης κατασκευής από την εταιρία AOSONG και μπορεί να μετρήσει θερμοκρασία με ακρίβεια ενός βαθμού °C και με εύρος από -40 °C εώς και 80 °C και υγρασία περιβάλλοντος από 0% εώς και 99.9% με ακρίβεια 2%.
Τα χαρακτηριστικά αυτά μας καλύπτουν πλήρως και σε συνδυασμό με το πολύ χαμηλό κόστος του είναι μια εξαιρετική επιλογή.
Η ονομαστική τάση λειτουργίας του είναι από 3.3V εώς και 5V που είναι ανεκτό από τις δυνατότητες του συστήματος, και θα χρησιμοποιηθεί η 5V παροχή. Το σήμα που δίνει είναι ψηφιακό.

3.2 Αισθητήρας Θερμοκρασίας/Υγρασίας

- Βαρόμετρο

  Εδώ επιλέχθηκε ο νέας γενιάς αισθητήρας της Bosch BMP280, οποίος έχει ενσωματωθεί σε ένα module από την εταιρία Adafruit κάνοντάς τον εύχρηστο και προσβάσιμο στο ευρύ κοινό. Έχει την δυνατότητα ακριβούς θερμομέτρησης με εύρος ±1 βαθμού ºC και βαρομέτρησης με ±1 hPa. Μέσω του module της Adafruit έχουμε την δυνατότητα να του δώσουμε τάση λειτουργίας είτε στα 3.3v είτε στα 5v, ανάλογα με την διαθεσιμότητα του συστήματός μας. Η έξοδός του είναι ψηφιακό σήμα φυσικά.

3.3 *Βαρομετρικός Αισθητήρας*

- Ανεμόμετρο

  Εδώ προμηθευτήκαμε ένα ανεμόμετρο από την Adafruit, με δυνατότητα μέτρησης ανέμων εώς και 32.4 m/s. Είναι στιβαρή κατασκευή και επιστρέφει

αναλογικό σήμα (0.4v – 2.0v), οι τιμές του οποίου με απλή αναλογία αντιστοιχούν στην ελάχιστη (0) και στην μεγιστη τιμή ανέμου. Η τάση λειτουργίας του είναι ευρύτατη, από 7V εώς και 24V συνεχούς ρεύματος (DC). Εμείς εδώ θα τροφοδοτήσουμε τον εν λόγω αισθητήρα με 12V μιας τόσο είναι και η τάση της μπαταρίας που τροφοδοτεί το δίκτυο του σκάφους.

Εικόνα 3.4 Ανεμόμετρο

- Αισθητήρας Κατάκλισης

  Ουσιαστικά είναι ένας αισθητήρας στάθμης νερού, που λειτουργεί με μεταβλητή αντίσταση. Τροφοδοτείται από 3.3v είτε από 5v και ανάλογα το ποσοστό της βρεγμένης επιφάνειας μεταβάλετε η αντίσταση του αισθητήρα και ως εκ τουτου μεταβάλεται ανάλογα και η τιμή του αναλογικού σήματος που δίνει σαν έξοδο ο αισθητήρας.

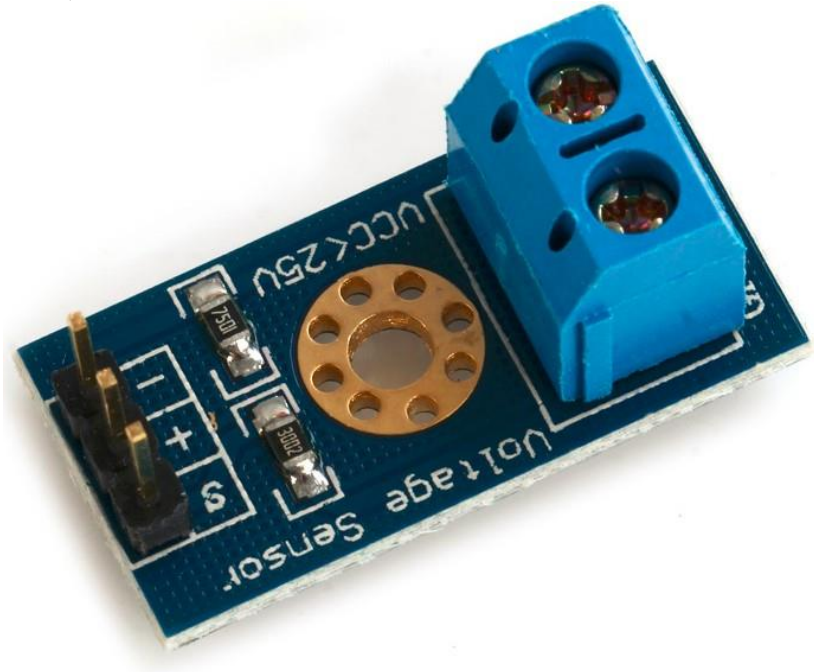
  Εμείς στην συγκεκριμένη περίπτωση τον χρησιμοποιήσαμε χωρίς να κάνουμε κάποια μετατροπή, καθώς οποιαδήποτε τιμή του σήματος μας δείχνει ότι έχουμε παρουσία νερού σε κάποιον συγκεκριμένο χώρο και άρα έχουμε alarm on.

Εικόνα 3.5 Water sensor

- Αισθητήρες ενδείξεων Μηχανής
  Όπως είναι γνωστό κάθε μηχανή σκάφους έχει από κατασκευής της κάποια μετρητικά όργανα / αισθητήρες που συνδέονται με το κεντρικό πάνελ του καπετάνιου. Αυτά συνήθως είναι, αλλά δεν περιορίζονται, θερμοκρασία και πίεση νερού ψύξης μηχανής, θερμοκρασία και πίεση λαδιού μηχανής, στροφές μηχανής και κάποια alarm που ουσιαστικά ενεργοποιούνται όταν οι προηγούμενοι αισθητήρες ξεπεράσουν κάποια προκαθορισμένη τιμή.
  Αποφασίσαμε πως θα ήταν χρήσιμο, μιας και στην δική μας περίπτωση το μικρό σκάφος αναψυχής που χρησιμοποιούμε στα πλαίσια της διπλωματικής έχει αναλογικές ενδείξεις των ενλόγω τιμών, να πάρουμε το αναλογικό σήμα από την καλωδίωση του σκάφους και να το περάσουμε και αυτό και στην τελική οθόνη μαζί με όλες τις υπόλοιπες μετρήσεις. Έτσι θα διευκολύνουμε τον ενδιαφερόμενο να διαβάσει πιο γρήγορα την τιμή που θέλει.

- Ενδείξεις μπαταρίας.
  Είναι σημαντικό για τον καπετάνιο να μπορεί να έχει πρόσβαση σε κάθε δεδομένη χρονική στιγμή στις ενδείξεις της μπαταρίας. Εμείς χρησιμοποιώντας έναν απλό διαιρέτη τάσης παίρνουμε κατευθείαν ρεύμα από το ηλεκτρικό κύκλωμα του σκάφους και έτσι μπορούμε να ξέρουμε την τάση του συστήματος σε κάθε δεδομένη χρονική στιγμή.
  Ο διαιρέτης τάσης δεν είναι τίποτα άλλο από μια απλή σύνδεση 2 συγκεκριμένων αντιστάσεων, που ανάλογα την τιμή τους, διαιρούν με συγκεκριμένη τιμή την τάση εισόδου. Συνεπώς τα 12 v διαιρούνται και φτάνουν

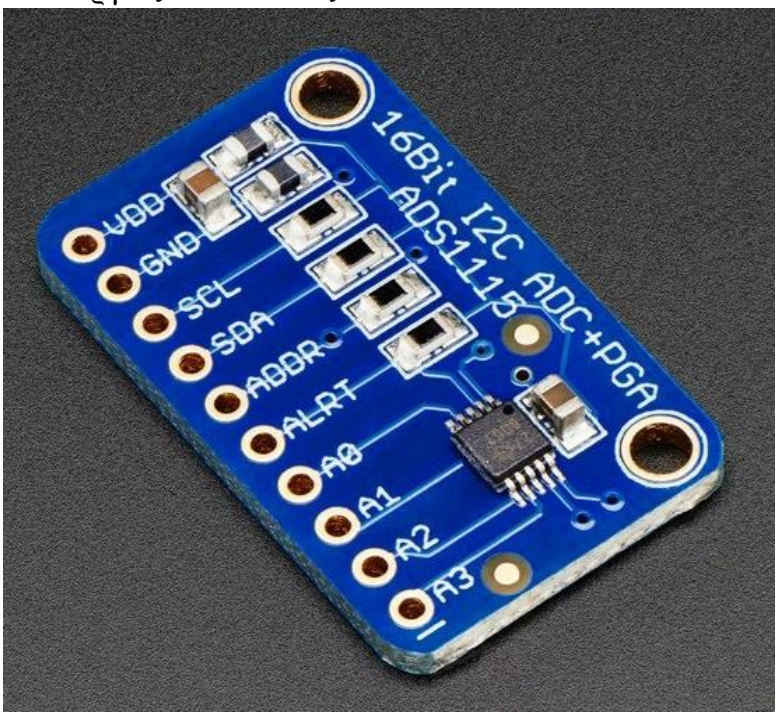
σε επίπεδα που μπορεί να διαχειριστεί ο μικροεπεξεργαστής μας και ύστερα η τιμή τους πολλαπλασιάζεται με την αντίστροφη διαδικασία και αποστέλλεται στον τελικό χρήστη/χρήστρια. Στην δική μας περίπτωση χρησιμοποιήθηκαν διαιρέτες 1/5.



Εικόνα 3.5 Voltage sensor

Για την μέτρηση της έντασης του ρεύματος στο κύκλωμα του σκάφους, χρησιμοποιήσαμε ένα ήδη εγκατεστημένο shunt resistor. Αυτό είναι μία αντίσταση με πάρα πολύ μικρή τιμή, που παράγει μια αντίστοιχα μικρή πτώση τάσης στο κύκλωμα.
Μετρώντας αυτήν την πτώση τάσης μπορούμε εύκολα να υπολογίσουμε την ένταση του ρεύματος.
Είναι αναγκαίο όμως να υπάρχει ένας ενισχυτής σήματος καθώς η μετατροπή ενός τόσο μικρού αναλογικού σήματος σε ψηφιακό σε τόσο μικρές τιμές (σαν αυτές που παίρνουμε από το shunt resistor) δεν έχουν αρκετή ακρίβεια. Συνεπώς εδώ χρησιμοποιήσαμε ένα module της Adafruit που είναι ενισχυτής σήματος και ταυτόχρονα μετατρέπει το αναλογικό σήμα σε ψηφικό. Επιπρόσθετα μας δίνει την δυνατότα για 4 εισόδους αναλογικού σήματος και μετατροπή τους σε ψηφιακό, γεγονός που μας βοηθάει πολύ στην επίλυση του ζητήματος των περιορισμένων αναλογικών θηρών στις πλακετες των μικροεπεξεργαστών NodeMCU.

Συγκεκριμένα είναι το ADS1115 16-Bit ADC , με ονομαστική τάση λειτουργίας από 2V εώς 5.5V.



Εικόνα 3.6 ADC CONVERTER – AMPLIFIER

Εικόνα 3.7 SHUNT RESISTOR

3.4 Μονάδες επεξεργασίας

Ολόκληρο το σύστημα που κατασκευάστηκε αποτελείται από 16 αισθητήρες, είτε αναλογικού σήματος είτε ψηφιακού. Το σήμα του κάθε αισθητήρα πρέπει να σταλεί ενσύρματα σε έναν μικροεπεξεργαστή NodeMCU ώστε να μετατραπεί σε πληροφορία και από να σταλεί προς τον τελικό χρήστη. Αμέσως δημιουργείται ένα πρόβλημα προς επίλυση που είναι το πως θα χωριστούν αυτοί οι αισθητήρες καθώς και βρίσκονται σε απομακρυσμένα μεταξύ τους μέρη, αλλά και πως θα ξεπεραστεί το πρόβλημα με τις περιορισμένες αναλογικές εισόδους στον κάθε μικροεπεξεργαστή NodeMCU.

Για την ρευματοδόσητη της κάθε μονάδας έπρεπε να βρεθεί ένας τρόπος ώστε να μην χρησιμοποιηθούν μπαταρίες. Το ηλεκτρικό κύκλωμα του σκάφους λειτουργεί στα 12v. Χρησιμοποιήθηκε λοιπόν μία πλακέτα μετατροπής ρεύματος με δυνατότητα αποδοχής σαν ρεύμα εισόδου ένα εύρος τάσεων από 9v εώς και 36v, και έξοδο σταθερή τιμή στα 5v (DC-DC Converter Step-Down) με μεγάλη απόδοση χωρίς απώλειες και δυνατότητα εξόδου με θύρα USB αλλά και με ακροδέκτες για καλωδίωση. Εδώ χρησιμοποιήθηκε το καλώδιο χαλκού καθώς ήταν πιο στιβαρή κατασκευή και θα ανταποκρινόταν καλύτερα στους κραδασμούς τους σκάφους σε αντίθεση με μία σύνδεση σε θύρα USB.
Αποφασίστηκε να κατασκευαστούν 4 μονάδες επεξεργασίας που η κάθε μία θα έχει να λαμβάνει σήμα από έναν αριθμό αισθητήρων. Η κάθε μία έχει τις δικές της ιδιατερότητες και γιαυτό θα αναπτυχθούν αναλυτικά παρακάτω.

- MODULE A

  Η πρώτη μονάδα επεξεργασίας είναι συνδεδεμένη με τον αισθητήρα νερού για την κατάκλιση του WC, καθώς πολλές φορές λόγω κραδασμών υπήρχε αποσύνδεση κάποιας παροχής νερού και ταυτόχρονα λόγω αμέλειας ήταν σε λειτουργία η αντλία νερού του σκάφους με αποτέλεσμα να έχουμε κατάκλιση στο WC. Ακόμα είναι συνδεμένη με ένα αισθητήρα θερμοκρασίας – υγρασίας στην καμπίνα του σκάφους, έξω δηλαδή από το WC και τέλος με έναν ακόμα αισθητήρα θερμοκρασίας και υγρασίας στο μηχανοστάσιο του σκάφους, κυρίως για την παρατήρηση της θερμοκρασίας του μηχανοστασίου.
  Η συνδεσμολογία έγινε παράλληλα με την ήδη υπάρχουσα καλωδίωση καθώς τα 3 σημεία που αναφέρθηκαν ήταν αρκετά κοντά και χωρισμένα απλώς από καποια διαχωριστικά χώρων από κόντρα πλακέ θαλάσσης.

- MODULE B

  Η δεύτερη μονάδα επεξεργασίας δεδομένων είναι υπεύθυνη στο να διαχειρίζεται αφενώς τα χαρακτηριστικά της μπαταρίας και αφετέρου για την κατάκλιση του Engine Room. Τοποθετήθηκε στον χώρο του μηχανοστασίου όπου σε πολύ κοντινή της απόσταση τοποθετήθηκε ο αισθητήρας ανίχνευσης νερού. Ύστερα απευθείας από τους πόλους της μπαταρίας γίνεται η λήψη της τιμής της τάσης της, φυσικά αφού πρώτα γίνεται Step Down στα επίπεδα που μπορεί να διαχειριστεί η πλακέτα (>5V). Τέλος μέσω του shunt resistor που υπάρχει τοποθετημένο πριν την μπαταρία, λαμβάνουμε την τιμή για την ένταση του ρεύματος. Αυτή καθότι πολύ μικρή, της τάξης των mV ενισχύεται από τον ADC converter – ενισχυτή σήματος και ύστερα ψηφιοποιείται για να διαχεριστεί από τον μικροεπεξεργαστή.

- MODULE C

  Η τρίτη μονάδα επεξεργασίας εκτελεί χρέη μετεωρολογικού σταθμού, και είναι τοποθετημένη για λόγους οικονομίας καλωδίωσης στον εξωτερικό χώρο δίπλα από το πάνελ οργάνων του καπετάνιου. Πάνω στην πλακέτα υπάρχει ένα θερμόμετρο και ένα βαρόμετρο, ένα θερμουγρασιόμετρο και τέλος είναι συνδεδεμένο και ένα ανεμόμετρο που έχει τοποθετηθεί στα 2 μέτρα από το σκάφος, καθώς δεν υπήρχε εύκολη πρόσβαση στο κατάρτι.

- MODULE D

  Η τέταρτη και τελευταία μονάδα επεξεργασίας είναι αυτή που λαμβάνει τα σήματα από τους ήδη εγκατεστημένους αισθητήρες της μηχανής. Είναι οι 2 αισθητήρες θερμοκρασίας νερού μηχανής και πίεσης λαδιού καθώς και τα 2 αλάρμ τα οποία λειτουργούν στα 4V. Πάλι υπάρχει μία πλακέτα ADC Converter η οποία αυτή την φορά βοηθάει στην διαχείριση πολλαπλών αναλογικών σημάτων χωρίς να προσφέρει κάποια ενίσχυση στο σήμα. Τα ρεύματα από τους 2 αισθητήρες περνάνε φυσικά από Step Down για να έρθουν στα επίπεδα που είναι αποδεκτά από την μονάδα επεξεργασίας (<5V).

3.5 Επιλογή Διακομιστή Δικτύου

Η επικοινωνία μεταξύ των αισθητήρων και των NodeMCU γίνεται ενσύρματα ακολουθώντας ένα αποκεντρωμένο μοντέλο σύνδεσης για λόγους απλότητας και ευκολίας κατασκευής. Η συγκέντρωση της πληροφορίας για την τελική χρήστρια ή τον τελικό χρήστη, αποτελεί το συγκεντρωτικό κομμάτι της διαδικασίας της μεταβίβασης

της πληροφορίας. Αυτό αποφασίστηκε να γίνει ασύρματα καθώς έτσι δεν θα αποτελέσει ποτέ εμπόδιο στην αυξομείωση των μονάδων επεξεργασίας. Γιαυτό επιλέχθηκε φυσικά και η χρήση του NodeMCU όπως προαναφέρθηκε. Προκύπτει έτσι η ανάγκη για την ύπαρξη ενός δικτύου για την μεταφορά της πληροφορίας μεταξύ των κόμβων του. Για λόγους απλότητας της υλοποίησης αλλά και για να μην υπάρχει ζήτημα συνδεσιμότητας λόγω ελλιπούς κάλυψης του δικτύου κινητής τηλεφωνίας, υλοποιήθηκε ένα τοπικό WiFi δίκτυο χωρίς πρόσβαση στο internet με αποκλειστική λειτουργία την σύνδεση των πολλαπλών NodeMCU αλλά και της συσκευής Android της καπετάνιου.

Η επίλυση αυτού του προβλήματος έγινε με την χρήση ενός RaspberryPi .
Το RaspberryPi είναι ένας κανονικός ηλεκτρονικός υπολογιστής, εξαιρετικά μικρού μεγέθους, πλήρως λειτουργικός, με μεγάλη φορητότητα. Είναι σε μέγεθος μικρότερο μιας παλάμης χεριού και έχει τα βασικά τεχνικά χαρακτηριστικά ενός λάπτοπ περιορισμένων δυνατοτήτων. Χρησιμοποιείται ευρέως για DIY projects, έχει μία τεράστια γκάμα περιφερειακών και φυσικά υποστηρίζεται από μία τεράστια κοινότητα ανθρώπων στο διαδίκτυο που μπορεί κανεί να ανατρέξει σε αυτή για οποιαδήποτε πληροφορία.

Το RaspberryPi λειτουργεί με λειτουργικό Linux, πράγμα που σημαίνει ότι με λίγες βασικές γνώσεις προγραμματισμού και αρχιτεκτονικής υπολογιστών μπορεί κανείς να αξιοποιήσει τις αμέτρητες δυνατότητες του ανοικτού λογισμικού αυτού. Επίσης η πραγματικά απίστευτα μεγάλη κοινότητα για τα Linux μπορεί να παρέχει λύση σε κάθε πρόβλημα που μπορεί να προκύψει.

Σε αυτά ακριβώς τα πλαίσια λοιπόν φάνηκε πως είναι η ιδανική λύση για να μπορέσουμε να σηκώσουμε ένα τοπικό δίκτυο μικρής εμβέλειας που να καλύπτει τα τετραγωνικά του σκάφους, που εκ των πραγμάτων είναι πολύ περιορισμένα και συνεπώς δεν υπάρχουν μεγάλες απαιτήσεις, γεφυρώνοντας έτσι τις συσκευές των 2 άκρων. Υπάρχει ενσωματωμένη κεραία WiFi στο RaspberryPi και από δέκτης σήματος μπορεί να γίνει πομπός.

Η ρευματοδότησή του γίνεται πολύ εύκολα με καλώδιο usb από μία προεγκατεστημένη παροχή τέτοιου είδους που υπάρχει στο σκάφος.

3.6 Ρευματοδότηση

Όπως προαναφέρθηκε όλα τα συστήματα που απαρτίζουν το project μας λαμβάνουν το ρεύμα τους από την παροχή του σκάφους, μετασχηματίζοντας το πριν από αυτά από 12V σε 5V μέσω των πλακετών Step down.

Είναι προγραμματισμένα με τέτοιο τρόπο ώστε να εκκινούν όλες οι λειτουργίες αυτόματα κατά την έναρξη της λειτουργίας τους, τόσο το RaspberryPi όσο και τα nodeMCU. Έτσι διαμορφώνεται μία συνθήκη αυτοματοποίησης της διαδικασίας

εκκίνησης που πηγαίνει παράλληλα με την εκκίνηση των βοηθητικών συστημάτων του σκάφους.

3.7 Πρωτόκολλο Μεταφοράς Δεδομένων MQTT

Η τελική πληροφορία, δηλαδή όλα τα ήδη επεξεργασμένα δεδομένα από τις μονάδες επεξεργασίας πρέπει μέσω δικτύου να μεταφερθούν στην τελική χρήστρια ή χρήστη. Αυτό πρέπει να γίνει με αποτελεσματικό τρόπο, γρήγορο, χωρίς να αυξάνει τον φόρτο του δικτύου, που εκ των πραγμάτων έχει ήδη περιορισμένο εύρος ζώνης (bandwidth) με τον μεγάλο όγκο πληροφορίας των αισθητήρων που μεταφέρεται ανά δευτερόλεπτο σχεδόν.

Το πρωτόκολλο MQTT ή Queue Telemetry Transport είναι ένα ελαφρύ συμπαγές πρωτόκολλο ανταλλαγής μηνυμάτων για την μεταφορά δεδομένων σε απομακρυσμένες τοποθεσίες όπου απαιτείται "αποτύπωμα μικρού κώδικα" ή το εύρος ζώνης του δικτύου είναι περιορισμένο. Αυτά τα πλεονεκτήματα επιτρέπουν την εφαρμογή αυτού του πρωτοκόλλου στα συστήματα M2M (Machine to Machine) και IIoT (Industrial Internet of Things), αλλά είναι ευρέως διαδεδομένη η χρήση του στο Home Automation καθώς είναι η ιδανική λύση για γρήγορη επικοινωνία μεταξύ αισθητήρων.

Στο επίπεδο της εφαρμογής, το πρωτόκολλο MQTT λειτουργεί πάνω από το πρωτόκολλο TCP / IP και χρησιμοποιεί την θύρα 1883 (8883 εάν συνδέεται μέσω SSL) ως προεπιλογή. Η ανταλλαγή των μηνυμάτων πραγματοποιείται μεταξύ του Client, που μπορεί να είναι Publisher ή Subscriber των μηνυμάτων, και του Broker των μηνυμάτων (π.χ. Mosquitto MQTT).

Ο Publisher στέλνει τα δεδομένα σε Broker MQTT ορίζοντας ένα συγκεκριμένο θέμα στο μήνυμα. Οι Subscribers μπορούν να λαμβάνουν διάφορα δεδομένα από πολλούς Publishers, ανάλογα με τη συνδρομή σε αντίστοιχα θέματα που έχουν δημιουργηθεί.

Τα θέματα είναι χαρακτήρες με κωδικοποίηση UTF-8. Η ιεραρχία των θεμάτων έχει μορφή δέντρου, διευκολύνοντας έτσι την οργάνωση και την πρόσβαση στα δεδομένα. Τα θέματα μπορούν να αποτελούνται από ένα ή περισσότερα επίπεδα που χωρίζονται με το σύμβολο «/».

Σύμφωνα με την διπλωματική εργασία της φίλης και συναδέλφισσας Αθηνάς Πλασκασοβίτη, «Ανάπτυξη Υποδομής MQTT για IoT Εφαρμογές», αναφέρεται μέγιστος αριθμός συνδέσεων, σε δυσμενείς κακόβουλες συνθήκες, περίπου 4000, χωρίς απώλειες, με νέες συνδέσεις κάθε 0.5s και αποστολή μηνυμάτων κάθε 50ms.

Στην δική μας περίπτωση τα θέματα είναι χωρισμένα ανά αισθητήρα για μεγαλύτερη διευκόλυνση κατά τον προγραμματισμό.

Οι Clients λοιπόν είναι οι μονάδες επεξεργασίας nodeMCU και ο broker είναι το RaspberryPi στην δική μας περίπτωση.

Ένα απλό αντίστοιχο παράδειγμα θα ήταν να είχαμε κατηγοριοποιήσει σε μία περίπτωση αυτοματισμού σπιτιού, τα θέματα ανάλογα τους χώρους. Έτσι για παράδειγμα ένα θέμα θα ήταν το μπάνιο :

- /home/bathroom

και από εκεί θα είχαμε επέκταση δέντρου με τον κάθε αισθητήρα :

- /home/bathroom/temperature για την θερμοκρασία του μπάνιου
- /home/bathroom/humidity για την υγρασία του μπάνιου
- /home/bathroom/lights για το αν είναι ανοιχτά τα φώτα του
- /home/bathroom/person για τον αν βρίσκεται κάποιος μέσα

Έτσι θα μπορούσε η κάθε ενδιαφερόμενη συσκευή σαν publisher να κάνει post διάφορες τιμές στο θέμα που τον αφορά. Πχ ένας αισθητήρας κίνησης να αναφέρει κίνηση στο /home/bathroom/person ή ένα θερμουγρασιόμετρο να αναφέρει την θερμοκρασία του μπάνιου στο αντίστοιχο θέμα.

Από την άλλη μεριά μία συσκευή θα μπορούσε να είναι subscriber σε κάποιο θέμα και να λαμβάνει μόνο την πληροφορία που έχει αναρτήσει κάποιος publisher, και από εκεί και ύστερα να την διαχειρίζεται είτε ενεργητικά, εκτελώντας μια κίνηση είτε παθητικά, απλώς αποθηκεύοντας την τιμή για μελλοντική χρήση.

Για παράδειγμα ένας subscriber στο /home/bathroom/temperature θα μπορούσε να ενεργοποιεί την θέρμανση του μπάνιου όταν η θερμοκρασία πέφτει κάτω από ένα συγκεκριμένο όριο, ενώ ένας άλλος subscriber θα μπορούσε να κλείνει το φως όταν στο /home/bathroom/person υπάρχει μία μεταβλητή Boolean που υποδεικνύει απουσία ανθρώπου(πχ True).

3.8 Σχεδιασμός Κουτιών Προστασίας

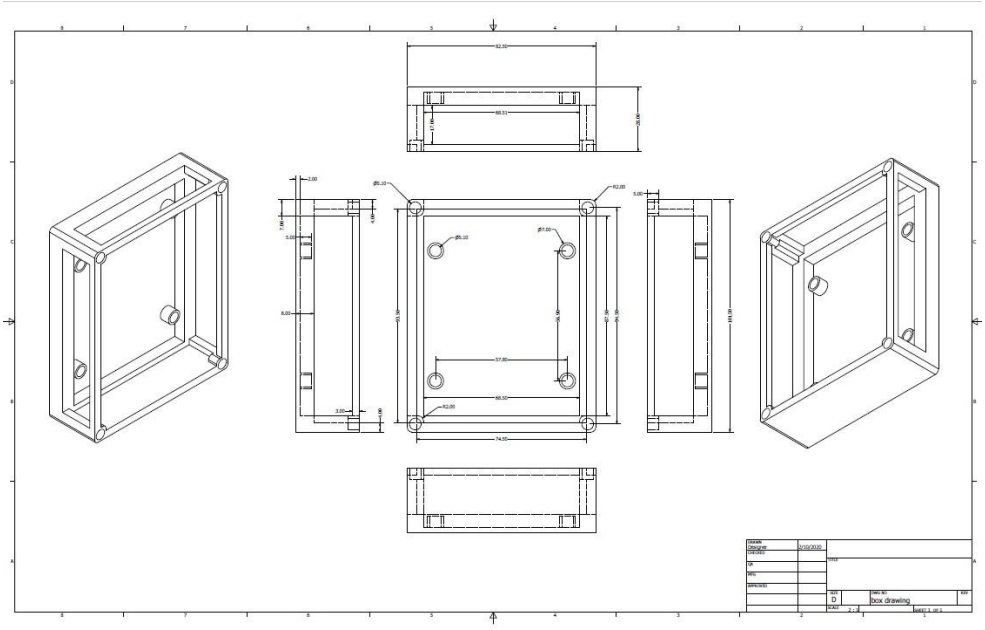
Μέσα στο περιβάλλον του σκάφους είναι σαφές ότι δεν θα μπορούσαμε να έχουμε εκτεθειμένα όλα τα μέρη του συστήματός μας. Είναι ένα περιβάλλον με πολλούς κραδασμούς, και με κινδύνους όπως έκθεση σε υγρά ή και σε χτυπήματα από κάποιο πέρασμα επιβάτη, καθώς το σκάφος είναι σχετικά μικρού μεγέθους. Επίσης ακολουθώντας την ιδέα το όλο σύστημα εν τέλει κάποια στιγμή στο μέλλον να

μπορέσει να μετασχηματιστεί σε ένα εμπορικό προιόν, φυσικά δεν θα ήταν δυνατή η πώλησή του χωρίς κάποια θήκη.

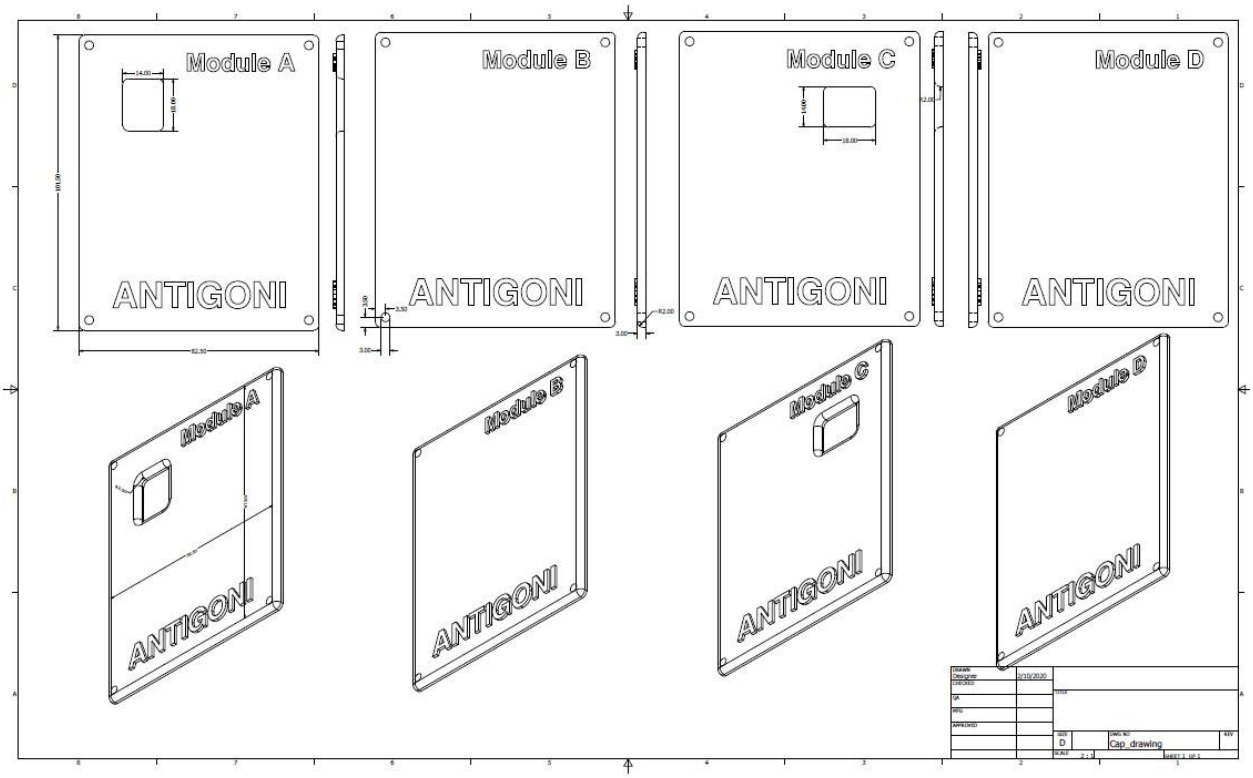
Η μόνη βιώσιμη λύση για το πρόβλημά μας ήταν να γίνει μία ιδιοκατασκευή των κουτιών. Φυσικά αποφύγαμε κάποια αμοιγώς χειροποίητη προσπάθεια καθώς δεν υπάρχουν ούτε οι πόροι και τα εργαλεία, ούτε και η εμπειρία για να κατασκευαστούν 4 κουτιά τα οποία θα μπορούν να τοποθετηθούν ακριβώς μέσα οι πλακέτες μας.

Επιλέχθηκε λοιπόν να εκτυπωθούν σε 3D printer. Με αυτόν τον τρόπο με πολύ απλά βήματα και βασικές γνώσεις σχεδιασμού σε πρόγραμμα τρισδιάστασης μοντελοποίσης μπόρεσαν να σχεδιαστούν 4 κουτιά μαζί με τα καπάκια. Αφού πρώτα έγιναν οι απαραίτητες μετρήσεις στις πλακέτες που θέλαμε να εγκαταστήσουμε μέσα τους, κατασκευάστηκαν ανάλογα την περίπτωση, είτε με πλήρη καπάκια, είτε με τρύπα ώστε να μπορεί να χωράει ο αισθητήρας θερμοκρασίας-υγρασίας και να μπορεί να προεξέχει από πάνω.

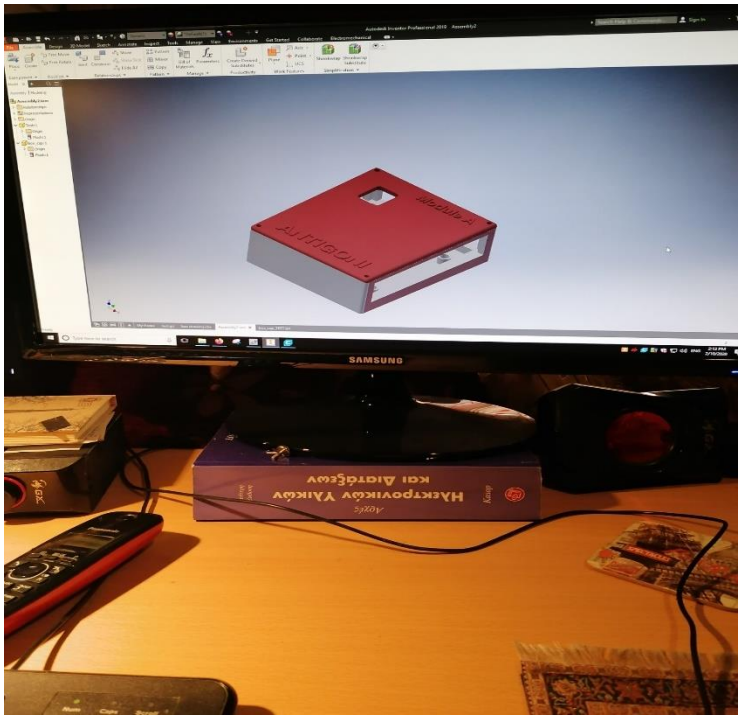
Μέσα στα κουτιά τοποθετήθηκαν τυφλές οπές στις οποίες προσαρμόστηκαν σπειρώματα ώστε να μπορούν να βιδωθούν πάνω τους με ασφάλεια οι πλακέτες μας. Το ίδιο έγινε και με τα καπάκια, τα οποία βιδώνονται στο σώμα του κουτιού με βίδες M3 με κεφαλή Alen.



ΕΙΚΟΝΑ 3.8 ΣΧΕΔΙΟ ΚΟΥΤΙΟΥ



Εικονα 3.9 Σχέδιο από το καπάκι του κουτιού



ΕΙΚΟΝΑ 3.10 Τρισδιαστατο Σχέδιο Κουτιού

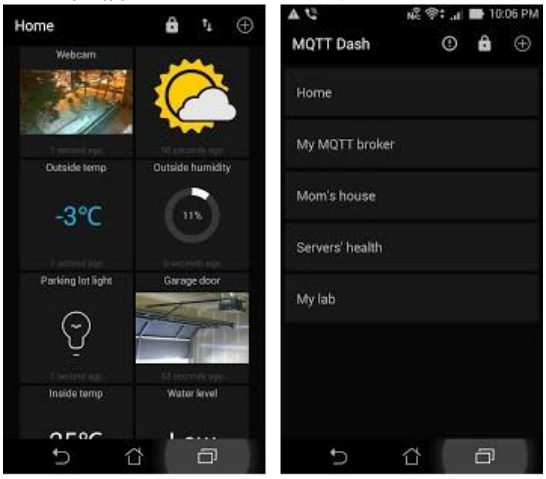
3.9 End User Interface

Στο τελευταίο κομμάτι της αρχιτεκτονικής του συστήματος θα καλύψουμε την επιλογή της εφαρμογής που χρησιμοποιήσαμε.

Οι ανάγκες εξαρχής ήταν πολύ συγκεκριμένες. Θέλαμε μία εφαρμογή που να τρέχει σε λειτουργικό Android ώστε να μπορεί να υπάρχει πρόσβαση σε αυτή από τον καθένα και την καθεμία εύκολα και άμεσα. Επίσης θέλαμε κάτι που να είναι ελαφρύ για να μην υπάρχει περιορισμός από τις δυνατότητες του κινητού, καθώς όντας καπετάνιος, πλέον είναι σημαντικό να υπάρχει πλήρης λειτουργικότητα του κινητού ανά πάσα στιγμή. Τέλος γνώμονας της επιλογής μας ήταν η εφαρμογή να έχει απλό, κατανοητό και εύχρηστο περιβάλλον διεπαφής ανθρώπου και μηχανής ώστε να μπορεί η χρήση του να είναι δυνατή από τον καθένα και την καθεμία, χωρίς απαραίτητα να είναι γνώστης καλής χρήσης κινητών. Μην ξεχνάμε ότι υπάρχουν και καπετάνιοι μεγάλοι σε ηλικία άλλωστε.

Επιλέχθηκε η εφαρμογή MQTT DASH από την ROUTIX SOFTWARE, μία εφαρμογή ευρέως διαδεδομένη για αυτές τις περιπτώσεις. Είναι πλήρως λειτουργική και εύχρηστη παρά τα λίγο χρόνια ύπαρξής της. Είναι ένας MQTT Client που παρακολουθεί και καταγράφει στην οθόνη όποια θέματα / topics επιλέγει ο χρήστης. Η διαμόρφωση της επιφάνειας εργασίας του είναι με απλά πλακίδια που καταγράφουν τα νέα μυνήματα που έρχονται στο κάθε θέμα. Συνεπώς η τελική χρήστρια ή χρήστης θα έχει ένα dashboard το οποίο θα ανανεώνεται διαρκώς με τα νέα δεδομένα σε κάθε δεδομένη στιγμή που αυτά θα προκύπτουν.

Αντίστοιχα για τις λειτουργίες Alarm, υπάρχει ειδική επιλογή που μπορεί κανείς να προγραμματίσει συγκεκριμένα πεδία να αλλάζουν χρώματα με παλμικό τρόπο , είτε όταν η τιμή γίνεται μία συγκεκριμένη, για παράδειγμα όταν το θέμα/topic της κατάκλισης λάβει την τιμή True, είτε όταν ξεπεράσει κάποια τιμή όταμ για παράδειγμα αυξηθεί η θερμοκρασία του μηχανοστασίου πέρα από κάποια ορισμένη κρίσιμη τιμή.



ΕΙΚΟΝΑ 3.11 MQTT DASH

Κεφάλαιο 4 - Αναλυτική περιγραφή του κατασκευαστικού μέρους του πειράματος.

4.1 Εισαγωγή - Η έννοια του Prototyping

Το Prototyping είναι η παραγωγική διαδικασία που μεταφέρει με αργά και σταθερά βήματα, μία ιδέα, από το πεδίο της σκέψης και της φαντασίας, στην πραγματικότητα μέσω των διαρκών πειραματισμών, δοκιμών και φυσικά λαθών.

Είναι η μετάβαση από το χαρτί στην ψηφιακή ή υλική μορφή του πειράματος. Η διαδικασία αυτή χαρακτηρίζεται από δημιουργικότητα και πειραματισμό, καθώς εφαρμόζεται στην πράξη σχεδόν κάθε πτυχή ενός σχεδίου που βρίσκεται στα πρώιμα στάδιά του για να ελεγχεί το κατά πόσο είναι λειτουργικό σαν σύστημα.

Επίσης είναι και ένα σημαντικό στάδιο καθώς μπορεί κανείς να λάβει τα πρώτα feedback από πιθανούς τελικούς χρήστες. Η διαδικασία αυτή προσφέρει πλήθος πληροφοριών προς βελτίωση του συστήματος καθώς αξιολογείται στην πράξη η λειτουργία ενός συστήματος από άτομα τα οποία πιθανός να μην έχουν ξανασχοληθεί με τα συγκεκριμένα στοιχεία και αυτό προσδίδει στην ανάδρασή τους μία αξία, καθώς η αντικειμενικότητα της γνώμης τους είναι αυξημένη.

Αξιοποιώντας λοιπόν τα παραπάνω στοιχεία, μέσω του Prototyping γίνονται τα πρώτα βήματα για την κατασκευή και την υλοποίηση ενός συστήματος, βήματα που ορίζουν την κατεύθυνση και παίζουν σημαντικό ρόλο στην αναγνώριση πραγματικών προβλημάτων που μπορεί να προκύψουν στο μέλλον και να μην τα έχει αντιληφθεί ο δημιουργός.

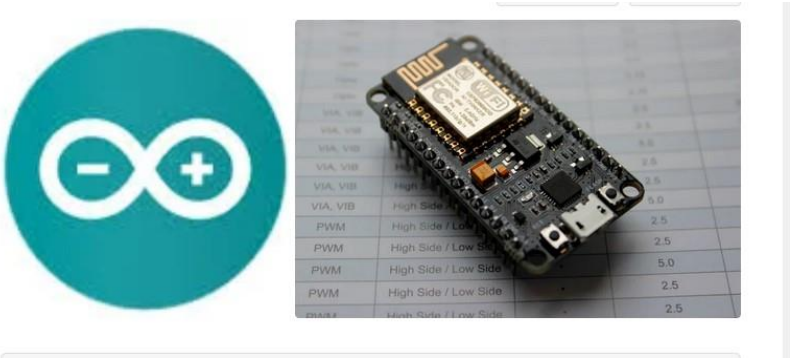
4.2 Δοκιμές Μονάδων Επεξεργασίας

Το πρώτο στάδιο της διαδικασίας είναι η δοκιμή στην πράξη του κάθε αισθητήρα ξεχωριστά, και γενικότερα του κάθε component σαν αυτούσιο κομμάτι του όλου συστήματος.

Με αυτόν τον τρόπο, πρακτικά επιμερίζεται το πρόβλημα, σε μικρά διαχειρίσιμα κομμάτια τα οποία ένα ένα ελέγχονται για την λειτουργικότητά τους, την πρακτικότητά τους και φυσικά για το εάν καλύπτουν στην πράξη τις ανάγκες μας, ή αν θα πρέπει να αντικατασταθούν με κάτι άλλο συγγενές.

Αρχικά έγιναν βασικά πειράματα για την εκμάθηση στην πράξη των NodeMCU σαν μικροεπεξεργαστές. Χρήση απλής συνδεσμολογίας, απλός κώδικας, πράξεις και βασικά υπολογιστικά προγράμματα ήταν τα πρώτα βήματα.

4.2.1 Πλατφόρμα Προγραμματισμού Arduino IDE

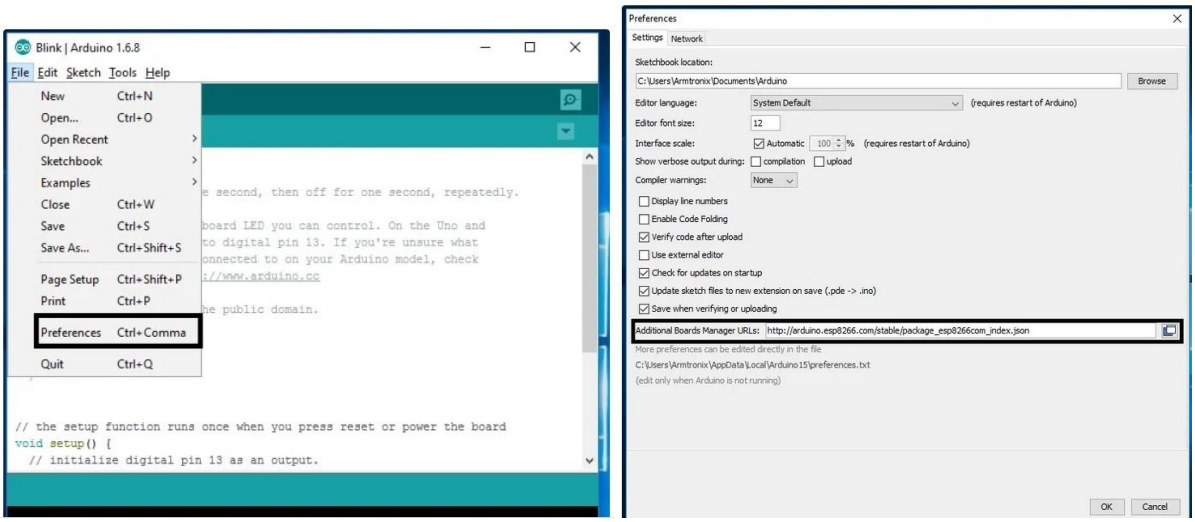


Εικόνα 4.1 – Node MCU

Το πρώτο στάδιο της εκκίνησης είναι η προσαρμογή του προγράμματος προγραμματισμού Arduino IDE ώστε να μπορεί να συνδεθεί με το NodeMCU.

Το συγκεκριμένο πρόγραμμα, που είναι και το πλέον διαδεδομένο για αυτήν την χρήση, είναι συμβατό με ένα τεράστιο εύρος μικροεπεξεργαστών. Οι απαραίτητες ενέργειες για την σύνδεσή τους είναι να γίνει download από το διαδίκτυο το απαραίτητο υλικό και βιβλιοθήκες, και να ρυθμιστούν οι σωστές πόρτες usb για να μπορεί να υπάρχει απρόσκοπτη επικοινωνία μεταξύ ενός ηλεκτρονικού υπολογιστή και του μικροεπεξεργαστή μας.

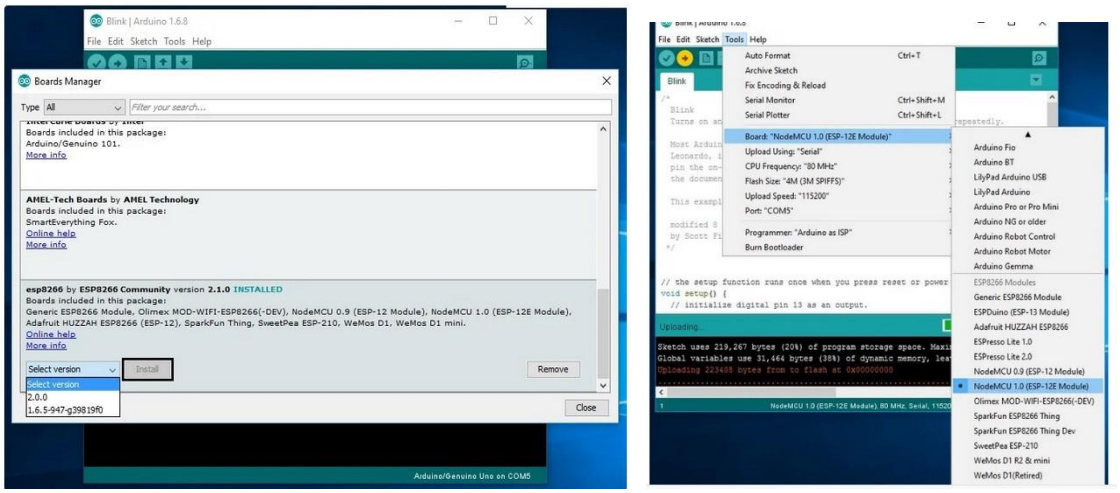
Αρχικά ανοίγοντας το πρόγραμμα και πηγαίνοντας στις ρυθμίσεις του, επιλέγουμε να προσθέσουμε επιπλέον διαχειριστές μονάδων (Board Managers). Στην σχετική επιλογή προσθέτουμε έναν σύνδεσμο που κατεβάζει τα απαραίτητα πακέτα για αυτήν την επιλογή.



Εικόνα 4.2-4.3

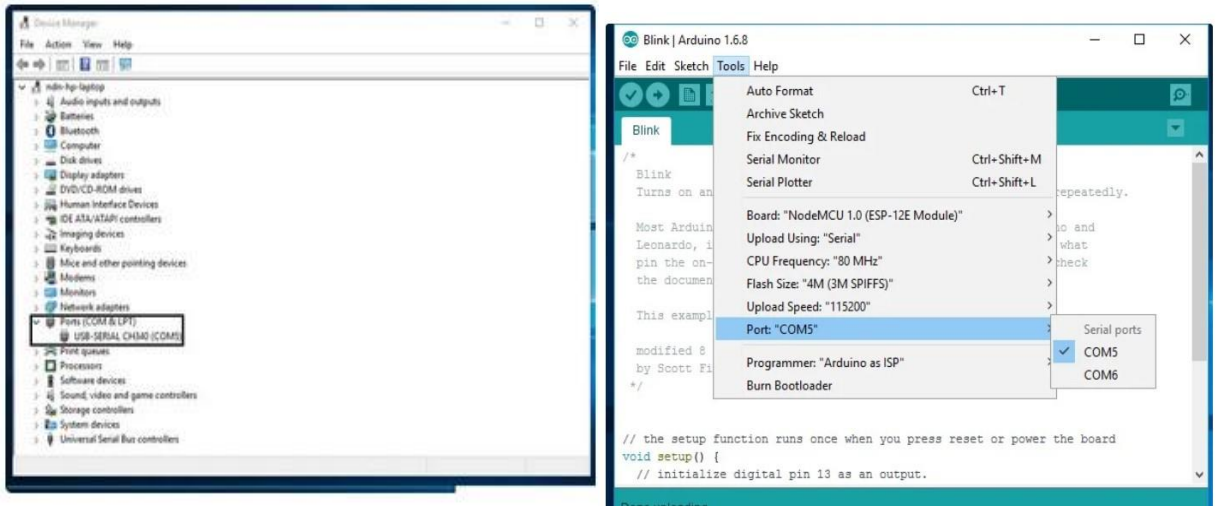
Ύστερα με πολύ απλά βήματα διαλέγουμε από το drop down menu τον μικροεπεξεργαστή που μας ενδιαφέρει, εδώ σε εμάς φυσικά είναι το NodeMCU, και κατεβάζουμε το απαραίτητο πακέτο βιβλιοθηκών.

Step 7: ESP8266 Board Package



Εικόνα 4.4-4.5

Το τελευταίο στάδιο είναι να κάνουμε τις κατάλληλες ρυθμίσεις για την επιλογής της σειριακής πόρτας επικοινωνίας του υπολογιστή μας και του μικροεπεξεργαστή μας. Από τις ρυθμίσεις του υπολογιστή μας για τις εξωτερικές συσκευές βλέπουμε σε ποια σειρικαή πόρτα έχει δημιουργηθεί η σύνδεση και πολύ απλά την επιλέγουμε στο Arduino IDE .



Εικόνα 4.6 – 4.7

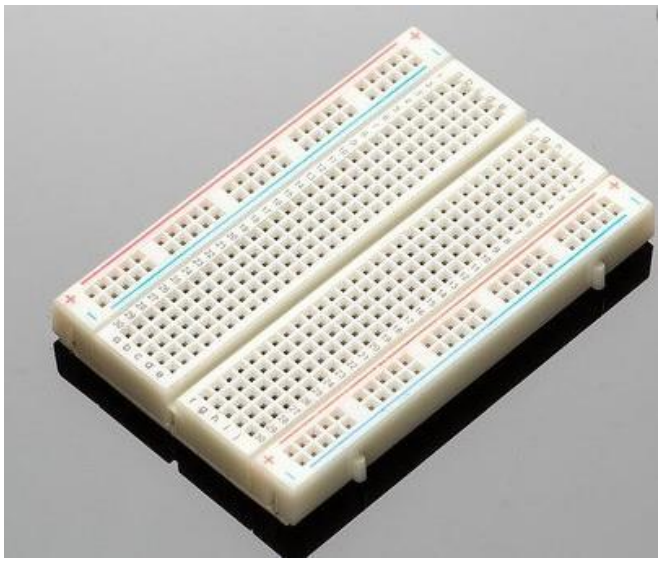
Τα βήματα είναι πολύ απλά και κατανοητά από την καθεμία και τον καθένα, και φυσικά υπάρχει τεράστιο μέγεθος υλικού στο διαδίκτυο για την υλοποίησή τους. Αφού γίνουν και οι τελευταίες επιλογές, μπορούμε πλέον να ανεβάσουμε στον μικροεπεξεργαστή μας το πρόγραμμά μας.

4.2.2 Βασική Συνδεσμολογία και Πρώτες δοκιμές.

4.2.2.1 Δοκιμές στο BreadBoard.

Το επόμενο στάδιο της διαδικασίας των δοκιμών είναι να δοκιμάσουμε στην πράξη ξεχωριστά τον κάθε έναν αισθητήρα, να εντοπίσουμε τα προβλήματα που προκύπτουν κατά την χρήση του και να καταλάβουμε την συμπεριφορά του. Αυτό είναι ένα πολύ σημαντικό κομμάτι καθώς είναι αδύνατον να συνδέσει κανείς όλα τα στοιχεία του συστήματός μας κατευθείαν και να προσπαθήσει να τα κάνει να συνεργαστούν μεταξύ τους.

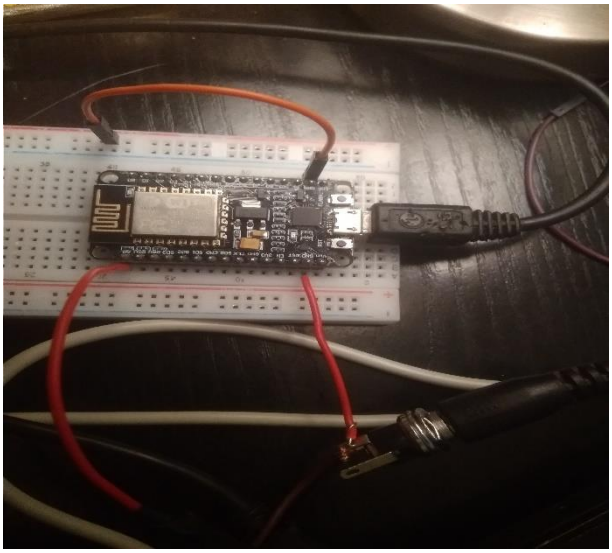
Η διαδικασία που ακολουθήθηκε είναι με την χρήση ενός breadboard. Το breadboard είναι μία πλακέτα με πολλαπλές υποδοχές καλωδίων, κατασκευασμένη με τέτοιο τρόπο ώστε να μπορεί να δεχτεί πάνω της πλήθος καλωδίων με ακροδέκτες τύπου βελόνας. Από την πίσω μεριά του υπάρχει συνδεσμολογία κατά γραμμές. Έτσι λοιπόν έχει κανείς την δυνατότητα να μπορεί να πειραμιστεί χωρίς να χρειάζεται να κάνει κολλήσεις καλωδίων και έτσι να μπορεί να εναλλάσει την συνδεσμολογία του. Η χρήση του breadboard είναι αναπόσπαστο κομμάτι του protyping.



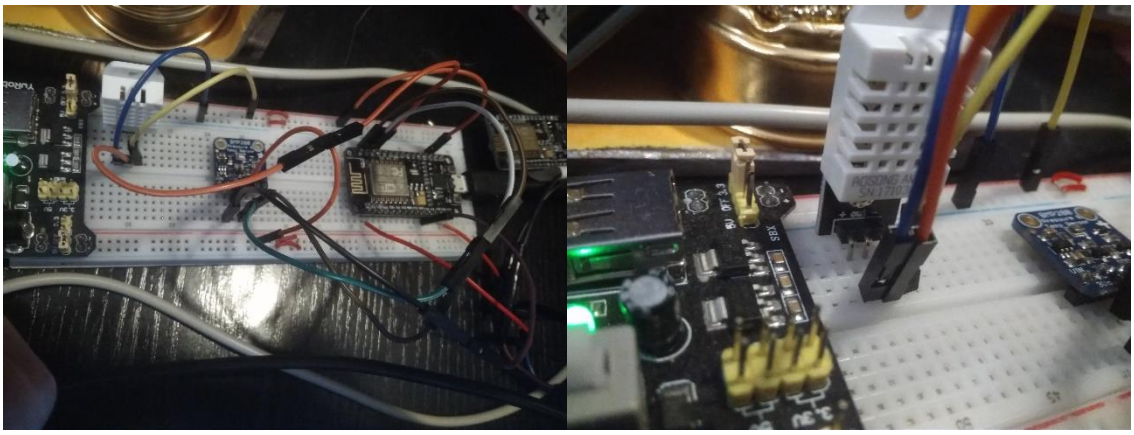
Εικόνα 4.8 - Breadboard

Σε πρώτη φάση δοκιμάστηκε με απλό κώδικα και ακολουθώντας τα βήματα της προηγούμενης παραγράφου να συνδεθεί και να λειτουργήσει σαν μονάδα ένα NodeMCU. Χωρίς πρόσθετη συνδεσμολογία, μόνο με το καλώδιο παροχής ρεύματος, που είναι και το ίδιο με το καλώδιο σύνδεσης με τον ηλεκτρονικό υπολογιστή, συνδέσαμε τον μικροεπεξεραστή μας και τον προγραμματίσαμε να αναβοσβήνει ένα απλό led. Έτσι μετρήσαμε χρόνους απόκρισης και εκκίνησης, και αξιοπιστία. Από την έναρξη της παροχής ρεύματος μέχρι την λειτουργία ο χρόνος που διαμεσολάβησε ήταν της τάξης των τριών (3) δευτερολέπτων.

Στο επόμενο στάδιο λοιπόν δοκιμάστηκαν ξεχωριστά οι αισθητήρες μας. Γράφτηκε ο βασικός κώδικας για τον καθένα, και ελέγχθηκε η λειτουργία του καθώς και το ότι μας δίνει τα σωστά αποτελέσματα. Από την αρχή υπήρχε ο φόβος λανθασμένων μετρήσεων καθώς οι αισθητήρες ήταν αγορασμένοι κατά κύριο λόγο από την Κίνα με πολύ χαμηλό κόστος, και ως εκ τούτου μειωμένη αξιοπιστία. Ευτυχώς όλοι αποδείχτηκαν απροβλημάτιστα λειτουργικοί.Παρακάτω φαίνεται η δοκιμαστική λειτουργία του βαρόμετρου και του θερμοϋγρασιόμετρου καθώς και η σύνδεση με την παροχή ρεύματος για το ανεμόμετρο.



Εικόνα 4.9 - Συνδεσμολογία με ανεμόμετρο

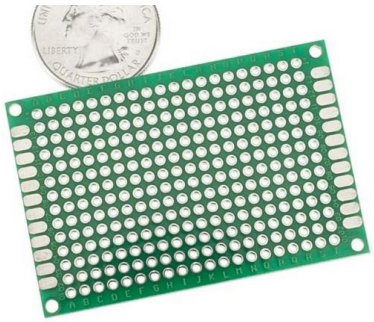


Εικόνα 4.10 - Δοκιμαστική λειτουργία θερμοϋγρασιόμετρου και θερμοβαρόμετρου

Αφού έγιναν οι κατάλληλες δοκιμές ανά αισθητήρα, στο επόμενο στάδιο ακολούθησε η ολοκληρωμένη σύνδεση της κάθε μονάδας επεξεργασίας σε δοκιμαστικό επίπεδο. Έτσι το κάθε ένα από τα Modules που είχαν σχεδιαστεί δοκιμάστηκε και στην πράξη. Δημιουργήθηκε ο συνδιαστικός κώδικας ο οποίος ενσωμάτωνε την λήψη και την επεξεργασία των δεδομένων από τους αισθητήρες, την σύνδεση στο δίκτυο καθώς και την αποστολή των δεδομένων στα κατάλληλα topics μέσω της χρήσης του MQTT πρωτοκόλλου.

4.2.2.2 Χρήση Perfboard και Ολοκλήρωση Κατασκευής

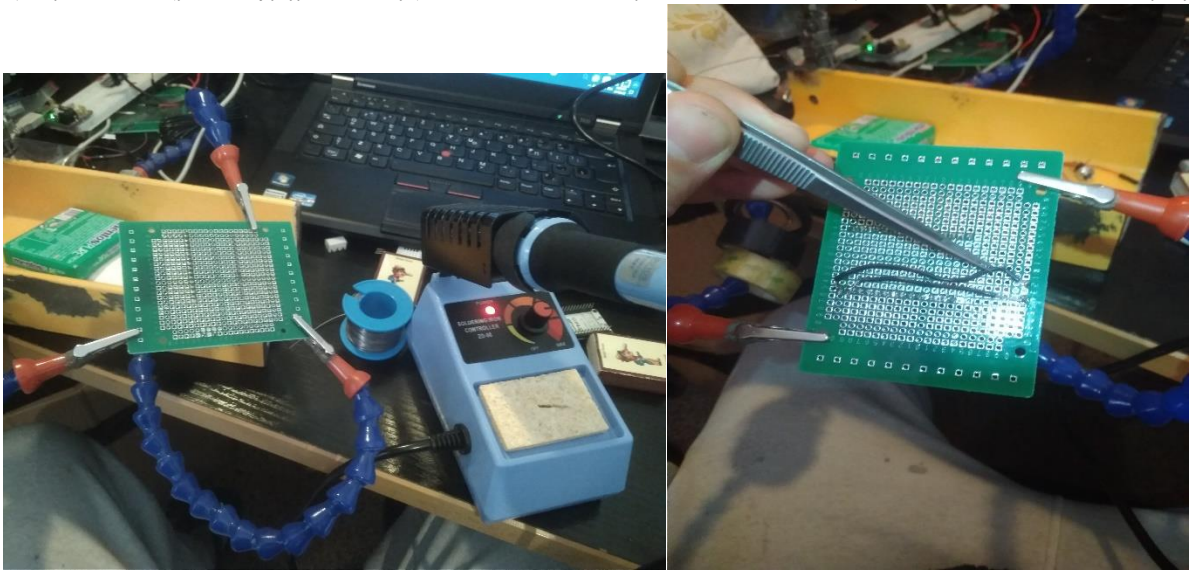
Το επόμενο στάδιο ήταν η μεταφορά των συστημάτων μας σε μία πιο στιβαρή έκδοση του εαυτού τους, καθώς οι δοκιμές μας δεν θα μπορούσαν να εφαρμοστούν στην πράξη. Εδώ υπήρχαν 2 επιλογές, με την πρώτη να είναι η πιο οικονομικά επώδυνη αλλά ανώδυνη από πλευράς χρόνου και σωματικής κούρασης, και την δεύτερη ακριβώς το ανάποδο. Η πρώτη επιλογή λοιπόν ήταν η κατασκευή μέσα από πρόγραμμα υπολογιστή της συνδεσμολογίας που απαιτείται και η αποστολή σε ειδικό κατασκευαστή PCB πλακετών για εκτύπωση ακριβώς αυτού που χρειαζόταν για το σύστημά μας. Η δεύτερη επιλογή η οποία ήταν και η δική μας φυσικά, ήταν η κατασκευή του συστήματός μας χρησιμοποιώντας Perfboard. Το Perfboard είναι μία διάτρητη πλακέτα, με προκαθορισμένα κενά ανάμεσα στις οπές της, ώστε να χωράει πλήθος των ανά τον κόσμο ηλεκτρονικών στοιχείων. Τα στοιχεία τοποθετούνται στις οπές, κολλούνται με ηλεκτρολογικό καλάι στην θέση τους, και από την πίσω όψη της πλακέτας γίνεται σύνδεση με χρήση καλωδίων και καλάι πάλι. Είναι μια διαδικασία που χρειάζεται υπομονή και επιμονή και κάθε λάθος κοστίζει σε ώρες εργασίας. Παρ'όλ'αυτά υπάρχει η δυνατότητα να αναστρέψεις ένα λάθος αφαιρώντας το καλάι, σε αντίθεση με την χρήση του PCB καθώς τότε θα έπρεπε να ξαναγίνει παραγγελία της πλακέτας, που σημαίνει αυτόματα κατακόρυφη αύξηση του ήδη υψηλού κόστους.



Εικόνα 4.11 - Perfboard

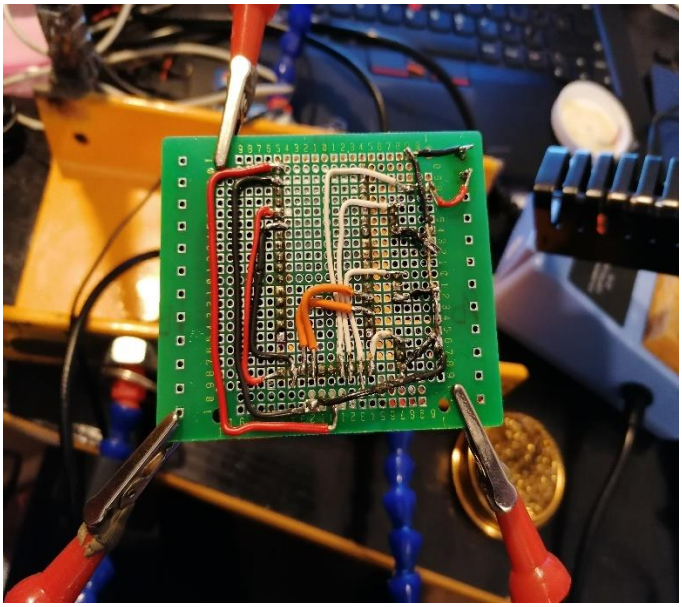
Ξεκινώντας λοιπόν την διαδικασία κολλήσεων όλων των ηλεκτρολογικών κομματιών, αρχικά έπρεπε να καθοριστούν οι θέσεις τους ώστε να χωράνε όσο το δυνατόν πιο αποτελεσματικά γίνεται πάνω στο perfboard. Μετά από πολλές δοκιμές τοποθετήθηκαν στην σωστή θέση ώστε να χωράει επαρκώς και η μονάδα μετατροπής του

ρεύματος (μετασχηματιστής) αλλά και φυσικά οι ακροδέκτες των αισθητήρων.

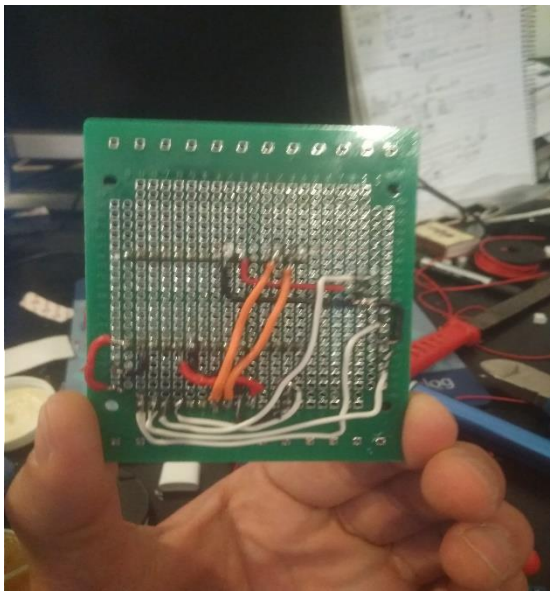


Εικόνα 4.12 – 4.13 - Οι πρώτες κολλήσεις

Η συνδεσμολογία έγινε σταδιακά και με δοκιμές σωστής κόλλησης με χρήση ενός δοκιμαστικού συνδεσιμότητας με χρήση πολυμέτρου. Κολλήθηκε σχεδόν ο κάθε ακροδέκτης του εκάστοτε NodeMCU με το αντίστοιχο component ώστε να υπάρχει λειτουργία του κάθε Module.

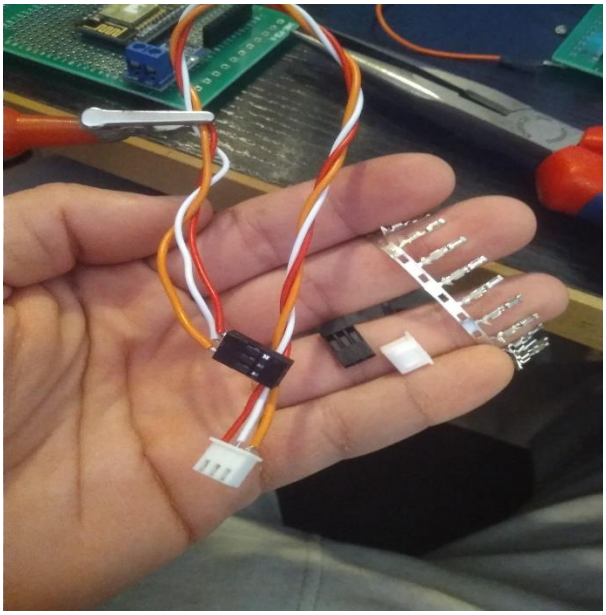


Εικόνα 4.14 – Τελειωμένες κολλήσεις



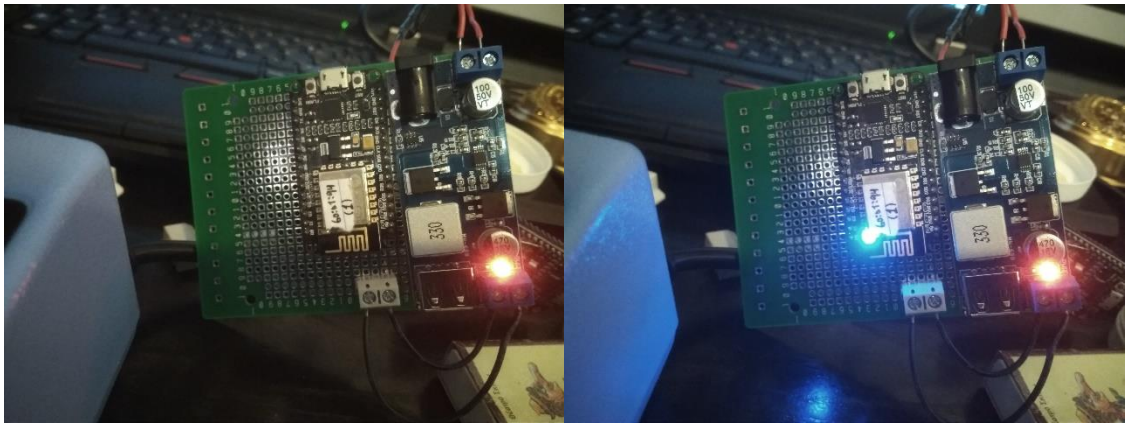
Εικόνα 4.15 – Τελειωμένες κολλήσεις

Οι αισθητήρες αποφασίστηκε να είναι αποσπώμενοι ώστε να υπάρχει η δυνατότητα αντικατάστασής τους εάν χρειαστεί. Τοποθετήθηκαν ακροδέκτες με «κλιπ» ώστε να μπορούν να παραμένουν στην θέση τους ακόμα και σε καθεστώς έντονων κραδασμών, όπως ακριβώς θα συνέβαινε σε περιβάλλον σκάφους. Ύστερα πλέχθηκαν τα απαραίτητα καλώδια μεταξύ τους ώστε να είναι ένα σώμα και να μπορούν με ευκολία να προσαρμοστούν παράλληλα με την υπάρχουσα καλωδίωση του σκάφους. Επίσης «πρεσαρίστηκαν» οι κατάλληλοι ακροδέκτες σε αυτά τα καλώδια ώστε να ενισχυθεί η στιβαρότητά τους.



Εικόνα 4.16 – Τελειωμένες κολλήσεις

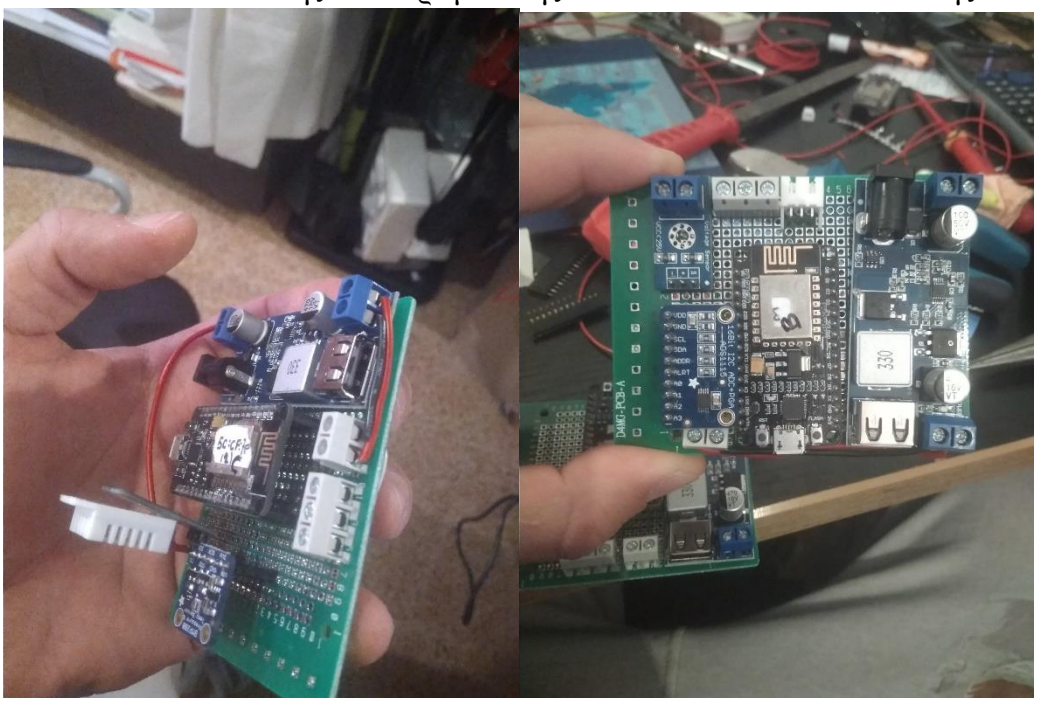
Η κάθε μονάδα επεξεργασίας ανεξαρτήτως λειτουργίας έχει κοινή δομική βάση με όλες τις άλλες. Αποτελείται από το NodeMCU, το perfboard και από τον μετασχηματιστή ρεύματος. Έχουν ίδια συνδεσμολογία και έχουν τοποθετηθεί τα παραπάνω στις ίδιες ακριβώς θέσεις. Επίσης επειδή δεν υπήρχαν οι κατάλληλες οπές στους μετασχηματιστές και στα perfboards τα 2 αυτά μεταξύ τους έχουν στερεωθεί χρησιμοποιώντας αυτοκόλλητα Velcro ώστε να μπορούν να τοποθετούνται και να αφαιρούνται ανάλογα με τις ανάγκες, που σε αντίθετη περίπτωση δεν θα υπήρχε αυτή η δυνατότητα εάν επιλέγαμε να τα στερεώσουμε με κάποια μόνιμη λύση όπως για παράδειγμα χρησιμοποιώντας κόλλα.



Εικόνα 4.15 – 4.16 - Perfboard – NodeMCU – Μετασχηματιστής ρεύματος

Εικόνα 4.17 - Χρήση Velcro για στερέωση του μετασχηματιστή

Αφού λοιπόν τοποθετήθηκαν όλα τα βασικά components σε κάθε μονάδα επεξεργασίας και έγινε η κατάλληλη σχεδίαση και προμελέτη για την τοποθέτηση των υπολοίπων ηλεκτρολογικών στοιχείων που χρειάζονταν ανά μονάδα, στο τέλος έγινε η τελική συναρμολόγηση και τοποθέτηση των αντίστοιχων on board αισθητήρων καθώς και των ακροδεκτών για τους αισθητήρες που θα τοποθετούνταν σε απομακρυσμένη θέση από την μονάδα επεξεργασίας. Ακολούθησε η αντίστοιχη όδευση των καλωδίων στις πίσω μεριές του perfboard και η κόλλησή τους. Έτσι έγιναν οι τελικές δοκιμές στα ολοκληρωμένα πλέον Module μας και η βάση του συστήματός μας πλέον είχε περάσει από τα στάδια της προμελέτης στα στάδια της ολοκλήρωσης.

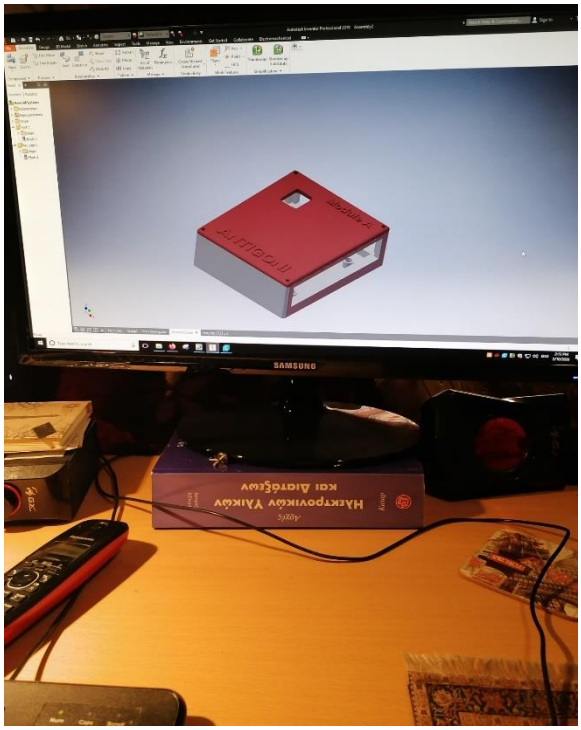


Εικόνες 4.18-4.19-4.20 – Τελειωμένα τα Boards

4.3 Κατασκευή Κουτιών / Θηκών.

Αφού ολοκληρώθηκε η κατασκευαστική διαδικασία συναρμολόγησης της κάθε μονάδας επεξεργασίας, και υπήρχαν τα ακριβή μεγέθη στα χέρια μας, ξεκίνησε η διαδικασία δημιουργίας των κουτιών τους. Όπως προαναφέρθηκε σε προηγούμενο κεφάλαιο αποφασίστηκε η χρήση ενός 3D Printer ώστε να δημιουργήσουμε κάτι το οποίο θα ανταποκρίνεται 100% στις ανάγκες μας ενώ παράλληλα θα έχει χαμηλό κόστος και ευκολία κατασκευής, δεδομένου βέβαια ότι υπάρχει διαθέσιμος ένας κατάλληλος εκτυπωτής.

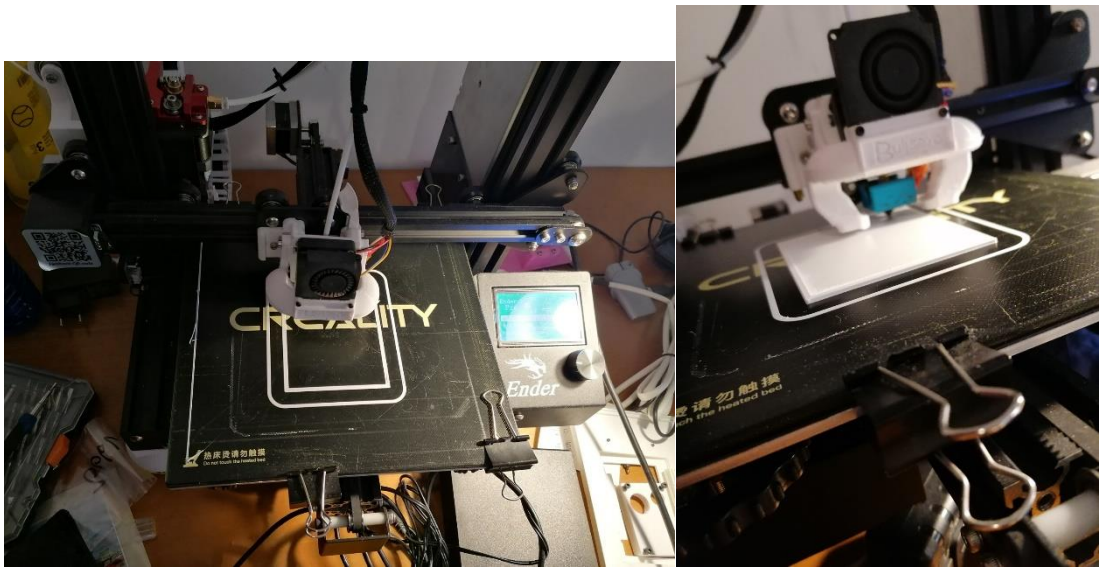
Μετρήθηκαν λοιπόν οι διαστάσεις της κάθε πλακέτας χρησιμοποιώντας παχύμετρο και σχεδιάστηκαν σε κατάλληλο πρόγραμμα τα αντίστοιχα σχέδια για την κάθε πλακέτα (Autodesk Inventor).



Εικόνα 4.21 – Κουτί στο σχεδιαστικό πρόγραμμα

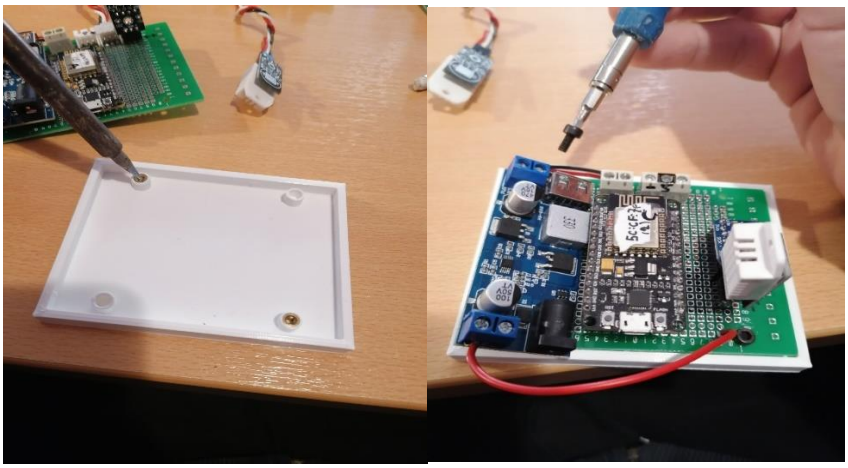
Τα σχέδια αυτά περάστηκαν στο αντίστοιχο λογισμικό που είναι υπεύθυνο για την λειτουργία του 3D Printer, έγιναν οι κατάλληλες παραμετροποιήσεις και ξεκίνησε η διαδικασία εκτύπωσης των κουτιών. Η εκτύπωση διήρκησε περίπου 4 ώρες ανά κουτί και περίπου 1.5 ώρες ανά καπάκι και το συνολικό κόστος των υλικών ανέρχεται περίπου στα 10 ευρώ συνολικά για όλα τα τεμάχια, μην υπολογίζοντας την κατανάλωση του ρεύματος

για την λειτουργία του εκτυπωτή καθώς και ούτε τις εργατοώρες που καταναλώθηκαν για τον σχεδιασμό.



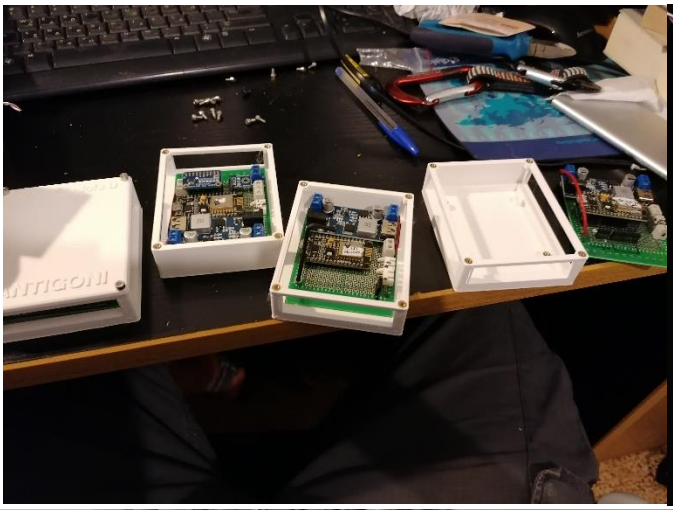
Εικόνες 4.22 – 4.23 – Χρήση τρισδιάστατου εκτυπωτή

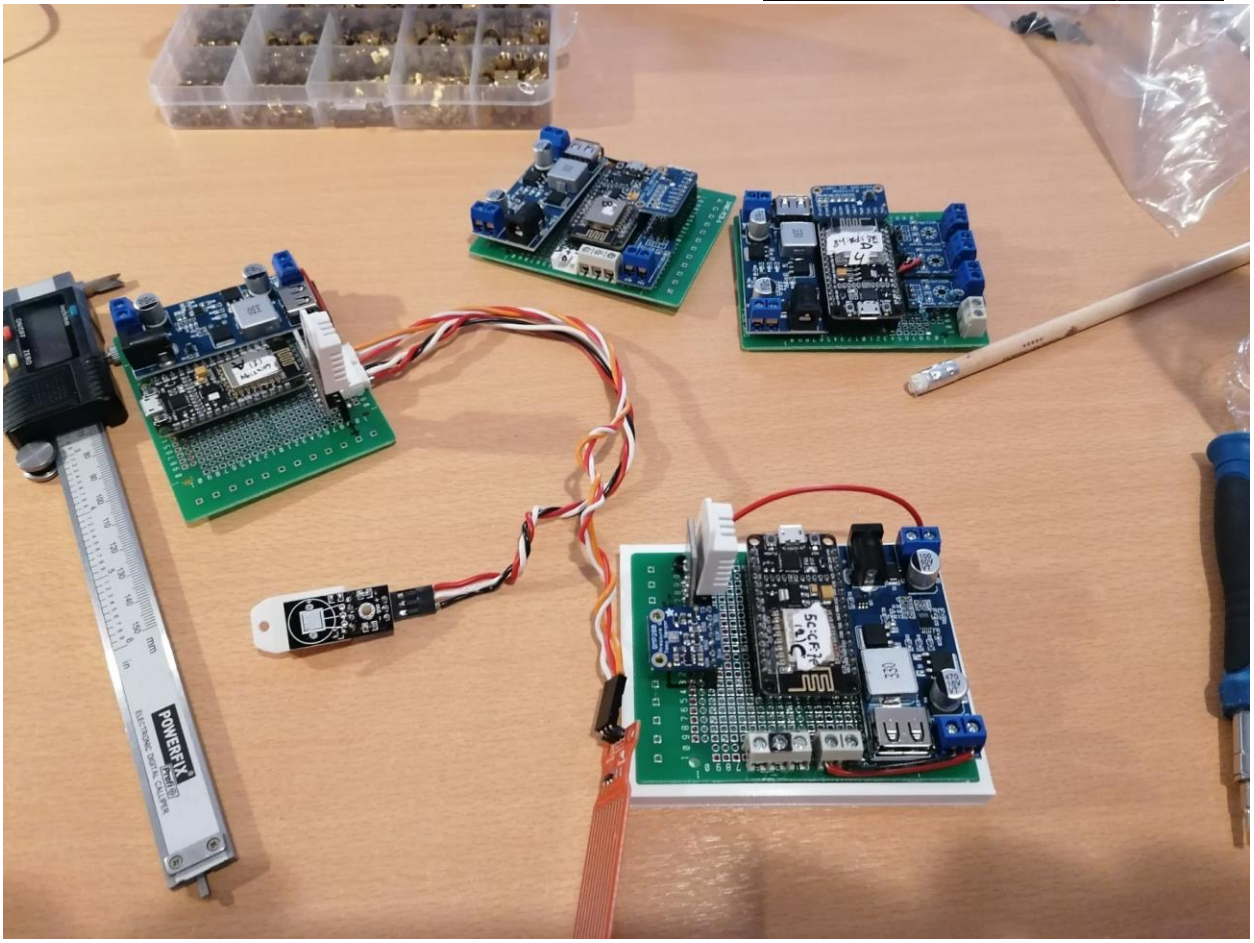
Στην συνέχεια προσαρμόστηκαν στις ήδη υπάρχουσες οπές ειδικού τύπου θυλικά σπειρώματα, κατάλληλα για χρήση σε τριασδιάστατα εκτυπωμένα κομμάτια. Η τοποθέτησή τους γίνεται με την χρήση θερμότητας, δηλαδή πιέζοντας με το κολλητήρι τα μεταλλικά αυτά στοιχεία μέσα στις οπές, οι οποίες εξαρχής έχουν τυπωθεί με περιορισμένες ανοχές. Έτσι η θερμότητα μεταφέρεται στα μεταλλικά σπειρώματα, αυτά με την σειρά τους λιώνουν το πλαστικό της οπής, και αφού τοποθετηθούν στην κατάλληλη θέση, αυτό κρυώνει και δημιουργεί μία σφιχτή συναρμογή. Έτσι μπορούμε να στερεώσουμε πάνω με απλές μηχανολογικές βίδες M3 με κεφαλή alen είτε τις πλακέτες, είτε τα καπάκια από τα κουτιά.



Εικόνα 4.24 – 4.25 – Εισαγωγή βιδών και στερέωση πλακέτας στο κουτί.

Μετά ακολουθεί η τελική συναρμολόγηση όλων των κομματιών της κατασκευής, που είναι και το τελευταίο στάδιο πριν την διαδικασία τοποθέτησης/εγκατάστασης του συστήματος και την δοκιμή όλου του πειράματος στην πράξη, πάνω στο σκάφος.





Εικόνες 4.26- 4.27 -4.28 – Τελειωμένα τα κομμάτια

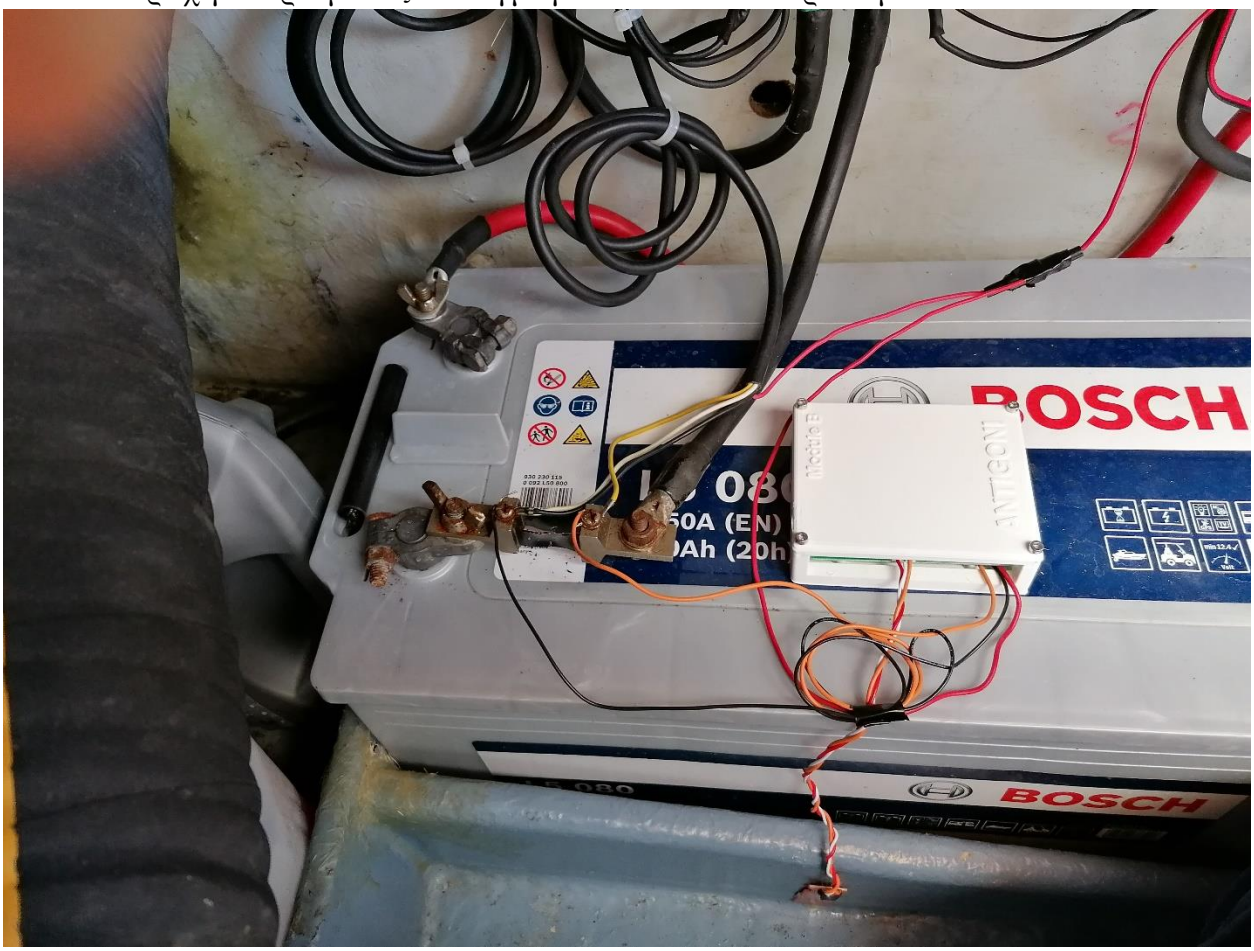
4.4 Εγκατάσταση/ Δοκιμή/ Λειτουργία σε πραγματικές συνθήκες

Το επόμενο στάδιο ήταν τοποθέτηση όλων των επιμέρους κομματιών που απαρτίζουν το σύστημά μας, πάνω στο μικρό σκάφος αναψυχής το οποίο αφετηριακά ήταν και το μοντέλο πάνω στο οποίο έγινε ολόκληρη η μελέτη.

4.4.1 Τοποθέτηση MODULE B

Ξεκινήσαμε με την τοποθέτηση του MODULE B που είναι υπεύθυνο για τις μετρήσεις της μπαταρίας (τάση ρεύματος και ένταση ρεύματος) και για την κατάκλιση του μηχανοστασίου (ανίχνευση υγρού στοιχείου στον πυθμένα του μηχανοστασίου σε ύψος 5cm). Το συγκεκριμένο τοποθετήθηκε πάνω στην μπαταρία για άμεση πρόσβαση στους ακροδέκτες της και στο shunt resistor και από εκεί κατέβηκε καλωδίωση για τον αισθητήρα ανίχνευσης υγρού.

Παροχή ρεύματος λήφθηκε από παρακείμενο καλώδιο 12V.

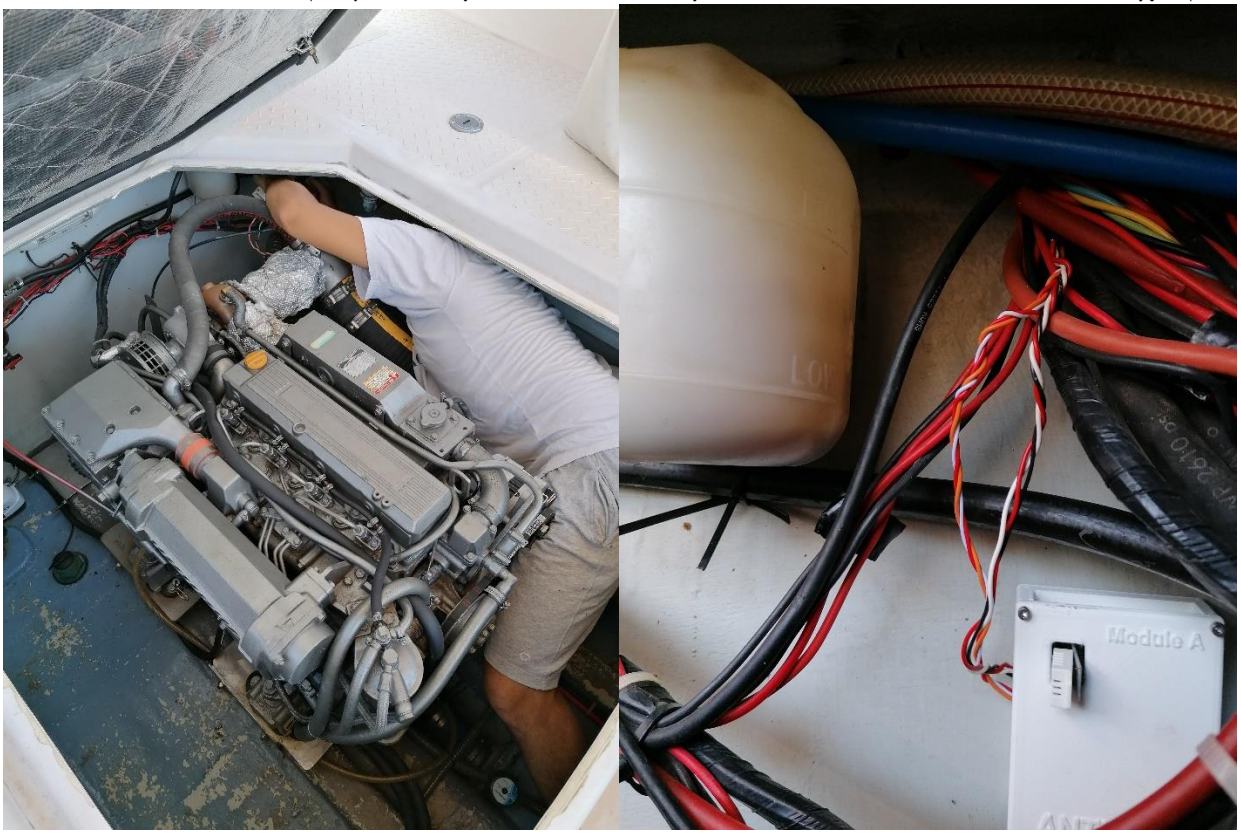


Εικόνα 4.29 – Τοποθέτηση Module B

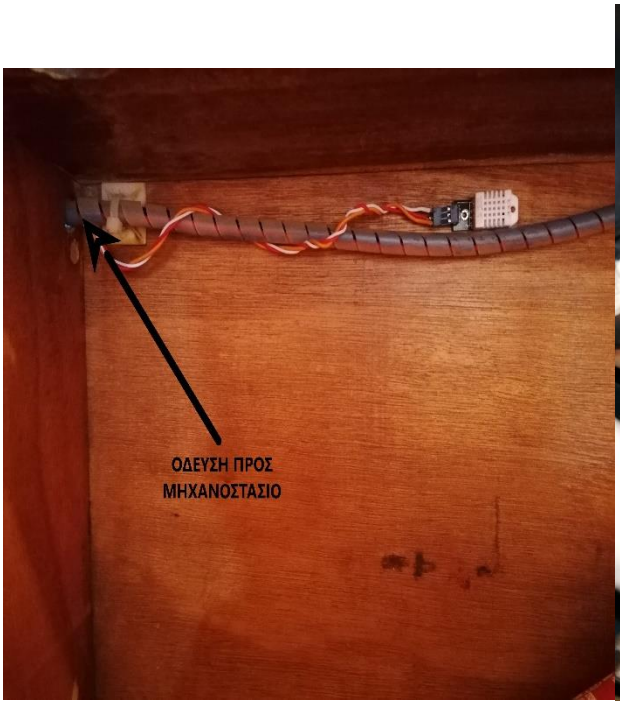
4.4.2 Τοποθέτηση MODULE A

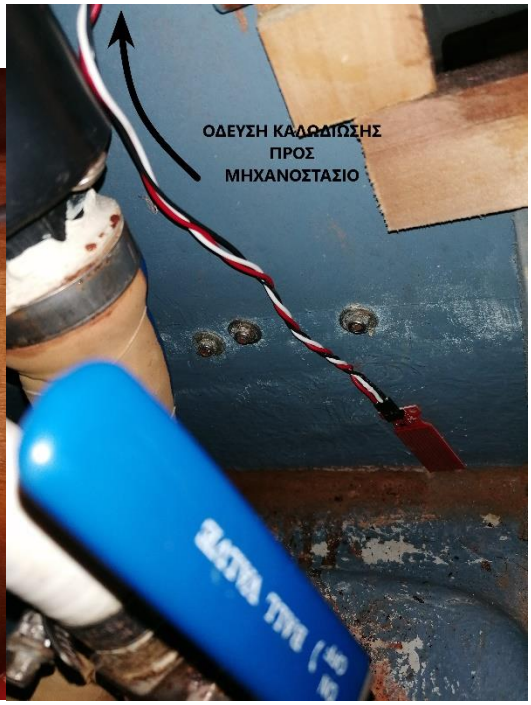
Επόμενο βήμα ήταν η τοποθέτηση πάλι στον χώρο του μηχανοστασίου, μιας και ήταν ήδη ανοικτό και είχαμε πρόσβαση, του MODULE A. Το συγκεκριμένο Module είναι υπεύθυνο για την θερμοκρασιοϋγρασιομέτρηση του μηχανοστασίου, και συνεπώς βόλευε να τοποθετηθεί στον ίδιο χώρο μιας και οι σχετικοί αισθητήρες βρίσκονται πάνω στην πλακέτα, και να τραβηχτεί καλωδίωση μέσα από την φρακτή, παράλληλα με την ήδη υπάρχουσα, αφενώς για να λαμβάνει μετρήσει θερμοκρασίας και υγρασίας από τον γειτονικό χώρο – το καθιστικό του σκάφους - και για να παίρνει μετρήσεις κατάκλισης του μικρού αντλιοστασίου πίσω από τον ψευδοτοίχο του WC του σκάφους.

Τοποθετήθηκε λοιπόν στην απέναντι μεριά από την προηγούμενη πλακέτα, σε ένα αρκετά δυσπρόσιτο σημείο, αλλά αναγκαίο καθώς ήταν η θέση που ήταν πιο κοντά στις οπές που πέρναγε η καλωδίωση στους διπλανούς χώρους.



Εικόνα 4.30 – Τοποθέτηση Module A

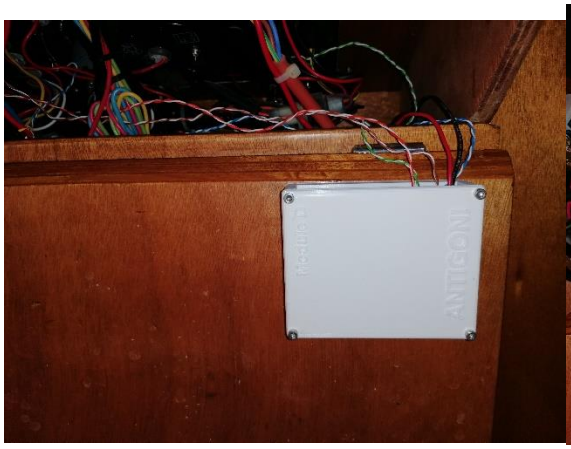


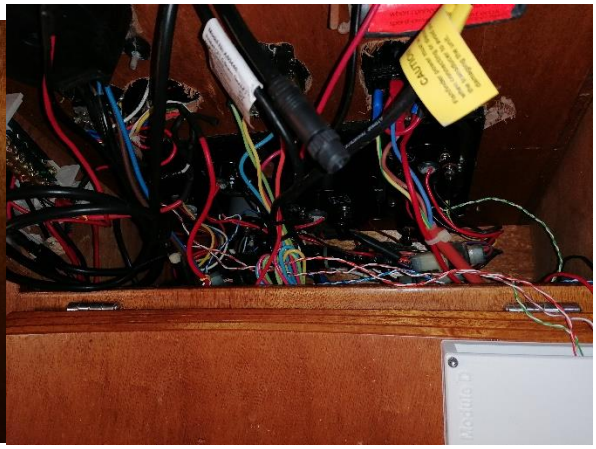


Εικόνες 4.31 – 4.32 - Αριστερά η μέτρηση θερμοκρασίας υγρασίας στον κύριο χώρο και δεξιά ο αισθητήρας κατάκλισης στο αντλιοστάσιο του WC

4.4.3 Τοποθέτηση του MODULE D

Η τοποθέτηση του Module D ήταν από τις πιο απαιτητικές, καθώς είχε αρκετή λεπτοδουλειά και επίσης επειδή ήρθαμε σε επαφή με την καλωδίωση του control panel της/του καπετάνιου, υπήρχε κίνδυνος να προκληθεί πρόβλημα σε κάποιο από τα όργανα. Ουσιαστικά για τις 4 ενδείξεις που λαμβάνουμε, παίρνουμε 4 σήματα από την κάθε καλωδίωση του εκάστοτε αντίστοιχου αναλογικού όργανου. Τοποθετήσαμε το module μας στο πίσω μέρος του πίνακα οργάνων ώστε να μην φαίνεται αφενώς και να μην είναι μακριά από τις πηγές του σήματός μας.





Εικόνες 4.32 – 4.33 – Τοποθέτηση Module D

Εικόνα 4.34 – Τοποθέτηση Module D

4.4.4 Τοποθέτηση MODULE C

Το τελευταίο και πιο απλό κομμάτι του συστήματός μας είναι το module που είναι υπεύθυνο για την λήψη μετεωρολογικών δεδομένων από το περιβάλλον του σκάφους. Έχει τοποθετηθεί στον εξωτερικό χώρο, κάτω από ένα φωτοβολταϊκό σύστημα που υπάρχει στο σκάφος, για προστασία από τις καιρικές συνθήκες. Εκεί αφενώς είναι υπο σκιά και συνεπώς οι μετρήσεις θερμοκρασίας και υγρασίας είναι ορθότερες από αυτές που θα λαμβάναμε τοποθετώντας το σε επαφή με την ηλιακή ακτινοβολία και αφετέρου βρίσκεται κοντά στην καλωδίωση του ανεμόμετρου.

Εικόνα 4.35 – Τοποθέτηση Module C

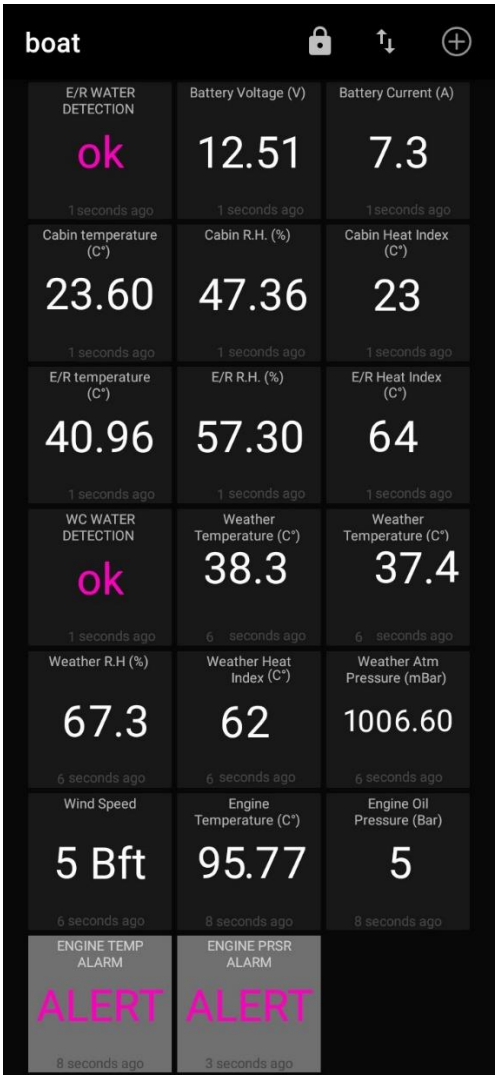
4.4.5 Τοποθέτηση του RaspberryPi

Το απλούστερο όλων ήταν φυσικά να βρεθεί μία θέση για να μπει η μονάδα που είναι υπεύθυνη για την τοπική δικτύωση. Αποφασίστηκε να τοποθετηθεί στο κέντρο του σκάφους ώστε να μην υπάρξει κάποιο πρόβλημα συνδεσιμότητας λόγω απόστασης ή εμποδίων. Η εύρεση αυτής της θέσης ήταν εύκολη καθώς ακριβώς στην είσοδο του χώρου ενδιαίτησης υπήρχε και η σχετική παροχή USB και μία θέση για την στερέωσή του.

Εικόνα 4.36 – Τοποθέτηση RaspberryPi

4.5 To End User Interface στην πράξη.

Η εφαρμογή που επιλέχτηκε όπως είπαμε, επιλέχθηκε με βάση την ευχρηστία και την εύκολη πρόσβαση στην πληροφορία από την/τον τελική χρήστρια/στη που στην συγκεκριμένη θα είναι ο/η καπετάνιος του σκάφους. Σκοπός λοιπόν είναι με μία ματιά να μπορεί να έχει πλήρη εποπτεία των μετρήσεων των αισθητήρων καθώς και να μπορεί να αντιληφθεί την ύπαρξη κάποιου επικίνδυνου σημείου στις μετρήσεις. Με πολύ απλό τρόπο η εφαρμογή προγραμματίστηκε να ενεργοποιεί φωτεινή ένδειξη όταν για παράδειγμα ενεργοποιείται κάποιο από τα alarm της μηχανής, πχ της θερμοκρασίας του νερού ή της πίεσης του λαδιού της, ή όταν υπάρχει κάπου ανίχνευση ύδατος από τους σχετικούς αισθητήρες κατάκλισης. Η εφαρμογή κατά την χρήση της φαίνεται παρακάτω.



Εικόνα 4.37 – End User Interface

Κεφάλαιο 5 - Συμπεράσματα

Στην παρούσα διπλωματική εργασία έγινε προσπάθεια να μεταφερθεί η ευρέως χρησιμοποιούμενη τεχνολογία του Internet of Things από τους οικιακούς χώρους στην ζωή εν πλω. Μεταβαίνοντας ουσιαστικά από την οικιακή ευχρηστία που προσφέρει σε απλό πολιτικό επίπεδο, σε μία τεχνική ευχρηστία σε επίπεδο χειρισμού σκάφους, προσφέροντας βοήθεια σε μία θέση ευθύνης που χρήζει την μεγαλύτερη δυνατή πρόσβαση σε πληροφορίες ανά δεδομένη χρονική στιγμή.

Η απόπειρα αυτή βασίστηκε επίσης στο γεγονός ότι όλα τα αντίστοιχα συστήματα ελέγχου για τα σκάφη αναψυχής έχουν αδικαιολόγητα αυξημένο κόστος. Παρουσιάστηκε λοιπόν η ανάγκη να αποδειχτεί στην πράξη πως υπάρχει η δυνατότητα να κατασκευαστεί ένα σύστημα με χαμηλό κόστος παραγωγής και συνεπώς αντίστοιχα χαμηλό κόστος κτήσης και παράλληλα χαμηλό κόστος συντήρησης.

Το σύστημα που δημιουργήσαμε έχει συνολικό κατασκευαστικό κόστος χαμηλότερο των 50-100 ευρώ και κάλυψε τις ανάγκες ενός μικρού σκάφους αναψυχής. Η αναγωγή αυτού σε εμπορική κλίμακα δείχνει ότι μπορεί να κατασκευαστεί κάτι σε ανάλογο κόστος.

Στο ίδιο πλαίσιο, έγινε προσπάθεια μέσα από το θέμα της διπλωματικής να κατασκευαστεί ένα σύστημα το οποίο σε έναν βαθμό θα μπορεί να επεκτείνεται κατά την βούληση της ιδιοκτήτριάς του ή του ιδιοκτήτη του, χωρίς να είναι απαραίτητος ένας «ειδικός», όπως στα περισσότερα κομμάτια πλέον της ζωής μας. Έγινε προσπάθεια να γίνουν τα πρώτα βήματα να κατασκευαστεί ένα modular system που θα δίνει ακριβώς αυτή την δυνατότητα. Στο μέλλον, σε μία ενδεχόμενη εμπορική χρήση του συστήματός μας θα μπορούσαν να κατασκευαστούν «τμήματα» ή «κομμάτια» τα οποία με αυτοματοποιημένο τρόπο θα μπορούσαν να συνδεθούν και να γίνουν κομμάτι του συνόλου χωρίς να είναι απαραίτητη η παρουσία ειδικών ή τεχνικών. Έτσι ο καθένας θα μπορούσε να προμηθευτεί την επέκταση του συστήματος που επιθυμεί και να την προσαρμώσει στο σκάφος του με τρόπο που ικανοποιεί τον ίδιο, στους χρόνους του και στα μέτρα που αυτός επιθυμεί. Έτσι η καθεμία ενδιαφερόμενη γίνεται κύρια της ιδιοκτησίας της σε έναν βαθμό και μπορεί να την δομήσει όπως εκείνη νομίζει, φέρνοντάς την πιο κοντά στην διαδικασία εγκατάστασης ενός βοηθητικού συστήματος που θα προσφέρει αυτά που χρειάζεται σαν καπετάνιος και πιθανώς να δώσει σε κάποιον τα εφόδια να αντιληφθεί σε μεγαλύτερο βαθμό την λειτουργία του συστήματος, έναντι μιας περίπτωσης που η εγκατάσταση έρχεται προμελετημένη και ο τελικός χρήστης λαμβάνει σκέτα δεδομένα χωρίς να αντιλαμβάνεται την διαδρομή τους και την αφετηρία τους.

Παράρτημα - Κώδικες

```
//---------BOARD A---------//
//-------------------------//

#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#include <DHT.h>
#include <Wire.h>

#define analogIN A0 // analog input for Water sensor
#define LED (2) // use onboard LED for convenience
#define DHTPIN1 D3 // Digital pin connected to the ENGINE ROOM DHT sensor. To Vin του dht μπορει να ειναι ειτε 3.3 είτε 5. δουλευει και με τα 2. ας επιλεξουμε 3.3 αν γινεται για ασφαλεια.
#define DHTPIN2 D2 // Digital pin connected to the CABIN DHT sensor
#define MAX_MSG_LEN (128)// maximum received message length
#define DHTTYPE DHT22   // DHT 22  AM2321
#define TESTDELAY 4000
#define RECOMMENDEDDELAY 10000


//const char* ssid = "tospitiepese"; //gia spiti
//const char* pass = "y1R0pVAV*";
const char* ssid = "Antigoni";
const char* pass = "kapetan!";
// MQTT Configuration
// if you have a hostname set for the MQTT server, you can use it here
//const char *serverHostname = "192.168.2.6";
// otherwise you can use an IP address like this
const IPAddress serverIPAddress(192, 168, 4, 1);// the topic we want to use
const char *topic1 = "er/temp";
const char *topic2 = "er/humid";
const char *topic3 = "er/hic";
```

```
const char *topic4 = "cabin/temp";
const char *topic5 = "cabin/humid";
const char *topic6 = "cabin/hic";
const char *topic7 = "wc/water";
float tempHumiditydata[3] = {0, 0, 0};



WiFiClient espClient;
PubSubClient client(espClient);
DHT dht1(DHTPIN1, DHTTYPE);
DHT dht2(DHTPIN2, DHTTYPE);

void setup() {
  // LED pin as output
  pinMode(LED, OUTPUT);
  digitalWrite(LED, HIGH);//LED --> OFF (reverse logic in NODEMCU)

  // Configure serial port for debugging
  Serial.begin(115200);

  dht1.begin(); //DHT1 SENSOR
  dht2.begin(); //DHT2 SENSOR

  connectWifi();// Initialise wifi connection - this will wait until connected

  // connect to MQTT server
  client.setServer(serverIPAddress, 1883);
  client.setCallback(callback);
}
```

```
//------------CONNECT TO WIFI------------//
//---------------------------------------//
void connectWifi() {
  delay(10);
  // Connecting to a WiFi network
  Serial.printf("\nConnecting to %s\n", ssid);
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, pass);
  while (WiFi.status() != WL_CONNECTED) {
    delay(250);
    Serial.print(".");
  }
  Serial.println("");
  Serial.print("WiFi connected on IP address ");
  Serial.println(WiFi.localIP());
  Serial.print("MAC: ");
  Serial.println(WiFi.macAddress());
}

//------------CONNECT TO MQTT SERVER------------//
//----------------------------------------------//
void connectMQTT() {
  // Wait until we're connected
  while (!client.connected()) {
    String clientId = "BOARD A";
    Serial.printf("MQTT connecting as client %s...\n", clientId.c_str());
    // Attempt to connect
    if (client.connect(clientId.c_str())) {
      Serial.println("MQTT connected");
      // Once connected, publish an announcement...
      client.publish(topic1, "temp test");
      client.publish(topic2, "humid test");
```

```
      client.publish(topic3, "hic test");
      client.publish(topic4, "temp test");
      client.publish(topic5, "humid test");
      client.publish(topic6, "hic test");
      client.publish(topic7, "water test");
      // ... and resubscribe
      client.subscribe(topic1);
      client.subscribe(topic2);
      client.subscribe(topic3);
      client.subscribe(topic4);
      client.subscribe(topic5);
      client.subscribe(topic6);
      client.subscribe(topic7);
      delay(1000);
      client.publish(topic1, "");// delay 1 sec, and then post a space so that the water alert tile is empty and not with a constant water test value
    } else {
      Serial.printf("MQTT failed, state %d, retrying...\n", client.state());
      // Wait before retrying
      delay(2500);
    }
  }
}

void callback(char *msgTopic, byte *msgPayload, unsigned int msgLength) {
  // copy payload to a static string
  static char message[MAX_MSG_LEN + 1];

  strncpy(message, (char *)msgPayload, msgLength);
  message[msgLength] = '\0';

  //Serial.printf("topic %s, message received: %s\n", topic, message);
}
```

```
//------------LED------------//
//-------------------------- //
void setLedState(boolean state) {
  // LED logic is inverted, low means on
  digitalWrite(LED, !state);
}

//------------DHT HUMIDITY-TEMPERATURE SENSOR------------//
//-----------------------------------------------------//
void humidTempSensor(float *data, int sensorNumber) {
  delay(5000);
  if (sensorNumber == 1) {
    data[0] = dht1.readTemperature(); //reading temperature
    data[1] = dht1.readHumidity(); // reading humidity
    data[2] = dht1.computeHeatIndex( data[0], data[1], false); // Compute heat index in Celsius. Δείκτης Δυσφορίας.
    if (isnan(data[0]) || isnan(data[1])) {
      Serial.println(F("Failed to read from DHT1 sensor!"));
      setLedState(true);
    } else {
      //LED PULSING INDICATING THAT THE SYSTEM IS WORKING//
      setLedState(false);
      delay(100);
      setLedState(true);
      delay(250);
      setLedState(false);
    }
  } else {
    data[0] = dht2.readTemperature(); //reading temperature
    data[1] = dht2.readHumidity(); // reading humidity
    data[2] = dht2.computeHeatIndex( data[0], data[1], false); // Compute heat index in Celsius. Δείκτης Δυσφορίας.
    if (isnan(data[0]) || isnan(data[1])) {
```

```
      Serial.println(F("Failed to read from DHT2 sensor!"));
      setLedState(true);
    } else {
      //LED PULSING INDICATING THAT THE SYSTEM IS WORKING//
      setLedState(false);
      delay(100);
      setLedState(true);
      delay(250);
      setLedState(false);
    }
  }
}

//------------MAIN LOOP------------//
//---------------------------------//
void loop() {
  if (!client.connected()) {
    connectMQTT();
  }
  // this is ESSENTIAL!
  client.loop();

  //DHT1 DATA//
  humidTempSensor(tempHumiditydata, 1);
  Serial.printf("t1 %f , h1 %f , hic1 %f\n", tempHumiditydata[0], tempHumiditydata[1], tempHumiditydata[2]);
  String temp1 = String(tempHumiditydata[0]);
  String humid1 = String(tempHumiditydata[1]);
  String hic1 = String(tempHumiditydata[2]);

  //DHT2 DATA//
  humidTempSensor(tempHumiditydata, 2);
  Serial.printf("t2 %f , h2 %f , hic2 %f\n", tempHumiditydata[0], tempHumiditydata[1], tempHumiditydata[2]);
```

```
  String temp2 = String(tempHumiditydata[0]);
  String humid2 = String(tempHumiditydata[1]);
  String hic2 = String(tempHumiditydata[2]);

  //MQTT PUBLISHING//
  client.publish(topic1, temp1.c_str());
  client.publish(topic2, humid1.c_str());
  client.publish(topic3, hic1.c_str());
  client.publish(topic4, temp2.c_str());
  client.publish(topic5, humid2.c_str());
  client.publish(topic6, hic2.c_str());
  //WATER SENSOR DATA//
  float  sensorValue = analogRead(analogIN);
  if (sensorValue > 400) {
    client.publish(topic7, "ALERT");
  }
}
```

```
//---------BOARD B---------//
//-------------------------//


#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#include <Wire.h>
#include <Adafruit_ADS1015.h>


#define LED (2) // use onboard LED for convenience
#define MAX_MSG_LEN (128)// maximum received message length
#define TESTDELAY 100
#define RECOMMENDEDDELAY 100



//const char* ssid = "tospitiepese";
//const char* pass = "y1R0pVAV*";
const char* ssid = "Antigoni";
const char* pass = "kapetan!";
// MQTT Configuration
// if you have a hostname set for the MQTT server, you can use it here
//const char *serverHostname = "192.168.2.6";
// otherwise you can use an IP address like this
const IPAddress serverIPAddress(192, 168, 4, 1);// the broker we want to use
const char *topic1 = "er/water";
const char *topic2 = "battery/voltage";
const char *topic3 = "battery/current";
const int shuntResistance = 0.0005;
```

```
WiFiClient espClient; //WiFi
PubSubClient client(espClient);// MQTT
Adafruit_ADS1115 ads; // ADC



void setup() {
  // LED pin as output
  pinMode(LED, OUTPUT);
  digitalWrite(LED, HIGH); //LED --> OFF (reverse logic in NODEMCU)

  // Configure serial port for debugging
  Serial.begin(115200);

  connectWifi();// Initialise wifi connection - this will wait until connected

  // connect to MQTT server
  client.setServer(serverIPAddress, 1883);
  client.setCallback(callback);

  // ----------ADS1115 SETUP GUIDE---------//
  // ads.setGain(GAIN_TWOTHIRDS);  // 2/3x gain +/- 6.144V  1 bit = 0.1875mV (default)
  ads.setGain(GAIN_ONE);        // 1x gain   +/- 4.096V  1 bit = 0.125mV
  // ads.setGain(GAIN_TWO);        // 2x gain   +/- 2.048V  1 bit = 0.0625mV
  // ads.setGain(GAIN_FOUR);       // 4x gain   +/- 1.024V  1 bit = 0.03125mV
  // ads.setGain(GAIN_EIGHT);      // 8x gain   +/- 0.512V  1 bit = 0.015625mV
  // ads.setGain(GAIN_SIXTEEN);    // 16x gain  +/- 0.256V  1 bit = 0.0078125mV

  ads.begin();
}
```

```
//------------CONNECT TO WIFI------------//
//---------------------------------------//
void connectWifi() {
  delay(10);
  // Connecting to a WiFi network
  Serial.printf("\nConnecting to %s\n", ssid);
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, pass);
  while (WiFi.status() != WL_CONNECTED) {
    delay(250);
    Serial.print(".");
  }
  Serial.println("");
  Serial.print("WiFi connected on IP address ");
  Serial.println(WiFi.localIP());
  Serial.print("MAC: ");
  Serial.println(WiFi.macAddress());
}

//------------CONNECT TO MQTT SERVER------------//
//----------------------------------------------//
void connectMQTT() {
  // Wait until we're connected
  while (!client.connected()) {
    String clientId = "BOARD B";
    Serial.printf("MQTT connecting as client %s...\n", clientId.c_str());

    // Attempt to connect
    if (client.connect(clientId.c_str())) {
      Serial.println("MQTT connected");
      // Once connected, publish an announcement...
      client.publish(topic1, "water test");
```

```
      client.publish(topic2, "volt test");
      client.publish(topic3, "current test");
      // ... and resubscribe
      client.subscribe(topic1);
      client.subscribe(topic2);
      client.subscribe(topic3);
      delay(1000);
      client.publish(topic1, "");// delay 1 sec, and then post a space so that the water alert tile is empty and not with a constant water test value
    } else {
      Serial.printf("MQTT failed, state %d, retrying...\n", client.state());
      // Wait before retrying
      delay(2500);
    }
  }
}

void callback(char *msgTopic, byte *msgPayload, unsigned int msgLength) {
  // copy payload to a static string
  static char message[MAX_MSG_LEN + 1];

  strncpy(message, (char *)msgPayload, msgLength);
  message[msgLength] = '\0';

  //Serial.printf("topic %s, message received: %s\n", topic, message);
}



//------------LED------------//
//-------------------------- //
void setLedState(boolean state) {
  // LED logic is inverted, low means on
  digitalWrite(LED, !state);
```

```
}

//------------MAIN LOOP------------//
//--------------------------------//
void loop() {
  if (!client.connected()) {
    connectMQTT();
  }
  // this is ESSENTIAL!
  client.loop();

  // idle
  delay(TESTDELAY);

  //BATTERY VOLTAGE//
  float fadc0;
  fadc0 = (ads.readADC_SingleEnded(0) * 0.125 * 5 * 0.001); //make it mV, reverse step down. then make it V./// the input reading is 2.5V approximately.
  Serial.println(ads.readADC_SingleEnded(0) * 0.125 * 5);//for debugging
  Serial.println(fadc0, 3);//for debugging

  String  Stringfadc0 = String(fadc0);
  client.publish(topic2, Stringfadc0.c_str()); // PUBLISHES BATTERY VOLTAGE

  //LED INDICATING THAT THE SYSTEM IS WORKING//
  setLedState(false);
  delay(100);
  setLedState(true);
  delay(250);
  setLedState(false);

  //WATER SENSOR//
  //(SAME GAIN AS BATTERY VOLTAGE)//
```

```
 float fadc1;
 fadc1 = ads.readADC_SingleEnded(1) * 0.125;
 if (fadc1 > 700) {
   client.publish(topic1, "ALERT");
   // 5 SECONDS LED STROBE FOR VISUAL ALARM //
   for (int i = 1 ; i < 50 ; i++) {
     setLedState(false);
     delay(50);
     setLedState(true);
     delay(50);
     setLedState(false);


   }
 }
 // BATTERY CURRENT DRAW //
   int16_t results;
 float multiplier = 0.125F; // multiplier depending on gain selected. specify that 0.125 is float
 results = ads.readADC_Differential_2_3(); // differential reading between A2 and A3
 float voltage_drop = results * multiplier * 5; // make it mV and reverse step down.
 float current = voltage_drop / shuntResistance ;
 String Strcurrent = String(current);
 client.publish(topic3, Strcurrent.c_str());
}
```

```
//---------BOARD C---------//
//-------------------------//

#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#include <DHT.h>
#include <Wire.h>
#include <Adafruit_BMP280.h>



#define LED (2) // use onboard LED for convenience
#define DHTPIN D3 // Digital pin connected to the DHT sensor. mhn ksexasw na einai D3 kai oxi sketo 3. To Vin του dht μπορει να ειναι ειτε 3.3 είτε 5. δουλευει και με τα 2. ας επιλεξουμε 3.3 αν γινεται για ασφαλεια.
#define MAX_MSG_LEN (128)// maximum received message length
#define DHTTYPE DHT22   // DHT 22  AM2321
#define TESTDELAY 4000
#define RECOMMENDEDDELAY 10000
#define BMP_SCK  (D1)// I2C PINS GIA TO BAROMETRIC AND TEMP SENSOR
#define BMP_SDI (D2)// I2C PINS GIA TO BAROMETRIC AND TEMP SENSOR


//const char* ssid = "tospitiepese"; //gia spiti
//const char* pass = "y1R0pVAV*";
const char* ssid = "Antigoni";
const char* pass = "kapetan!";
// MQTT Configuration
// if you have a hostname set for the MQTT server, you can use it here
//const char *serverHostname = "192.168.2.6";
// otherwise you can use an IP address like this
const IPAddress serverIPAddress(192, 168, 4, 1);// the topic we want to use
const char *topic1 = "weather/temp";
const char *topic2 = "weather/humid";
const char *topic3 = "weather/hic";
const char *topic4 = "weather/temp2";
```

```
const char *topic5 = "weather/barometer";
const char *topic6 = "weather/anemometer";
float tempHumiditydata[3] = {0, 0, 0};
float windSpeed = 0;


WiFiClient espClient;
PubSubClient client(espClient);
DHT dht(DHTPIN, DHTTYPE);
Adafruit_BMP280 bmp;


void setup() {
  // LED pin as output
  pinMode(LED, OUTPUT);
  digitalWrite(LED, HIGH);

  // Configure serial port for debugging
  Serial.begin(115200);

  dht.begin(); //DHT SENSOR

  connectWifi();// Initialise wifi connection - this will wait until connected

  // connect to MQTT server
  client.setServer(serverIPAddress, 1883);
  client.setCallback(callback);

  // SETTING UP AND TESTING BMP280 SENSOR//
  Serial.println(F("BMP280 test"));
  if (!bmp.begin()) {
    Serial.println(F("Could not find a valid BMP280 sensor, check wiring!")); //IF BMP SENSOR IS NOT FOUND THEN DO NOT PROCEED
```

```
    while (1);
  }

  /* Default settings from datasheet. */
  bmp.setSampling(Adafruit_BMP280::MODE_NORMAL,     /* Operating Mode. */
                  Adafruit_BMP280::SAMPLING_X2,     /* Temp. oversampling */
                  Adafruit_BMP280::SAMPLING_X16,    /* Pressure oversampling */
                  Adafruit_BMP280::FILTER_X16,      /* Filtering. */
                  Adafruit_BMP280::STANDBY_MS_500); /* Standby time. */
}

//------------CONNECT TO WIFI------------//
//---------------------------------------//
void connectWifi() {
  delay(10);
  // Connecting to a WiFi network
  Serial.printf("\nConnecting to %s\n", ssid);
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, pass);
  while (WiFi.status() != WL_CONNECTED) {
    delay(250);
    Serial.print(".");
  }
  Serial.println("");
  Serial.print("WiFi connected on IP address ");
  Serial.println(WiFi.localIP());
  Serial.print("MAC: ");
  Serial.println(WiFi.macAddress());
}
```

```
//------------CONNECT TO MQTT SERVER------------//
//----------------------------------------------//
void connectMQTT() {
  // Wait until we're connected
  while (!client.connected()) {
    String clientId = "BOARD c";
    Serial.printf("MQTT connecting as client %s...\n", clientId.c_str());
    // Attempt to connect
    if (client.connect(clientId.c_str())) {
      Serial.println("MQTT connected");
      // Once connected, publish an announcement...
      client.publish(topic1, "temp test");
      client.publish(topic2, "humid test");
      client.publish(topic3, "hic test");
      client.publish(topic4, "bartemp test");
      client.publish(topic5, "barometer test");
      client.publish(topic6, "anemometer test");
      // ... and resubscribe
      client.subscribe(topic1);
      client.subscribe(topic2);
      client.subscribe(topic3);
      client.subscribe(topic4);
      client.subscribe(topic5);
      client.subscribe(topic6);
    } else {
      Serial.printf("MQTT failed, state %d, retrying...\n", client.state());
      // Wait before retrying
      delay(2500);
    }
  }
}
```

```
void callback(char *msgTopic, byte *msgPayload, unsigned int msgLength) {
  // copy payload to a static string
  static char message[MAX_MSG_LEN + 1];

  strncpy(message, (char *)msgPayload, msgLength);
  message[msgLength] = '\0';

  //Serial.printf("topic %s, message received: %s\n", topic, message);
}

//------------LED------------//
//-------------------------- //
void setLedState(boolean state) {
  // LED logic is inverted, low means on
  digitalWrite(LED, !state);
}

//------------DHT HUMIDITY-TEMPERATURE SENSOR------------//
//-----------------------------------------------------//
void humidTempSensor(float *data) {
  delay(5000);
  data[0] = dht.readTemperature(); //reading temperature
  data[1] = dht.readHumidity(); // reading humidity
  data[2] = dht.computeHeatIndex( data[0], data[1], false); // Compute heat index in Celsius. Δείκτης Δυσφορίας.
  if (isnan(data[0]) || isnan(data[1])) {
    Serial.println(F("Failed to read from DHT sensor!"));
    setLedState(true);
  } else {
    //LED PULSING INDICATING THAT THE SYSTEM IS WORKING//
    setLedState(false);
    delay(100);
    setLedState(true);
```

```
    delay(250);
    setLedState(false);
  }
}

//------------ANEMOMETER SENSOR------------//
//-----------------------------------------//
String  anemometer() {
  int total = 0;
  for ( int i = 0; i <= 4; i++) {
    total = total + analogRead(A0);//διαβαζουμε την τιμη του ανεμομετρου και την προσθετουμε στο συνολο/
    delay(1000); //περιμένουμε 1 δευτερολεπτο και ξαναδιαβαζουμε. ετσι θα παρουμε μετα μεση τιμη για 5 δευτερλοπετα μετρησεων
  }
  float averagesensorvalue = total * 0.2; // μεσος όρος 5 μετρήσεων του ανεμομετρου για 5 δευτερολεπτα
  float voltage = averagesensorvalue *  0.003222657; // μετατροπη απο 0-1024 σε 0-3.3 volt
  //CALCULATING SPEED FROM VOLTAGE//
  if (voltage <= 0.43) {
    windSpeed = 0;
  } else {
    windSpeed = ((voltage - 0.4) * 32 / (2 - 0.4)) * 2.232694;
  }
  //BEAUFORT SCALING//
  Serial.println(windSpeed);
  if (windSpeed <= 1) {
    Serial.println("0 Bft");
    String windSpeedBft = "0 Bft";
    return windSpeedBft;
  } else if (windSpeed > 1 && windSpeed <= 1.5) {
    Serial.println("1 Bft");
    String windSpeedBft = "1 Bft";
    return windSpeedBft;
  } else if (windSpeed > 1.5 && windSpeed <= 3.3) {
```

```
  Serial.println("2 Bft");
  String windSpeedBft = "2 Bft";
  return windSpeedBft;
} else if (windSpeed > 3.3 && windSpeed <= 5.5) {
  Serial.println("3 Bft");
  String windSpeedBft = "3 Bft";
  return windSpeedBft;
} else if (windSpeed > 5.5 && windSpeed <= 7.9) {
  Serial.println("4 Bft");
  String windSpeedBft = "4 Bft";
  return windSpeedBft;
} else if (windSpeed > 7.9 && windSpeed <= 10.7) {
  Serial.println("5 Bft");
  String windSpeedBft = "5 Bft";
  return windSpeedBft;
} else if (windSpeed > 10.7 && windSpeed <= 13.8) {
  Serial.println("6 Bft");
  String windSpeedBft = "6 Bft";
  return windSpeedBft;
} else if (windSpeed > 13.8 && windSpeed <= 17.1) {
  Serial.println("7 Bft");
  String windSpeedBft = "7 Bft";
  return windSpeedBft;
} else if (windSpeed > 17.2 && windSpeed <= 20.7) {
  Serial.println("8 Bft");
  String windSpeedBft = "8 Bft";
  return windSpeedBft;
} else if (windSpeed > 20.7 && windSpeed <= 24.4) {
  Serial.println("9 Bft");
  String windSpeedBft = "9 Bft";
  return windSpeedBft;
} else if (windSpeed > 24.4 && windSpeed <= 28.4) {
```

```
    Serial.println("10 Bft");
    String windSpeedBft = "10 Bft";
    return windSpeedBft;
  } else if (windSpeed > 28.4 && windSpeed <= 32.6) {
    Serial.println("11 Bft");
    String windSpeedBft = "11 Bft";
    return windSpeedBft;
  } else {
    Serial.println("12 Bft");
    String windSpeedBft = "12 Bft";
    return windSpeedBft;
  }
}

//------------MAIN LOOP------------//
//---------------------------------//
void loop() {
  if (!client.connected()) {
    connectMQTT();
  }
  // this is ESSENTIAL!
  client.loop();

  //DHT DATA//
  humidTempSensor(tempHumiditydata);
  Serial.printf("t %f , h %f , hic %f\n", tempHumiditydata[0], tempHumiditydata[1], tempHumiditydata[2]);
  String temp = String(tempHumiditydata[0]);
  String humid = String(tempHumiditydata[1]);
  String hic = String(tempHumiditydata[2]);

  //BMP DATA//
  String pressure = String(bmp.readPressure() * 0.01); // ΣE mBAR
```

```
  String bartemp = String(bmp.readTemperature());
  //ANEMOMETER DATA//
  String windSpeedBft = anemometer();//το βγαζει η συναρτηση σε string

  //MQTT PUBLISHING//
  client.publish(topic1, temp.c_str());
  client.publish(topic2, humid.c_str());
  client.publish(topic3, hic.c_str());
  client.publish(topic4, bartemp.c_str());
  client.publish(topic5, pressure.c_str());
  client.publish(topic6, windSpeedBft.c_str());
}
```

```
//---------BOARD D---------//
//-------------------------//

#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#include <Wire.h>
#include <Adafruit_ADS1015.h>


#define LED (2) // use onboard LED for convenience
#define MAX_MSG_LEN (128)// maximum received message length
#define TESTDELAY 100
#define RECOMMENDEDDELAY 100



//const char* ssid = "tospitiepese";
//const char* pass = "y1R0pVAV*";
const char* ssid = "Antigoni";
const char* pass = "kapetan!";
// MQTT Configuration
// if you have a hostname set for the MQTT server, you can use it here
//const char *serverHostname = "192.168.2.6";
// otherwise you can use an IP address like this
const IPAddress serverIPAddress(192, 168, 4, 1);// the broker we want to use
const char *topic1 = "engine/temp";
const char *topic2 = "engine/oilpressure";
const char *topic3 = "engine/tempalarm";
const char *topic4 = "engine/pressurealarm";


WiFiClient espClient; //WiFi
```

```
PubSubClient client(espClient);// MQTT
Adafruit_ADS1115 ads; // ADC



void setup() {
  // LED pin as output
  pinMode(LED, OUTPUT);
  digitalWrite(LED, HIGH); //LED --> OFF (reverse logic in NODEMCU)

  // Configure serial port for debugging
  Serial.begin(115200);

  connectWifi();// Initialise wifi connection - this will wait until connected

  // connect to MQTT server
  client.setServer(serverIPAddress, 1883);
  client.setCallback(callback);

  // ----------ADS1115 SETUP GUIDE---------//
  // ads.setGain(GAIN_TWOTHIRDS);  // 2/3x gain +/- 6.144V  1 bit = 0.1875mV (default)
  ads.setGain(GAIN_ONE);        // 1x gain   +/- 4.096V  1 bit = 0.125mV
  // ads.setGain(GAIN_TWO);        // 2x gain   +/- 2.048V  1 bit = 0.0625mV
  // ads.setGain(GAIN_FOUR);       // 4x gain   +/- 1.024V  1 bit = 0.03125mV
  // ads.setGain(GAIN_EIGHT);      // 8x gain   +/- 0.512V  1 bit = 0.015625mV
  // ads.setGain(GAIN_SIXTEEN);    // 16x gain  +/- 0.256V  1 bit = 0.0078125mV

  ads.begin();
}
```

```
//------------CONNECT TO WIFI------------//
//---------------------------------------//
void connectWifi() {
  delay(10);
  // Connecting to a WiFi network
  Serial.printf("\nConnecting to %s\n", ssid);
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, pass);
  while (WiFi.status() != WL_CONNECTED) {
    delay(250);
    Serial.print(".");
  }
  Serial.println("");
  Serial.print("WiFi connected on IP address ");
  Serial.println(WiFi.localIP());
  Serial.print("MAC: ");
  Serial.println(WiFi.macAddress());
}

//------------CONNECT TO MQTT SERVER------------//
//----------------------------------------------//
void connectMQTT() {
  // Wait until we're connected
  while (!client.connected()) {
    String clientId = "BOARD D";
    Serial.printf("MQTT connecting as client %s...\n", clientId.c_str());

    // Attempt to connect
    if (client.connect(clientId.c_str())) {
      Serial.println("MQTT connected");
      // Once connected, publish an announcement...
      client.publish(topic1, "temp test");
```

```
      client.publish(topic2, "oilprs test");
      client.publish(topic3, "tempalarm test");
      client.publish(topic4, "prsralarm test");
      // ... and resubscribe
      client.subscribe(topic1);
      client.subscribe(topic2);
      client.subscribe(topic3);
      client.subscribe(topic4);
    } else {
      Serial.printf("MQTT failed, state %d, retrying...\n", client.state());
      // Wait before retrying
      delay(2500);
    }
  }
}

void callback(char *msgTopic, byte *msgPayload, unsigned int msgLength) {
  // copy payload to a static string
  static char message[MAX_MSG_LEN + 1];

  strncpy(message, (char *)msgPayload, msgLength);
  message[msgLength] = '\0';

  //Serial.printf("topic %s, message received: %s\n", topic, message);
}

//------------LED------------//
//-------------------------- //
void setLedState(boolean state) {
  // LED logic is inverted, low means on
  digitalWrite(LED, !state);
}
```

```
//------------MAIN LOOP------------//
//--------------------------------//
void loop() {
  if (!client.connected()) {
    connectMQTT();
  }
  // this is ESSENTIAL!
  client.loop();

  // idle
  delay(TESTDELAY);

  //ENGINE WATER TEMPERATURE//
  float fadc0;
  fadc0 = ads.readADC_SingleEnded(0) * 0.125 * 5;//bits to volts. reverse step down. the input reading is 2.5V approximately.
  float temp = fadc0;
  String  Stringfadc0 = String(temp);
  client.publish(topic1, Stringfadc0.c_str()); // PUBLISHES ENGINE WATER TEMPERATURE

  //LED INDICATING THAT THE SYSTEM IS WORKING//
  setLedState(false);
  delay(100);
  setLedState(true);
  delay(250);
  setLedState(false);

  //OIL PRESSURE SENSOR//
  float fadc1;
  fadc0 = ads.readADC_SingleEnded(1) * 0.125 * 5;//bits to volts. reverse step down. the input reading is 2.5V approximately.
  float oil_pressure = fadc1;   // multiplied by something MISSING CONVERTION FROM VOLTS TO TEMPERATURE //
```

```
  String Stringfadc1 = String(oil_pressure);
  client.publish(topic2, Stringfadc0.c_str()); // PUBLISHES ENGINE OIL PRESSURE


  // TEMPERATURE ALARM //
  float fadc2;
  fadc2 = ads.readADC_SingleEnded(2) * 0.125 * 5; //bits to volts. reverse step down. the input reading is 0.2V-0.8V approximately.
  if (fadc2 > 1000) { // if the input is greater than 1 volts, it means we have an alarm signal.
    client.publish(topic3, "ALERT");
    // 5 SECONDS LED STROBE FOR VISUAL ALARM //
    for (int i = 1 ; i < 50 ; i++) {
      setLedState(false);
      delay(50);
      setLedState(true);
      delay(50);
      setLedState(false);
    }
  }
  // OIL PRESSURE ALARM //
  float fadc3;
  fadc3 = ads.readADC_SingleEnded(3) * 0.125 * 5; //bits to volts. reverse step down. the input reading is 0.2V-0.8V approximately.
  if (fadc3 > 1000) { // if the input is greater than 1 volt, it means we have an alarm signal.
    client.publish(topic4, "ALERT");
    // 5 SECONDS LED STROBE FOR VISUAL ALARM //
    for (int i = 1 ; i < 50 ; i++) {
      setLedState(false);
      delay(50);
      setLedState(true);
      delay(50);
      setLedState(false);


    }
```

Βιβλιογραφία

https://www.adafruit.com/product/2651

https://www.sparkfun.com/datasheets/Sensors/Temperature/DHT22.pdf

https://cdn-shop.adafruit.com/datasheets/BST-BMP280-DS001-11.pdf

https://www.adafruit.com/product/1733

https://www.hellasdigital.gr/electronics/sensors/humidity-sensor/water-level-sensor-depth-of-detection-for-arduino/

https://www.hellasdigital.gr/electronics/sensors/current-sensors/voltage-sensor-module-for-robot-arduino-dc-0-25-v/

https://www.adafruit.com/product/1085

https://grobotronics.com/dc-dc-converter-step-down-9-36v-5a.html

https://en.wikipedia.org/wiki/Raspberry_Pi

https://www.techpress.gr/index.php/archives/85470

https://en.wikipedia.org/wiki/Shunt_(electrical)

https://cy.ipc2u.com/articles/articles-and-reviews/ti-einai-to-mqtt-kai-giati-to-chreiazomaste-sto-iiot/

https://en.wikipedia.org/wiki/MQTT

«Ανάπτυξη Υποδομής MQTT για IoT Εφαρμογές», Αθηνά Πλασκασοβίτη

http://artemis.cslab.ece.ntua.gr:8080/jspui/handle/123456789/17577

https://play.google.com/store/apps/details?id=net.routix.mqttdash&hl=en&gl=US

https://www.interaction-design.org/literature/topics/prototyping

https://www.instructables.com/Steps-to-Setup-Arduino-IDE-for-NODEMCU-ESP8266-WiF/