Τμήμα Ηλεκτρολόγων Μηχανικών
και Μηχανικών Υπολογιστών
Πανεπιστήμιο Πελοποννήσου

Πτυχιακή Εργασία

Σχεδιασμός και ανάπτυξη ενσωματωμένου συστήματος για τη διευκόλυνση της μελισσοκομίας.

Γεώργιος Ρουμελιώτης

Αριθμός Μητρώου 3103

Επιβλέπων καθηγητής: ΧΡΙΣΤΟΔΟΥΛΟΥ ΣΩΤΗΡΙΟΣ

Επιτροπή Αξιολόγησης: ΤΖΗΜΑΣ ΙΩΑΝΝΗΣ
ΠΑΡΑΣΚΕΥΑΣ ΜΗΧΑΗΛ

«Εάν η μέλισσα εξαφανιζόταν από την επιφάνεια του πλανήτη, τότε ο άνθρωπος θα είχε μόνο τέσσερα χρόνια ζωής. Όχι άλλες μέλισσες, όχι άλλη επικονίαση, όχι άλλα φυτά, όχι άλλα ζώα, όχι άλλος άνθρωπος.»

- Άλμπερτ Αϊνστάιν

ΠΡΟΛΟΓΟΣ

Η παρούσα πτυχιακή εργασία έχει εκπονηθεί στο Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Πανεπιστημίου Πελοποννήσου και στόχος της είναι η κατασκευή μιας μελισσοκομικής ζυγαριάς η οποία έχει την δυνατότητα να ενημερώνει τους μελισσοκόμους μέσω SMS ή και μέσω online IoT πλατφόρμας για την κατάσταση της κυψέλης τους, δηλαδή το βάρος, την θερμοκρασία και την υγρασία.

Η διαδικασία που αναγκάζονται να ακολουθούν οι μελισσοκόμοι για να ελέγξουν τις κυψέλες τους, δηλαδή να πηγαίνουν ξεχωριστά σε κάθε μέρος όπου έχουν τοποθετήσει μελίσσι και έπειτα να το επιθεωρούν για να καταλάβουν σε τι κατάσταση βρίσκεται και τι ανάγκες έχει εκείνη την συγκεκριμένη χρονική στιγμή, είναι εξαιρετικά δύσκολη, χρονοβόρα αλλά και κοστοβόρα. Έκτος από αυτό, δεν τους παρέχει πλήρη ενημέρωση αφού αδυνατούν να επιβλέπουν τα μελίσσια σε τακτά χρονικά διαστήματα. Η λύση της δημιουργίας και κατασκευής μιας μελισσοκομικής ζυγαριάς, που χρησιμοποιεί αισθητήρες για την επίτευξη των παραπάνω δυνατοτήτων, καθιστά την επίβλεψη των κυψελών πολύ πιο εύκολη, εξασφαλίζει την εξοικονόμηση χρόνου ενώ ταυτόχρονα αυξάνει το κέρδος της εκάστοτε επιχείρησης. Αυτό είναι δυνατό διότι ο χρήστης δεν χρειάζεται να παραβρίσκεται συνέχεια στο μέρος όπου είναι τοποθετημένες οι κυψέλες, καθώς ενημερώνεται διαρκώς για την κατάσταση τους όπου και αν βρίσκεται.

Ευχαριστώ θερμά τον κ. Χριστοδούλου Σωτήριο, τόσο για την ευκαιρία που μου έδωσε να ασχοληθώ με ένα τόσο ενδιαφέρον θέμα, όσο και για την αμέριστη βοήθεια που μου πρόσφερε καθόλη την διάρκεια της φοιτητικής μου πορείας.

ΠΕΡΙΛΗΨΗ

Σκοπός της παρούσας πτυχιακής εργασίας είναι η μελέτη της ανάπτυξης ενός ενσωματωμένου συστήματος για την διευκόλυνση της μελισσοκομίας και η υλοποίησή του.

Ο σχεδιασμός του συστήματος αποτελείται από δύο κύρια μέρη. Πρώτα, από πλευράς υλικού με την χρήση EDA εργαλείου και του μικροελεγκτή ESP32. Έπειτα από πλευράς software με την χρήση γλωσσών προγραμματισμού, όπως C/C++, HTML5, CSS3 και JavaScript.

Η εργασία αυτή αναπτύσσεται σε επτά κεφάλαια. Στο πρώτο κεφάλαιο, γίνεται μια εισαγωγή στα προβλήματα που αντιμετωπίζει ο κλάδος της μελισσοκομίας καθώς και στις τεχνολογίες που θα χρησιμοποιηθούν για την επίλυση τους.

Στο δεύτερο κεφάλαιο, αναλύεται η μεθοδολογία υλοποίησης ενός ενσωματωμένου συστήματος, οι προδιαγραφές απαιτήσεων υλικού και λογισμικού, όπως επίσης και τα υλικά, οι τεχνικές και οι υπηρεσίες που θα χρησιμοποιηθούν για την ανάπτυξη του συστήματος.

Έπειτα, το τρίτο κεφάλαιο περιγράφει αναλυτικά τα μέρη της υλοποίησης ενός κυκλώματος, πιο συγκεκριμένα το PCB, το schematic, το Gerber file και το BOM.

Στην συνέχεια, το τέταρτο κεφάλαιο ασχολείται με τα απαραίτητα εργαλεία που χρειάζονται για την ολοκληρωμένη υλοποίηση του project. Αυτά είναι, το EasyEDA για την ανάπτυξη του κυκλώματος και το Platform IO το όποιο είναι υπεύθυνο για την ανάπτυξη λογισμικού.

Στο πέμπτο κεφάλαιο παρουσιάζονται και γίνεται ανάλυση σε όλα τα αρχεία που χρησιμοποιήθηκαν για την δημιουργία του συστήματος δηλαδή το σχηματικό διάγραμμα, το Gerber file, το PCB layout καθώς και ο κώδικας της συσκευής.

Μετά από αυτό, στο έκτο κεφάλαιο εξηγείται η χρήση του συστήματος, ειδικότερα η διαδικασία που πρέπει να ακολουθηθεί για την ρύθμιση της μελισσοκομικής ζυγαριάς και την ρύθμιση της IoT εφαρμογής, ThingSpeak.

Τέλος, το έβδομο και τελευταίο κεφάλαιο κλείνει με τα σχετικά συμπεράσματα που προκύπτουν και αναφέρει τα σχέδια για τις πιθανές μελλοντικές βελτιώσεις της μελισσοκομικής ζυγαριάς.

Abstract

The purpose of this thesis is to study the development of an embedded system to facilitate beekeeping and its implementation.

The design of the system consists of two main parts. First, in terms of hardware, the EDA tool and the ESP32 microcontroller are used. Then in terms of software, programming languages such as C / C ++, HTML5, CSS3 and JavaScript are used.

This work is developed in seven chapters. In the first chapter, an introduction is made to the problems faced by the beekeeping industry as well as to the technologies that will be used to solve them.

The second chapter analyzes the methodology of implementation of an embedded system, the specifications of hardware and software requirements, as well as the materials, techniques and services that will be used for the development of the system.

Next, the third chapter describes in detail the parts of the implementation of a circuit, more specifically the PCB, schematic, Gerber file and BOM.

Then, the fourth chapter deals with the necessary tools needed for the complete implementation of the project. These are EasyEDA for circuit development and Platform IO which is responsible for the software development.

The fifth chapter presents and analyzes all the files used to create the system, i.e. the schematic diagram, the Gerber file, the PCB layout and the device code.

After that, the sixth chapter explains the usage of the system, in particular the procedure to be followed for the adjustment of the beehive scale and the adjustment of the IoT application, ThingSpeak.

Finally, the seventh and final chapter closes with the relevant conclusions that emerge and mentions the plans for possible future improvements of the beehive scale.

Περιεχόμενα

ΚΕΦΑΛΑΙΟ 1
Εισαγωγή

1.1 Τα Προβλήματα της μελισσοκομίας και τρόποι αντιμετώπισης

Είναι ευρύτατα διαδεδομένη η άποψη ότι υπάρχουν διάφοροι παράγοντες που καθιστούν σημαντική τη τοποθέτηση μελισσοκομικής ζυγαριάς σε μια κυψέλη. Δεν υπάρχει αμφιβολία ότι είναι ικανή να απαλλάξει τους μελισσοκόμους από αρκετές ανησυχίες και άγχη που τους περιτριγυρίζουν, όπως επίσης να δώσει λύσεις σε πολυάριθμα προβλήματα που γεννιούνται στον κλάδο της μελισσοκομίας, με στόχο πάντα να προσφέρει την μέγιστη δυνατή εξυπηρέτηση και ιδανικά αποτελέσματα.

Αρχικά, μια από τις κύριες έγνοιές των μελισσοκόμων είναι η διατήρηση υγιών μελισσών στη κυψέλη τους με αποτέλεσμα να τους διαθέτουν την καλύτερη και την πιο πλούσια παραγωγή μελιού που μπορούν. Κάτι τέτοιο, μπορεί να φανεί σοβαρή πρόκληση για τους μελισσοκόμους αφού πολλοί από αυτούς δεν έχουν μόνο μια, αλλά δεκάδες, καμία φορά και εκατοντάδες κυψέλες, διασκορπισμένες σε διαφορετικά μέρη, μακριά η μια από την άλλη, τις οποίες πρέπει να επιβλέπουν και να φροντίζουν, ώστε να έχουν την μεγαλύτερη δυνατή παραγωγή. Δυστυχώς όμως, είναι περιορισμένος ο αριθμός κυψελών που μπορούν να επιθεωρήσουν σε μικρό χρονικό διάστημα, αφού θα πρέπει να σπαταλάνε πολύτιμο χρόνο, από μέρες έως εβδομάδες, και αρκετές φορές ακόμα περισσότερο από αυτό για να πηγαίνουν σε κάθε περιοχή όπου έχουν τοποθετηθεί μελίσσια, με αποτέλεσμα να μην ξέρουν την κατάσταση και τις ανάγκες που θα έχουν δημιουργηθεί στις κυψέλες τους. Στην περίπτωση αυτή, είναι απαραίτητο να επισημανθεί ότι οι απαιτήσεις που μπορούν να παρουσιαστούν από ένα μοναδικό μελίσσι ποικίλουν, διαφέρουν και αποτελούν αξιοσημείωτους συντελεστές για την ομαλή εξέλιξη και πρόοδο της εκάστοτε κυψέλης. Πιο συγκεκριμένα, ο χειμώνας είναι μια δύσκολη και απαιτητική εποχή για τις κυψέλες, αφού οι μέλισσες δεν μπορούν να βγουν έξω στο κρύο και εξαρτώνται από το μέλι τους για να παραμείνουν ζεστές και για να έχουν τροφή. Κατά συνέπεια, εάν το μέλι τους είναι ελλιπές και δεν αρκεί για να μείνουν υγιής, είναι δουλειά του μελισσοκόμου να τους παρέχει σιρόπι ώστε να συνεχιστεί η αύξηση της παραγωγής, πράγμα που φαντάζει δύσκολο εάν δεν μπορεί να ξέρει άμεσα σε τι κατάσταση βρίσκεται η κυψέλη. Έκτος από αυτό, για έναν μελισσοκόμο είναι αξιοσημείωτα ωφέλιμο να γνωρίζει ανά πάσα ώρα και στιγμή την απόδοση των σημείων όπου έχει τοποθετήσει τα μελίσσια του, ώστε να διαπράξει τις ανάλογες διαδικασίες. Για παράδειγμα, εάν μια κυψέλη δεν είναι παραγωγική και δεν έχει την απαιτούμενη απόδοση, είναι εξαιρετικά πιθανό να ωφελείται στο ότι στο σημείο όπου είναι τοποθετημένη, υπάρχουν ακραία καιρικά φαινόμενα όπως υπερβολικά έντονος αέρας και αρκετά δυνατές και συνεχείς βροχοπτώσεις. Ένας διαφορετικός λόγος που μπορεί να υπάρχει ελλιπείς παραγωγικότητα λόγω του σημείου που βρίσκεται, είναι το μη ικανοποιητικό ποσοστό θερμοκρασίας και υγρασίας, που διαφέρει από περιοχή σε περιοχή. Επομένως, σε τέτοιες περιπτώσεις ο μελισσοκόμος θα πρέπει να μετακινήσει τα μελίσσια του σε ένα διαφορετικό σημείο, πιο αποδοτικό από το προηγούμενο, ώστε να του προσφέρει την παραγωγή που επιθυμεί και χρειάζεται. Ειδάλλως, υπάρχει και το ενδεχόμενο το σημείο όπου είναι τοποθετημένη μια ζυγαριά να είναι εξεζητημένα

αποδοτικό και να παρέχει άψογα αποτελέσματα. Σε μια τέτοια περίπτωση, ο μελισσοκόμος ενδέχεται να θελήσει να μεταφέρει απευθείας και τα υπόλοιπα μελίσσια του σε εκείνη την περιοχή για να εκμεταλλευτεί το ιδανικό ποσοστό παραγωγικότητας που του προσφέρεται και να αυξήσει τις πιθανότητες μιας επιτυχημένης συγκομιδής.

Εύκολα, λοιπόν, οδηγούμαστε στο συμπέρασμα ότι η μελισσοκομία σαν επάγγελμα απαιτεί συνεχής και άμεση προσοχή, ενημέρωση και επίβλεψη της κάθε κυψέλης ξεχωριστά για να υπάρχουν άρτια αποτελέσματα, γεγονός που είναι δύσκολα επιτεύξιμο για έναν μελισσοκόμο. Επομένως, αληθεύει ότι ο ρόλος που μπορεί να διαδραματίσει μια μελισσοκομική ζυγαριά είναι εξαιρετικά σημαντικός, αφού διευκολύνει σε μεγάλο βαθμό όλες τις παραπάνω προϋποθέσεις. Αρχικά, μια μελισσοκομική ζυγαριά δίνει την δυνατότητα καταγραφής του βάρους της κυψέλης, δηλαδή τη ποσότητα του μελιού που παράγεται, ανά τακτά χρονικά διαστήματα, χωρίς να χρειάζεται ο μελισσοκόμος να πάει σε κάθε μελίσσι ξεχωριστά, αφού του προσφέρεται η μέτρηση σαν μήνυμα στο κινητό του ή σε online πλατφόρμα. Αυτό είναι ένα από τα πιο σημαντικά πράγματα που μπορεί να γνωρίζει ένας μελισσοκόμος για τα μελίσσια του, καθώς τον προστατεύει από διάφορους κίνδυνους που μπορούν να εμφανιστούν. Ένας από αυτούς είναι να ξέρει πότε πρέπει να φροντίσει τα μελίσσια του τον χειμώνα, όπως αναφέρθηκε και προηγουμένως. Επίσης, εάν παρατηρήσει ραγδαία μείωση βάρους στα μηνύματα, υπάρχει περίπτωση κάποιο ζώο να έχει ρίξει την κυψέλη ή κάποιες μέλισσες να έχουν εγκαταλείψει το μελίσσι για να πάνε σε άλλο μέρος. Ο μελισσοκόμος θα ενημερωθεί άμεσα για την μείωση βάρους και θα μπορέσει να ενεργήσει εγκαίρως, χωρίς να δημιουργηθούν μεγάλες απώλειες στο μελίσσι του. Ένα άλλο πλεονέκτημα είναι ότι όταν έρθει η ώρα τις συγκομιδής, θα γνωρίζει ακριβώς πόσο μέλι θα πρέπει να μαζέψει και πόσο να αφήσει μέσα στην κυψέλη, ώστε να παραμείνουν υγιής οι μέλισσες του. Επιπλέον, θα αποτελούσε σοβαρή παράλειψη να μην τονιστεί ότι εκτός από την καταγραφή βάρους, η ζυγαριά περιέχει αισθητήρες θερμοκρασίας και υγρασίας, το οποίο συμβάλει στην εύκολη παρακολούθηση κλίματος και καιρού από τον μελισσοκόμο. Επομένως, στις έντονες καιρικές συνθήκες που θέτουν σε κίνδυνο και την απαραίτητη παραγωγή μελιού αλλά και την υγειά των μελισσών, ο μελισσοκόμος πάλι θα πληροφορηθεί αρκετά γρήγορα ώστε να είναι σε θέση να πράξει αναλόγως για το καλό της κυψέλης, μεταφέροντας την σε μια άλλη περιοχή που τηρεί τις προϋποθέσεις για την καλύτερη ποιότητα ζωής των μελισσών. Παράλληλα, είναι εφικτή και η σύγκριση προηγουμένων μετρήσεων βάρους, θερμοκρασίας και υγρασίας για τυχόν αλλαγές και αυξομειώσεις που μπορούν να επηρεάσουν αξιοσημείωτα το μέλλον της απόδοσης των κυψελών. Τέλος, αξίζει να αναφερθεί ότι τα έσοδα των μελισσοκόμων θα αυξηθούν, όχι μόνο λόγω της μεγαλύτερης παραγωγής αλλά και τις εξοικονόμησης οικονομικών πόρων από έξοδα μεταφορών και των απωλειών παραγωγής που θα μπορούσαν να προκύψουν.

Εν κατακλείδι, είναι σαφές ότι για τους παραπάνω λόγους η χρήση ενός τέτοιου είδους συστήματος είναι απαραίτητη για τον επαγγελματία μελισσοκόμο, εφόσον λύνει την πλειοψηφία των προβλημάτων που υπάρχουν στο επάγγελμα του.

1.2 Διαδίκτυο [1]

Το Διαδίκτυο (αγγλικά: Internet) είναι παγκόσμιο σύστημα διασυνδεδεμένων δικτύων υπολογιστών, οι οποίοι χρησιμοποιούν καθιερωμένη ομάδα πρωτοκόλλων, η οποία συχνά αποκαλείται "TCP/IP" (αν και αυτή δεν χρησιμοποιείται από όλες τις υπηρεσίες του Διαδικτύου) για να εξυπηρετεί δισεκατομμύρια χρήστες καθημερινά σε ολόκληρο τον κόσμο. Οι διασυνδεδεμένοι ηλεκτρονικοί υπολογιστές ανά τον κόσμο, οι οποίοι βρίσκονται σε ένα κοινό δίκτυο επικοινωνίας, ανταλλάσσουν μηνύματα (πακέτα) με τη χρήση διαφόρων πρωτοκόλλων (τυποποιημένοι κανόνες επικοινωνίας), τα οποία υλοποιούνται σε επίπεδο υλικού και λογισμικού. Το κοινό αυτό δίκτυο καλείται Διαδίκτυο.

1.2.1 Η Ιστορία του Διαδικτύου

Οι πρώτες απόπειρες για την δημιουργία ενός διαδικτύου ξεκίνησαν στις ΗΠΑ κατά την διάρκεια του ψυχρού πολέμου. Η Σοβιετική Ένωση είχε ήδη στείλει στο διάστημα τον δορυφόρο Σπούτνικ 1 κάνοντας τους Αμερικανούς να φοβούνται όλο και περισσότερο για την ασφάλεια της χώρας τους. Θέλοντας λοιπόν να προστατευτούν από μια πιθανή πυρηνική επίθεση των Ρώσων δημιούργησαν την υπηρεσία προηγμένων αμυντικών ερευνών *ARPA* (Advanced Research Project Agency) γνωστή ως DARPA (*Defense Advanced Research Projects Agency*) στις μέρες μας. Αποστολή της συγκεκριμένης υπηρεσίας ήταν να βοηθήσει τις στρατιωτικές δυνάμεις των ΗΠΑ να αναπτυχθούν τεχνολογικά και να δημιουργηθεί ένα δίκτυο επικοινωνίας το οποίο θα μπορούσε να επιβιώσει σε μια ενδεχόμενη πυρηνική επίθεση.
Το αρχικό θεωρητικό υπόβαθρο δόθηκε από τον Τζ. Λικλάιντερ (*J.C.R. Licklider*) που ανέφερε σε συγγράμματά του το "γαλαξιακό δίκτυο". Η θεωρία αυτή υποστήριζε την ύπαρξη ενός δικτύου υπολογιστών που θα ήταν συνδεδεμένοι μεταξύ τους και θα μπορούσαν να ανταλλάσσουν γρήγορα πληροφορίες και προγράμματα. Το επόμενο θέμα που προ έκυπτε ήταν ότι το δίκτυο αυτό θα έπρεπε να ήταν αποκεντρωμένο έτσι ώστε ακόμη κι αν κάποιος κόμβος του δεχόταν επίθεση να υπήρχε δίοδος επικοινωνίας για τους υπόλοιπους υπολογιστές. Τη λύση σε αυτό έδωσε ο Πολ Μπάραν (*Paul Baran*) με τον σχεδιασμό ενός κατανεμημένου δικτύου επικοινωνίας που χρησιμοποιούσε την ψηφιακή τεχνολογία. Πολύ σημαντικό ρόλο έπαιξε και η θεωρία ανταλλαγής πακέτων του Λέοναρντ Κλάινροκ (*Leonard Kleinrock*), που υποστήριζε ότι πακέτα πληροφοριών που θα περιείχαν την προέλευση και τον προορισμό τους μπορούσαν να σταλούν από έναν υπολογιστή σε έναν άλλο.
Στηριζόμενο λοιπόν σε αυτές τις τρεις θεωρίες δημιουργήθηκε το πρώτο είδος διαδικτύου γνωστό ως ARPANET. Εγκαταστάθηκε και λειτούργησε για πρώτη φορά το 1969 με 4 κόμβους μέσω των οποίων συνδέονται 4 μίνι υπολογιστές (mini computers 12k): του πανεπιστημίου της Καλιφόρνια στην Σάντα Μπάρμπαρα του πανεπιστημίου της Καλιφόρνια στο Λος Άντζελες, το SRI στο Στάνφορντ και το πανεπιστήμιο της Γιούτα. Η ταχύτητα του δικτύου έφθανε τα 50 kbps και έτσι επιτεύχθηκε η πρώτη *dial up*

σύνδεση μέσω γραμμών τηλεφώνου. Μέχρι το 1972 οι συνδεδεμένοι στο ARPANET υπολογιστές έχουν φτάσει τους 23, οπότε και εφαρμόζεται για πρώτη φορά το σύστημα διαχείρισης ηλεκτρονικού ταχυδρομείου (*e-mail*).
Παράλληλα δημιουργήθηκαν και άλλα δίκτυα, τα οποία χρησιμοποιούσαν διαφορετικά πρωτόκολλα(όπως το x.25 και το UUCP) τα οποία συνδέονταν με το ARPANET. Το πρωτόκολλο που χρησιμοποιούσε το ARPANET ήταν το NCP (Network Control Protocol), το οποίο, όμως, είχε το μειονέκτημα ότι λειτουργούσε μόνο με συγκεκριμένους τύπους υπολογιστών. Έτσι, δημιουργήθηκε η ανάγκη στις αρχές του 1970 για ένα πρωτόκολλο που θα ένωνε όλα τα δίκτυα που είχαν δημιουργηθεί μέχρι τότε. Το 1974 λοιπόν, δημοσιεύεται η μελέτη των Βιντ Σερφ (*Vint Cerf*) και Μπομπ Κάαν (Bob Kahn) από την οποία προέκυψε το πρωτόκολλο TCP (Transmission Control Protocol) που αργότερα το 1978 έγινε TCP/IP, προσετέθη δηλαδή το Internet Protocol (IP), ώσπου το 1983 έγινε το μοναδικό πρωτόκολλο που ακολουθούσε το ARPANET.
Το 1984 υλοποιείται το πρώτο *DNS* (Domain Name System) σύστημα στο οποίο καταγράφονται 1000 κεντρικοί κόμβοι και οι υπολογιστές του διαδικτύου πλέον αναγνωρίζονται από διευθύνσεις κωδικοποιημένων αριθμών. Ένα ακόμη σημαντικό βήμα στην ανάπτυξη του Διαδικτύου έκανε το Εθνικό Ίδρυμα Επιστημών (National Science Foundation, NSF) των ΗΠΑ, το οποίο δημιούργησε την πρώτη διαδικτυακή πανεπιστημιακή ραχοκοκκαλιά (backbone), το NSFNet, το 1986. Ακολούθησε η ενσωμάτωση άλλων σημαντικών δικτύων, όπως το Usenet, το Fidonet και το Bitnet.
Ο όρος Διαδίκτυο/Ίντερνετ ξεκίνησε να χρησιμοποιείται ευρέως την εποχή που συνδέθηκε το ARPANET με το NSFNet και Internet σήμαινε οποιοδήποτε δίκτυο χρησιμοποιούσε TCP/IP. Η μεγάλη άνθιση του Διαδικτύου όμως, ξεκίνησε με την εφαρμογή της υπηρεσίας του Παγκόσμιου Ιστού από τον Τιμ Μπέρνερς-Λι στο ερευνητικό ίδρυμα CERN το 1989, ο οποίος είναι στην ουσία, η "πλατφόρμα", η οποία κάνει εύκολη την πρόσβαση στο Ίντερνετ, ακόμη και στη μορφή που είναι γνωστό σήμερα.

1.3 Internet Of Things (IoT) [2]

Το Διαδίκτυο των Πραγμάτων (Internet Of Things)ορίζεται σύμφωνα με τους Sundmaeker et al ως μία δυναμική παγκόσμια δικτυακή υποδομή με αυτορρυθμιζόμενες δυνατότητες που βασίζονται σε πρότυπα και διαλειτουργικά πρωτόκολλα επικοινωνίας, στα οποία τα φυσικά και εικονικά «πράγματα» έχουν ταυτότητα, φυσικά χαρακτηριστικά και εικονικές προσωπικότητες, ενώ χρησιμοποιούν έξυπνες διεπαφές και ενσωματώνονται απρόσκοπτα στο δίκτυο πληροφοριών.
Ο ορισμός «Διαδίκτυο των Πραγμάτων (Internet Of Things)» επινοήθηκε από τον Βρετανό επιχειρηματία Kevin Ashton το 1999. Ωστόσο η ιδέα συσκευών συνδεδεμένων μεταξύ τους ήταν αρκετά παλιότερη. Τότε, η ιδέα ονομαζόταν συχνά ως «ενσωματωμένο Διαδίκτυο (embedded internet)» ή «διάχυτος υπολογισμός (pervasive computing)». Η ευρεία χρήση του όρου Internet of Things παρατηρήθηκε περίπου το 2010, σύμφωνα με την εταιρία Cisco.

1.3.1 Εφαρμογές του Internet of Things

Το IoT προσφέρει νέες πηγές δεδομένων και νέα επιχειρηματικά μοντέλα που μπορούν να ενισχύσουν την παραγωγικότητα σε διάφορους κλάδους. Μερικοί από αυτούς είναι:

- Υγειονομική Περίθαλψη

Πολλοί άνθρωποι έχουν ήδη υιοθετήσει wearable συσκευές για να παρακολουθούν την φυσική τους κατάσταση, τον ύπνο ή άλλες συνήθειες τους. Αυτά θα μπορούσαν να είναι ένα πολύ μικρό δείγμα του πώς το IoT συνδυάζεται με τον κλάδο της υγείας. Συσκευές παρακολούθησης ασθενών, ηλεκτρονικά αρχεία και άλλα έξυπνα αξεσουάρ μπορούν να σώσουν ζωές.

- Βιομηχανική Παραγωγή

Πρόκειται για τον κλάδο που επωφελείται περισσότερο από το IoT. Αισθητήρες συλλογής δεδομένων μπορούν να επικοινωνήσουν μεταξύ τους ή να παρακολουθούν τη χρήση των πόρων ενός εργοστασίου σε πραγματικό χρόνο, καθιστώντας ευκολότερη την εργασία των ανθρώπων, πιο αποτελεσματικά και με μειωμένο κόστος.

- Λιανεμπόριο

Τόσο οι καταναλωτές όσο και τα καταστήματα μπορούν να επωφεληθούν από IoT. Τα καταστήματα, για παράδειγμα, θα μπορούσαν να χρησιμοποιήσουν IoT για σκοπούς παρακολούθησης των αποθεμάτων ή της ασφάλειας ενώ οι καταναλωτές μπορούν να έχουν μία περισσότερο εξατομικευμένη εμπειρία αγορών μέσω των δεδομένων που συλλέγονται από τους αισθητήρες ή τις κάμερες.

- Τηλεπικοινωνίες

Ο κλάδος των τηλεπικοινωνιών θα επηρεαστεί σημαντικά από το IoT, εφόσον θα είναι ο κλάδος που θα διατηρεί όλα τα δεδομένα που χρησιμοποιεί. Τα smartphones και άλλες προσωπικές συσκευές πρέπει να είναι σε θέση να διατηρούν μια αξιόπιστη σύνδεση στο διαδίκτυο για να λειτουργήσει αποτελεσματικά το Internet of Things.

- Μεταφορές

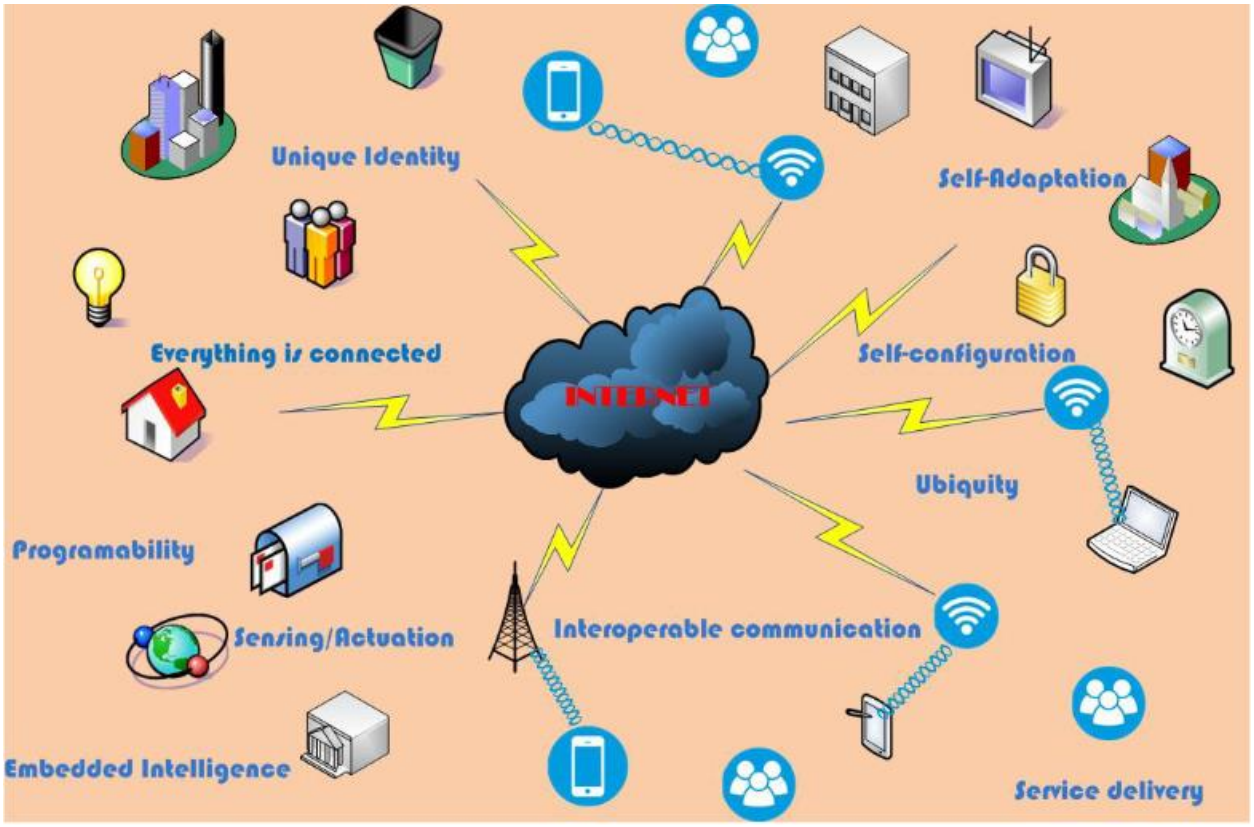
Το IoT επηρεάζει επίσης το κλάδο των μεταφορών σε μεγάλη κλίμακα: οι εταιρείες διανομής μπορούν να παρακολουθούν το στόλο τους με τη χρήση GPS λύσεων και οι δρόμοι μπορούν να παρακολουθούνται μέσω αισθητήρων για να είναι όσο το δυνατόν ασφαλέστεροι.

- Ενέργεια

Οι έξυπνοι μετρητές (smart meters), όχι μόνο συλλέγουν δεδομένα αυτόματα, αλλά καθιστούν και δυνατή την εφαρμογή analytics για την παρακολούθηση και τη διαχείριση της χρήσης της ενέργειας. Παρομοίως, αισθητήρες σε συσκευές όπως οι ανεμόμυλοι μπορούν να παρακολουθούν τα δεδομένα και να χρησιμοποιούν προγνωστική μοντελοποίηση ώστε να προγραμματιστεί η διακοπή λειτουργίας για πιο αποδοτική χρήση της ενέργειας.

1.3.2 Χαρακτηριστικά και απαιτήσεις IoT

Το Internet of Things δεν είναι μία μεμονωμένη τεχνολογία, αλλά μια μίξη διαφορετικών τεχνολογιών που παρέχει λύσεις βασιζόμενες στην πληροφορική και στις τηλεπικοινωνίες (information technology and communication technology). Η πληροφορική προσφέρει τα συστατικά στοιχεία του IoT, δηλαδή τις συσκευές και το σύνολο των λογισμικών προγραμμάτων (hardware and software) που χρησιμοποιούνται για την αποθήκευση, την λήψη και την επεξεργασία δεδομένων ενώ οι τηλεπικοινωνίες φροντίζουν για την αποτελεσματική επικοινωνία μεταξύ των στοιχειών αυτών (Patel et al., 2016). Στην εικόνα 1 βλέπουμε μια γενικευμένη σύνοψη των χαρακτηριστικών και των απαιτήσεων του IoT:



Εικόνα 1: Χαρακτηριστικά και απαιτήσεις του IoT.

1.4 Ενσωματωμένο σύστημα [3]

Ένα ενσωματωμένο σύστημα είναι σύστημα υπολογιστή με ειδική λειτουργία μέσα σε μεγαλύτερο μηχανικό ή ηλεκτρικό σύστημα, συχνά με περιορισμούς υπολογισμών σε πραγματικό χρόνο. Είναι ενσωματωμένο ως μέρος μιας πλήρους συσκευής που συχνά περιλαμβάνει υλικό και μηχανικά μέρη. Τα ενσωματωμένα συστήματα ελέγχουν πολλές συσκευές σε κοινή χρήση σήμερα. Το 98% όλων των μικροεπεξεργαστών κατασκευάζονται ως συστατικά ενσωματωμένων συστημάτων.

1.4.1 Παραδείγματα

Παραδείγματα ιδιοτήτων των τυπικών ενσωματωμένων υπολογιστών σε σύγκριση με τους ομολόγους τους γενικής χρήσης είναι η χαμηλή κατανάλωση ενέργειας, το μικρό

μέγεθος, τα εύρωστα λειτουργικά εύρη και το χαμηλό κόστος ανά μονάδα. Αυτό έρχεται στην τιμή των περιορισμένων πόρων επεξεργασίας, γεγονός που τους καθιστά πολύ πιο δύσκολο να προγραμματιστούν και να αλληλοεπιδρούν. Ωστόσο, με την οικοδόμηση μηχανισμών πληροφοριών πάνω από το υλικό, εκμεταλλευόμενοι τους πιθανούς υπάρχοντες αισθητήρες και την ύπαρξη ενός δικτύου ενσωματωμένων μονάδων, μπορεί κανείς να διαχειριστεί άριστα τους διαθέσιμους πόρους σε επίπεδο μονάδας και δικτύου, καθώς και να παράσχει αυξημένες λειτουργίες πολύ πιο πέρα από τις διαθέσιμες. Για παράδειγμα, μπορούν να σχεδιαστούν έξυπνες τεχνικές για τη διαχείριση της κατανάλωσης ενέργειας των ενσωματωμένων συστημάτων.

Τα σύγχρονα ενσωματωμένα συστήματα συχνά βασίζονται σε μικροελεγκτές (δηλ. κεντρικές μονάδες επεξεργασίας με ενσωματωμένη μνήμη ή περιφερειακές διεπαφές), αλλά οι συνηθισμένοι μικροεπεξεργαστές (χρησιμοποιώντας εξωτερικά τσιπ για μνήμη και περιφερειακά κυκλώματα διασύνδεσης) είναι επίσης συνηθισμένοι, ειδικά σε πιο σύνθετα συστήματα. Και στις δύο περιπτώσεις, οι χρησιμοποιούμενοι επεξεργαστές μπορεί να είναι τύποι που κυμαίνονται από γενικούς σκοπούς έως αυτούς που εξειδικεύονται σε συγκεκριμένες κατηγορίες υπολογισμών ή ακόμη και προσαρμοσμένοι για την εφαρμογή που βρίσκεται στο χέρι. Μια κοινή πρότυπη κατηγορία αποκλειστικών επεξεργαστών είναι ο επεξεργαστής ψηφιακών σημάτων.

Δεδομένου ότι το ενσωματωμένο σύστημα είναι αφιερωμένο σε συγκεκριμένες εργασίες, οι μηχανικοί σχεδιασμού μπορούν να το βελτιστοποιήσουν για να μειώσουν το μέγεθος και το κόστος του προϊόντος και να αυξήσουν την αξιοπιστία και την απόδοση. Ορισμένα ενσωματωμένα συστήματα είναι μαζικής παραγωγής, επωφελούμενα από οικονομίες κλίμακας.

1.4.2 Είδη ενσωματωμένων συστημάτων

Τα ενσωματωμένα συστήματα κυμαίνονται από φορητές συσκευές όπως ψηφιακά ρολόγια και συσκευές αναπαραγωγής MP3 έως μεγάλες σταθερές εγκαταστάσεις όπως φανάρια κυκλοφορίας, εργοστασιακές μονάδες ελέγχου και πολύπλοκα συστήματα, όπως τα υβριδικά οχήματα, η μαγνητική τομογραφία και η αεροηλεκτρονική. Η πολυπλοκότητα ποικίλλει από χαμηλή, με ένα ενιαίο μικροελεγκτή τσιπ, έως πολύ υψηλή με πολλαπλές μονάδες, περιφερειακά και δίκτυα τοποθετημένα μέσα σε ένα μεγάλο πλαίσιο ή περίβλημα.

1.5 Μικροελεγκτής [4]

Ο μικροελεγκτής (αγγλικά, microcontroller) είναι ένας τύπος επεξεργαστή, ουσιαστικά μια παραλλαγή μικροεπεξεργαστή, ο οποίος μπορεί να λειτουργήσει με ελάχιστα εξωτερικά εξαρτήματα, λόγω των πολλών ενσωματωμένων υποσυστημάτων που διαθέτει. Χρησιμοποιείται ευρύτατα σε όλα τα ενσωματωμένα συστήματα (embedded systems) ελέγχου χαμηλού και μεσαίου κόστους, όπως αυτά που χρησιμοποιούνται σε αυτοματισμούς, ηλεκτρονικά καταναλωτικά προϊόντα (από ψηφιακές φωτογραφικές μηχανές έως παιχνίδια), ηλεκτρικές συσκευές και κάθε είδους αυτοκινούμενα τροχοφόρα οχήματα.

1.5.1. Διαφορές από τον μικροεπεξεργαστή

Στους σύγχρονους μικροεπεξεργαστές για μη ενσωματωμένα συστήματα (π.χ. τους μικροεπεξεργαστές των προσωπικών υπολογιστών), δίνεται έμφαση στην υπολογιστική ισχύ. Η ευελιξία ανάπτυξης διαφορετικών εφαρμογών είναι μεγάλη, καθώς η λειτουργικότητα του τελικού συστήματος καθορίζεται από τα εξωτερικά περιφερειακά τα οποία διασυνδέονται με την κεντρική μονάδα (μικροεπεξεργαστή), η οποία δεν είναι εξειδικευμένη. Αντίθετα, στους μικροεπεξεργαστές για ενσωματωμένα συστήματα (μικροελεγκτές), οι οποίοι έχουν μικρότερες ή και μηδαμινές δυνατότητες συνεργασίας με εξωτερικά περιφερειακά, αυτού του είδους, η ευελιξία είναι περιορισμένη, καθώς και η υπολογιστική ισχύς. Οι μικροελεγκτές δίνουν έμφαση στο μικρό αριθμό ολοκληρωμένων κυκλωμάτων που απαιτείται για τη λειτουργία μιας συσκευής, το χαμηλό κόστος και την εξειδίκευση.
Αναλυτικά, τα πλεονεκτήματα των μικροελεγκτών είναι:

- Αυτονομία, μέσω της ενσωμάτωσης σύνθετων περιφερειακών υποσυστημάτων όπως μνήμες και θύρες επικοινωνίας. Έτσι πολλοί μικροελεγκτές δεν χρειάζονται κανένα άλλο ολοκληρωμένο κύκλωμα για να λειτουργήσουν.
- Η ενσωμάτωση περιφερειακών σημαίνει ευκολότερη υλοποίηση εφαρμογών λόγω των απλούστερων διασυνδέσεων. Επίσης, οδηγεί σε χαμηλότερη κατανάλωση ισχύος, μεγιστοποιώντας τη φορητότητα και ελαχιστοποιεί το κόστος της συσκευής στην οποία ενσωματώνεται ο μικροελεγκτής.
- Χαμηλό κόστος.
- Μεγαλύτερη αξιοπιστία, και πάλι λόγω των λιγότερων διασυνδέσεων.
- Μειωμένες εκπομπές ηλεκτρομαγνητικών παρεμβολών και μειωμένη ευαισθησία σε αντίστοιχες παρεμβολές από άλλες ηλεκτρικές και ηλεκτρονικές συσκευές. Το πλεονέκτημα αυτό προκύπτει από το μικρότερο αριθμό και μήκος εξωτερικών διασυνδέσεων καθώς και τις χαμηλότερες ταχύτητες λειτουργίας.
- Περισσότεροι διαθέσιμοι ακροδέκτες για ψηφιακές εισόδους-εξόδους (για δεδομένο μέγεθος ολοκληρωμένου κυκλώματος), λόγω της μη δέσμευσής τους για τη σύνδεση εξωτερικών περιφερειακών.
- Μικρό μέγεθος συνολικού υπολογιστικού συστήματος.

Η βασική αρχιτεκτονική των μικροελεγκτών δεν διαφέρει από αυτή των κοινών μικροεπεξεργαστών, αν και στους πρώτους απαντάται συχνά η αρχιτεκτονική μνήμης τύπου Harvard, η οποία χρησιμοποιεί διαφορετικές αρτηρίες σύνδεσης της μνήμης προγράμματος και της μνήμης δεδομένων (π.χ. οι σειρές AVR από την Atmel και PIC από την Microchip). Στους κοινούς μικροεπεξεργαστές συνηθίζεται η ενιαία διάταξη μνήμης τύπου φον Νόιμαν.

Συνήθη υποσυστήματα

Στον μικροεπεξεργαστή, το ολοκληρωμένο κύκλωμα που τον αποτελεί περιέχει μόνο την Λογική και Αριθμητική Μονάδα (ALU), στοιχειώδεις καταχωρητές (registers), προσωρινή μνήμη RAM πολύ υψηλής ταχύτητας (cache memory) και, κάποιες φορές, τον ελεγκτή μνήμης (memory controller). Όμως, για τη λειτουργία ενός πλήρους ενσωματωμένου υπολογιστικού συστήματος, απαιτούνται πολλά εξωτερικά υποσυστήματα και περιφερειακά. Τέτοια είναι:

- Κύκλωμα συνδετικής λογικής (glue logic) για τη σύνδεση των εξωτερικών μνημών και άλλων περιφερειακών παράλληλης σύνδεσης στην αρτηρία δεδομένων (data bus) του επεξεργαστή.
- Μνήμη προγράμματος (τύπου ROM, FLASH, EPROM κλπ) η οποία περιέχει το λογισμικό του συστήματος. Σε κάποια μοντέλα, είναι δυνατό το κλείδωμα αυτής της μνήμης, μετά την εγγραφή της, ώστε να προστατευτεί το περιεχόμενό της από αντιγραφή.
- Μεγάλο μέγεθος μνήμης RAM.
- Μόνιμη μνήμη αποθήκευσης παραμέτρων λειτουργίας (τύπου EEPROM ή NVRAM) η οποία να μπορεί να γράφεται στον πυρήνα του μικροελεγκτή. Αυτή η μνήμη έχει, έναντι της FLASH, το πλεονέκτημα της δυνατότητας διαγραφής και εγγραφής οποιουδήποτε μεμονωμένου byte.
- Κύκλωμα αρχικοποίησης (reset).
- Διαχειριστή αιτήσεων διακοπής (interrupt request controller) από τα περιφερειακά.
- Κύκλωμα επιτήρησης τροφοδοσίας (brown-out detection) το οποία παρακολουθεί την τροφοδοσία και αρχικοποιεί ολόκληρο το σύστημα όταν αυτή πέσει κάτω από τα ανεκτά όρια, προλαμβάνοντας έτσι την αλλοίωση των δεδομένων.
- Κύκλωμα επιτήρησης λειτουργίας (watchdog timer) το οποίο αρχικοποιεί το σύστημα, αν αυτό εμφανίσει σημάδια δυσλειτουργίας λόγω κολλήματος (hang).
- Τοπικό ταλαντωτή για την παροχή παλμών χρονισμού (clock).
- Έναν ή περισσότερους χρονιστές-απαριθμητές υψηλής ταχύτητας (hardware timer-counter) για τη δημιουργία καθυστερήσεων, μέτρηση διάρκειας γεγονότων, απαρίθμηση γεγονότων και άλλων λειτουργιών ακριβούς χρονισμού.
- Ρολόι πραγματικού χρόνου (Real Time Clock, RTC) το οποίο τροφοδοτείται από ανεξάρτητη μπαταρία και γι αυτό πρέπει να έχει πολύ χαμηλή κατανάλωση ρεύματος.
- Σειρά ανεξάρτητων ψηφιακών εισόδων και εξόδων (Parallel Input-Output, PIO).

Γενικά, όλες οι οικογένειες μικροελεγκτών ενσωματώνουν τα περισσότερα από τα παραπάνω περιφερειακά, με διαφοροποιήσεις κυρίως στην ύπαρξη ή μη εσωτερικής μνήμης προγράμματος και στο είδος της. Έτσι, υπάρχουν:

- Μικροελεγκτές χωρίς μνήμη προγράμματος, οι οποίοι χαρακτηρίζονται ως ROM-less. Αυτοί παρέχουν πάντοτε μια παράλληλη αρτηρία (bus) δεδομένων, πάνω στην οποία συνδέονται εξωτερικές μνήμες προγράμματος και RAM. Τέτοιοι τύποι μικροελεγκτών προορίζονται για πιο ισχυρά υπολογιστικά συστήματα ελέγχου, με μεγαλύτερες απαιτήσεις μνήμης.
- Μικροελεγκτές με μνήμη ROM, η οποία κατασκευάζεται με το λογισμικό της (Mask ROM) ή γράφεται μόνο μια φορά (One Time Programmable, OTP). Παρέχουν τη δυνατότητα πολύ χαμηλού κόστους, όταν αγοράζονται σε πολύ μεγάλες ποσότητες.
- Μικροελεγκτές με μνήμη FLASH, οι οποία μπορούν συνήθως να προγραμματιστεί πολλές φορές. Αυτή είναι η πιο διαδεδομένη κατηγορία. Συχνά ο προγραμματισμός της μνήμης μπορεί να γίνει ακόμη και πάνω στο κύκλωμα της ίδιας της ενσωματωμένης (embedded) εφαρμογής (δυνατότητα In Circuit Programming). Αυτοί οι μικροελεγκτές έχουν ουσιαστικά αντικαταστήσει τους παλαιότερους τύπους EPROM που έσβηναν με υπεριώδη ακτινοβολία (από το ειδικό τζαμάκι).

Πρόσθετες λειτουργίες

Ανάλογα με την εφαρμογή για την οποία προορίζεται ένας μικροελεγκτής, μπορεί να περιέχει και:

- Μία ή περισσότερες ασύγχρονες σειριακές θύρες επικοινωνίας (Universal Asynchronous Receiver Transmitter, UART).
- Σύγχρονες σειριακές θύρες επικοινωνίας (π.χ. I2C, SPI, Ethernet).
- Ολόκληρα υποσυστήματα για την άμεση υποστήριξη από υλικολογισμικό (firmware) των πιο σύνθετων πρωτοκόλλων επικοινωνίας όπως CAN, HDLC, ISDN, ADSL.
- Μονάδα άμεσης εκτέλεσης πράξεων κινητής υποδιαστολής (Floating Point Processing Unit, FPU), η οποία είναι πάντοτε πιο γρήγορη από την ALU του επεξεργαστή. Τέτοιες μονάδες χαρακτηρίζουν τους μικροελεγκτές με δυνατότητες ψηφιακής επεξεργασίας σήματος (Digital Signal Processing, DSP). Τα τελευταία χρόνια, με την ευρύτατη διάδοση των φορητών συσκευών ήχου και εικόνας, παρατηρείται μια τάση σύγκλισης των μικροελεγκτών με τους DSP.[4]
- Περισσότερες από μία εισόδους για μετατροπή αναλογικού σήματος σε ψηφιακό (Analog to Digital converter, ADC).
- Μετατροπέα ψηφιακού σε αναλογικό σήμα (Digital to Analog converter, DAC).
- Ελεγκτή οθόνης υγρών κρυστάλλων (Liquid Crystal Display, LCD).
- Υποσύστημα προγραμματισμού πάνω στο κύκλωμα (τύπου ISP, βλ. παραπάνω). Χάρη σε αυτό το κύκλωμα, είναι δυνατός ο επαναπρογραμματισμός (αναβάθμιση λογισμικού) της εφαρμογής, συνδέοντας

στη συσκευή μια εξωτερική συσκευή προγραμματισμού (συνήθως σε θύρα UART RS-232) ή ακόμη και από το διαδίκτυο. Αυτή η δυνατότητα απαιτεί την προϋπαρξη λογισμικού υποδοχής (bootstrap) μέσα στη μνήμη προγράμματος και επομένως δεν μπορεί να γίνει σε τελείως άδεια μνήμη προγράμματος.

- Υποσύστημα προγραμματισμού (τύπου ISP) και διάγνωσης (συνήθως είναι το καθιερωμένο πρότυπο JTAG). Χάρη σε αυτό, είναι δυνατός ο προγραμματισμός της μνήμης προγράμματος χωρίς να προαπαιτείται κάποιο πρόγραμμα υποδοχής. Γι αυτό το λόγο, είναι ιδιαίτερα χρήσιμο στον αρχικό προγραμματισμό, π.χ. κατά τη συναρμολόγηση, ή σε περίπτωση σφάλματος (bug) στο λογισμικό υποδοχής το οποίο να καθιστά αδύνατη την κανονική αναβάθμιση.

1.5.2 Διαδεδομένες κατηγορίες μικροελεγκτών

Λόγω του ισχυρότατου ανταγωνισμού αλλά και της τάσης ενσωμάτωσης των μικροελεγκτών σε κάθε ηλεκτρική και ηλεκτρονική συσκευή, η βιομηχανία μικροελεγκτών έχει καταλήξει στην παραγωγή ανταγωνιστικών μοντέλων μαζικής παραγωγής καθώς και μικροελεγκτών για πιο εξειδικευμένες εφαρμογές. Έτσι διακρίνονται οι εξής κυρίως κατηγορίες:

- Μικροελεγκτές (καμμιά φορά 4-bit αλλά συνήθως 8-bit) πολύ χαμηλού κόστους, γενικής χρήσης, με πολύ μικρό αριθμό ακροδεκτών (ακόμη και λιγότερους από 8). Σχεδιάζονται με έμφαση στη χαμηλή κατανάλωση ισχύος και την αυτάρκεια, ώστε να χρειάζονται ελάχιστα ή και καθόλου εξωτερικά εξαρτήματα και να μη μπορεί να αντιγραφεί εύκολα το εσωτερικό λογισμικό τους. Απουσιάζει η δυνατότητα επέκτασης της μνήμης τους. Μερικά μοντέλα είναι ευρέως γνωστά στους ερασιτέχνες ηλεκτρονικούς, όπως π.χ. οι περισσότεροι μικροελεγκτές των σειρών PIC (Microchip), AVR (Atmel) και 8051 (Intel, Atmel, Dallas κα).
- Μικροελεγκτές (συνήθως 8-bit αλλά και 16 ή 32-bit) χαμηλού κόστους, γενικής χρήσης, με μέτριο έως σχετικά μεγάλο αριθμό ακροδεκτών. Διαθέτουν μεγάλο αριθμό κοινών περιφερειακών, όπως θύρες UART, I2C, SPI ή CAN, μετατροπείς αναλογικού σε ψηφιακό και ψηφιακού σε αναλογικό. Στους κατασκευαστές της Άπω Ανατολής (Ιαπωνία, Κορέα), συνηθίζεται η ενσωμάτωση ελεγκτών οθόνης υγρών κρυστάλλων και πληκτρολογίου. Μερικές φορές παρέχουν δυνατότητα εξωτερικής επέκτασης της μνήμης τους.
- Μικροελεγκτές (κυρίως 32-bit) μέσου κόστους, γενικής χρήσης, με μεγάλο αριθμό ακροδεκτών. Χαρακτηρίζονται από έμφαση στην ταχύτητα εκτέλεσης εντολών, υψηλή αυτάρκεια περιφερειακών και μεγάλες δυνατότητες εσωτερικής ή εξωτερικής μνήμης προγράμματος (FLASH) και RAM. Στο χώρο αυτό έχουν ισχυρή παρουσία οι αρχιτεκτονικές με υψηλή μεταφερσιμότητα λογισμικού (portability) από τον ένα στον άλλο κατασκευαστή. Π.χ. μεταξύ των μικροελεγκτών τύπου ARM ή MIPS, το σύνολο των βασικών εντολών που αναγνωρίζει η ALU είναι ακριβώς το ίδιο, μειώνοντας έτσι τις μεγάλες αλλαγές στο λογισμικό, όταν στο μέλλον ο πελάτης υιοθετήσει ένα μικροελεγκτή άλλου

κατασκευαστή (αρκεί, φυσικά, να υποστηρίζει κι αυτός το σύνολο εντολών ARM ή MIPS, αντίστοιχα).

- Μικροελεγκτές εξειδικευμένων εφαρμογών, οι οποίοι ενσωματώνουν συνήθως κάποιο εξειδικευμένο πρωτόκολλο επικοινωνίας το οποίο υλοποιείται πάντοτε σε hardware. Τέτοιοι μικροελεγκτές χρησιμοποιούνται σε τηλεπικοινωνιακές συσκευές όπως τα μόντεμ.

Η μεγάλη μερίδα πωλήσεων των μικροελεγκτών εξακολουθεί να αφορά αυτούς των 8-bit, καθώς είναι η κατηγορία με το χαμηλότερο κόστος και το μικρότερο μέγεθος λογισμικού για το ίδιο αποτέλεσμα, ιδίως επειδή οι σύγχρονες οικογένειες μικροελεγκτών 8-bit έχουν πολύ βελτιωμένες επιδόσεις σε σχέση με το παρελθόν.

1.5.3 Γλώσσες προγραμματισμού και εργαλεία ανάπτυξης

Η επιτυχία μιας οικογένειας μικροελεγκτών καθορίζεται σε μεγάλο βαθμό από τη διαθεσιμότητα και την ευχρηστία των σχετικών εργαλείων ανάπτυξης, όπως μεταφραστές από γλώσσες υψηλού επιπέδου σε γλώσσα κατανοητή από τον μικροελεγκτή (assembly), προγραμματιστές της εσωτερικής μνήμης και εργαλεία εκσφαλμάτωσης (debuggers). Στους μικροελεγκτές, τα εργαλεία αυτά δεν αποτελούνται μόνον από λογισμικό, καθώς δεν υπάρχει τυποποιημένος τρόπος επικοινωνίας με αυτούς. Έτσι διατίθενται προγραμματιστές της εσωτερικής μνήμης (συνήθως μέσω θύρας JTAG ή USB) καθώς και έτοιμες πλακέτες (evaluation boards) με ψηφιακές ή και αναλογικές εξόδους. Οι πλακέτες αυτές έχουν ως κεντρική τους μονάδα τον εκάστοτε μικροελεγκτή, δυνατότητα εύκολου προγραμματισμού και συνήθως συνοδεύονται από λογισμικό ανάπτυξης εφαρμογών με έτοιμα παραδείγματα. Ετσι ο χρήστης μπορεί να δοκιμάσει τις περισσότερες δυνατότητες του μικροελεγκτή πριν καν σχεδιάσει τη δική του πλακέτα. Στον τομέα των εργαλείων ανάπτυξης, δραστηριοποιούνται όχι μόνο οι ίδιοι οι κατασκευαστές μικροελεγκτών αλλά και εξειδικευμένες εταιρείες.

Η πιο διαδεδομένη γλώσσα προγραμματισμού των μικροελεγκτών είναι η C, η C++ και οι παραλλαγές τους. Σε τμήματα του λογισμικού όπου απαιτείται ταχύτητα η μικρό μέγεθος χρησιμοποιούμενης μνήμης, μπορεί να χρησιμοποιείται η Assembly. Όμως οι μεγαλύτερες απαιτήσεις σε λειτουργικότητα και η ευκολία προγραμματισμού της C έναντι της assembly, σε συνδυασμό με την επάρκεια μνήμης των σύγχρονων μικροελεγκτών, έχουν γενικά εκτοπίσει την Assembly από τις περισσότερες εφαρμογές.

1.5.4 Κατασκευαστές

Μερικοί από τους γνωστότερους κατασκευαστές μικροελεγκτών είναι οι

- ARM (δεν κατασκευάζει αλλά παραχωρεί δικαιώματα χρήσης του πυρήνα)
- Atmel
- Epson
- Freescale Semiconductor (πρώην Motorola)
- Hitachi
- Maxim (μετά την εξαγορά της Dallas)
- Microchip
- NEC
- Toshiba
- Texas Instruments
- National Instruments
- Espressif Systems

ΚΕΦΑΛΑΙΟ 2
Ανάπτυξη Ενσωματωμένου Συστήματος

2.1 Μεθοδολογία υλοποίησης ενός ενσωματωμένου συστήματος [5]

Οι μεθοδολογία υλοποίησης ενός ενσωματωμένου συστήματος όπως περιγράφετε και στην *Εικόνα 2* αποτελείτε από 6 επίπεδα στα οποία πρέπει να λαμβάνουμε υπόψη μας και το υλικό και το λογισμικό τα επίπεδα περιγράφονται παρακάτω.

Σχεδίαση (Planning)

Υλικό
- Προδιαγραφές απαιτήσεων υλικού

Λογισμικό
- Προδιαγραφές απαιτήσεων λογισμικού

Ανάλυση (Analysis)

Υλικό
- Μελέτη σκοπιμότητας και αγοράς
- Βελτιστοποίηση λογαριασμού υλικών (BOM)

Λογισμικό
- Μελέτη σκοπιμότητας
- Αναγνώριση πόρων
- Έγγραφο προγραμματισμένων δοκιμών

Σχεδιασμός (Design)

Υλικό
- Σχεδιασμός υλικού
- Σχηματικό διάγραμμα
- Σχεδιασμός PCB

Λογισμικό
- Αναλυτικό έγγραφο για την ανάπτυξη του λογισμικού
- Έγγραφο στρατηγικής για τις δοκιμές του λογισμικού

Υλοποίηση (Implementation)

Υλικό
- Προμήθεια εξαρτημάτων
- Κατασκευή PCB
- Συναρμολόγηση πλακέτας

Λογισμικό

- Βιβλιοθήκες κώδικα λογισμικού
- Κώδικας της εφαρμογής και κριτική
- Δοκιμή μονάδας

Δοκιμές και Ολοκλήρωση (Testing and Integration)

Υλικό

- Παρουσίαση κυκλώματος
- Επαλήθευση και επικύρωση σχεδιασμού
- Πιστοποιήσεις (εάν χρειάζονται)

Λογισμικό

- Δοκιμή ομαλής λειτουργίας μεταξύ (υλικού - λογισμικού)
- Δοκιμή συστήματος
- Δοκιμή αποδοχής

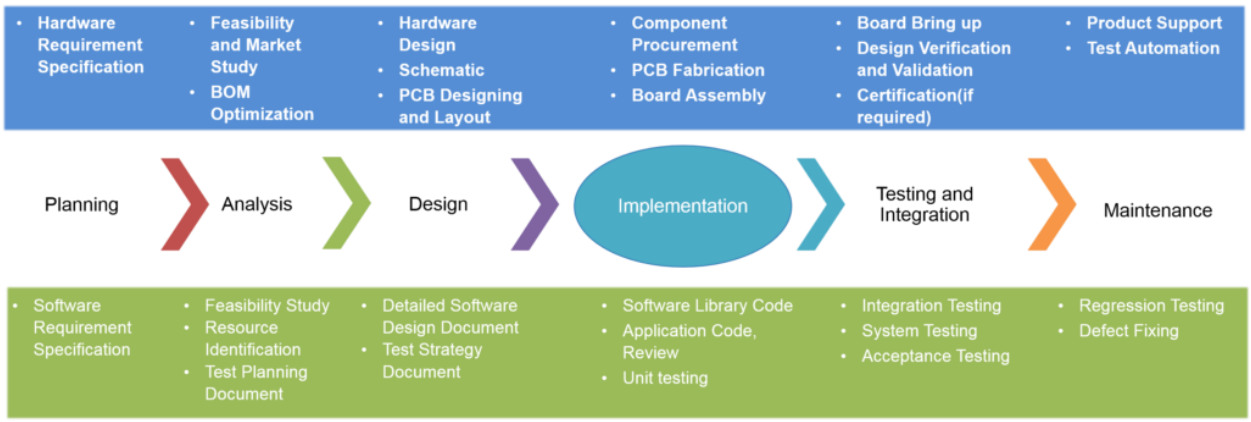
Συντήρηση (Maintenance)

Υλικό

- Υποστήριξη προϊόντος
- Δοκιμές αυτοματισμού

Λογισμικό

- Αναδρομή παλαιών δοκιμών
- Επιδιόρθωση ελαττωμάτων



Εικόνα 2: Μεθοδολογία υλοποίησης ενός ενσωματωμένου συστήματος

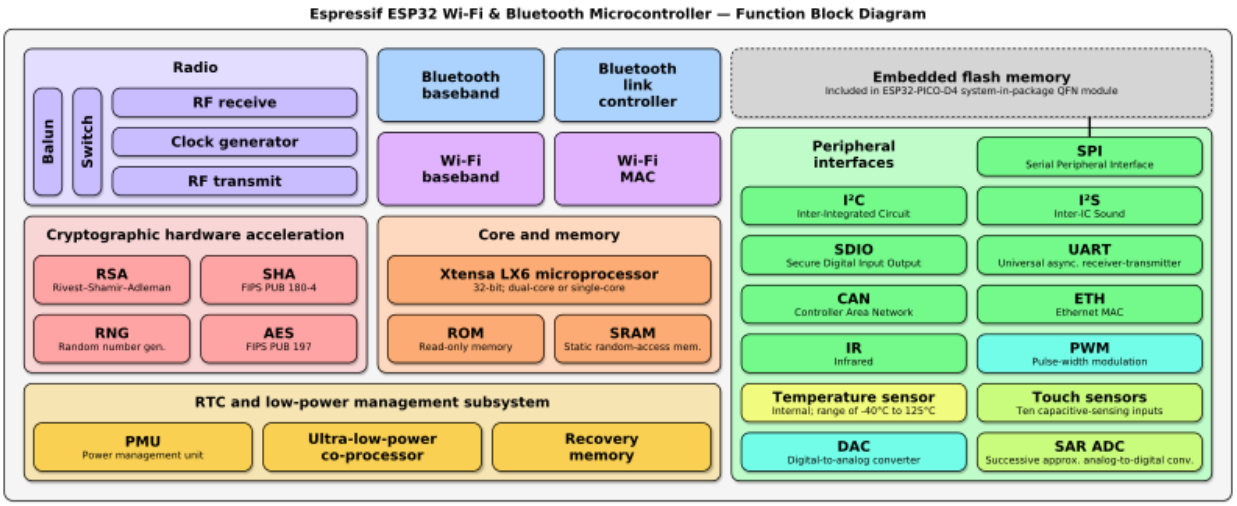
2.2 Προδιαγραφές απαιτήσεων υλικού

Το ενσωματωμένο σύστημα που υλοποιείτε (μελισσοκομική ζυγαριά) πρέπει να έχει ακρίβεια στο ζύγισμα του βάρους ± *20 γραμμάρια* και πρέπει να έχει δυνατότητα μέτρησης της υγρασίας και της θερμοκρασίας του περιβάλλοντος. Το σύστημα θα επικοινωνεί μέσω SMS ή θα ανεβάζει τα δεδομένα που καταγράφει στο ThingSpeak μέσο σύνδεσης GSM/GPRS. Πρέπει να λειτουργεί με φωτοβολταϊκό panel για αυτονομία μπαταρίας. Το σύστημα χρειάζεται να έχει κάποιο εύκολο τρόπο ρύθμισης από το χρήστη, η ρυθμίσεις αυτές θα πραγματοποιούνται με τη βοήθεια μιας WEB εφαρμογής που θα τρέχει πάνω στη ζυγαριά.

2.2.1 Ο Μικροελεγκτής ESP32 [6]

Το ESP32 είναι μια σειρά συστήματος χαμηλού κόστους και χαμηλής κατανάλωσης σε μικροελεγκτές chip με ενσωματωμένο Wi-Fi και Bluetooth διπλής λειτουργίας. Η σειρά ESP32 χρησιμοποιεί τον μικροεπεξεργαστή Tensilica Xtensa LX6 τόσο σε παραλλαγές δύο πυρήνων όσο και σε ένα πυρήνα και περιλαμβάνει ενσωματωμένους διακόπτες κεραίας, balun RF, ενισχυτή ισχύος, ενισχυτή λήψης χαμηλού θορύβου, φίλτρα και μονάδες διαχείρισης ισχύος. Το ESP32 δημιουργήθηκε και αναπτύχθηκε από την Espressif Systems, μια κινεζική εταιρεία με έδρα τη Σαγκάη. Είναι διάδοχος του μικροελεγκτή ESP8266.

ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ



Εικόνα 3: Block διάγραμμα για τη λειτουργία του ESP32

- Επεξεργαστές:
    - CPU: Xtensa dual-core (or single-core) 32-bit LX6 microprocessor, operating at 160 or 240 MHz and performing at up to 600 DMIPS Ultra
    - low power (ULP) co-processor

- Μνήμη:520 KiB SRAM, 448 KiB ROM

- Ασύρματη συνδεσιμότητα:

    - Wi-Fi: 802.11 b/g/n
    - Bluetooth: v4.2 BR/EDR and BLE (shares the radio with Wi-Fi)

- Περιφερειακές διεπαφές:

    - 34 × programmable GPIOs
    - 12-bit SAR ADC up to 18 channels
    - 2 × 8-bit DACs
    - 10 × touch sensors (capacitive sensing GPIOs)
    - 4 × SPI
    - 2 × I²S interfaces
    - 2 × I²C interfaces
    - 3 × UART
    - SD/SDIO/CE-ATA/MMC/eMMC host controller
    - SDIO/SPI slave controller
    - Ethernet MAC interface with dedicated DMA and IEEE1588 Precision Time Protocol support
    - CAN bus 2.0
    - Infrared remote controller (TX/RX, up to 8 channels)
    - Motor PWM
    - LED PWM (up to 16 channels)
    - Hall effect sensor
    - Ultra low power analog pre-amplifier

- Ασφάλεια:

    - IEEE 802.11 standard security features all supported, including WPA, WPA/WPA2 and WAPI
    - Secure boot

- Flash encryption
- 1024-bit OTP, up to 768-bit for customers
- Cryptographic hardware acceleration: AES, SHA-2, RSA, elliptic curve cryptography (ECC), random number generator (RNG)

- Διαχείριση ισχύος:

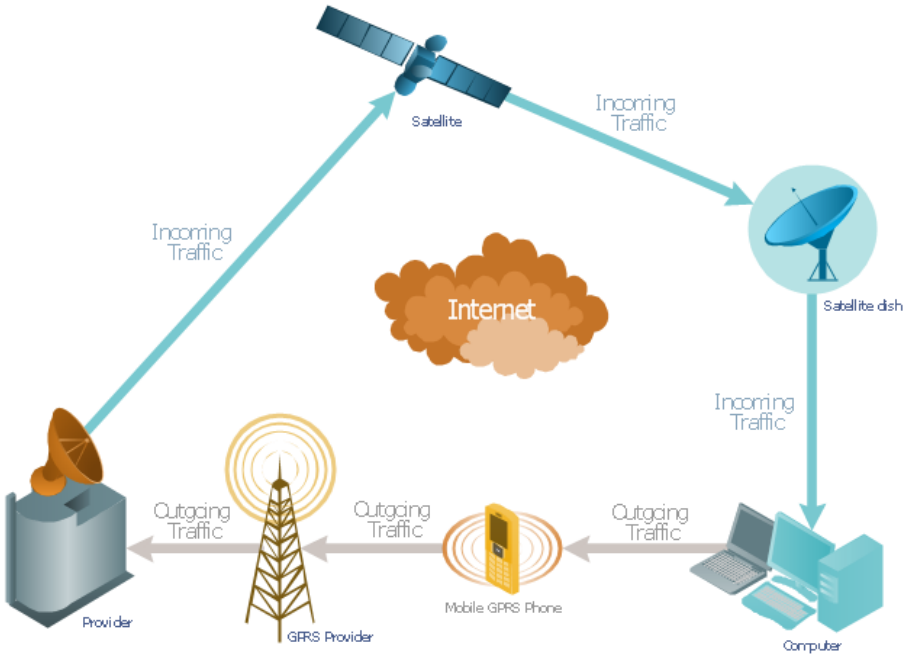
    - Internal low-dropout regulator
    - Individual power domain for RTC
    - 5 μA deep sleep current

Wake up from GPIO interrupt, timer, ADC measurements, capacitive touch sensor interrupt

2.2.2GSM/GPRS [7]

Το Global System for Mobile Communications (GSM) είναι ένα πρότυπο που αναπτύχθηκε από το European Telecommunications Standards Institute (ETSI) για να περιγράψει τα πρωτόκολλα για ψηφιακά κινητά δίκτυα δεύτερης γενιάς (2G) που χρησιμοποιούνται από κινητές συσκευές, όπως κινητά τηλέφωνα και tablet. Χρησιμοποιήθηκε για πρώτη φορά στη Φινλανδία τον Δεκέμβριο του 1991. Μέχρι τα μέσα της δεκαετίας του 2010, έγινε παγκόσμιο πρότυπο για τις κινητές επικοινωνίες που επιτυγχάνουν μερίδιο αγοράς άνω του 90% και λειτουργεί σε περισσότερες από 193 χώρες και εδάφη. Τα δίκτυα 2G αναπτύχθηκαν ως αντικατάσταση αναλογικών κινητών δικτύων πρώτης γενιάς (1G). Το πρότυπο GSM αρχικά περιέγραψε ένα ψηφιακό circuit-switched δίκτυο που βελτιστοποιήθηκε για τη πλήρη αμφίδρομη φωνητική τηλεφωνία. Αυτό επεκτάθηκε με την πάροδο του χρόνου για να συμπεριλάβει data communications, πρώτα με circuit-switched μεταφορά, στη συνέχεια με μεταφορά δεδομένων με τη χρήση πακέτων μέσω της υπηρεσίας General Packet Radio Service (GPRS) και Enhanced Data Rates for GSM Evolution (EDGE).

Έπειτα , το 3GPP ανέπτυξε πρότυπα UMTS τρίτης γενιάς (3G), ακολουθούμενα από το LTE Advanced τέταρτης γενιάς (4G) και τα πρότυπα πέμπτης γενιάς 5G, τα οποία δεν αποτελούν μέρος του προτύπου ETSI GSM.

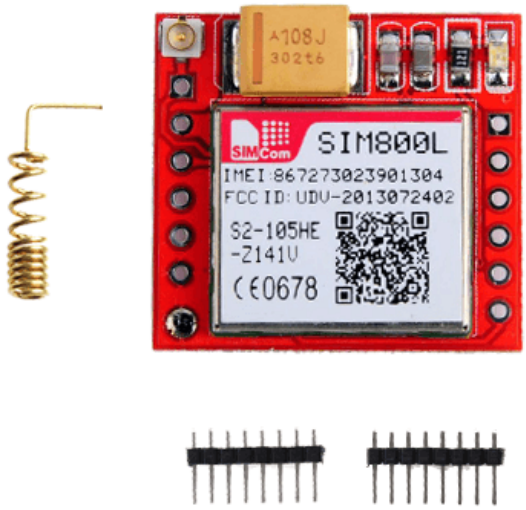


Εικόνα 4: Λειτουργία GPRS

2.2.3 SIM800L [8]

Το SIM800L είναι μια μονάδα GSM / GPRS τετραπλής ζώνης, που λειτουργεί σε συχνότητες GSM850MHz, EGSM900MHz, DCS1800MHz και PCS1900MHz. Οι SIM800L διαθέτουν GPRS multi-slot class 12 / class 10 (προαιρετικά) και υποστηρίζουν τα σχήματα κωδικοποίησης GPRS CS-1, CS-2, CS-3 και CS-4.

Με μια μικρή διαμόρφωση 17,8 * 15,8 * 2,4 mm, το SIM800L μπορεί να ικανοποιήσει σχεδόν όλες τις απαιτήσεις χώρου σε εφαρμογές πελατών, όπως smartphone, PDA και άλλες φορητές συσκευές. Το SIM800L είναι ένα πακέτο LGA με 88 τακάκια και παρέχει όλες τις διεπαφές υλικού μεταξύ της μονάδας και των πλακέτων των πελατών.



Εικόνα 5: SIM800L

- Υποστηρίζει 5 * 5 * 2 πληκτρολόγια
- Μία πλήρης σειριακή θύρα μόντεμ, ο πελάτης μπορεί να διαμορφώσει δύο σειριακές θύρες
- Ήχο που περιλαμβάνει δύο εισόδους μικροφώνου. έξοδος δέκτη και έξοδος ηχείου
- Προγραμματιζόμενη είσοδος και έξοδος γενικού σκοπού.
- Μία διεπαφή κάρτας SIM
- Υποστηρίζει FM
- Υποστηρίζει μία PWM SIM800L έχει σχεδιαστεί με τεχνική εξοικονόμησης ενέργειας έτσι ώστε η τρέχουσα κατανάλωση να είναι τόσο χαμηλή όσο 1,04mA σε κατάσταση αναστολής λειτουργίας

2.2.4 Ηλιακό πάνελ [9]

Ένα ηλιακό πάνελ, ή μια μονάδα φωτοβολταϊκών (PV), είναι ένα συγκρότημα φωτοβολταϊκών κυττάρων τοποθετημένων σε ένα πλαίσιο. Τα ηλιακά πάνελ χρησιμοποιούν το ηλιακό φως ως πηγή ενέργειας για την παραγωγή ηλεκτρικής ενέργειας συνεχούς ρεύματος (DC). Μια συλλογή φωτοβολταϊκών μονάδων ονομάζεται φωτοβολταϊκός πίνακας και ένα σύστημα πινάκων είναι μια συστοιχία. Οι συστοιχίες ενός φωτοβολταϊκού συστήματος παρέχουν ηλιακό ηλεκτρισμό σε ηλεκτρικό εξοπλισμό.

Πως λειτουργούν τα ηλιακά πάνελ

Με απλά λόγια, ένα ηλιακό πάνελ λειτουργεί επιτρέποντας στα φωτόνια, ή σωματίδια φωτός, να απαλλάξουν τα ηλεκτρόνια από τα άτομα, δημιουργώντας μια ροή ηλεκτρικής ενέργειας. Τα ηλιακά πάνελ περιλαμβάνουν στην πραγματικότητα πολλές, μικρότερες μονάδες που ονομάζονται φωτοβολταϊκά κύτταρα. (Φωτοβολταϊκά σημαίνει ότι μετατρέπουν το ηλιακό φως σε ηλεκτρισμό.) Πολλά κύτταρα που συνδέονται μεταξύ τους αποτελούν ένα ηλιακό πάνελ.

Κάθε φωτοβολταϊκό κύτταρο είναι βασικά ένα σάντουιτς που αποτελείται από δύο φέτες ημι-αγώγιμου υλικού, συνήθως πυριτίου - το ίδιο που χρησιμοποιείται στη μικροηλεκτρονική.

Για να λειτουργήσουν, τα φωτοβολταϊκά κύτταρα πρέπει να δημιουργήσουν ένα ηλεκτρικό πεδίο. Όπως ένα μαγνητικό πεδίο, που συμβαίνει λόγω αντίθετων πόλων, ένα ηλεκτρικό πεδίο συμβαίνει όταν διαχωρίζονται τα αντίθετα φορτία. Για να αποκτήσουν αυτό το πεδίο, οι κατασκευαστές αναμιγνύουν πυρίτιο με άλλα υλικά, δίνοντας σε κάθε φέτα του σάντουιτς θετικό ή αρνητικό ηλεκτρικό φορτίο.

Συγκεκριμένα, εμφυτεύουν φωσφόρο στο ανώτερο στρώμα πυριτίου, το οποίο προσθέτει επιπλέον ηλεκτρόνια, με αρνητικό φορτίο, σε αυτό το στρώμα. Εν τω μεταξύ, το κάτω στρώμα λαμβάνει μια δόση βορίου, το οποίο έχει ως αποτέλεσμα λιγότερα ηλεκτρόνια ή θετικό φορτίο. Όλα αυτά δημιουργούν ένα ηλεκτρικό πεδίο στη διασταύρωση μεταξύ των στρώσεων πυριτίου. Στη συνέχεια, όταν ένα φωτόνιο του ηλιακού φωτός απαλλάξει ένα ηλεκτρόνιο, το ηλεκτρικό πεδίο θα ωθήσει αυτό το ηλεκτρόνιο έξω από τη διασταύρωση του πυριτίου.

Μερικά άλλα συστατικά του κυττάρου μετατρέπουν αυτά τα ηλεκτρόνια σε χρησιμοποιήσιμη ισχύ. Μεταλλικές αγώγιμες πλάκες στις πλευρές του κυττάρου συλλέγουν τα ηλεκτρόνια και τα μεταφέρουν σε καλώδια. Σε αυτό το σημείο, τα ηλεκτρόνια μπορούν να ρέουν όπως οποιαδήποτε άλλη πηγή ηλεκτρικής ενέργειας.

Πρόσφατα, οι ερευνητές παρήγαγαν εξαιρετικά λεπτά, ευέλικτα ηλιακά κύτταρα με πάχος μόλις 1,3 microns - περίπου το 1/100 του πλάτους μιας ανθρώπινης τρίχας - και είναι 20 φορές ελαφρύτερα από μια κόλλα χαρτιού A4. Επιστήμονες ανέφεραν, σε μια μελέτη που δημοσιεύθηκε το 2016 στο περιοδικό Organic Electronics, ότι ενώ τα κύτταρα είναι τόσο ελαφριά που μπορούν να κάθονται στην κορυφή μιας φυσαλίδας σαπουνιού, μπόρουν να παράγουν ενέργεια με την ίδια αποτελεσματικότητα όπως ηλιακά κύτταρα με βάση το γυαλί. Ελαφρύτερα, πιο εύκαμπτα ηλιακά κύτταρα όπως αυτά θα μπορούσαν να ενσωματωθούν στην αεροδιαστημική τεχνολογία ή ακόμα και στις φορέτες συσκεύες.

Υπάρχουν άλλοι τύποι τεχνολογίας για την ηλιακή ενέργεια - συμπεριλαμβανομένης της ηλιακής θερμικής και της συγκεντρωμένης ηλιακής ενέργειας (CSP) - που λειτουργούν με διαφορετικό τρόπο από τα φωτοβολταϊκα ηλιακα πάνελ, αλλά όλοι αξιοποιούν τη

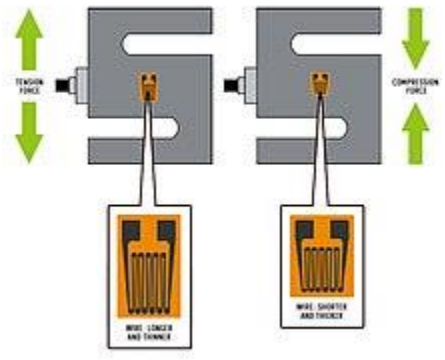
δύναμη του ηλιακού φωτός είτε για τη δημιουργία ηλεκτρικής ενέργειας είτε για τη θέρμανση νερού ή αέρα.

2.2.5 Αισθητήρας Load Cell [10]

Ένα load cell είναι ένας transducer δύναμης (μορφοτροπέας). Μετατρέπει μια δύναμη όπως τάση, συμπίεση, πίεση ή ροπή σε ηλεκτρικό σήμα που μπορεί να μετρηθεί και να τυποποιηθεί. Καθώς η δύναμη που ασκείται στο load cell αυξάνεται, το ηλεκτρικό σήμα αλλάζει αναλογικά. Οι πιο συνηθισμένοι τύποι load cell που χρησιμοποιούνται είναι μετρητές πίεσης, αέρος (pneumatic) και υδραυλικοί

Strain gauge load cell

Τα strain gauge load cells είναι το είδος που βρίσκεται συχνότερα στις βιομηχανίες. Είναι ιδανικό καθώς είναι πολύ ακριβές, ευέλικτο και οικονομικό. Δομικά, ένα load cell έχει ένα μεταλλικό σώμα στο οποίο έχουν ασφαλιστεί οι μετρητές τάσης (strain gauges). Το σώμα είναι συνήθως κατασκευασμένο από αλουμίνιο, κράμα χάλυβα ή ανοξείδωτο ατσάλι που το καθιστά πολύ ανθεκτικό αλλά ελάχιστα ελαστικό. Αυτή η ελαστικότητα δημιουργεί τον όρο "spring element", που αναφέρεται στο σώμα του load cell. Όταν ασκείται δύναμη στο load cell, το spring element παραμορφώνεται ελαφρώς, και εκτός εάν υπερφορτωθεί, επιστρέφει πάντα στο αρχικό του σχήμα. Καθώς το spring element παραμορφώνεται, οι μετρητές τάσης αλλάζουν επίσης σχήμα. Η προκύπτουσα αλλαγή στην αντίσταση στους μετρητές τάσης μπορεί να μετρηθεί σε Volt. Η μεταβολή της τάσης είναι ανάλογη με την ποσότητα δύναμης που ασκείται στον αισθητήρα, έτσι η ποσότητα δύναμης μπορεί να υπολογιστεί από την έξοδο του load cell.



Εικόνα 6: Strain gauge (Μετρητής Τάσης)

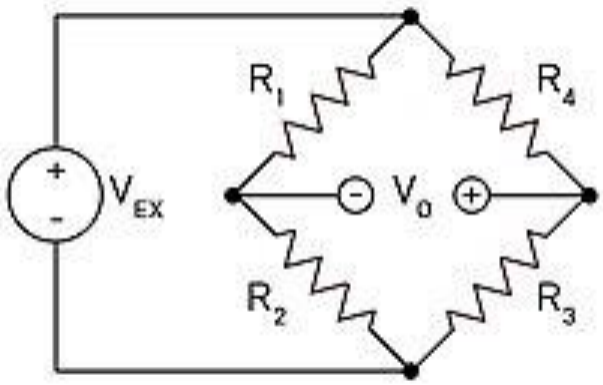
Ένα strain gauge είναι κατασκευασμένο από πολύ λεπτό καλώδιο, ή φύλλο αλουμινίου, τοποθετημένο σε σχέδιο πλέγματος και προσαρτημένο σε ένα εύκαμπτο στήριγμα.

Όταν αλλάζει το σχήμα του strain gauge, συμβαίνει μια αλλαγή στην ηλεκτρική του αντίσταση. Το καλώδιο ή το φύλλο αλουμινίου στο strain gauge είναι διατεταγμένο με τρόπο που, όταν ασκείται δύναμη προς μία κατεύθυνση, προκύπτει μια γραμμική αλλαγή στην αντίσταση. Η δύναμη έντασης τεντώνει ένα strain gauge, προκαλώντας το να γίνει πιο λεπτό και πιο μακρύ, με αποτέλεσμα την αύξηση της αντίστασης. Η δύναμη συμπίεσης κάνει το αντίθετο. Το strain gauge συμπιέζεται, γίνεται πιο παχύ και πιο κοντό, και η αντίσταση μειώνεται. Το strain gauge είναι προσαρτημένο σε ένα εύκαμπτο στήριγμα που επιτρέπει την εύκολη εφαρμογή του σε ένα load cell, αντικατοπτρίζοντας τις μικρές αλλαγές που πρέπει να μετρηθούν.

Δεδομένου ότι η μεταβολή στην αντίσταση που μετράτε από ένα μονό strain gauge είναι εξαιρετικά μικρή, είναι δύσκολο να μετρηθούν με ακρίβεια οι αλλαγές. Η αύξηση του αριθμού των strain gauges που εφαρμόζεται συλλογικά αυξάνει αυτές τις μικρές αλλαγές σε κάτι πιο μετρήσιμο. Ένα σετ 4 strain gauges σε ένα συγκεκριμένο κύκλωμα ονομάζεται γέφυρα Wheatstone.

Γέφυρα Wheatstone

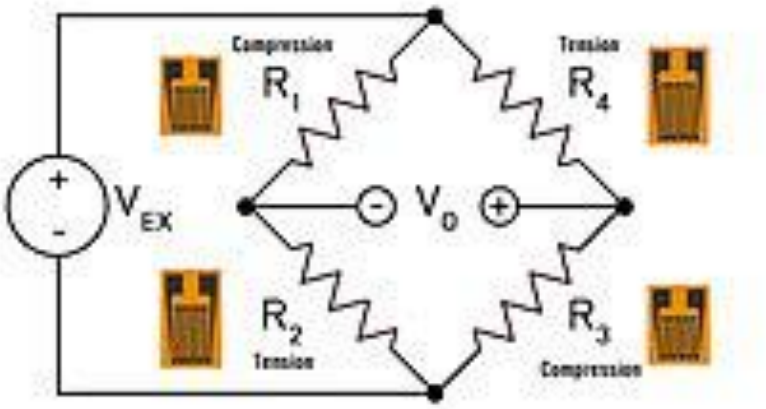
Η γέφυρα Wheatstone είναι μια διαμόρφωση τεσσάρων ισορροπημένων αντιστάσεων με μια γνωστή τάση διέγερσης που εφαρμόζεται όπως φαίνεται παρακάτω:



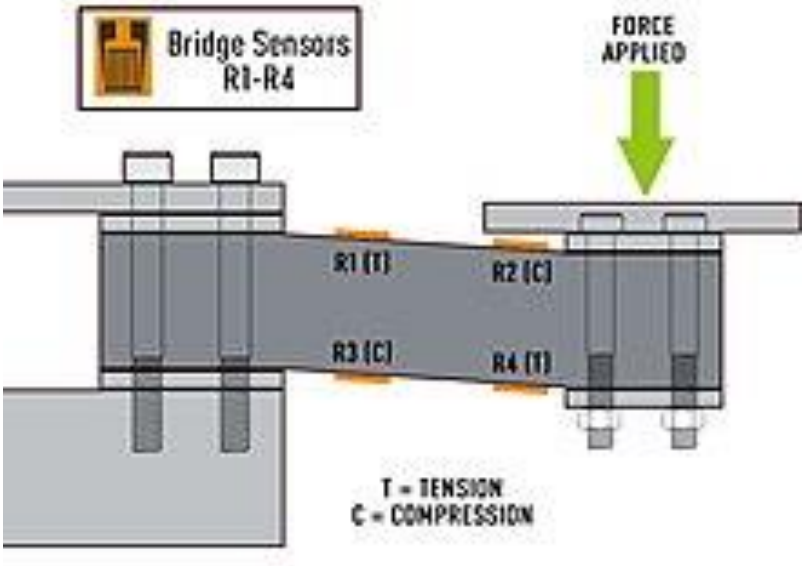
Εικόνα 7: Γέφυρα Wheatstone

Η τάση διέγερσης V_{EX} είναι μια γνωστή σταθερά και η τάση εξόδου V_0 είναι μεταβλητή ανάλογα με το σχήμα των strain gauges. Εάν όλες οι αντιστάσεις είναι ισορροπημένες, δηλαδή $R1/R2 = R4/R3$ τότε το V_0 είναι μηδέν. Εάν αλλάξει η αντίσταση ακόμη και σε μία από τις αντιστάσεις, τότε θα αλλάξει και το V_0. Η αλλαγή στο V_0μπορεί να μετρηθεί και να ερμηνευθεί χρησιμοποιώντας το νόμο του Ohm. Ο νόμος του Ohm δηλώνει ότι το ρεύμα (I, μετρούμενο σε αμπέρ) που διατρέχει έναν αγωγό μεταξύ δύο σημείων είναι ευθέως ανάλογο με την τάση V στα δύο σημεία. Η αντίσταση (R, μετρούμενη σε ohms)

εισάγεται ως η σταθερά σε αυτήν τη σχέση, ανεξάρτητα από το ρεύμα. Ο νόμος του Ohm εκφράζεται στην εξίσωση $I = V / R$.



Εικόνα 8: Γέφυρα Wheatstone συμπεριφορά αντιστάσεων



Εικόνα 9: Γέφυρα Wheatstone συμπεριφορά αντιστάσεων (2)

Όταν εφαρμοστεί στα 4 σκέλη του κυκλώματος γέφυρας Wheatstone, η προκύπτουσα εξίσωση είναι:

$$V_o = \left(\frac{R3}{R3 + R4} - \frac{R2}{R1 + R2} \right) V_{EX}$$

Εικόνα 10: Εξίσωση της Γέφυρας Wheatstone

Σε ένα load cell, οι αντιστάσεις αντικαθίστανται με strain gauges και διατάσσονται σε εναλλασσόμενο σχηματισμό τάσης και συμπίεσης. Όταν ασκείται δύναμη στο load cell, η δομή και η αντίσταση των strain gauges αλλάζει και μετράτε το V_o . Από τα προκύπτοντας δεδομένα, το V_o μπορεί εύκολα να προσδιοριστεί χρησιμοποιώντας την παραπάνω εξίσωση.

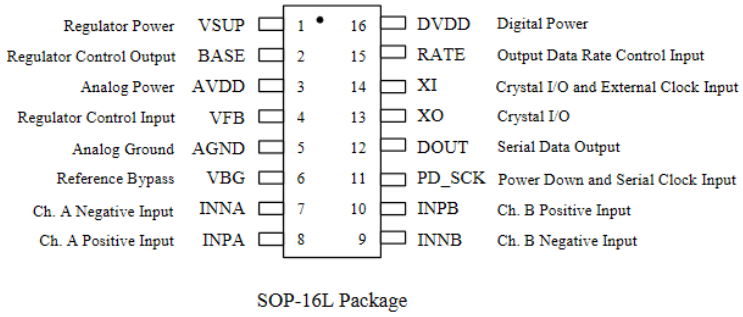
2.2.6 HX711 Αναλογικός μετατροπέας 24bit (ADC) για ζυγαριές [11]

ΠΕΡΙΓΡΑΦΗ

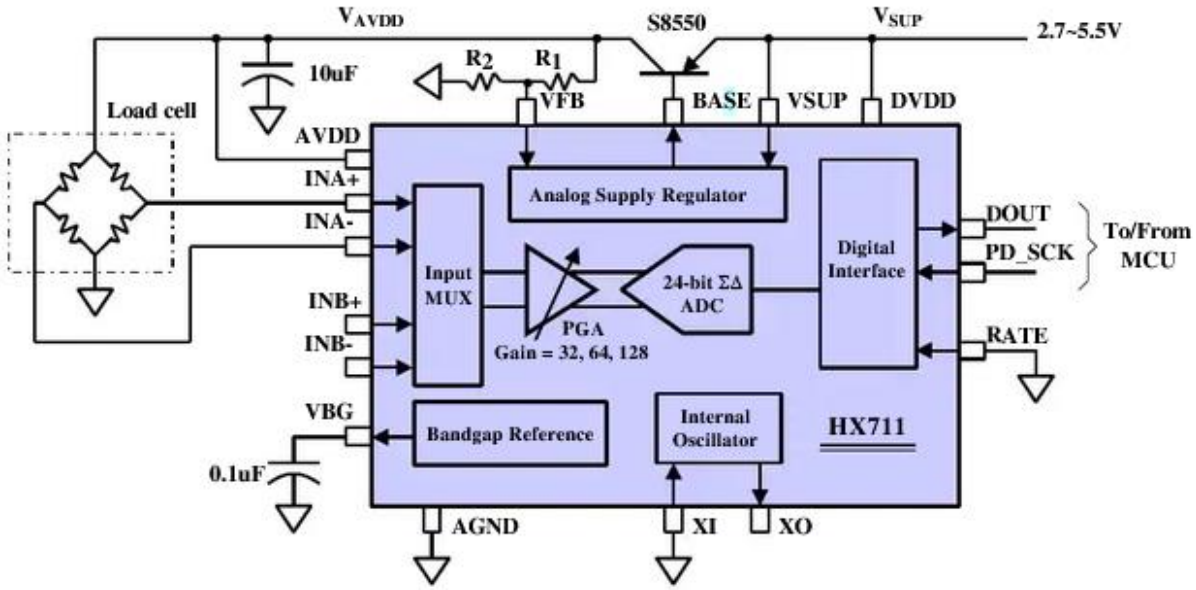
Το HX711 είναι ένας αναλογικός σε ψηφιακός μετασχηματιστής(ADC) 24 bit σχεδιασμένος για ζυγαριές και εφαρμογές ελέγχου σε βιομηχανίες για να αλληλοεπιδρά απευθείας με ένας αισθητήρα γέφυρας.

ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ

• Two selectable differential input channels
• On-chip active low noise PGA with selectable gain of 32, 64 and 128
• On-chip power supply regulator for load-cell and ADC analog power supply
• On-chip oscillator requiring no external component with optional external crystal
• On-chip power-on-reset
• Simple digital control and serial interface: pin-driven controls, no programming needed
• Selectable 10SPS or 80SPS output data rate
• Simultaneous 50 and 60Hz supply rejection
• Current consumption including on-chip analog power supply regulator: normal operation < 1.5mA, power down < 1uA
• Operation supply voltage range: 2.6 ~ 5.5V
• Operation temperature range: -40 ~ +85℃
• 16 pin SOP-16 package



Εικόνα 10: Το pinout του HX711



Εικόνα 11: Block διάγραμμα τυπικής χρήσης για ζυγαριά

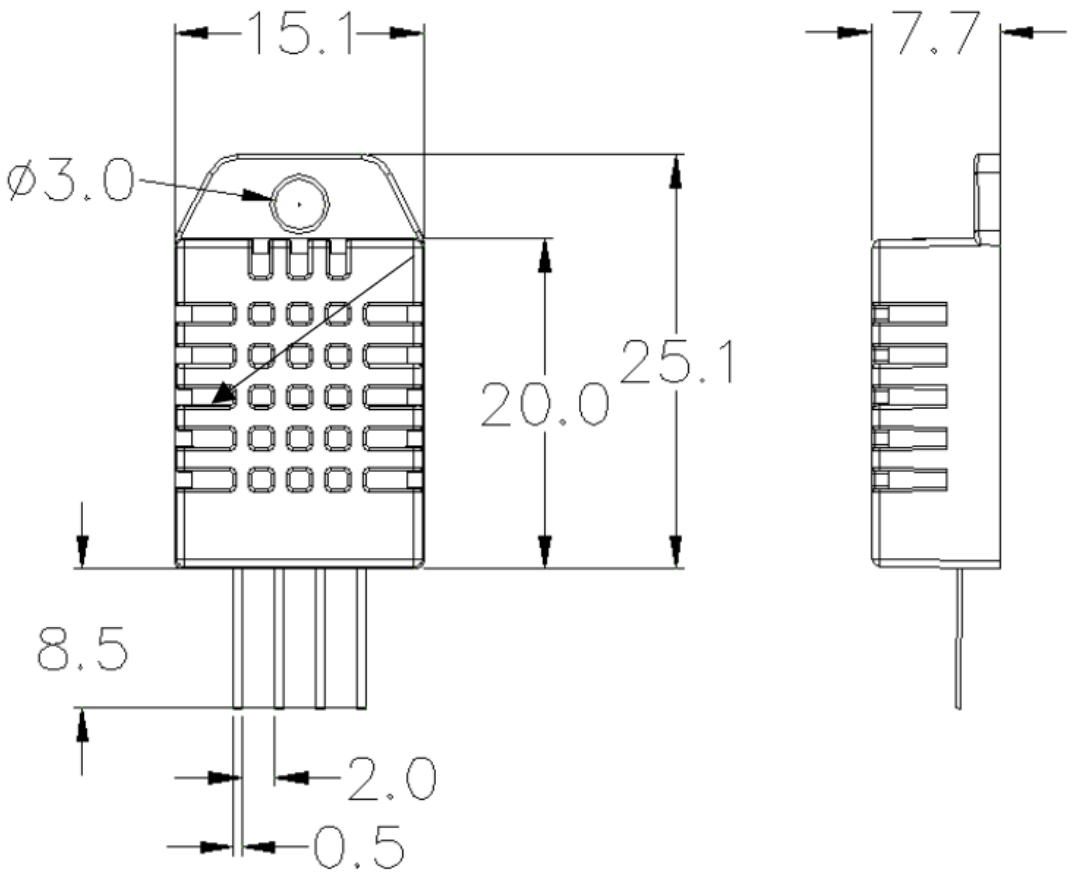
2.2.7 Αισθητήρας DHT22 (AM2302) [12]

ΠΕΡΙΓΡΑΦΗ

Ο αισθητήρας DHT22 ή αλλιώς AM2302 είναι ένας αισθητήρας υγρασίας και θερμοκρασίας ο οποίος έχει μικρό μέγεθος , χαμηλή κατανάλωση και μεγάλη απόσταση μετάδοσης της πληροφορίας έως και 20 μέτρα.

ΧΑΡΑΚΤΗΡΙΣΤΗΚΑ

Model	DHT22
Power supply	3.3-6V DC
Output signal	digital signal via single-bus
Sensing element	Polymer capacitor
Operating range	humidity 0-100%RH; temperature -40~80Celsius
Accuracy	humidity +-2%RH(Max +-5%RH); temperature <+-0.5Celsius
Resolution or sensitivity	humidity 0.1%RH; temperature 0.1Celsius
Repeatability	humidity +-1%RH; temperature +-0.2Celsius
Humidity hysteresis	+-0.3%RH
Long-term Stability	+-0.5%RH/year
Sensing period	Average: 2s
Interchangeability	fully interchangeable
Dimensions	small size 14*18*5.5mm; big size 22*28*5mm



Pin sequence number: 1 2 3 4 (from left to right direction).

Pin	Function
1	VDD----power supply
2	DATA--signal
3	NULL
4	GND

Εικόνα 12: Το pinout και οι διαστάσεις του αισθητήρα DHT22

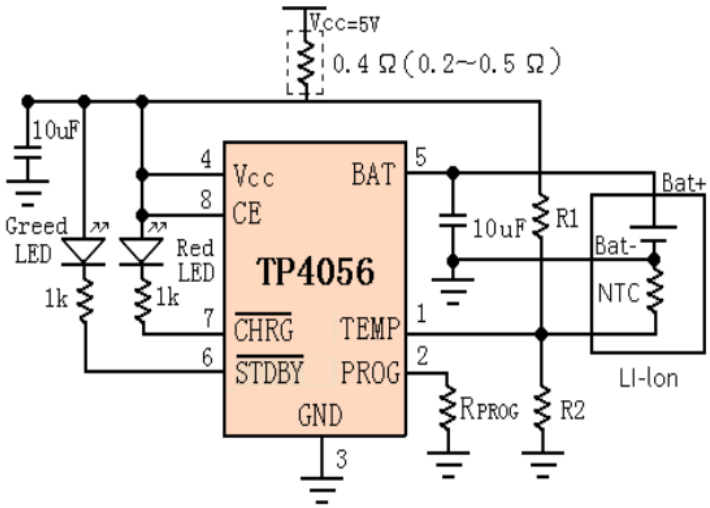
2.2.8 TP4056 [13]

ΠΕΡΙΓΡΑΦΗ

Το TP4056 είναι ένας πλήρης γραμμικός φορτιστής σταθερού ρεύματος/σταθερής τάσης για μόνο μια μπαταρία ιόντων λιθίου. Το πακέτο SOP και ο χαμηλός αριθμός εξωτερικών εξαρτημάτων καθιστούν το TP4056 ιδανικό για φορητές εφαρμογές. Επιπλέον, το TP4056 μπορεί να λειτουργήσει με χρήση USB και μετατροπέα ρεύματος τοίχου.

ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ

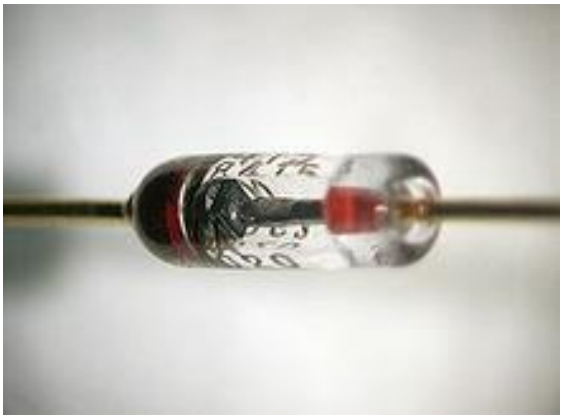
- Programmable Charge Current Up to 1000mA
- No MOSFET, Sense Resistor or Blocking Diode Required
- Complete Linear Charger in SOP-8 Package for Single Cell Lithium-Ion Batteries
- Constant-Current/Constant-Voltage
- Charges Single Cell Li-Ion Batteries Directly from USB Port
- Preset 4.2V Charge Voltage with 1.5% Accuracy
- Automatic Recharge
- two Charge Status Output Pins
- C/10 Charge Termination
- 2.9V Trickle Charge Threshold (TP4056)
- Soft-Start Limits Inrush Current
- Available Radiator in 8-Lead SOP Package,the Radiator need connect GND or impending



Εικόνα 13: Διάγραμμα τυπικής χρήσεις του TP4056

2.2.9 Δίοδοι [14]

Στην ηλεκτρονική, η δίοδος είναι ένα στοιχείο που περιορίζει τη κατευθυντήρια ροή των φορέων αγωγιμότητας (charge carriers). Ουσιαστικά, η δίοδος επιτρέπει στο ηλεκτρικό ρεύμα να περάσει από τη μια κατεύθυνση, αλλά μπλοκάρει την κίνηση από την αντίθετη κατεύθυνση. Έτσι, η δίοδος μπορεί να θεωρηθεί ως μια ηλεκτρονική εκδοχή της βαλβίδας, η οποία, για παράδειγμα, σε ένα σωλήνα νερού, δεν επιτρέπει ροή αντίθετη από την επιθυμητή. Οπότε, τα κυκλώματα που απαιτούν ροή προς μία μόνο κατεύθυνση περιλαμβάνουν μία ή περισσότερες διόδους στη σχεδίαση του κυκλώματος. Οι πρώτες δίοδοι περιλάμβαναν κρυστάλλους cat's whisker και λυχνίες κενού. Σήμερα, οι περισσότερες δίοδοι είναι κατασκευασμένες από υλικά ημιαγωγών όπως πυρίτιο ή γερμάνιο.



Εικόνα 14: Κοντινό πλάνο μιας διόδου πυριτίου. Η άνοδος είναι στη δεξιά πλευρά και η κάθοδος βρίσκεται στην αριστερή πλευρά (όπου σημειώνεται με μια μαύρη ταινία). Ο τετράγωνος κρύσταλλος πυριτίου μπορεί να φανεί μεταξύ των δύο αγωγών.

Κύριες λειτουργίες

Η πιο συνηθισμένη λειτουργία μιας δίοδού είναι να επιτρέψει στο ηλεκτρικό ρεύμα να περάσει προς μία κατεύθυνση, ενώ το εμποδίζει στην αντίθετη κατεύθυνση. Ως εκ τούτου, η δίοδος μπορεί να θεωρηθεί ως ηλεκτρονική έκδοση βαλβίδας. Αυτή η μονοκατευθυντική συμπεριφορά ονομάζεται ανόρθωση και χρησιμοποιείται για τη μετατροπή εναλλασσόμενου ρεύματος (AC) σε συνεχές ρεύμα (DC).

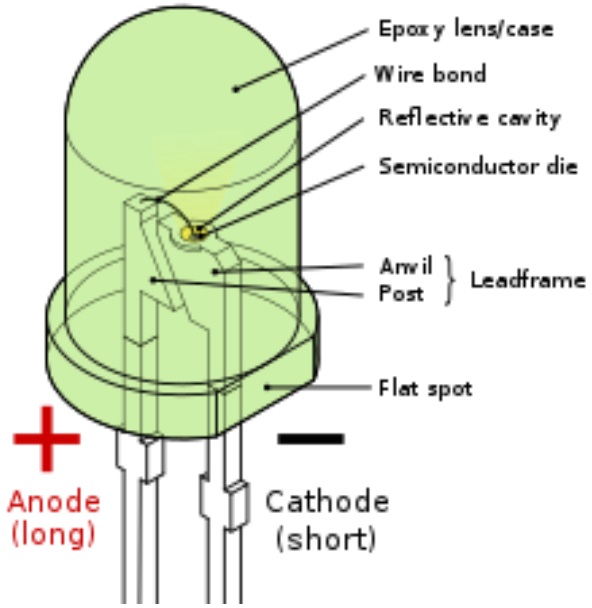
2.2.10 Δίοδος εκπομπής φωτός (LEDs) [15]

Μια δίοδος εκπομπής φωτός (LED) είναι μια πηγή φωτός ημιαγωγού που εκπέμπει φως όταν το ρεύμα ρέει μέσα από αυτό. Τα ηλεκτρόνια στον ημιαγωγό ανασυνδυάζονται με τρύπες ηλεκτρονίων, απελευθερώνοντας ενέργεια με τη μορφή φωτονίων. Το χρώμα του φωτός (που αντιστοιχεί στην ενέργεια των φωτονίων) καθορίζεται από την ενέργεια που απαιτείται για να διασχίσουν τα ηλεκτρόνια το διάκενο ζώνης του ημιαγωγού. Το λευκό φως λαμβάνεται με τη χρήση πολλαπλών ημιαγωγών ή ενός στρώματος φωσφόρου που εκπέμπει φως στη συσκευή ημιαγωγών.

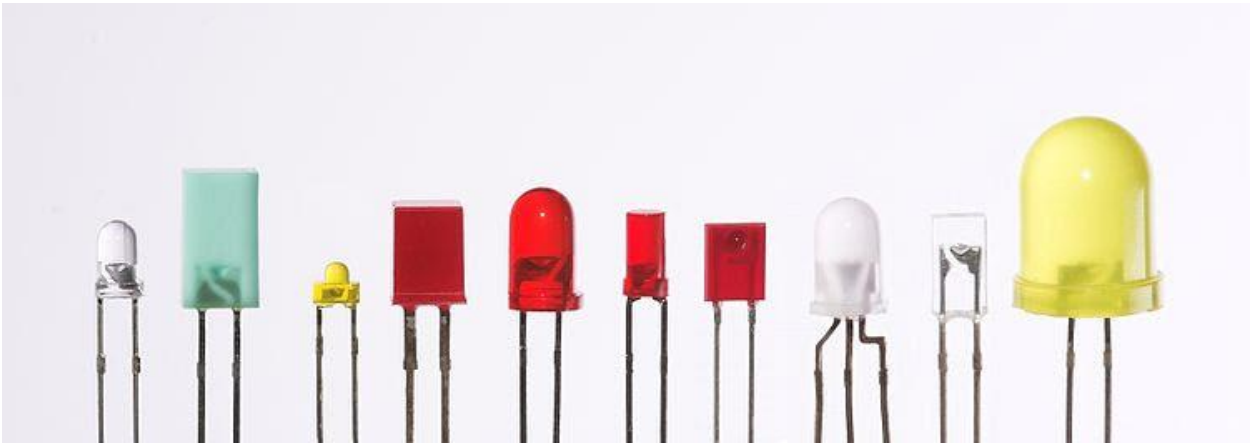
Εμφανιζόμενα ως πρακτικά ηλεκτρονικά εξαρτήματα το 1962, τα πρώτα LED εκπέμπουν φως υπέρυθρης ακτινοβολίας χαμηλής έντασης (IR). Οι υπέρυθρες λυχνίες LED χρησιμοποιούνται σε κυκλώματα τηλεχειριστηρίου, όπως αυτά που χρησιμοποιούνται με μεγάλη ποικιλία ηλεκτρονικών ειδών ευρείας κατανάλωσης. Τα πρώτα LED ορατού φωτός ήταν χαμηλής έντασης και περιορίστηκαν σε κόκκινο. Τα μοντέρνα LED είναι διαθέσιμα σε ορατά, υπεριώδη (UV) και υπέρυθρα μήκη κύματος, με υψηλή απόδοση φωτός.

Τα πρώιμα LED χρησιμοποιούνται συχνά ως ενδεικτικές λυχνίες, αντικαθιστώντας μικρούς λαμπτήρες πυρακτώσεως και σε οθόνες επτά τμημάτων. Οι πρόσφατες εξελίξεις έχουν παράγει LED λευκού φωτός υψηλής απόδοσης κατάλληλα για φωτισμό δωματίου και εξωτερικού χώρου. Τα LED οδήγησαν σε νέες οθόνες και αισθητήρες, ενώ οι υψηλοί ρυθμοί μεταγωγής τους είναι χρήσιμοι στην προηγμένη τεχνολογία επικοινωνιών.

Τα LED έχουν πολλά πλεονεκτήματα έναντι των πυρακτωμένων πηγών φωτός, όπως χαμηλότερη κατανάλωση ενέργειας, μεγαλύτερη διάρκεια ζωής, βελτιωμένη φυσική αντοχή, μικρότερο μέγεθος και ταχύτερη εναλλαγή. Οι λυχνίες LED χρησιμοποιούνται σε εφαρμογές τόσο διαφορετικές όπως ο φωτισμός της αεροπορίας, τα fairy lights, οι προβολείς αυτοκινήτων, η διαφήμιση, ο γενικός φωτισμός, τα σήματα κυκλοφορίας, τα φλας της κάμερας, η ταπετσαρία με φωτισμό, τα φώτα καλλιέργειας κηπουρικής και οι ιατρικές συσκευές.



Εικόνα 15: Μέρη ενός συμβατικού LED


Εικόνα 16: Διάφορα είδη LED

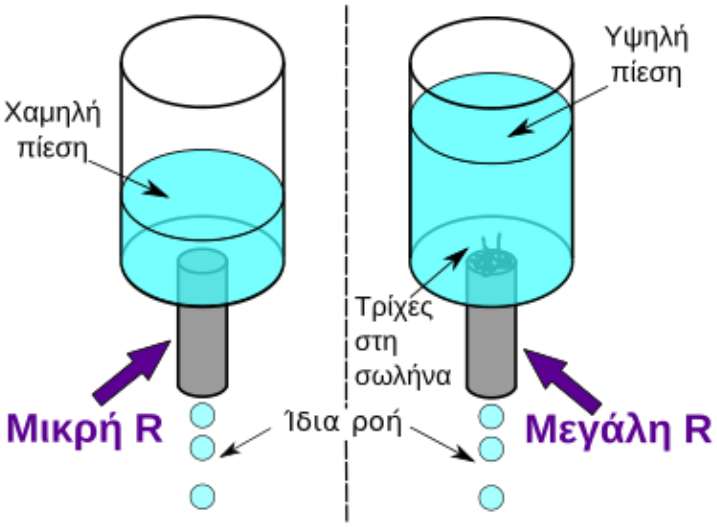
Τα LED παράγονται σε διάφορα σχήματα και μεγέθη. Το χρώμα του πλαστικού φακού είναι συχνά το ίδιο με το πραγματικό χρώμα του εκπεμπόμενου φωτός, αλλά όχι πάντα. Για παράδειγμα, το μωβ πλαστικό χρησιμοποιείται συχνά για υπέρυθρες λυχνίες LED και οι περισσότερες μπλε συσκευές έχουν άχρωμο περίβλημα. Τα σύγχρονα LED υψηλής ισχύος βρίσκονται συνήθως σε τεχνολογία επιφανείας (SMT).

2.2.11 Αντιστάτης [16]

Ο αντιστάτης είναι ένα ηλεκτρολογικό/ηλεκτρονικό εξάρτημα το οποίο χρησιμοποιείται σε διάφορα κυκλώματα για τον έλεγχο της ροής του ρεύματος. Ο αντιστάτης μερικές φορές λέγεται και ηλεκτρική αντίσταση, αν και ηλεκτρική αντίσταση είναι ένα φαινόμενο. Συχνές συνδεσμολογίες αντιστατών που συναντά κανείς στα ηλεκτρικά ή ηλεκτρονικά κυκλώματα είναι η σύνδεση αντιστάσεων σε σειρά και η σύνδεση αντιστάσεων παράλληλα. Η πρώτη συνδεσμολογία ονομάζεται διαιρέτης τάσης, ενώ η δεύτερη ονομάζεται διαιρέτης ρεύματος

Εικόνα 17: Μία συνηθισμένη αντίσταση

Ηλεκτρική αντίσταση άνθρακα, 10.000 Ω ή 10kΩ, ανοχής ±5%. Μπορεί να αναγνωριστεί από τον χρωματικό κώδικα αντιστάσεων. Το πολύμετρο μπορεί να χρησιμοποιηθεί για τον προσδιορισμό της τιμής.



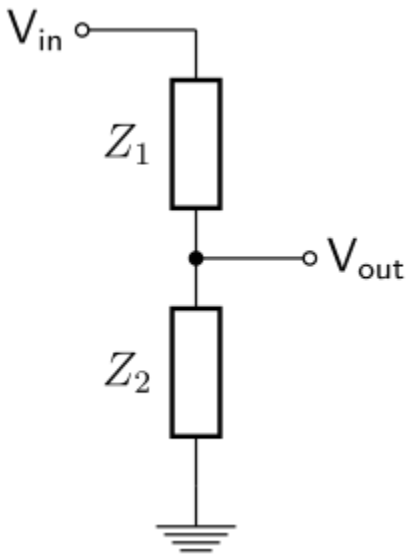
Εικόνα 18: Περιγραφή λειτουργίας αντιστάτη

Το ρεύμα που διαρρέει ένα κύκλωμα μπορεί να συγκριθεί με το νερό που διαρρέει ένα κύκλωμα σωληνώσεων. Η ωμική αντίσταση μπορεί να συγκριθεί με τις τρίχες οι οποίες

εμποδίζουν την ροή του νερού. Για να έχουμε την ίδια ροή νερού έχοντας την αντίσταση των τριχών πρέπει να αυξήσουμε την πίεση του νερού. Η πίεση του νερού μπορεί να συγκριθεί με την τάση Volts του ηλεκτρικού κυκλώματος

2.2.12 Διαιρέτης τάσης [17]

Στα ηλεκτρονικά, ένας διαχωριστής τάσης (επίσης γνωστός ως δυνητικός διαχωριστής) είναι ένα παθητικό γραμμικό κύκλωμα που παράγει μια τάση εξόδου (V_{out}) που είναι ένα κλάσμα της τάσης εισόδου του (V_{in}).Η διαίρεση τάσης είναι το αποτέλεσμα της κατανομής της τάσης εισόδου μεταξύ των εξαρτημάτων του διαχωριστή. Ένα απλό παράδειγμα διαχωριστή τάσης είναι δύο αντιστάσεις συνδεδεμένες σε σειρά, με την τάση εισόδου να εφαρμόζεται σε όλο το ζεύγος αντιστάσεων και η τάση εξόδου να προκύπτει από τη σύνδεση μεταξύ τους. Τα διαχωριστικά τάσης αντιστάσεων χρησιμοποιούνται συνήθως για τη δημιουργία τάσεων αναφοράς ή για τη μείωση του μεγέθους μιας τάσης ώστε να μπορεί να μετρηθεί.



Εικόνα 19: Ένας απλός διαχωριστής τάσης

Γενική Περίπτωση

Ένας διαχωριστής τάσης που αναφέρεται στο έδαφος δημιουργείται συνδέοντας δύο ηλεκτρικές αντιστάσεις σε σειρά, όπως φαίνεται στην Εικόνα 19.Η τάση εισόδου εφαρμόζεται σε όλες τις σειρές αντίστασης Z_1 και Z_2 και η έξοδος είναι η τάση σε Z_2. Τα Z_1 και Z_2 μπορεί να αποτελούνται από οποιονδήποτε συνδυασμό στοιχείων όπως αντιστάσεις, επαγωγείς και πυκνωτές.

Εάν το ρεύμα στο καλώδιο εξόδου είναι μηδέν, τότε η σχέση μεταξύ της τάσης εισόδου, $V_{in,}$ και της τάσης εξόδου, V_{out} , είναι:

$$V_{\text{out}} = V_{\text{in}} \cdot \frac{Z_2}{Z_1 + Z_2}$$

2.3 Προδιαγραφές απαιτήσεων λογισμικού

Το λογισμικό θα πρέπει να τηρεί τις παρακάτω προδιαγραφές. Πρώτα από όλα πρέπει να υλοποιηθούν συναρτήσεις οι οποίες θα είναι ικανές να διαβάζουν τις τιμές θερμοκρασίας , υγρασίας , βάρους και ποσοστού της μπαταρίας. Έπειτα, χρειάζεται να δημιουργηθούν οι κατάλληλες διαδικασίες για την αποστολή των παραπάνω δεδομένων στο ThingSpeak ή ακόμα και μέσω SMS, ενώ στην συνέχεια πρέπει να ακολουθήσει η διαδικασία η οποία θα βάζει το μικροελεγκτή ESP32 σε Deep Sleep Mode. Κατόπιν, ο χρήστης θα πρέπει να κάνει configure τη ζυγαριά με ευκολία, πράγμα το οποίο θα γίνει εφικτό με την χρήση του ESP32 ως web server και με την χρήση μιας web εφαρμογής η οποία θα επικοινωνεί με τη βοήθεια web sockets. Με αυτόν τον τρόπο, ο χρήστης θα είναι ικανός να ρυθμίσει τη ζυγαριά από οποιαδήποτε συσκευή με πρόσβαση σε browser και τα δεδομένα θα αποθηκεύονται στη μνήμη του ESP32. Τέλος, η πρόσβαση στη web εφαρμογή θα μπορεί να γίνει ευκολά από τον χρήστη, δημιουργώντας ένα hotspot με τη βοήθεια του ESP32.

Προδιαγραφές λογισμικού αναφορικά

- Web Application (scale’s configuration page)
- Web Socket Communication
- Web Server Handler
- ESP32 WiFi HotSpot
- Αποθήκευση δεδομένων
- Ανάγνωση Βάρους
- Ανάγνωση Θερμοκρασία
- Ανάγνωση Υγρασίας
- Ανάγνωση του ποσοστού της μπαταρίας
- Αποστολή μετρήσεων μέσω SMS
- Αποστολή μετρήσεων στο ThingSpeak
- ESP32 Deep Sleep Mode

2.3.1 Short Message Service [18]

Short Message Service γνωστό και ως SMS, είναι υπηρεσία της κινητής τηλεφωνίας, με την οποία ο χρήστης έχει τη δυνατότητα να αποστείλει ή να παραλάβει σύντομο γραπτό μήνυμα από άλλους χρήστες, στην οθόνη του κινητού του τηλεφώνου.

Η λέξη είναι ακρωνύμιο των αγγλικών λέξεων Short Message Service που σημαίνει Υπηρεσία Σύντομου Μηνύματος και καθιερώθηκε για πρώτη φορά από τις εταιρείες κινητής τηλεφωνίας το 1992. Ένα απλό γραπτό μήνυμα ορίζεται στους 160 αλφαριθμητικούς χαρακτήρες, συμπεριλαμβανομένων και των κενών διαστημάτων. Πέραν του ορίου αυτού, ένα μήνυμα ανάλογα με τις δυνατότητες κινητού τηλεφώνου μπορεί να συνεχιστεί σε περισσότερους χαρακτήρες υπολογιζόμενο στη χρέωση του ως δεύτερο μήνυμα κ.ο.κ.

Έχει υπολογιστεί πως το 74% των χρηστών κινητού τηλεφώνου παγκοσμίως χρησιμοποιούν αυτή την υπηρεσία, κάτι που μεταφράζεται σε περίπου 2,4 δισεκατομμύρια SMS ημερησίως. Στις περισσότερες περιπτώσεις χρησιμοποιείται αυτή η μέθοδος επικοινωνίας διότι το κόστος αποστολής ενός γραπτού μηνύματος είναι κατά πολύ μικρότερο από την απ' ευθείας συνομιλία.

Τα τελευταία χρόνια πολλές εταιρείες παροχής κινητής τηλεφωνίας δίνουν τη δυνατότητα στους χρήστες να αποστείλουν SMS και μέσω ηλεκτρονικού υπολογιστή. Σ' αυτές τις περιπτώσεις η υπηρεσία παρέχεται δωρεάν ή με μικρή χρέωση. Επίσης πολύ διαδεδομένες τα τελευταία χρόνια είναι και οι υπηρεσίες μαζικής αποστολής SMS από τους χρήστες για εμπορικούς και ψηφοθηρικούς σκοπούς (παιχνίδια, διαγωνισμοί, κληρώσεις, συμμετοχή σε τηλεψηφοφορίες κλπ.) όπου το εκάστοτε κόστος ορίζεται από τον παροχέα και συνήθως είναι ακριβότερο (έως και πολλαπλάσιο) ενός συνηθισμένου μηνύματος.

2.3.2 ThingSpeak [19] [20]

Σύμφωνα με τους προγραμματιστές του, το "ThingSpeak είναι μια εφαρμογή Internet of Things (IoT) ανοιχτού κώδικα και API για την αποθήκευση και ανάκτηση δεδομένων από πράγματα χρησιμοποιώντας το πρωτόκολλο HTTP και MQTT μέσω του Διαδικτύου ή μέσω τοπικού δικτύου. Το ThingSpeak επιτρέπει τη δημιουργία εφαρμογής για τη καταγραφή μετρήσεων από αισθητήρες, εφαρμογές παρακολούθησης της τοποθεσίας και ένα κοινωνικό δίκτυο πραγμάτων με ενημερώσεις των καταστάσεών τους ". Το ThingSpeak κυκλοφόρησε αρχικά από το ioBridge το 2010 ως υπηρεσία υποστήριξης εφαρμογών IoT. Το ThingSpeak έχει ενσωματωμένη υποστήριξη από το λογισμικό MATLAB από τη MathWorks, επιτρέποντας στους χρήστες του ThingSpeak να αναλύουν και να οπτικοποιούν τα μεταφορτωμένα δεδομένα χρησιμοποιώντας το Matlab χωρίς να απαιτούν την αγορά άδειας Matlab από το Mathworks. Το ThingSpeak έχει στενή σχέση με την Mathworks, Inc. Στην πραγματικότητα, όλα τα έγγραφα τεκμηρίωσης ThingSpeak ενσωματώνονται στον ιστότοπο τεκμηρίωσης Matworks της

Mathworks και επιτρέπουν ακόμη και τους εγγεγραμμένους λογαριασμούς χρηστών του Mathworks ως έγκυρα διαπιστευτήρια σύνδεσης στον ιστότοπο του ThingSpeak. Οι όροι υπηρεσίας και η πολιτική απορρήτου του ThingSpeak.com βρίσκονται μεταξύ του χρήστη που συμφωνεί και της Mathworks, Inc.

Εικόνα 20: Λογότυπο του ThingSpeak

2.3.3 Deep Sleep Mode [21]

Το ESP32 μπορεί να χρησιμοποιήσει διάφορους τρόπους κατανάλωσης ενέργειας.

- Active mode
- Modem Sleep mode
- Light Sleep mode
- Deep Sleep mode
- Hibernation mode

Μπορείτε να συγκρίνετε τις πέντε διαφορετικές λειτουργίες στον παρακάτω πίνακα από το ESP32 Espressif datasheet.

Power mode	Active	Modem-sleep	Light-sleep	Deep-sleep	Hibernation
Sleep pattern	Association sleep pattern			ULP sensor-monitored pattern	-
CPU	ON	ON	PAUSE	OFF	OFF
Wi-Fi/BT baseband and radio	ON	OFF	OFF	OFF	OFF
RTC memory and RTC peripherals	ON	ON	ON	ON	OFF
ULP co-processor	ON	ON	ON	ON/OFF	OFF

Εικόνα 21: Πίνακας σύγκρισης διαφορετικών λειτουργιών.

Το ESP32 Espressif datasheet παρέχει επίσης έναν πίνακα που συγκρίνει την κατανάλωση ενέργειας των διαφορετικών τρόπων λειτουργίας.

Power mode	Description	Power consumption
Active (RF working)	Wi-Fi Tx packet 14 dBm ~ 19.5 dBm	Please refer to Table 10 for details.
	Wi-Fi / BT Tx packet 0 dBm	
	Wi-Fi / BT Rx and listening	
Modem-sleep	The CPU is powered on.	Max speed 240 MHz: 30 mA ~ 50 mA
		Normal speed 80 MHz: 20 mA ~ 25 mA
		Slow speed 2 MHz: 2 mA ~ 4 mA
Light-sleep	-	0.8 mA
Deep-sleep	The ULP co-processor is powered on.	150 μA
	ULP sensor-monitored pattern	100 μA @1% duty
	RTC timer + RTC memory	10 μA
Hibernation	RTC timer only	5 μA
Power off	CHIP_PU is set to low level, the chip is powered off	0.1 μA

Εικόνα 22: Πίνακας σύγκρισης κατανάλωσης ενέργειας

Επίσης, εδώ είναι ο πίνακας 10 για σύγκριση της κατανάλωσης ενέργειας σε ενεργή λειτουργία (Active mode):

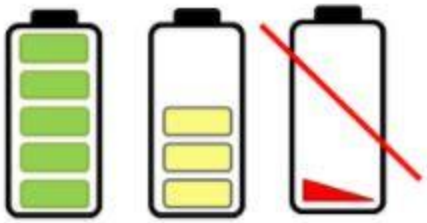
Table 10: RF Power-Consumption Specifications

Mode	Min	Typ	Max	Unit
Transmit 802.11b, DSSS 1 Mbps, POUT = +19.5 dBm	-	240	-	mA
Transmit 802.11b, OFDM 54 Mbps, POUT = +16 dBm	-	190	-	mA
Transmit 802.11g, OFDM MCS7, POUT = +14 dBm	-	180	-	mA
Receive 802.11b/g/n	-	95 ~ 100	-	mA
Transmit BT/BLE, POUT = 0 dBm	-	130	-	mA
Receive BT/BLE	-	95 ~ 100	-	mA

Εικόνα 23: Πίνακας σύγκρισης κατανάλωσης ενέργειας (2)

Χρησιμότητα

Το να λειτουργεί το ESP32 σε active mode με μπαταρίες δεν είναι ιδανικό, καθώς η ισχύς από τις μπαταρίες θα εξαντληθεί πολύ γρήγορα.



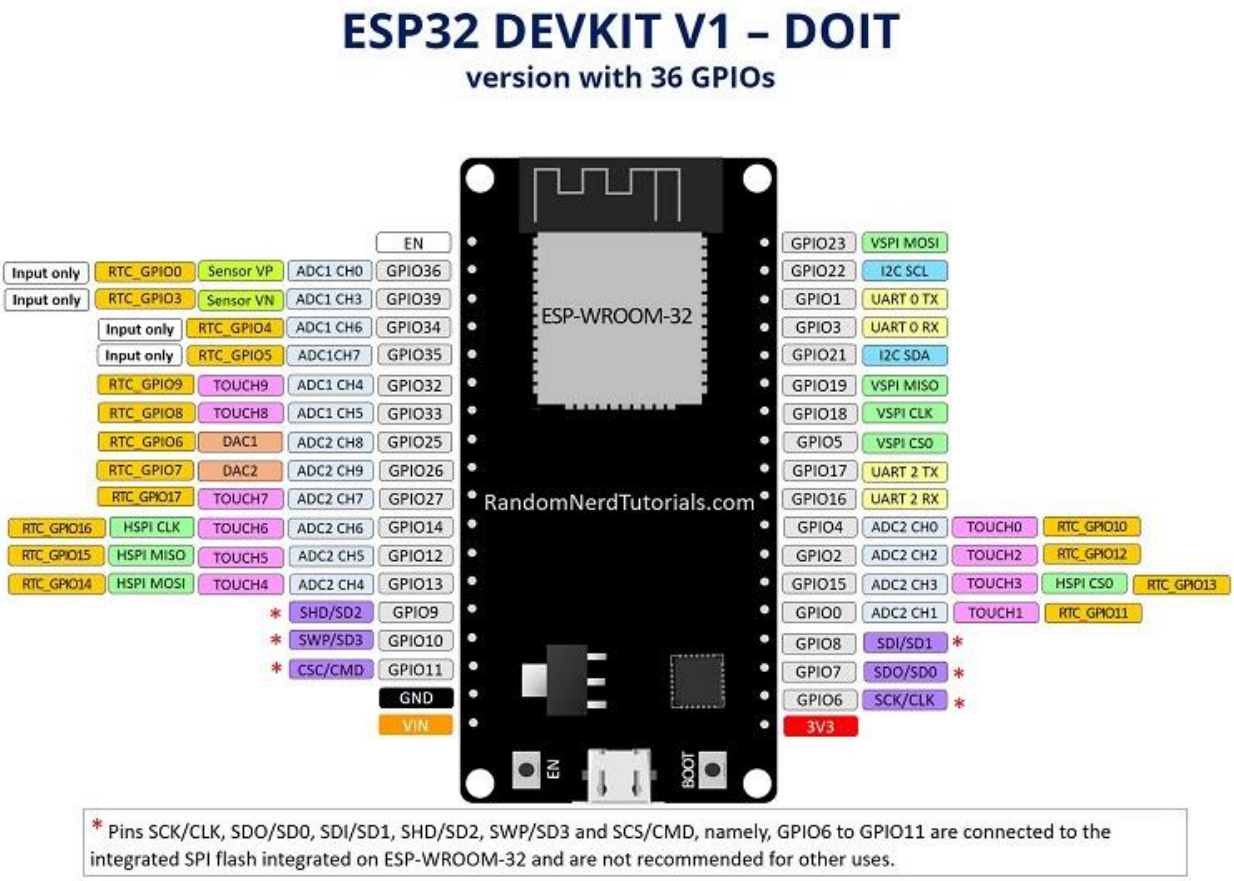
Εικόνα 24.

Εάν τεθεί το ESP32 σε deep sleep mode, θα μειώσει την κατανάλωση ενέργειας και οι μπαταρίες του θα διαρκέσουν περισσότερο.Έχοντας το ESP32 σε deep sleep mode σημαίνει ότι κόβεται τις δραστηριότητες που καταναλώνουν περισσότερη ισχύ κατά τη λειτουργία, αλλά αφήνετε αρκετή δραστηριότητα για να ξυπνήσετε τον επεξεργαστή όταν συμβεί κάτι ενδιαφέρον.Στο deep sleep mode δεν πραγματοποιούνται δραστηριότητες CPU ή Wi-Fi, αλλά ο συν-επεξεργαστής Ultra Low Power (ULP) εξακολουθεί να είναι ενεργοποιημένος.Ενώ το ESP32 βρίσκεται σε deep sleep mode, η RTC memory παραμένει επίσης ενεργοποιημένη, οπότε μπορούμε να γράψουμε ένα πρόγραμμα για τον επεξεργαστή ULP και να το αποθηκεύσουμε στη RTC memory για πρόσβαση σε περιφερειακές συσκευές, εσωτερικούς χρονοδιακόπτες και εσωτερικούς αισθητήρες.Αυτός ο τρόπος λειτουργίας είναι χρήσιμος εάν πρέπει να ξυπνήσετε την κύρια CPU με εξωτερικό συμβάν, χρονοδιακόπτη ή και τα δύο, διατηρώντας παράλληλα την ελάχιστη κατανάλωση ενέργειας.

RTC_GPIO pins

Κατά τη διάρκεια του deep sleep mode, κάποια από τα pins του ESP32 μπορούν να χρησιμοποιηθούν από τον συν-επεξεργαστή ULP, δηλαδή τα pins RTC_GPIO και τα pins αφής. Το ESP32 datasheet παρέχει έναν πίνακα που προσδιορίζει τα pins RTC_GPIO.

Μπορείτε να ρίξετε μια ματιά στο παρακάτω pinout για να εντοπίσετε τα διαφορετικά pins RTC_GPIO. Τα pins RTC_GPIO επισημαίνονται με ένα πορτοκαλί ορθογώνιο κουτί.



Εικόνα 25: ESP32 Pinout

Τρόποι αφύπνισης

Αφού θέσετε το ESP32 σε deep sleep mode, υπάρχουν διάφοροι τρόποι για να το ξυπνήσετε:

- Μπορείτε να χρησιμοποιήσετε το χρονόμετρο, ξυπνάτε το ESP32 χρησιμοποιώντας προκαθορισμένες χρονικές περιόδους.
- Μπορείτε να χρησιμοποιήσετε τα pins αφής.
- Μπορείτε να χρησιμοποιήσετε δύο δυνατότητες εξωτερικής αφύπνισης: είτε μία εξωτερική αφύπνιση, είτε πολλές διαφορετικές εξωτερικές αφύπνιση.
- Μπορείτε να χρησιμοποιήσετε τον επεξεργαστή ULP.

Γράφοντας κώδικα για το deep sleep mode

Για να γράψετε ένα κώδικα που θέτει το ESP32 σε deep sleep mode και, στη συνέχεια, το ξυπνάει, πρέπει να έχετε υπόψη ότι:

Πρώτα, πρέπει να ορίσετε τους τρόπους αφύπνισης. Αυτό σημαίνει το τι θα ξυπνήσει το ESP32. Μπορεί να χρησιμοποιηθεί παραπάνω από ένας τρόπος αφύπνισης.

Μπορείτε να αποφασίσετε ποια περιφερειακά θα απενεργοποιήσετε ή θα συνεχίσετε κατά τη διάρκεια του deep sleep mode. Ωστόσο, από προεπιλογή, το ESP32 απενεργοποιεί αυτόματα τα περιφερειακά που δεν χρειάζονται με τον τρόπο αφύπνισης που ορίσατε.

Τέλος, χρησιμοποιείστε τη συνάρτηση **esp_deep_sleep_start ()** για να θέσετε το ESP32 σε deep sleep mode.

Ξύπνημα σε καθορισμένη χρονική στιγμή

Το ESP32 μπορεί να μεταβεί deep sleep mode και στη συνέχεια να ξυπνήσει σε προκαθορισμένες χρονικές περιόδους. Αυτή η λειτουργία είναι ιδιαίτερα χρήσιμη αν εκτελείτε έργα που απαιτούν χρονοσήμανση ή καθημερινές εργασίες, διατηρώντας παράλληλα χαμηλή κατανάλωση ενέργειας. Ο ελεγκτής ESP32 RTC διαθέτει ένα ενσωματωμένο χρονόμετρο που μπορείτε να χρησιμοποιήσετε για να ενεργοποιήσετε το ESP32 μετά από ένα προκαθορισμένο χρονικό διάστημα.

Ενεργοποίηση του ξυπνήματος προκαθορισμένης ώρας

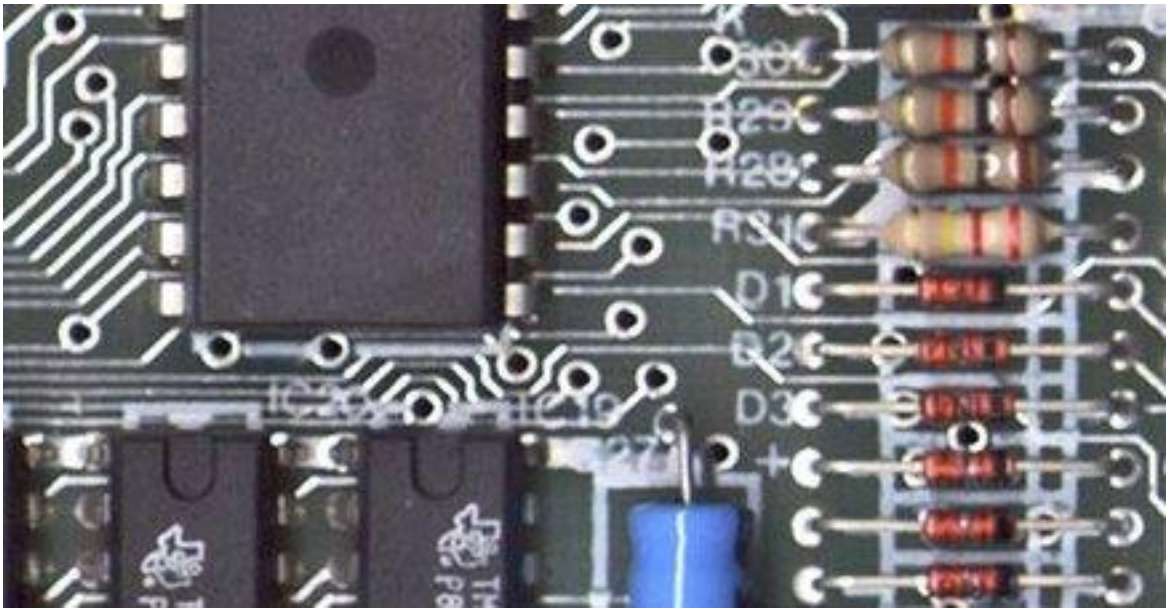
Η δυνατότητα του ESP32 να ξυπνήσει μετά από ένα προκαθορισμένο χρονικό διάστημα είναι πολύ απλή. Στο Arduino IDE, απλώς πρέπει να καθορίσετε τον χρόνο ύπνου σε μικροδευτερόλεπτα στην ακόλουθη συνάρτηση:

esp_sleep_enable_timer_wakeup(time_in_us);

ΚΕΦΑΛΑΙΟ 3
Σχεδιασμός κυκλώματος

3.1 Πλακέτα Τυπωμένου Κυκλώματος (PCB) [22] [23] [24]

Μια πλακέτα τυπωμένου κυκλώματος (PCB), είναι μια πλακέτα η οποία χρησιμοποιείται για την υποστήριξη και διασύνδεση ηλεκτρονικών στοιχείων μέσω αγώγιμων μονοπατιών τα οποία τυπώνονται πάνω στην πλακέτα. Μια τέτοια πλακέτα μπορεί να είναι μονής όψης, διπλής όψης είτε να αποτελείται από πολλαπλά επίπεδα. Οι πλακέτες πολλαπλών στρώσεων επιτρέπουν την τοποθέτηση περισσότερων στοιχείων στην ίδια επιφάνεια. Οι πιο εξελιγμένες πλακέτες ενσωματώνουν ηλεκτρονικά στοιχεία μέσα σε μια στρώση.



Εικόνα 26: Μέρος μιας πλακέτας υπολογιστών Sinclair ZX Spectrum του 1984, ενός PCB, που εμφανίζει τα αγώγιμα ίχνη, τις διατομές (τις διαδρομές διαμέσου των οπών προς την άλλη επιφάνεια).

ΣΦΑΙΡΙΚΗ ΕΙΚΟΝΑ

Ένα βασικό PCB αποτελείται από ένα επίπεδο φύλλο μονωτικού υλικού και ένα στρώμα χαλκού φύλλου, που είναι πλαστικοποιημένο στο υπόστρωμα. Η χημική χάραξη χωρίζει τον χαλκό σε ξεχωριστές αγώγιμες γραμμές που ονομάζονται ίχνη ή ίχνη κυκλώματος, μαξιλάρια για συνδέσεις, σημεία για διέλευση συνδέσεων μεταξύ στρωμάτων χαλκού και χαρακτηριστικά όπως συμπαγείς αγώγιμες περιοχές για ηλεκτρομαγνητική θωράκιση ή άλλους σκοπούς.Τα κομμάτια λειτουργούν ως καλώδια σταθερά στη θέση τους, και μονώνονται μεταξύ τους από τον αέρα και το υλικό του υποστρώματος. Η επιφάνεια ενός PCB μπορεί να έχει μια επίστρωση που προστατεύει το χαλκό από τη διάβρωση και μειώνει τις πιθανότητες των solder shorts μεταξύ ιχνών ή ανεπιθύμητης ηλεκτρικής επαφής με αδέσποτα γυμνά καλώδια. Λόγω της λειτουργίας του στην αποτροπή των

solder shorts, η επικάλυψη ονομάζεται κολλητική αντίσταση ή κολλητική μάσκα.(solder resist ή solder mask)
Μια πλακέτα τυπωμένου κυκλώματος μπορεί να έχει πολλαπλά στρώματα χαλκού. Ένας πίνακας δύο στρωμάτων έχει χαλκό και στις δύο πλευρές. πολυεπίπεδες πλακέτες που κάνουν σάντουιτς επιπλέον στρώματα χαλκού μεταξύ στρώσεων μονωτικού υλικού. Οι αγωγοί σε διαφορετικά στρώματα συνδέονται με vias, οι οποίες είναι χάλκινες οπές που λειτουργούν ως ηλεκτρικές σήραγγες μέσω του μονωτικού υποστρώματος. Τα καλώδια διαμέσου οπών μερικές φορές λειτουργούν επίσης αποτελεσματικά ως vias. Μετά τα PCB δύο επιπέδων, το επόμενο βήμα είναι συνήθως τέσσερα στρώματα. Συχνά δύο στρώματα αφιερώνονται ως τροφοδοσία και επίπεδα γείωσης, και τα άλλα δύο χρησιμοποιούνται για την καλωδίωση σήματος μεταξύ των εξαρτημάτων.
Τα εξαρτήματα "μέσω οπών" (Through hole) στερεώνονται από τα καλώδια τους που περνούν από την πλακέτα και συγκολλούνται σε ίχνη στην άλλη πλευρά. Τα εξαρτήματα "επιφανειακής βάσης" συνδέονται με τους αγωγούς τους σε ίχνη χαλκού στην ίδια πλευρά της πλακέτας. Ένας πίνακας μπορεί να χρησιμοποιήσει και τις δύο μεθόδους για την τοποθέτηση εξαρτημάτων. Τα PCB με μόνο εξαρτήματα που έχουν τοποθετηθεί μέσω οπών είναι πλέον ασυνήθιστα. Η τοποθέτηση επιφάνειας χρησιμοποιείται για τρανζίστορ, διόδους, τσιπ IC, αντιστάσεις και πυκνωτές. Η τοποθέτηση μέσω οπών (Through hole) μπορεί να χρησιμοποιηθεί για ορισμένα μεγάλα εξαρτήματα, όπως ηλεκτρολυτικοί πυκνωτές και σύνδεσμοι.
Το σχέδιο που χαράσσεται σε κάθε στρώση χαλκού ενός PCB ονομάζεται "έργο τέχνης" (artwork). Η χάραξη γίνεται συνήθως με χρήση φωτοανθεκτικού που επικαλύπτεται στο PCB, και στη συνέχεια εκτίθεται σε φως που προβάλλεται στο σχέδιο του έργου τέχνης (artwork). Το υλικό αντίστασης προστατεύει τον χαλκό από τη διάλυση στο διάλυμα χάραξης. Κατόπιν καθαρίζεται η χαραγμένη πλακέτα. Ένας σχεδιασμός PCB μπορεί να αναπαραχθεί μαζικά με τρόπο παρόμοιο με τον τρόπο όπου οι φωτογραφίες μπορούν να αναπαραχθούν μαζικά από αρνητικά φιλμ χρησιμοποιώντας έναν φωτογραφικό εκτυπωτή.
Σε πολυστρωματικές πλακέτες, τα στρώματα του υλικού στρωματοποιούνται μαζί σε ένα εναλλασσόμενο σάντουιτς: χαλκός, υπόστρωμα, χαλκός, υπόστρωμα, χαλκός κ.λπ. Κάθε επίπεδο χαλκού είναι χαραγμένο και κάθε εσωτερική vias (που δεν θα εκτείνεται και στις δύο εξωτερικές επιφάνειες της τελικής πλατφόρμας πολλαπλών στρωμάτων) επικαλύπτεται, πριν τα στρώματα συγκολληθούν μαζί. Τα εσωτερικά στρώματα χαλκού προστατεύονται από τα παρακείμενα στρώματα υποστρώματος.
Το εποξειδικό γυαλί FR-4 είναι το πιο κοινό μονωτικό υπόστρωμα. Ένα άλλο υλικό υποστρώματος είναι βαμβακερό χαρτί εμποτισμένο με φαινολική ρητίνη, συχνά καφέ.
Όταν ένα PCB δεν έχει εγκατεστημένα εξαρτήματα, ονομάζεται τυπωμένη πλακέτα καλωδίωσης (printed wiring board, PWB) ή χαραγμένη πλακέτα καλωδίωσης (etched wiring board). Ωστόσο, ο όρος "τυπωμένη πλακέτα καλωδίωσης" δεν έχει χρησιμοποιηθεί. Ένα PCB με ηλεκτρονικά εξαρτήματα ονομάζεται συγκρότημα τυπωμένου κυκλώματος (printed circuit assembly , PCA), συγκρότημα πλακέτας τυπωμένου κυκλώματος (printed circuit board assembly) ή συγκρότημα PCB (PCB assembly, PCBA). Στην άτυπη χρήση, ο όρος "πλακέτα τυπωμένου κυκλώματος" συνήθως σημαίνει "συγκρότημα τυπωμένου κυκλώματος" (με εξαρτήματα). Ο προτιμώμενος όρος IPC για συναρμολογημένες πλακέτες είναι το συγκρότημα κάρτας

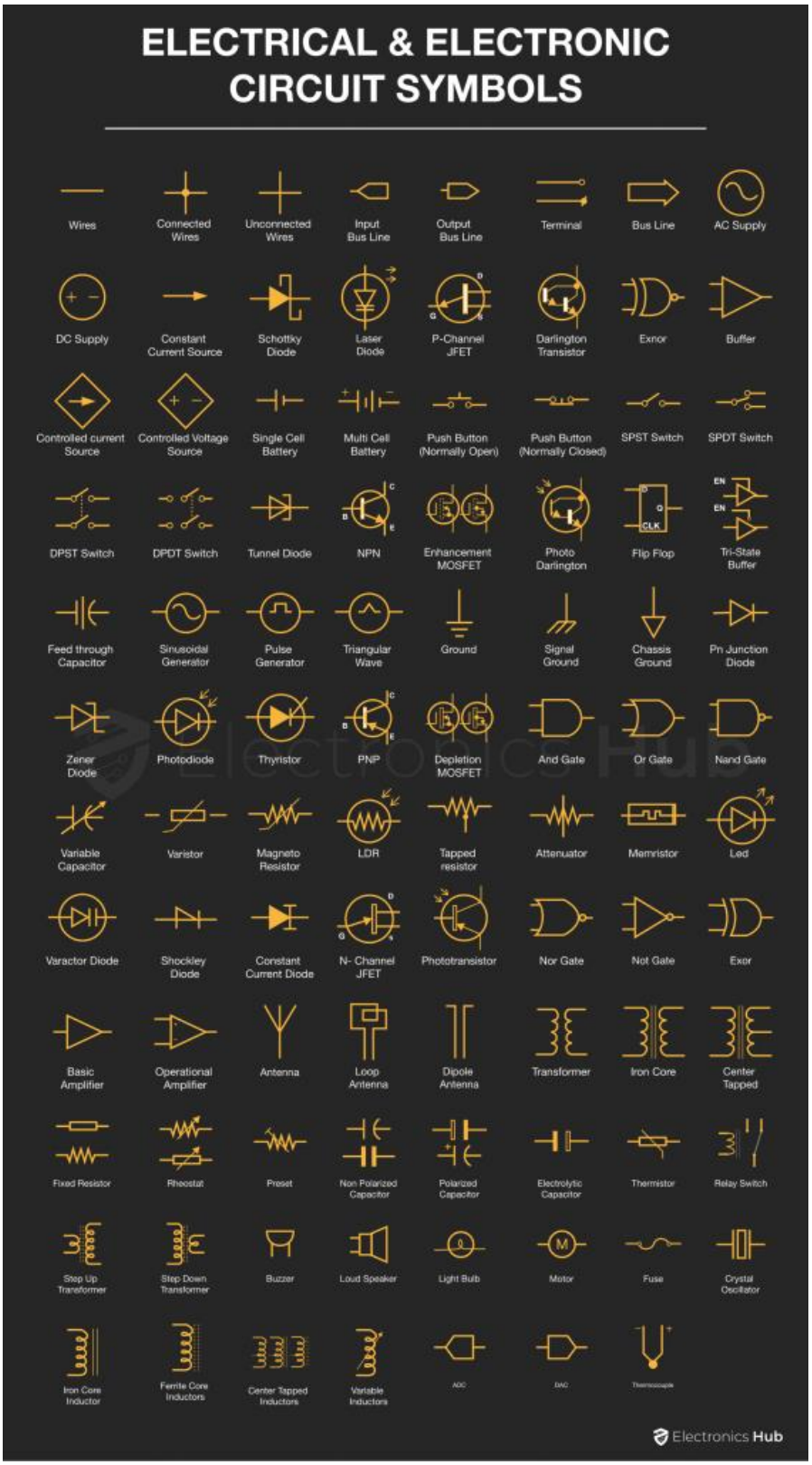
κυκλώματος (circuit card assembly, CCA), και για συναρμολογημένα πλαϊνά επίπεδα είναι συγκροτήματα πλάτους (backplane assemblies).
Το "Κάρτα" ("Card") είναι ένας άλλος ευρέως χρησιμοποιούμενος άτυπος όρος για ένα "συγκρότημα τυπωμένων κυκλωμάτων". Για παράδειγμα, κάρτα επέκτασης.

3.2 Διάγραμμα κυκλώματος (Circuit diagram) [25] [26]

Ένα διάγραμμα κυκλώματος (ηλεκτρικό διάγραμμα, στοιχειώδες διάγραμμα, ηλεκτρονικό σχηματικό) είναι μια γραφική αναπαράσταση ενός ηλεκτρικού κυκλώματος. Ένα εικονογραφικό διάγραμμα κυκλώματος χρησιμοποιεί απλές εικόνες συστατικών, ενώ ένα σχηματικό διάγραμμα δείχνει τα στοιχεία και τις διασυνδέσεις του κυκλώματος χρησιμοποιώντας τυποποιημένες συμβολικές αναπαραστάσεις. Η παρουσίαση των διασυνδέσεων μεταξύ εξαρτημάτων κυκλώματος στο σχηματικό διάγραμμα δεν αντιστοιχεί απαραίτητα στις φυσικές διευθετήσεις της τελικής συσκευής. Σε αντίθεση με ένα διάγραμμα μπλοκ ή διάγραμμα διάταξης, ένα διάγραμμα κυκλώματος δείχνει τις πραγματικές ηλεκτρικές συνδέσεις. Ένα σχέδιο που προορίζεται να απεικονίσει τη φυσική διάταξη των καλωδίων και των εξαρτημάτων που συνδέουν ονομάζεται έργο τέχνης ή διάταξη, φυσικό σχέδιο ή διάγραμμα καλωδίωσης. Τα διαγράμματα κυκλώματος χρησιμοποιούνται για το σχεδιασμό (σχεδιασμό κυκλώματος), την κατασκευή (όπως διάταξη PCB) και τη συντήρηση ηλεκτρικού και ηλεκτρονικού εξοπλισμού.

Σύμβολα

Τα διαγράμματα κυκλώματος είναι εικόνες με σύμβολα που διαφέρουν από χώρα σε χώρα και έχουν αλλάξει με την πάροδο του χρόνου, αλλά τώρα σε μεγάλο βαθμό είναι διεθνώς τυποποιημένα. Παρακάτω παρουσιάζονται κάποια από τα πιο συνηθισμένα σύμβολα .



Εικόνα 27: Schematic Symbols

3.3 Μορφή Gerber (Gerber format) [27]

Το Gerber format είναι μια ανοιχτή διανυσματική μορφή ASCII για σχέδια πλακέτας τυπωμένου κυκλώματος (PCB). Είναι το de facto πρότυπο που χρησιμοποιείται από το λογισμικό της βιομηχανίας PCB για την περιγραφή των εικόνων της πλακέτας τυπωμένου κυκλώματος. Ο επίσημος ιστότοπος περιέχει τις προδιαγραφές, τα δοκιμαστικά αρχεία, τις σημειώσεις και το Reference Gerber Viewer για την υποστήριξη χρηστών και ιδιαίτερα προγραμματιστών λογισμικού Gerber.

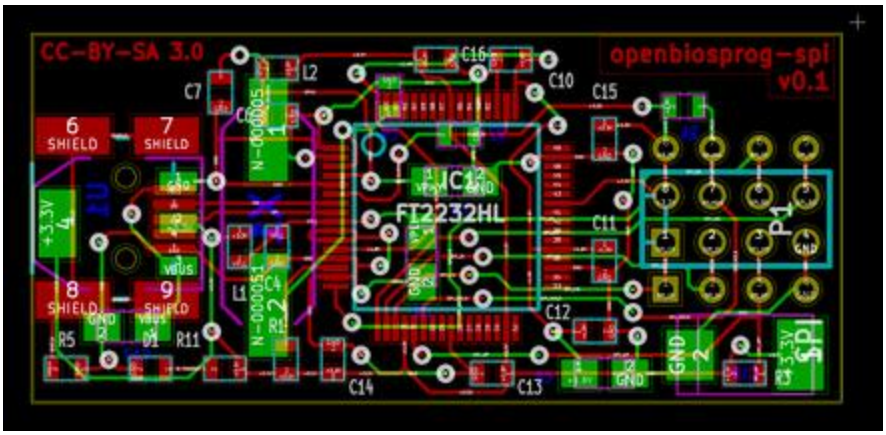
Υπάρχουν δύο μεγάλες γενιές Gerber format :

- Extended Gerber, ή RS-274X. Αυτή είναι η τρέχουσα μορφή Gerber. Το 2014, η μορφή γραφικών επεκτάθηκε με την επιλογή προσθήκης μετα-πληροφοριών στα αντικείμενα γραφικών. Τα αρχεία με χαρακτηριστικά ονομάζονται αρχεία X2 ενώ αυτά χωρίς χαρακτηριστικά είναι αρχεία X1.
- Standard Gerber, ή RS-274-D. Αυτή η παρωχημένη μορφή ανακλήθηκε.

Η τυπική επέκταση αρχείου είναι .GBR ή .gbr αν και χρησιμοποιούνται και άλλες επεκτάσεις.

Συνήθως, όλα αυτά τα αρχεία "συμπιέζονται" σε ένα αρχείο που αποστέλλεται στο κατάστημα κατασκευής PCB. Ο κατασκευαστής τα φορτώνει σε ένα σύστημα κατασκευής με τη βοήθεια υπολογιστή (computer-aided manufacturing, CAM) για την προετοιμασία δεδομένων για κάθε βήμα της διαδικασίας παραγωγής PCB.

Το χαρακτηριστικό .FileFunction είναι η τυποποιημένη μέθοδος για τη σύνδεση κάθε επιπέδου στο PCB με το αντίστοιχο αρχείο Gerber στα δεδομένα κατασκευής. Εάν τα χαρακτηριστικά δεν υποστηρίζονται, διατίθενται μόνο άτυπες μέθοδοι. Μια απλή άτυπη μέθοδος είναι να εκφράσετε με σαφήνεια τη συνάστηση του αρχείου στο όνομα του αρχείου. Μερικές φορές κατάχρηση της επέκτασης αρχείου υποδεικνύει τη συνάρτηση του αρχείου - π.χ. .BOT για το κάτω επίπεδο αντί για την τυπική επέκταση .GBR. Στη βιομηχανία αυτό θεωρείται κακή πρακτική και οι μηχανικοί θα πρέπει να χρησιμοποιούν το κατάλληλο χαρακτηριστικό X2.



Εικόνα 28: Αρχείο Gerber

3.4 Bill of materials (BOM) [28]

Ένας λογαριασμός υλικών ή μια δομή προϊόντος (μερικές φορές λογαριασμός υλικού, BOM ή σχετικός κατάλογος) είναι μια λίστα με τις πρώτες ύλες, τα υποσυστήματα, τα ενδιάμεσα συστήματα, τα υποσυστατικά, τα μέρη και τις ποσότητες από αυτά που χρειάζονται για την κατασκευή ενός τελικού προϊόντος . Ένα BOM μπορεί να χρησιμοποιηθεί για επικοινωνία μεταξύ κατασκευαστικών εταίρων ή μπορεί και να περιοριστεί σε ένα μόνο εργοστάσιο παραγωγής. Ένας λογαριασμός υλικών συνδέεται συχνά με μια εντολή παραγωγής της οποίας η έκδοση μπορεί να δημιουργήσει κρατήσεις για εξαρτήματα στον λογαριασμό υλικών που βρίσκονται σε απόθεμα και αιτήσεις για συστατικά που δεν είναι διαθέσιμα. Υπάρχουν δύο τύποι λογαριασμών υλικού.

Ένα BOM μπορεί να ορίσει προϊόντα όπως είναι σχεδιασμένα (λογαριασμός υλικών μηχανικής), όπως παραγγέλλονται (λογαριασμός πωλήσεων υλικών), καθώς κατασκευάζονται (λογαριασμός κατασκευής υλικών), ή καθώς διατηρούνται (λογαριασμός υλικών υπηρεσιών) Οι διαφορετικοί τύποι εξαρτώνται από την επιχειρηματική ανάγκη και τη χρήση για την οποία προορίζονται. Στις βιομηχανίες διεργασιών, το BOM είναι επίσης γνωστό ως τύπος, συνταγή ή λίστα συστατικών. Η φράση "bill of material" (ή "BOM") χρησιμοποιείται συχνά από μηχανικούς για να μην αναφέρεται στον κυριολεκτικό λογαριασμό, αλλά στην τρέχουσα διαμόρφωση παραγωγής ενός προϊόντος, για να το διακρίνει από τροποποιημένες ή βελτιωμένες εκδόσεις υπό μελέτη ή σε δοκιμή.

Μερικές φορές ο όρος "ψευδό-λογαριασμός υλικών("pseudo-bill of materials")" ή "ψευδο-BOM" χρησιμοποιείται για να αναφέρεται σε μια πιο ευέλικτη ή απλοποιημένη έκδοση. Συχνά χρησιμοποιείται ένας αριθμός ανταλλακτικού για να αντιπροσωπεύει μια

ομάδα σχετικών (συνήθως τυπικών) μερών που έχουν κοινά χαρακτηριστικά και είναι εναλλάξιμα στο πλαίσιο αυτού του BOM.

Στα ηλεκτρονικά, το BOM αντιπροσωπεύει τη λίστα των εξαρτημάτων που χρησιμοποιούνται στην τυπωμένη πλακέτα καλωδίων ή στην πλακέτα τυπωμένων κυκλωμάτων. Μόλις ολοκληρωθεί ο σχεδιασμός του κυκλώματος, η λίστα BOM μεταβιβάζεται στον μηχανικό διάταξης PCB καθώς και στον μηχανικό εξαρτημάτων που θα προμηθεύσει τα εξαρτήματα που απαιτούνται για το σχεδιασμό.

ΚΕΦΑΛΑΙΟ 4
Εργαλεία ανάπτυξης

4.1 Platform IO [29]

Το PlatformIO είναι ένα cross-platform, cross-architecture, πολλαπλού πλαισίου, επαγγελματικό εργαλείο για μηχανικούς ενσωματωμένων συστημάτων και για προγραμματιστές λογισμικού που γράφουν εφαρμογές για ενσωματωμένα προϊόντα.

Εικόνα 29: Λογότυπο Platform IO

Φιλοσοφία

Η μοναδική φιλοσοφία του PlatformIO στην ενσωματωμένη αγορά παρέχει στους προγραμματιστές ένα σύγχρονο ολοκληρωμένο περιβάλλον ανάπτυξης (Cloud & Desktop IDE) που λειτουργεί σε πολλαπλές πλατφόρμες, υποστηρίζει πολλά διαφορετικά κιτ ανάπτυξης λογισμικού (SDK) ή Frameworks και περιλαμβάνει εξελιγμένο debugging, unit testing, αυτόματη ανάλυση κώδικα (Static Code Analysis) και απομακρυσμένη διαχείριση (Remote Development). Είναι σχεδιασμένο για να μεγιστοποιεί την ευελιξία και την επιλογή από προγραμματιστές. Το PlatformIO είναι ένα απαραίτητο εργαλείο για επαγγελματίες μηχανικούς ενσωματωμένων συστημάτων που αναπτύσσουν λύσεις σε περισσότερες από μία συγκεκριμένες πλατφόρμες.

Τεχνολογίες

Το PlatformIO εφαρμόζει την πιο πρόσφατη επεκτάσιμη και ευέλικτη τεχνολογία λογισμικού στην ενσωματωμένη αγορά - μια περιοχή που παραδοσιακά εξυπηρετείται από πολύπλοκα εργαλεία λογισμικού που έχουν μάθει έμπειροι μηχανικοί υλικού με την πάροδο του χρόνου (συχνά οδυνηρά). Αντ 'αυτού, με το PlatformIO, οι χρήστες μπορούν να είναι χομπίστες ή επαγγελματίες. Μπορούν να εισαγάγουν το κλασικό

σκίτσο Arduino "Blink" (Arduino “Blink” sketch) ή να αναπτύξουν ένα εξελιγμένο ενσωματωμένο πρόγραμμα C χαμηλού επιπέδου (low-level embedded C program) για ένα εμπορικό προϊόν. Παράδειγμα κώδικα για οποιοδήποτε υποστηριζόμενο πλαίσιο (Example code for any supported framework) μπορεί να μεταγλωττιστεί και να μεταφορτωθεί σε μια πλατφόρμα (target platform) μέσα σε λίγα λεπτά.

Η δομή του συστήματος δόμησης (build system structure) επισημαίνει αυτόματα τα software dependencies και τις εφαρμόζει χρησιμοποιώντας μια αρθρωτή ιεραρχία (modular hierarchy) που αφαιρεί τη συνήθη πολυπλοκότητα. Οι προγραμματιστές δεν χρειάζεται πλέον να βρίσκουν και να συναρμολογούν χειροκίνητα ένα περιβάλλον εργαλείων, μεταγλωττιστών και εξαρτήσεων βιβλιοθηκών (toolchains, compilers, and library dependencies) για να αναπτύξουν εφαρμογές για έναν συγκεκριμένο στόχο. Με το PlatformIO, κάνοντας κλικ στο κουμπί μεταγλώττισης θα εμφανιστούν αυτόματα όλες οι απαραίτητες εξαρτήσεις (dependencies).

Πώς λειτουργεί

Χωρίς να εμβαθύνουμε πολύ στις λεπτομέρειες υλοποίησης του PlatformIO, ο κύκλος εργασίας του project που αναπτύχθηκε χρησιμοποιώντας το PlatformIO έχει ως εξής:

- Οι χρήστες επιλέγουν πίνακες που ενδιαφέρονται για το "platformio.ini" (Project Configuration File)
- Βάσει αυτής της λίστας πινάκων, το PlatformIO κατεβάζει τις απαραίτητες αλυσίδες εργαλείων (toolchains) και τις εγκαθιστά αυτόματα.

- Οι χρήστες αναπτύσσουν κώδικα και το PlatformIO διασφαλίζει ότι έχει μεταγλωττιστεί, προετοιμαστεί και μεταφορτωθεί σε όλους τους πίνακες ενδιαφέροντος.

4.2 EasyEDA [30]

Το EasyEDA είναι μια διαδικτυακή σουίτα εργαλείων EDA που επιτρέπει στους μηχανικούς υλικού (hardware engineers) να σχεδιάζουν, να προσομοιώνουν, να μοιράζονται - δημόσια και ιδιωτικά - και να συζητούν σχήματα (schematics), προσομοιώσεις και πίνακες τυπωμένων κυκλωμάτων. Άλλα χαρακτηριστικά περιλαμβάνουν τη δημιουργία ενός λογαριασμού υλικών, αρχείων Gerber, επιλογής και τοποθέτησης αρχείων και εξόδων σε μορφές PDF, PNG και SVG.

Κατασκευαστής	EasyEDA
Αρχική Έκδοση	Άυγουστος 2013.
Stable Release	6.4.19.2 / 29 Μαρτίου, 2021.
Πλατφόρμες	Linux, macOS, Windows
Διαθέσημο σε	13 γλώσσες
Τύπος	EDA software
Άδεια	Commercial software
Ιστοσελίδα	easyeda.com

Η EasyEDA επιτρέπει τη δημιουργία και επεξεργασία σχηματικών διαγραμμάτων, την προσομοίωση SPICE μικτών αναλογικών και ψηφιακών κυκλωμάτων (SPICE simulator of mixed analogue and digital circuits), καθώς και την δημιουργία και επεξεργασία διατάξεων πλακέτας τυπωμένων κυκλωμάτων, και προαιρετικά την κατασκευή των πλακετών τυπωμένου κυκλώματος.

Η δωρεάν συνδρομή προσφέρεται για δημόσιο και για περιορισμένο αριθμό ιδιωτικών έργων. Αυτός ο αριθμός μπορεί να αυξηθεί συνεισφέροντας δημόσια έργα υψηλής ποιότητας, σχηματικά σύμβολα και ίχνη PCB (PCB footprints) και / ή πληρώνοντας μια μηνιαία συνδρομή.

Οι εγγεγραμμένοι χρήστες μπορούν να κατεβάσουν δωρεάν αρχεία Gerber από το εργαλείο. Όμως αν είναι με χρέωση, η EasyEDA προσφέρει υπηρεσία κατασκευής PCB. Αυτή η υπηρεσία μπορεί επίσης να δέχεται εισόδους αρχείων Gerber από εργαλεία τρίτων.

Εικόνα 30: Λογότυπο EasyEDA

ΙΣΤΟΡΙΚΗ ΑΝΑΔΡΟΜΗ

Τον Ιούνιο του 2010, ο Dillon He και ο Eric Cui ξεκίνησαν μια έρευνα για τα εργαλεία EDA, για ένα δικό τους hardware project. Η αναζήτηση ενός εργαλείου EDA το οποίο έπρεπε να ήταν ικανό για design capture, προσομοίωση κυκλώματος και διάταξη τυπωμένου κυκλώματος σημείωσε περιορισμένη επιτυχία και ο He κατέληξε στο συμπέρασμα ότι ο σχεδιασμός του δικού τους εργαλείου EDA θα ήταν επωφελής.

Βασικοί τους στόχοι ήταν τα εργαλεία να είναι ανεξάρτητα απο την πλατφόρμα, δωρεάν, εύκολα στην μάθηση και στην χρήση.

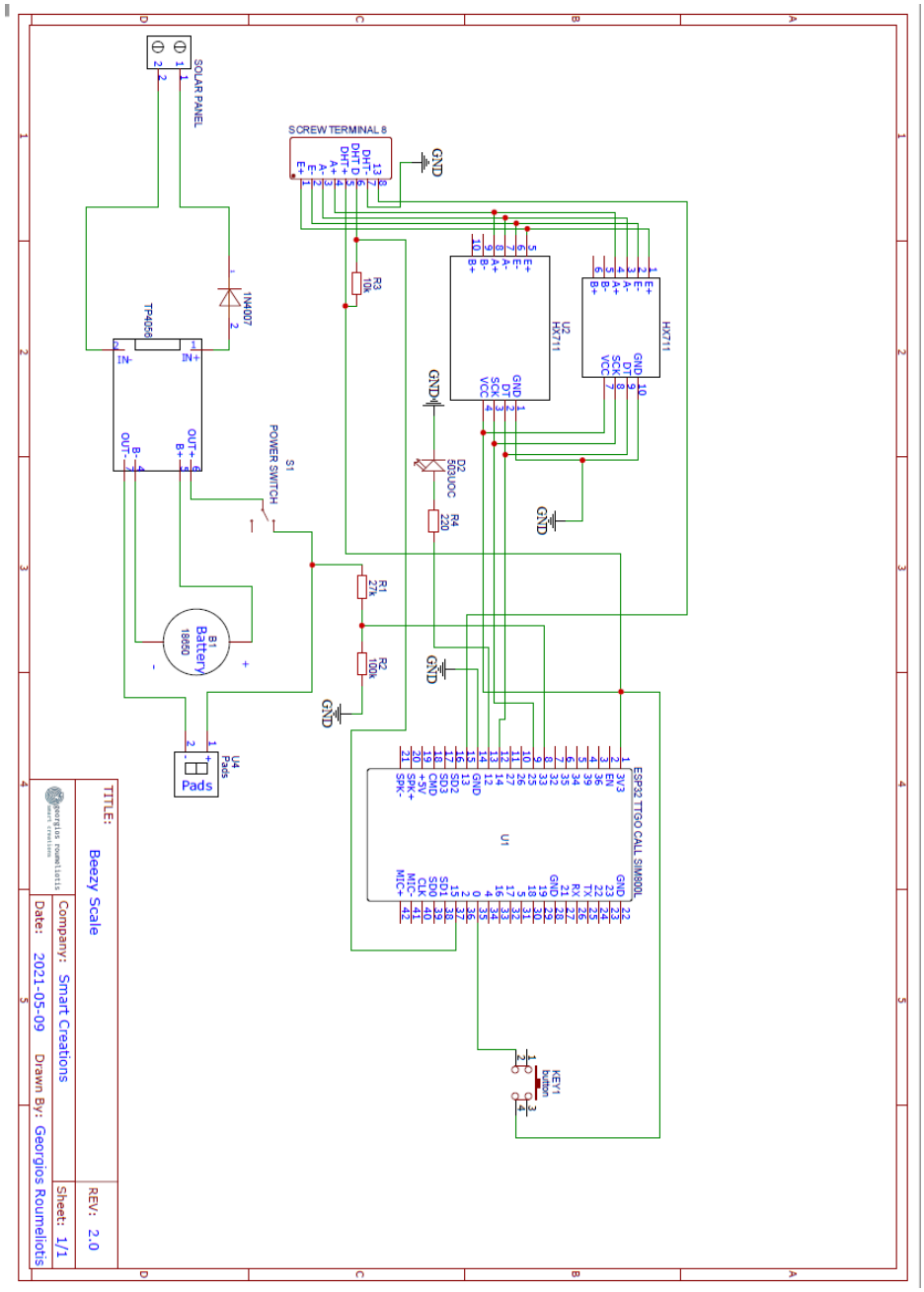
Η πρώτη έκδοση του εργαλείου εμφανίστηκε στο Διαδίκτυο τον Άυγουστο του 2013. Κυκλοφόρησε επίσημα τον Μάρτιο του 2014 και από τότε αναπτύχθηκε ραγδαία με πολλές σημαντικές αναθεωρήσεις και βελτιώσεις.

ΚΕΦΑΛΑΙΟ 5
Υλοποίηση

5.1 Λογαριασμός Υλικών

TTGO T-Call ESP32 with SIM800L GPRS Module	10.00
Load Cell Amplifier - HX711 (Soldered Header)	1.90
Humidity and Temperature Sensor - AM2302 (DHT22)	6.90
Load Cell - 100kg	14.90
Lithium Battery Charger and Protection Module 1A - TP4056	1.60
Interface Cable SMA Female to U.FL - 20cm	2.20
LED Diffused 5mm Red	0.08
Battery Holder 1x18650 for PCB	0.90
Battery Lithium 18650 3.7V 3350mAh - LG INR18650-3.35	6.20
Tact Switch 6x6mm 5mm 4pins	0.08
Slide Switch Mini SPDT - 0.3A/30VDC	0.20
Screw Terminal 2P 5mm	0.20
Screw Terminal 8P 2.54mm	1.80
Diode Rectifier - 1A 1000V 1N4007	0.02
Carbon Resistor 1/4W 5% 100Kohm	0.02
Carbon Resistor 1/4W 5% 10Kohm	0.02
Carbon Resistor 1/4W 5% 27Kohm	0.02
Carbon Resistor 1/4W 5% 220ohm	0.02
Project Box 105x105x60mm Grey	5.80
Solar Panel 3.5W 165x135mm	12.90
2x Cable Gland M12 - Light Grey	1.00
GSM Antenna 900-1800MHZ - 3M Meter with SMA Connector	2.01
ΣΥΝΟΛΟ	68.86

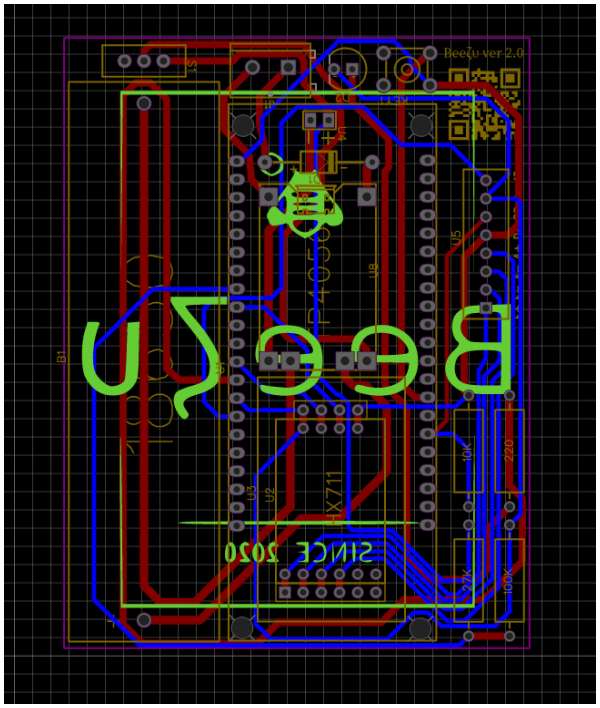
5.2 Σχηματικό Διάγραμμα



5.3 Gerber File

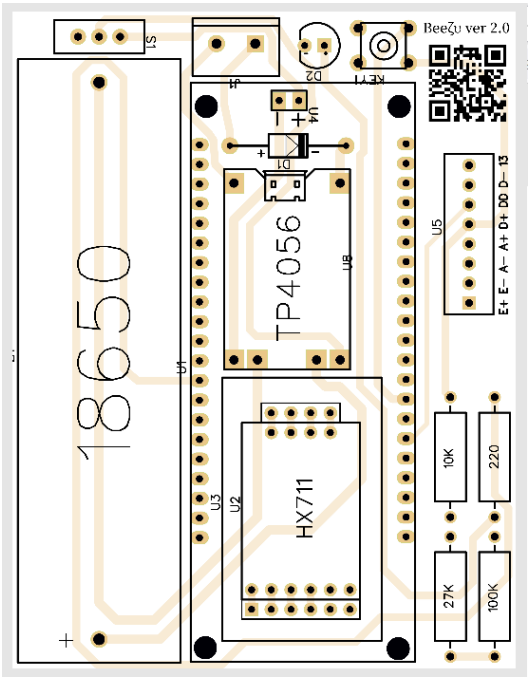
Gerber_Beezy Scale v2_2021-05-13.zip

Gerber_BoardOutline.GKO
Gerber_BottomLayer.GBL
Gerber_BottomSilkLayer.GBO
Gerber_BottomSolderMaskLayer.GBS
Gerber_Drill_NPTH.DRL
Gerber_Drill_PTH.DRL
Gerber_TopLayer.GTL
Gerber_TopSilkLayer.GTO
Gerber_TopSolderMaskLayer.GTS
How-to-order-PCB.txt

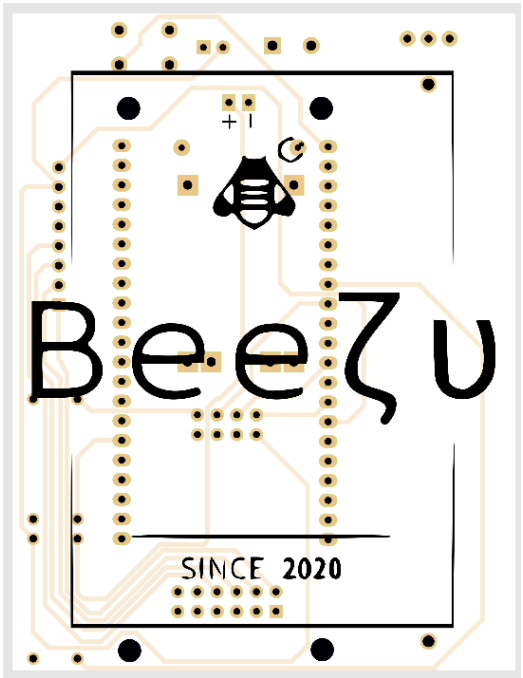


Εικόνα 31: Αναπαράσταση Gerber File

5.4 PCB Layout



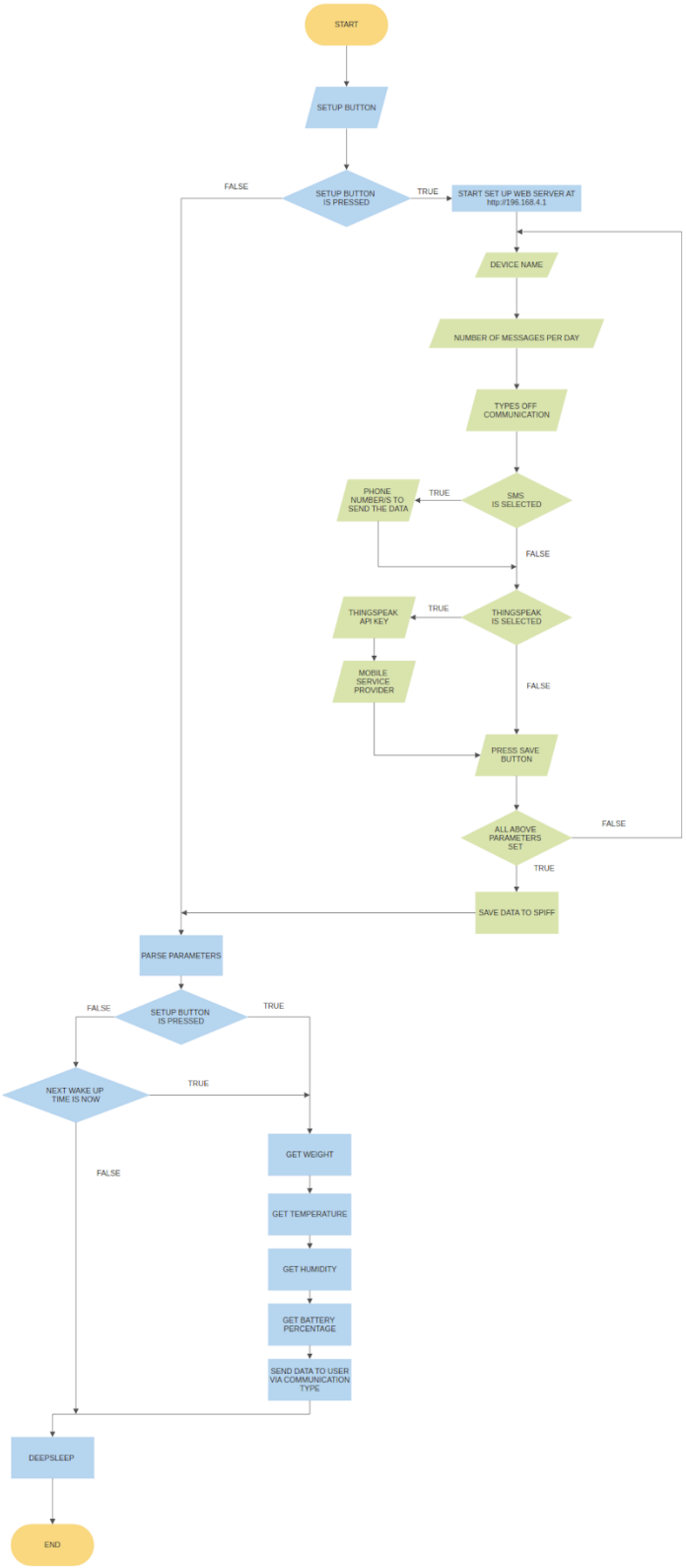
Εικόνα 32: Μπροστά μέρος PCB



Εικόνα 33: Πίσω μέρος PCB

5.5 Κώδικας [31] [32] [33] [34] [35] [36]

5.5.1 Διάγραμμα ροής



5.5.2 main.cpp

```
/*
*
*
*          BEEZY
*
*     BEE HIVE SCALE
*
*
*
* @author : Georgios Roumeliotis
* @version : 2.5.0
*
*/


/* DEFINES && CONST --> */

/* 1 MINUTE IN MICROSECONDS --> */
#define MINUTES 60000000
/* <-- 1 MINUTE IN MICROSECONDS */


/* ESP32 PINS --> */
#define LOADCELL_DOUT_PIN  14
#define LOADCELL_SCK_PIN  25
#define SETUP_PIN 0
#define INDICATOR 12
#define BATTERY_PIN 33
#define DHTPIN 15
#define DHTTYPE DHT22
/* <-- ESP32 PINS */

/* TTGO T-Call pins */
#define MODEM_RST            5
#define MODEM_PWKEY          4
#define MODEM_POWER_ON       23
#define MODEM_TX             27
#define MODEM_RX             26
#define I2C_SDA              21
#define I2C_SCL              22
/**/

/* Configure TinyGSM library */
#define TINY_GSM_MODEM_SIM800      /* Modem is SIM800 */
#define TINY_GSM_RX_BUFFER   1024  /* Set RX buffer to 1Kb */
```

```
/**/

/* Set serial for debug console (to Serial Monitor, default speed 115200) */
#define SerialMon Serial
/* Set serial for AT commands (to SIM800 module) */
#define SerialAT  Serial1

#define IP5306_ADDR          0x75
#define IP5306_REG_SYS_CTL0  0x00

/* This value is obtained using the SparkFun_HX711_Calibration sketch */
#define calibration_factor -5124
/* This large value is obtained using the SparkFun_HX711_Calibration sketch */
#define zero_factor 8315300

const char* ssid = "Beezy";
const char* password = "B3EZY!";
/* <-- DEFINES && CONST */

/* LIBRARIES --> */
#include <HX711.h> /* https://github.com/bogde/HX711 */
#include <soc/rtc.h>
#include <Wire.h>
#include <TinyGsmClient.h> /* https://github.com/vshymanskyy/TinyGSM */
#include <Adafruit_Sensor.h> /* https://github.com/adafruit/Adafruit_Sensor */
#include <DHT.h>   /* https://github.com/adafruit/DHT-sensor-library/blob/master/DHT.h */
/* <-- LIBRARIES */

/* MY LIBRARIES & HEADERS */
#include <BeezyManagerWebpage.h>
/* <-- MY LIBRARIES & HEADERS */

/* GLOBAL --> */
int SETUP_MODE = false;

String LOCATION = "";
String THINGSPEAK = "";
String PHONE = "";
String MESSAGES = "";
String APN = "";
String PREFIX = "";

int DEEP_SLEEP_MINS = 1;

float SCALE_WEIGHT = 0.0;
```

```
float HUMIDITY = 0.0;
float TEMPERATURE = 0.0;
float BATTERY_PER = 0.0;
/* <-- GLOBAL*/

/* OBJECTS DECLARATION --> */
TinyGsm modem(SerialAT);
HX711 scale;
TinyGsmClient client(modem);
DHT dht = DHT(DHTPIN, DHTTYPE);
/* <-- OBJECTS DECLARATION */

/* FUNCTIONS DECLARATION --> */
void indicator(int pin,int tempo);
bool setup_button(int pin, size_t time_to_wait);
bool save_data_to_file(String data, String file_path);
String get_data_from_file(String file_path);
void parse_data_to_global(String data);
void print_global(void);
bool iniialize_sim800l(String apn);
String sim800l_get_time(void);
bool setPowerBoostKeepOn(int en);
int next_message_wakeup(int messages, String time);
void deepsleep_handler(uint next_wakeup, uint messages);
bool send_sms(String target, String message);
String build_message(String name, int add_weight, int weight, float temperature, float humidity, float batterty_per );
bool upload_data_to_thingspeak(String api_key, int weight, float temperature, float humidity, float battery);
float last_scale_mesure(float w);
/* <-- FUNCTIONS DECLARATION */

/* SETUP FUNCTION --> */
void setup(){

  /* INITIALIAZE UART SERIAL COMMUNICATION WITH 115200 BAUDRATE -->*/
  Serial.begin(115200);
  /* <-- INITIALIAZE UART SERIAL COMMUNICATION WITH 115200 BAUDRATE */

  /* CONFIGURE PINS STATE --> */
  pinMode(SETUP_PIN,INPUT_PULLDOWN);
  pinMode(INDICATOR,OUTPUT);
  /* <-- CONFIGURE PINS STATE */

  /* BLINK LED ONCE FOR ONE SECOND --> */
  indicator(INDICATOR,1000);
  /* <-- BLINK LED ONCE FOR ONE SECOND */
```

```
/* WAIT FOR UART COMMUNICATION TO ESTABLISHED --> */
while(!Serial) { }
/* <-- WAIT FOR UART COMMUNICATION TO ESTABLISHED */

Serial.println("");
Serial.println("\n[+] --> JUST WAKE UP FROM DEEPSLEEP ");
Serial.println("\n[+]--> SPIFFS INITIALIZE");


/* INITIALIZE SPI FLASH FILE SYSTEM --> */
if(!SPIFFS.begin(true)){
  Serial.println("AN ERROR HAS OCCURED WHILE MOUNTING SPIFFS");
}
/* <-- INITIALIZE SPI FLASH FILE SYSTEM */

/* WAIT ONE SECOND --> */
delay(1000);
/* <-- WAIT ONE SECOND */

/* CHECK UP TO 5 SECONDS IF USER PRESS THE SET UP BUTTON --> */
SETUP_MODE = setup_button(SETUP_PIN,5);
/* <-- CHECK UP TO 5 SECONDS IF USER PRESS THE SET UP BUTTON */


Serial.println(String("[+]--> SETUP_BUTTON : ") + (SETUP_MODE? "TRUE" : "FALSE"));
Serial.println("\n[+]--> PARSE DATA FROM FILE");


/* GET CONFIGURATION DATA FROM SPIFFS AND PARSE THEM TO THE GLOBAL VARIABLES --> */
parse_data_to_global(get_data_from_file("/data.txt"));
/* <-- GET CONFIGURATION DATA FROM SPIFFS AND PARSE THEM TO THE GLOBAL VARIABLES */

/* IF SETUP BUTTON IS NOT PRESSED START SIM800L AND DEEPSLEEP HANDLER ELSE START CONFIGURATION WEB SERVER --> */
if(!SETUP_MODE){


  Serial.println("\n[+] --> WAIT SIM800L TO CONNECT \n");


  /* INITIALIZE SIM800L AND WAIT TO CONNECT TO THE NETWORK --> */
  iniialize_sim800l(APN);
  /* <-- INITIALIZE SIM800L AND WAIT TO CONNECT TO THE NETWORK */

  /**/
  Serial.println("\n[+]--> DEEP SLEEP HANDLER");
```

```
  /* GET MESSAGES NUMBER FROM THE CONFIGURATION FILE - GET TIME FROM THE SIM800L - CALCULATE WHEN THE NEXT MESSAGE
  IS GOING TO SEND AND PARSE THAT IN THE DEEPSLEEP HANDLER.IF IS NOT THE TIME FOR THE MESSAGE TO BE SEND DEEPSLEEP
  FOR THE REMAINING IF IT IS LESS THAN A HOUR ELSE DEEPSLEEP FOR ONE HOUR --> */
  deepsleep_handler(next_message_wakeup(MESSAGES.toInt(),sim800l_get_time()),MESSAGES.toInt());
  /* <-- GET MESSAGES NUMBER FROM THE CONFIGURATION FILE - GET TIME FROM THE SIM800L - CALCULATE WHEN THE NEXT MESSAGE
  IS GOING TO SEND AND PARSE THAT IN THE DEEPSLEEP HANDLER.IF IS NOT THE TIME FOR THE MESSAGE TO BE SEND DEEPSLEEP
  FOR THE REMAINING IF IT IS LESS THAN A HOUR ELSE DEEPSLEEP FOR ONE HOUR */
}
else{

  Serial.println("[+] --> START SET UP WEB SERVER ");


  /* START CONFIGURATION WEB SERVER AND SAVE CONFIGURATION DATA TO SPIFFS AND RESET LAST MEASURE VALUE --> */
  save_data_to_file(Beezy_Web_Server(ssid,password),"/data.txt");
  save_data_to_file("0.0", "/last_mesure.txt");
  /* <-- START CONFIGURATION WEB SERVER AND SAVE CONFIGURATION DATA TO SPIFFS AND RESET LAST MEASURE VALUE */

  /* SERIAL PRINTS FOR DEBBUGING REASONS --> */
  Serial.println("[+] --> CLOSE SET UP WEB SERVER ");
  /* <-- SERIAL PRINTS FOR DEBBUGING REASONS */

}
/* <-- IF SETUP BUTTON IS NOT PRESSED START SIM800L AND DEEPSLEEP HANDLER ELSE START CONFIGURATION WEB SERVER */

/* SLOW DOWN CPU CLOCK FREQUENCY TO 80 MHZ FOR BETTER READING FROM THE LOADCELL --> */
rtc_clk_cpu_freq_set(RTC_CPU_FREQ_80M);
/* <-- SLOW DOWN CPU CLOCK FREQUENCY TO 80 MHZ FOR BETTER READING FROM THE LOADCELL */

/* SERIAL PRINTS FOR  REASONS --> */
Serial.println("\n[+]--> SENSORS INITIALIZE");
Serial.println("[+]--> INITIALIZING DHT22");
/* <-- SERIAL PRINTS FOR DEBBUGING REASONS */

/* INITIALIZING DHT22 TEMPERATURE AND HUMIDITY SENSOR --> */
dht.begin();
/* INITIALIZING DHT22 TEMPERATURE AND HUMIDITY SENSOR --> */

/* INITIALIZING HX711 SENSOR --> */

Serial.println("[+]--> INITIALIZING SCALE");
scale.begin(LOADCELL_DOUT_PIN, LOADCELL_SCK_PIN);
Serial.println("[+]--> INITIALIZING SET SCALE");
```

```
  /*  THESE VALUES ARE OBTAINED USING THE SPARKFUN_HX711_CALIBRATION SKETCH
  (https://github.com/sparkfun/HX711-Load-Cell-
Amplifier/blob/master/firmware/SparkFun_HX711_Calibration/SparkFun_HX711_Calibration.ino) --> */
  scale.set_scale(calibration_factor);
  Serial.println("[+]--> INITIALIZING SET OFFSET");
  scale.set_offset(zero_factor); /* ZERO OUT THE SCALE USING PREVIOUSLY KNOWN ZERO FACTOR */
  /* <-- THESE VALUES ARE USING THE SPARKFUN_HX711_CALIBRATION SKETCH
  (https://github.com/sparkfun/HX711-Load-Cell-
Amplifier/blob/master/firmware/SparkFun_HX711_Calibration/SparkFun_HX711_Calibration.ino) */

  /* <-- INITIALIZING HX711 SENSOR */

  /* GET CONFIGURATION DATA FROM SPIFFS AND PARSE THEM TO THE GLOBAL VARIABLES --> */
  parse_data_to_global(get_data_from_file("/data.txt"));
  /* <-- GET CONFIGURATION DATA FROM SPIFFS AND PARSE THEM TO THE GLOBAL VARIABLES */

  /* PRINT GLOBAL VARIABLES FOR DEBUGGING --> */
  print_global();
  /* <-- PRINT GLOBAL VARIABLES FOR DEBUGGING */

  Serial.println("\n[+] --> GET READINGS FROM SENSORS  ");

  /* GATHER ALL DATA FROM THE SENSORS --> */
  HUMIDITY = dht.readHumidity();
  TEMPERATURE = dht.readTemperature();
  BATTERY_PER = map(analogRead(BATTERY_PIN), 0.0f, 4095.0f, 0, 100);
  SCALE_WEIGHT = scale.get_units()*453.592; /* TURN LIBS  TO KG */
  /* GATHER ALL DATA FROM THE SENSORS --> */

  /* CALCULATE HOW MUCH WEIGHT IS ADDED SINCE LAST MEASURE --> */
  float SCALE_ADD_WEIGHT = last_scale_mesure(SCALE_WEIGHT);
  /* <-- CALCULATE HOW MUCH WEIGHT IS ADDED SINCE LAST MEASURE */

  Serial.println("\n[+] --> WAIT SIM800L TO CONNECT \n");

  /* INITIALIZE SIM800L AND WAIT TO CONNECT TO THE NETWORK --> */
  iniialize_sim800l(APN);
  /* <-- INITIALIZE SIM800L AND WAIT TO CONNECT TO THE NETWORK */

  /* SEND MESSAGE VIA SMS OR / AND THINGSPEAK DEPENDING ON PREFIX VALUE --> */
  if(PREFIX == "sms" || PREFIX == "sms_thingspeak"){
    while(!send_sms("+30" + PHONE,build_message(LOCATION,SCALE_ADD_WEIGHT,SCALE_WEIGHT,TEMPERATURE,HUMIDITY,BATTERY_PER))
){
      Serial.println("\n[+] --> SENDING SMS ...");
      delay(500);
```

```
    }
    Serial.println("\n[+] --> SMS SEND");
    delay(1000);
  }

  if(PREFIX == "thingspeak" || PREFIX == "sms_thingspeak"){
    while(!upload_data_to_thingspeak(THINGSPEAK,SCALE_WEIGHT,TEMPERATURE,HUMIDITY,BATTERY_PER)){
      Serial.println("\n[+] --> SENDING DATA TO THINGSPEAK ... \n");
      delay(500);
    }
    Serial.println("\n[+] --> DATA TO THINGSPEAK SEND \n");
  }
  /* <-- SEND MESSAGE VIA SMS OR / AND THINGSPEAK DEPENDING ON PREFIX VALUE --> */

  /* CALCULATE NEXT WAKEUP IN MINUTES --> */
  int mins = next_message_wakeup(MESSAGES.toInt(),sim800l_get_time());
  /* <--  CALCULATE NEXT WAKEUP IN MINUTES */

  /* IF NEXT WAKEUP IS MORE THAN 60 MINUTES DEEPSLEEP FOR 60 MINUTES ELSE DEEPSLEEP FOR THE NUMBER OF MINUTES --> */
  if(mins > 60){
    esp_sleep_enable_timer_wakeup(3600000000);
    Serial.println("\n\n[+] --> DEEPSLEEP FOR 60 MINUTES");
  }
  else{
    esp_sleep_enable_timer_wakeup(mins*60000000);
    Serial.println("\n\n[+] --> DEEPSLEEP FOR "+String(mins)+" MINUTES");
  }
  esp_deep_sleep_start();
  /* <-- IF NEXT WAKEUP IS MORE THAN 60 MINUTES DEEPSLEEP FOR 60 MINUTES ELSE DEEPSLEEP FOR THE NUMBER OF MINUTES */

}
/* <-- SETUP FUNCTION */

/* LOOP FUNCTION --> */
void loop(){}
/* <-- LOOP FUNCTION */



/******************************************** CUSTOM FUNCTIONS ************************************************/


/* TOGGLE A PIN FOR A SPECIFIC TIME --> */
void indicator(int pin,int tempo){
  delay(tempo);
```

```
  digitalWrite(pin, HIGH);
  delay(tempo);
  digitalWrite(pin, LOW);
}
/* TOGGLE A PIN FOR A SPECIFIC TIME --> */

/* WAIT UP TO (time_to_wait) SECONDS FOR PIN TO GET HIGH (RETURNS PIN STATUS) --> */
bool setup_button(int pin, size_t time_to_wait){
  int status = 0;
  for (size_t i = 0; i < time_to_wait; i++) {
    status = digitalRead(pin);
    delay(10);
    if(status == 1){
      return status;
    }
    delay(990);
  }

  return status;
}
/* <-- WAIT UP TO (time_to_wait) SECONDS FOR PIN TO GET HIGH (RETURNS PIN STATUS) */

/* SAVE DATA TO A SPECIFIED FILE IN SPIFFS --> */
bool save_data_to_file(String data, String file_path){
  File write = SPIFFS.open(file_path, "w");
  write.print(data);
  write.close();
  return true;
}
/* <-- SAVE DATA TO A SPECIFIED FILE IN SPIFFS */

/*  RETURNS THE DATA OF A SPECIFIED FILE --> */
String get_data_from_file(String file_path){
  File read = SPIFFS.open(file_path, "r");
  String FILE_STRING = "";
  while (read.available()) {
    FILE_STRING +=  read.readString();
  }

  read.close();

  return FILE_STRING;
}
/* <--  RETURNS THE DATA OF A SPECIFIED FILE */
```

```
/* PARSE DATA TO THE GLOBAL VARIABLES --> */
void parse_data_to_global(String data){
  String msg = data;
  byte separator = msg.indexOf('|');
  LOCATION = msg.substring(0,separator);
  msg = msg.substring(separator+1);
  separator = msg.indexOf('|');
  THINGSPEAK = msg.substring(0,separator);
  msg = msg.substring(separator+1);
  separator = msg.indexOf('|');
  PHONE = msg.substring(0,separator);
  msg = msg.substring(separator+1);
  separator = msg.indexOf('|');
  MESSAGES = msg.substring(0,separator);
  msg = msg.substring(separator+1);
  separator = msg.indexOf('|');
  APN = msg.substring(0,separator);
  msg = msg.substring(separator+1);
  separator = msg.indexOf('|');
  PREFIX = msg.substring(0,separator);
}
/* <-- PARSE DATA TO THE GLOBAL VARIABLES */

/* PRINT GLOBAL VARIAL ON THE SERIAL MONITOR --> */
void print_global(void){
  Serial.print("LOCATION : ");
  Serial.println(LOCATION);
  Serial.print("THINGSPEAK : ");
  Serial.println(THINGSPEAK);
  Serial.print("PHONE : ");
  Serial.println(PHONE);
  Serial.print("MESSAGES : ");
  Serial.println(MESSAGES);
  Serial.print("APN : ");
  Serial.println(APN);
  Serial.print("PREFIX : ");
  Serial.println(PREFIX);
}
/* <-- PRINT GLOBAL VARIAL ON THE SERIAL MONITOR */

/* SET POWER  BOOST KEEP ON (ESP32 TTGO CALL USES THE IP5306 BATTERY MANAGER) --> */
bool setPowerBoostKeepOn(int en){
  Wire.beginTransmission(IP5306_ADDR);
  Wire.write(IP5306_REG_SYS_CTL0);
  if (en) {
```

```
    Wire.write(0x37); // Set bit1: 1 enable 0 disable boost keep on
  } else {
    Wire.write(0x35); // 0x37 is default reg value
  }
  return Wire.endTransmission() == 0;
}
/* <-- SET POWER  BOOST KEEP ON (ESP32 TTGO CALL USES THE IP5306 BATTERY MANAGER) */

/* INITIALIZE SIM800L AND START GPRS CONNECTION IF A APN CODE IS PROVIDED --> */
bool iniialize_sim800l(String apn = ""){
  /* INITIALIZE I2C COMMUNICATION FOR THE IP5306 --> */
  Wire.begin(I2C_SDA, I2C_SCL);
  delay(500);
  /* <-- INITIALIZE I2C COMMUNICATION FOR THE IP5306 */

  bool isOk = setPowerBoostKeepOn(1);
  SerialMon.println(String("IP5306 KEEP ON ") + (isOk ? "OK" : "FAIL"));

  /* SET MODEM RESET , ENSABLE AND POWER PINS --> */
  pinMode(MODEM_PWKEY, OUTPUT);
  pinMode(MODEM_RST, OUTPUT);
  pinMode(MODEM_POWER_ON, OUTPUT);
  digitalWrite(MODEM_PWKEY, LOW);
  digitalWrite(MODEM_RST, HIGH);
  digitalWrite(MODEM_POWER_ON, HIGH);
  /* <-- SET MODEM RESET , ENSABLE AND POWER PINS */

  /* SET GSM MODULE BAUDERATE AND UART PINS --> */
  SerialAT.begin(115200, SERIAL_8N1, MODEM_RX, MODEM_TX);
  delay(3000);
  /* <-- SET GSM MODULE BAUDERATE AND UART PINS */

  /* INITIALIZE MODEM --> */
  modem.init();
  /* <-- INITIALIZE MODEM */

  /* RETURN THE STATUS IF GPRS IS CONNECTED --> */
  if (!modem.gprsConnect(apn.c_str())) {
    return false;
  }
  else {
    return true;
  }
  /* <-- RETURN THE STATUS IF GPRS IS CONNECTED */
```

```
}
/* <-- INITIALIZE SIM800L AND START GPRS CONNECTION IF A APN CODE IS PROVIDED */

/* RETURN TIME FROM SIM800L --> */
String sim800l_get_time(void){
  String time = modem.getGSMDateTime(DATE_TIME);
  time = time.substring(0,8);
  return time;
}
/* <-- RETURN TIME FROM SIM800L */

/* RETURNS HOW MUCH TIME IS LEFT FOR THE NEXT MESSAGE TO BE SEND --> */
int next_message_wakeup(int messages, String time){

  int next_message = 0;

  int hour = 0;
  int minutes = 0;


  hour = time.substring(0,2).toInt();
  minutes = time.substring(3,5).toInt();
  Serial.print("HOUR : ");
  Serial.println(time.substring(0,2));

  Serial.print("MINUTES : ");
  Serial.println(time.substring(3,5));


  if(messages == 1){
    /* 21:00:00   24h */

    if(hour >= 21){
      next_message = ((24-hour+21)*60) - minutes;
    }
    else{
      next_message = (21*60) - ((hour*60)+minutes);
    }

  }
  else if(messages == 2){
    /* 09:00:00 - 21:00:00   12h */

    int next_09 = 0;
    int next_21 = 0;
```

```
  if(hour >= 9){
    next_09 = ((24-hour+9)*60) - minutes;
  }
  else{
    next_09 = (9*60) - ((hour*60)+minutes);
  }

  if(hour >= 21){
    next_21 = ((24-hour+21)*60) - minutes;
  }
  else{
    next_21 = (21*60) - ((hour*60)+minutes);
  }

  if(next_09 < next_21){
    next_message = next_09;
  }
  else{
    next_message = next_21;
  }

}
else if(messages == 3){

  int next_05 = 0;
  int next_21 = 0;
  int next_14 = 0;


  if(hour >= 5){
    next_05 = ((24-hour+5)*60) - minutes;
  }
  else{
    next_05 = (5*60) - ((hour*60)+minutes);
  }

  if(hour >= 21){
    next_21 = ((24-hour+21)*60) - minutes;
  }
  else{
    next_21 = (21*60) - ((hour*60)+minutes);
  }
```

```
  if(hour >= 14){
    next_14 = ((24-hour+14)*60) - minutes;
  }
  else{
    next_14 = (14*60) - ((hour*60)+minutes);
  }


  if(next_05 < next_21){
    next_message = next_05;
  }
  else{
    next_message = next_21;
  }


  if(next_14 < next_message){
    next_message = next_14;
  }

}
else if(messages == 4){

  int next_03 = 0;
  int next_15 = 0;
  int next_09 = 0;
  int next_21 = 0;


  if(hour >= 3){
    next_03 = ((24-hour+3)*60) - minutes;
  }
  else{
    next_03 = (3*60) - ((hour*60)+minutes);
  }

  if(hour >= 21){
    next_21 = ((24-hour+21)*60) - minutes;
  }
  else{
    next_21 = (21*60) - ((hour*60)+minutes);
  }

  if(hour >= 15){
    next_15 = ((24-hour+15)*60) - minutes;
```

```
    }
    else{
      next_15 = (15*60) - ((hour*60)+minutes);
    }

    if(hour >= 9){
      next_09 = ((24-hour+9)*60) - minutes;
    }
    else{
      next_09 = (9*60) - ((hour*60)+minutes);
    }


    if(next_03 < next_09){
      next_message = next_03;
    }
    else{
      next_message = next_09;
    }


    if(next_15 < next_message){
      next_message = next_15;
    }

    if(next_21 < next_message){
      next_message = next_21;
    }

  }


  Serial.print("NEXT MESSAGE : ");
  Serial.println(next_message);

  return next_message;

}
/* <-- RETURNS HOW MUCH TIME IS LEFT FOR THE NEXT MESSAGE TO BE SEND */

/* IF IT IS NOT THE TIME TO SEND THE MESSAGE IT STARTS DEEPSLEEP FOR THE REMAINING TIME MAX DEEPSLEEP TIME IS 60 MINUTES
--> */
void deepsleep_handler(uint next_wakeup, uint messages){

  if(messages == 1 && (next_wakeup <= 1440 && next_wakeup >= 1435)){
```

```
    return;
  }

  if(messages == 2 && (next_wakeup <= 720 && next_wakeup >= 715)){
    return;
  }

  if(messages == 3 && (next_wakeup <= 480 && next_wakeup >= 475)){
    return;
  }

  if(messages == 4 && (next_wakeup <= 360 && next_wakeup >= 355)){
    return;
  }

  if(next_wakeup >= 60){
    esp_sleep_enable_timer_wakeup(3600000000);
    Serial.println("\n\n[+] --> DEEPSLEEP FOR 1 HOUR");
    esp_deep_sleep_start();

  }
  else if(next_wakeup < 60 && next_wakeup != 0){
    Serial.println("\n[+] --> DEEPSLEEP FOR " + String(next_wakeup) + " MINUTES");
    esp_sleep_enable_timer_wakeup(next_wakeup*60000000);
    esp_deep_sleep_start();
  }
  else{
    return;
  }

}
/* <--
 IF IT IS NOT THE TIME TO SEND THE MESSAGE IT STARTS DEEPSLEEP FOR THE REMAINING TIME MAX DEEPSLEEP TIME IS 60 MINUTES */

/* SEND MESSAGE VIA SMS --> */
bool send_sms(String target, String message){
  if(modem.sendSMS(target, message)){
    return true;
  }
  else{
    return false;
  }
}
/* <-- SEND MESSAGE VIA SMS */
```

```
/* BUILD THE SMS MESSAGE TO BE SEND --> */
String build_message(String name, int add_weight, int weight, float temperature, float humidity, float batterty_per ){
  String BEEZY_SMS = "NAME : "+ name +
                     "\nADD WEIGHT : " + add_weight +" gr" +
                     "\nTOTAL WEIGHT : " + weight +" gr" +
                     "\nTEMPERATURE : " + temperature + " oC" +
                     "\nHUMIDITY : " + humidity + " %" +
                     "\nBATTERY : " + batterty_per + " %";
  return BEEZY_SMS;
}
/* <-- BUILD THE SMS MESSAGE TO BE SEND */

/* SEND DATA TO THINGSPEAK VIA GET REQUEST --> */
bool upload_data_to_thingspeak(String api_key, int weight, float temperature, float humidity, float battery){

  if (!client.connect("api.thingspeak.com", 80)) {
    return false;
  }

  String httpRequestData = "api_key=" + api_key + "&field1="+ String(weight) +
                                                  "&field2="+ String(temperature) +
                                                  "&field3="+ String(humidity) +
                                                  "&field4="+ String(battery);



  client.print(String("GET ") + "/update" + " HTTP/1.1\r\n");
  client.print(String("Host: ") + "api.thingspeak.com" + "\r\n");
  client.println("Connection: close");
  client.println("Content-Type: application/x-www-form-urlencoded");
  client.print("Content-Length: ");
  client.println(httpRequestData.length());
  client.println();
  client.println(httpRequestData);

  return true;
}
/* <-- SEND DATA TO THINGSPEAK VIA GET REQUEST */

/* RETURNS TO DIFFERENCE BETWEEN THE LAST WEIGHTING MEASUREMENT AND STORES LAST MEASURE TO SPIFFS --> */
float last_scale_mesure(float w){

  File read = SPIFFS.open("/last_mesure.txt","r");
  String weight = "";
  while (read.available()) {
```

```
    weight +=  read.readString();
  }
  read.close();

  float add_w = w - weight.toFloat();

  File write = SPIFFS.open("/last_mesure.txt","w");
  write.print(w);
  write.close();

  return add_w;
}
/* <-- RETURNS TO DIFFERENCE BETWEEN THE LAST WEIGHTING MEASUREMENT AND STORES LAST MEASURE TO SPIFFS */
```

5.5.3 BeezyManagerWebPage.h

```
/* LIBRARIES --> */
#include <WiFi.h>
#include <AsyncTCP.h>
#include <ESPAsyncWebServer.h>
#include <SPIFFS.h>
/* <-- LIBRARIES */

/* GLOBAL --> */
bool SETUP_MESSAGE = false;
String WEB_SERVER_DATA = "";
/* <-- GLOBAL */

/* ASYNC WEB SERVER OBJECT INITIALIZATION ON PORT 80 --> */
AsyncWebServer server(80);
/* <-- ASYNC WEB SERVER OBJECT INITIALIZATION ON PORT 80 */

/* ASYNC WEB SOCKET OBJECT INITIALIZATION --> */
AsyncWebSocket ws("/ws");
/* <-- ASYNC WEB SOCKET OBJECT INITIALIZATION */

/* FUNCTIONS DECLARATION --> */
void initWebSocket();
void onEvent(AsyncWebSocket *server, AsyncWebSocketClient *client, AwsEventType type,void *arg, uint8_t *data, size_t len
);
String Beezy_Web_Server(const char* ssid, const char* passwd);
/* <-- FUNCTIONS DECLARATION*/
```

```
/* WEB SERVER HANDLER --> */
String Beezy_Web_Server(const char* ssid, const char* passwd){

  /* SET ESP32 AS A WIFI ACCESS POINT --> */
  WiFi.mode(WIFI_AP);
  /* <-- SET ESP32 AS A WIFI ACCESS POINT */

  /* SET THE SSID AND PASSWORD FOR THE ACCESS POINT --> */
  WiFi.softAP(ssid, passwd);
  /* <-- SET THE SSID AND PASSWORD FOR THE ACCESS POINT */

  /* INITIALIZING WEB SOCKET --> */
  initWebSocket();
  /* <-- INITIALIZING WEB SOCKET */

  /* GET WEB PAGE FILES FOR THE WEB SERVER FROM SPIFFS --> */
  server.on("/", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send(SPIFFS, "/index.html","text/html");
  });
  server.on("/styles.css", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send(SPIFFS, "/styles.css", "text/css");
  });
  server.on("/script.js", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send(SPIFFS, "/script.js", "text/javascript");
  });
  server.on("/beezy.png", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send(SPIFFS, "/beezy.png", "image/png");
  });
  /* <-- GET WEB PAGE FILES FOR THE WEB SERVER FROM SPIFFS */

  /* START WEB SERVER --> */
  server.begin();
  /* <-- START WEB SERVER */

  /* WAIT FOR THE USER TO SEND CONFIGURATION DATA & BLINK THE INDICATION LED ON THE BOARD --> */
  while (SETUP_MESSAGE == false){
    delay(500);
    digitalWrite(12, HIGH);
    delay(500);
    digitalWrite(12, LOW);
    Serial.println("WAIT FOR SETUP TO FINISH");
  }
  /* <-- WAIT FOR THE USER TO SEND CONFIGURATION DATA & BLINK THE INDICATION LED ON THE BOARD */
```

```
  /* STOP WEB SERVER --> */
  server.end();
  /* <-- STOP WEB SERVER */

  /* STOP WIFI ACCESS POINT FROM ESP32 --> */
  WiFi.softAPdisconnect(true);
  /* <-- STOP WIFI ACCESS POINT FROM ESP32 */

  /* RETURN THE CONFIGURATION DATA --> */
  return WEB_SERVER_DATA;
  /* <-- RETURN THE CONFIGURATION DATA */

}
/* <-- WEB SERVER HANDLER */

/* IMPLEMENT THE WEB SOCKET EVENT --> */
void onEvent(AsyncWebSocket *server, AsyncWebSocketClient *client, AwsEventType type,
             void *arg, uint8_t *data, size_t len) {

   String msg = (const char*)  data; /* DATA SENDEND FROM THE USER VIA THE WEB PAGE */
   String write_file = msg;
   Serial.println(write_file);

   String location = "";
   String thingspeak = "";
   String phone = "";
   String messages = "";
   String apn = "";
   String prefix = "";

   /* SEPARATE DATA TO MULTIPLE VARIABLES --> */
   byte separator = msg.indexOf('|');
   location = msg.substring(0,separator);
   msg = msg.substring(separator+1);
   separator = msg.indexOf('|');
   thingspeak = msg.substring(0,separator);
   msg = msg.substring(separator+1);
   separator = msg.indexOf('|');
   phone = msg.substring(0,separator);
   msg = msg.substring(separator+1);
   separator = msg.indexOf('|');
   messages = msg.substring(0,separator);
   msg = msg.substring(separator+1);
   separator = msg.indexOf('|');
   apn = msg.substring(0,separator);
```

```
   msg = msg.substring(separator+1);
   separator = msg.indexOf('|');
   prefix = msg.substring(0,separator);
   /* <-- SEPARATE DATA TO MULTIPLE VARIABLES */

   /* CHECK IF DATA WHERE SEND CORRECTLY AND WE HAVE NOT DATA LOSES --> */
   if(location != "" && thingspeak != "" && phone != "" && messages != ""){
     WEB_SERVER_DATA = location+"|"+thingspeak+"|"+phone+"|"+messages+"|"+apn+"|"+prefix+"|";
     SETUP_MESSAGE = true;
   }
   /* <-- CHECK IF DATA WHERE SEND CORRECTLY AND WE HAVE NOT DATA LOSES */

}
/* <-- IMPLEMENT THE WEB SOCKET EVENT */

/* INITIALIZING WEB SOCKETS AND WEB SERVER --> */
void initWebSocket() {
  ws.onEvent(onEvent);
  server.addHandler(&ws);
}
/* <-- INITIALIZING WEB SOCKETS AND WEB SERVER */
```

5.5.4 HTML5

```
<!DOCTYPE html>
<html lang='en' >
<head>
  <meta charset='UTF-8'>
  <meta name='viewport' content='width=device-width, initial-scale=1'>
  <title>Beeζu - Set Up</title>
  <link rel="stylesheet" href="styles.css">
  <script type="text/javascript" src="script.js"></script>

</head>
<body>
<!-- partial:index.partial.html -->

<div class='login-box wrapper'>
  <h2><img src="beezy.png" width="150px" alt=""></h2>
  <form >
    <div class='user-box'>
      <input type='text' name='LOCATION'  id="place" required=''>
```

```
      <label>NAME</label>
    </div>
    <div class='user-box'>
      <input type="number" id="sms"  min="1" max="4" required><br><br>
      <label>MESSAGES</label>
    </div>
    <div class='user-box'>
      <select name="pref" id="pref" onchange="showFields()">
        <option value="sms_thingspeak">SMS & THINGSPEAK</option>
        <option value="sms">SMS</option>
        <option value="thingspeak">THINGSPEAK</option>
      </select>
      <br><br><br>
    </div>
    <div class='user-box'>
      <input type='text' name='APIKEY' id="thingspeak" required=''>
      <label id="tl">THINGSPEAK API KEY</label>
    </div>
    <div class='user-box'>
      <input type="tel" id="phone" name="phone" pattern="69[0-9]{8}" required><br><br>
      <label id="pl">PHONE NUMBER</label>
    </div>
    <div class='user-box'>
      <select name="phone_telecom" id="phone_telecom">
        <option value="internet">COSMOTE</option>
        <option value="internet.vodafone.gr">VODAFONE</option>
        <option value="gint.b-online.gr">WIND</option>
      </select>
    </div>

      <br><br>
    <a href='#'>
      <span></span>
      <span></span>
      <span></span>
      <span></span>
      <input id='sub'  onclick="send()" value='Save'/>
    </a>
  </form>
  <br>
  <small id="savem" style="margin-left:65%;font-size:0px;">DATA SAVED !</small>
</div>
</body>
</html>
```

5.5.5 CSS3

```
html {
  height: 100%;
}
body {
  margin:0;
  padding:0;
  font-family: sans-serif;
  background: #ffec00;
}

.login-box {
  position: absolute;
  top: 50%;
  left: 50%;
  width: 400px;
  padding: 40px;
  transform: translate(-50%, -50%);
  background: rgba(0,0,0,.4);
  box-sizing: border-box;
  box-shadow: 0 15px 25px rgba(0,0,0,.6);
  border-radius: 10px;
}

.login-box h2 {
  margin: 0 0 30px;
  padding: 0;
  color: #000;
  text-align: center;
}

.login-box .user-box {
  position: relative;
}

.login-box .user-box input {
  width: 100%;
  padding: 10px 0;
  font-size: 16px;
  color: #000;
  margin-bottom: 30px;
  border: none;
  border-bottom: 1px solid #000;
```

```
  outline: none;
  background: transparent;
}
.login-box .user-box label {
  position: absolute;
  top:0;
  left: 0;
  padding: 10px 0;
  font-size: 16px;
  color: #000;
  pointer-events: none;
  transition: .5s;
}

.login-box .user-box input:focus ~ label,
.login-box .user-box input:valid ~ label {
  top: -20px;
  left: 0;
  color: #ffec00;
  font-size: 12px;
}

.login-box form a {
  position: relative;
  display: inline-block;
  color: #ffec00;
  font-size: 16px;
  text-decoration: none;
  text-transform: uppercase;
  overflow: hidden;
  transition: .5s;
  margin-top: 40px;
  letter-spacing: 4px
}



.login-box a:hover {
  background: #ffec00;
  color: #fff;
  border-radius: 5px;
  box-shadow: 0 0 5px #ffec00,
              0 0 25px #ffec00,
              0 0 50px #ffec00,
              0 0 100px #ffec00;
```

```
}

.login-box a span {
  position: absolute;
  display: block;
}

.login-box a span:nth-child(1) {
  top: 0;
  left: -100%;
  width: 100%;
  height: 2px;
  background: linear-gradient(90deg, transparent, #ffec00);
  animation: btn-anim1 1s linear infinite;
}

@keyframes btn-anim1 {
  0% {
    left: -100%;
  }
  50%,100% {
    left: 100%;
  }
}

.login-box a span:nth-child(2) {
  top: -100%;
  right: 0;
  width: 2px;
  height: 100%;
  background: linear-gradient(180deg, transparent, #ffec00);
  animation: btn-anim2 1s linear infinite;
  animation-delay: .25s
}

@keyframes btn-anim2 {
  0% {
    top: -100%;
  }
  50%,100% {
    top: 100%;
  }
}

.login-box a span:nth-child(3) {
```

```
  bottom: 0;
  right: -100%;
  width: 100%;
  height: 2px;
  background: linear-gradient(270deg, transparent, #ffec00);
  animation: btn-anim3 1s linear infinite;
  animation-delay: .5s
}

@keyframes btn-anim3 {
  0% {
    right: -100%;
  }
  50%,100% {
    right: 100%;
  }
}

.login-box a span:nth-child(4) {
  bottom: -100%;
  left: 0;
  width: 2px;
  height: 100%;
  background: linear-gradient(360deg, transparent, #ffec00);
  animation: btn-anim4 1s linear infinite;
  animation-delay: .75s
}

@keyframes btn-anim4 {
  0% {
    bottom: -100%;
  }
  50%,100% {
    bottom: 100%;
  }
}

#sub{
  border: none;
  background-color: transparent;
  color: black;
  text-transform: uppercase;
  font-size: 16px;
  padding: 10px 20px;
  width: 45px;
```

```
}

select{
  background-color: transparent;
  font-size: 16px;
  padding-right:65%;
}

#pref{
  padding-right:45%;
}

@media only screen and (max-width:410px){

  body{
    height: 800px;
    overflow: visible;
    position: absolute;
    left: 50%
  }

  img{
    margin-top: 10vw;
  }

  select{
    padding-right:50%;
  }

  #pref{
    padding-right:20%;
  }

  .login-box {
    position: absolute;
    top: 50%;
    left: 50%;
    width: 300px;
    padding: 40px;
    transform: translate(-50%, -50%);
    background: rgba(0,0,0,0);
    box-sizing: border-box;
    box-shadow: 0 15px 25px rgba(0,0,0,0);
    border-radius: 10px;
```

```
}



.login-box .user-box input:focus ~ label,
.login-box .user-box input:valid ~ label {
  top: -20px;
  left: 0;
  color: #000;
  font-size: 12px;
}

.login-box form a {
  position: relative;
  display: inline-block;
  color: #000;
  font-size: 16px;
  text-decoration: none;
  text-transform: uppercase;
  overflow: hidden;
  transition: .5s;
  margin-top: 40px;
  letter-spacing: 4px
}



.login-box a:hover {
  background: #000;
  color: #fff;
  border-radius: 5px;
  box-shadow: 0 0 5px #000,
              0 0 25px #000,
              0 0 50px #000,
              0 0 100px #000;
}

.login-box a span {
  position: absolute;
  display: block;
}

.login-box a span:nth-child(1) {
  top: 0;
  left: -100%;
```

```
  width: 100%;
  height: 2px;
  background: linear-gradient(90deg, transparent, #000);
  animation: btn-anim1 1s linear infinite;
}

@keyframes btn-anim1 {
  0% {
    left: -100%;
  }
  50%,100% {
    left: 100%;
  }
}

.login-box a span:nth-child(2) {
  top: -100%;
  right: 0;
  width: 2px;
  height: 100%;
  background: linear-gradient(180deg, transparent, #000);
  animation: btn-anim2 1s linear infinite;
  animation-delay: .25s
}

@keyframes btn-anim2 {
  0% {
    top: -100%;
  }
  50%,100% {
    top: 100%;
  }
}

.login-box a span:nth-child(3) {
  bottom: 0;
  right: -100%;
  width: 100%;
  height: 2px;
  background: linear-gradient(270deg, transparent, #000);
  animation: btn-anim3 1s linear infinite;
  animation-delay: .5s
}

@keyframes btn-anim3 {
```

```
    0% {
      right: -100%;
    }
    50%,100% {
      right: 100%;
    }
  }

  .login-box a span:nth-child(4) {
    bottom: -100%;
    left: 0;
    width: 2px;
    height: 100%;
    background: linear-gradient(360deg, transparent, #000);
    animation: btn-anim4 1s linear infinite;
    animation-delay: .75s
  }

  @keyframes btn-anim4 {
    0% {
      bottom: -100%;
    }
    50%,100% {
      bottom: 100%;
    }
  }

  #sub:hover{
    border: none;
    background-color: transparent;
    color: #ffec00;
    text-transform: uppercase;
    font-size: 16px;
    padding: 10px 20px;
  }


}
```

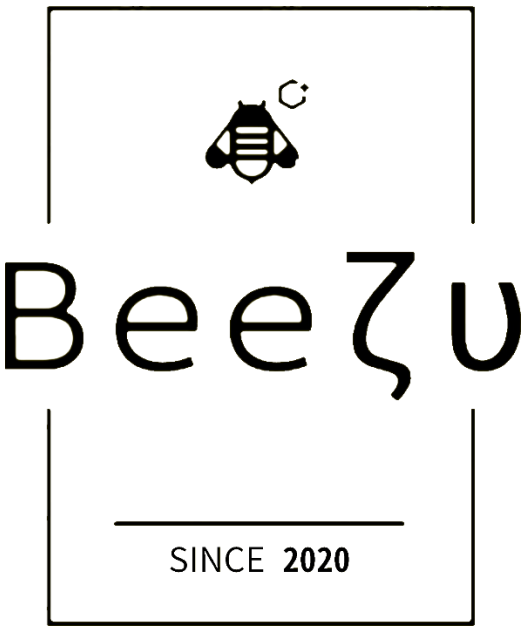
5.5.6 Javascript

```
/* SOCKET GATEWAY --> */
var gateway = 'ws://'+window.location.hostname+'/ws';
/* <-- SOCKET GATEWAY */

var websocket;

/* INITIALIZING WEB SOCKET --> */
function initWebSocket() {
  console.log('Trying to open a WebSocket connection...');
  websocket = new WebSocket(gateway);
  websocket.onopen    = onOpen;
  websocket.onclose   = onClose;
}
/* <-- INITIALIZING WEB SOCKET */


function onOpen(event) {
  console.log('Connection opened');
}

function onClose(event) {
  console.log('Connection closed');
  setTimeout(initWebSocket, 2000);
}

/* WEB PAGE EVENT LISTENER --> */
window.addEventListener('load', onLoad);
/* <-- WEB PAGE EVENT LISTENER */

/* CALL initWebSocket() --> */
function onLoad(event) {
  initWebSocket();
}
/* <-- CALL initWebSocket() --> */

/* SHOW NECESSARY FIELDS PER CHOICE --> */
function showFields(){
  var fields = document.getElementById("pref").value;
  if(fields == 'sms'){
    document.getElementById("thingspeak").style.display = "none";
    document.getElementById("tl").style.display = "none";
    document.getElementById("phone_telecom").style.display = "none";
```

```
    document.getElementById("phone").style.display = "block";
    document.getElementById("pl").style.display = "block";
  }
  else if(fields == 'thingspeak'){
    document.getElementById("thingspeak").style.display = "block";
    document.getElementById("tl").style.display = "block";
    document.getElementById("phone_telecom").style.display = "block";
    document.getElementById("phone").style.display = "none";
    document.getElementById("pl").style.display = "none";
  }
  else{
    document.getElementById("thingspeak").style.display = "block";
    document.getElementById("tl").style.display = "block";
    document.getElementById("phone_telecom").style.display = "block";
    document.getElementById("phone").style.display = "block";
    document.getElementById("pl").style.display = "block";
  }
}
/* <-- SHOW NECESSARY FIELDS PER CHOICE */

/* SEND DATA TO ESP32 VIA WEB SOCKET --> */
function send(){

  place = document.getElementById("place").value;
  thingspeak = document.getElementById("thingspeak").value;
  phone = document.getElementById("phone").value;
  sms = document.getElementById("sms").value;
  apn = document.getElementById("phone_telecom").value;
  prefix = document.getElementById("pref").value;

  document.getElementById("savem").style.fontSize = "0px";

  document.getElementById("place").value = "";
  document.getElementById("thingspeak").value = "";
  document.getElementById("phone").value = "";
  document.getElementById("sms").value = "";

  /* CHECK IF ALL NECESSERY FIELDS ARE COMPLETE --> */
  if(prefix == "sms_thingspeak"){
    if(place != "" && thingspeak != "" && phone != "" && sms != "" ){
      msg = place+"|"+thingspeak+"|"+phone+"|"+sms+"|"+apn+"|"+prefix+"|";
      console.log(msg);
      websocket.send(msg);
      document.getElementById("savem").style.fontSize = "10px";
    }
```

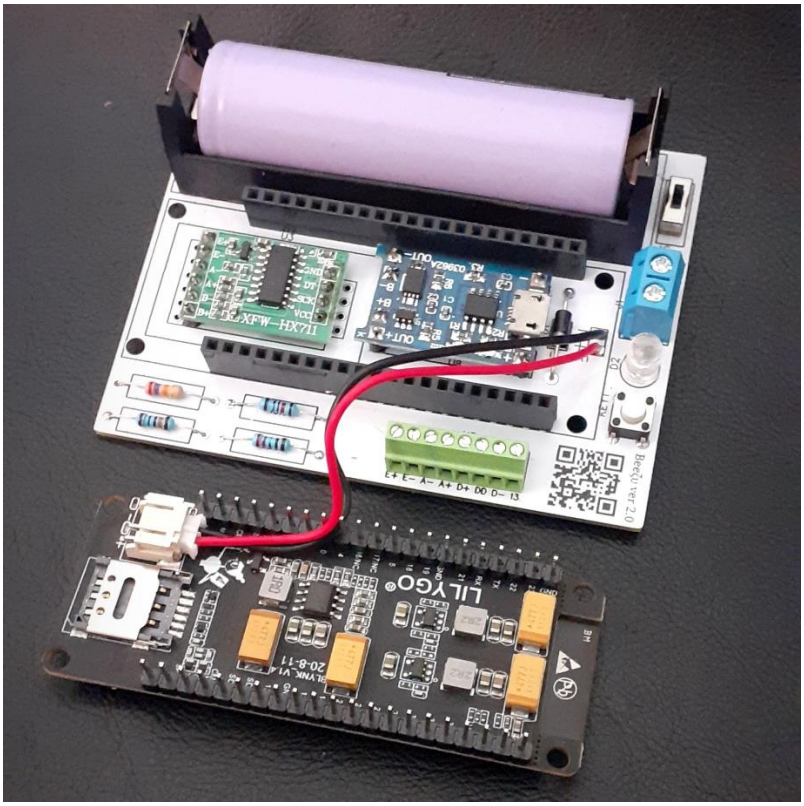
```
  }
  else if (prefix == "sms") {
    if(place != "" && phone != "" && sms != "" ){
      msg = place+"|"+"0"+"|"+phone+"|"+sms+"|"+"0"+"|"+prefix+"|";
      console.log(msg);
      websocket.send(msg);
      document.getElementById("savem").style.fontSize = "10px";
    }
  }
  else if (prefix == "thingspeak") {
    if(place != "" && thingspeak != "" && sms != "" ){
      msg = place+"|"+thingspeak+"|"+"0"+"|"+sms+"|"+apn+"|"+prefix+"|";
      console.log(msg);
      websocket.send(msg);
      document.getElementById("savem").style.fontSize = "10px";
    }
  }
  /* CHECK IF ALL NECESSERY FIELDS ARE COMPLETE --> */
}
/* <-- SEND DATA TO ESP32 VIA WEB SOCKET */
```

5.5.7 Εικόνα web application

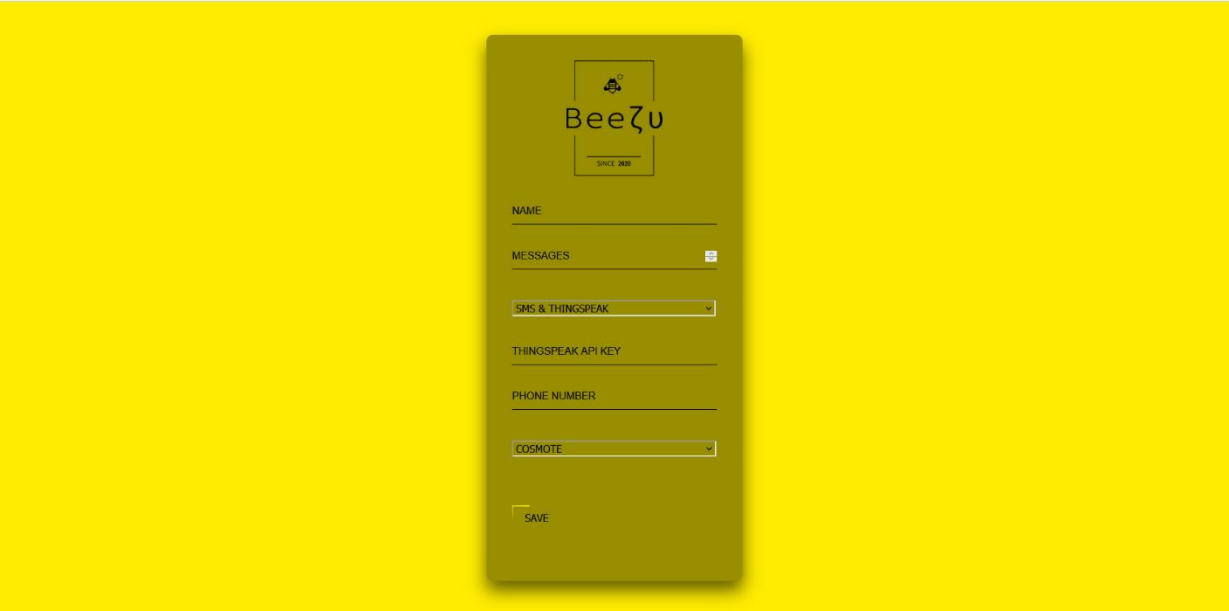


5.6 Φωτογραφικό υλικό

Εικόνα 33: PCB μελισσοκομικής ζυγαριάς (1)



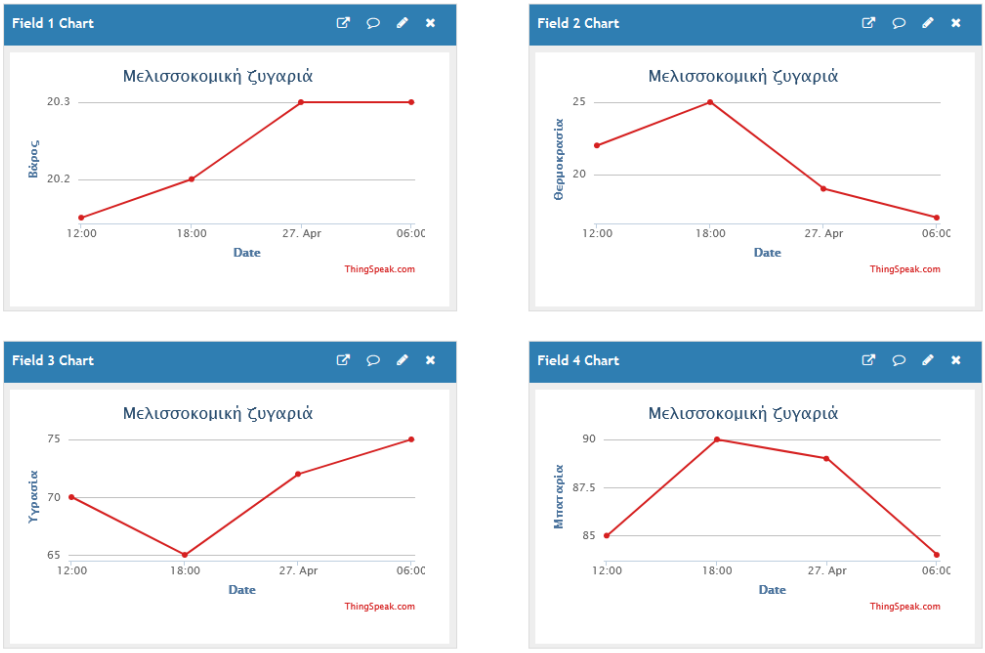
Εικόνα 34: PCB μελισσοκομικής ζυγαριάς (2)



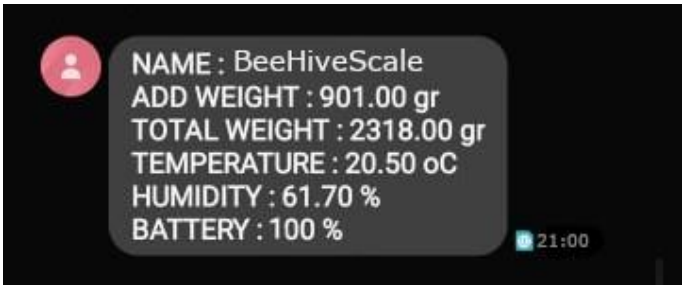
Εικόνα 35: Η web εφαρμογή για τη ρύθμιση της ζυγαριάς μέσω PC

Εικόνα 36: Η web εφαρμογή για τη ρύθμιση της ζυγαριάς μέσω Mobile

Εικόνα 37: Η έξυπνη μελισσοκομική ζυγαριά



Εικόνα 38: Προβολή δεδομένων από το ThingSpeak

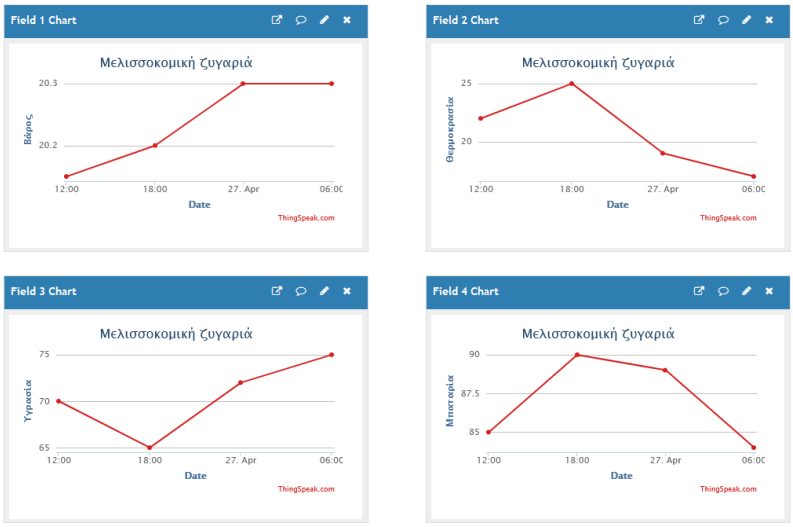


Εικόνα 39: Προβολή δεδομένων μέσο SMS

ΚΕΦΑΛΑΙΟ 6
Χρήση συστήματος

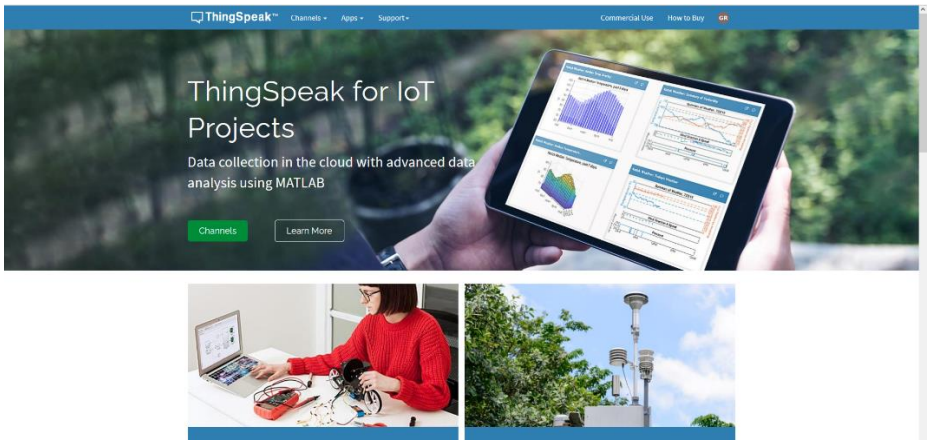
6.1 Δημιουργία λογαριασμού στο ThingSpeak

Το ThingSpeak όπως προαναφέρθηκε είναι μια IoT εφαρμογή μέσω της οποίας ο χρήστης θα έχει τη δυνατότητα να ενημερώνεται για τις μετρήσεις της ζυγαριάς και θα μπορεί να τις συγκρίνει με αυτές προηγούμενων ημερών, αφού το ThingSpeak χρησιμοποιεί line charts για την οπτικοποίηση των δεδομένων του. Παρακάτω λοιπόν, αναλύεται ο τρόπος με τον οποίο ρυθμίζεται η εφαρμογή για να είναι συμβατή με την μελισσοκομική ζυγαριά.

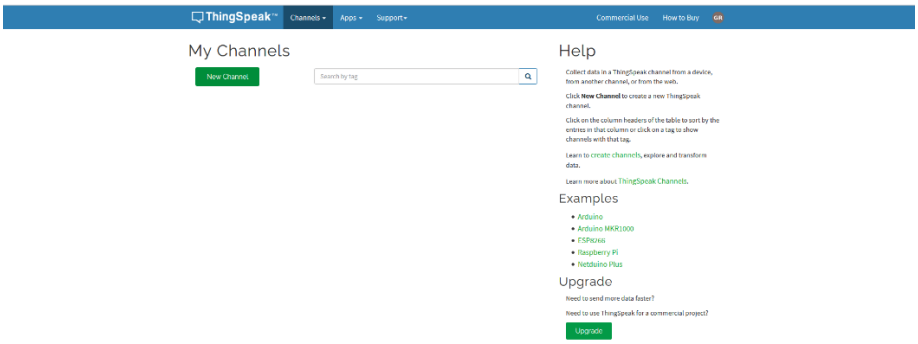


Εικόνα 40 : Οπτικοποίηση δεδομένων στο ThingSpeak

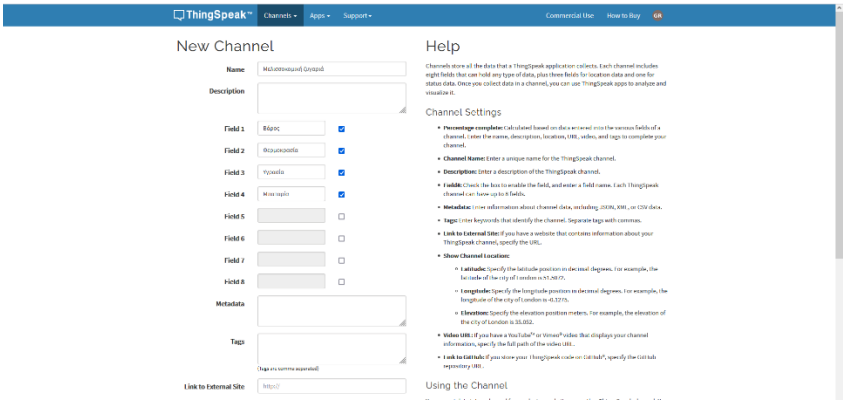
Αρχικά, ο χρήστης πρέπει να κατευθυνθεί στην ιστοσελίδα https://www.thingspeak.com και να δημιουργήσει ένα λογαριασμό συμπληρώνοντας τα στοιχεία του στα αντίστοιχα πεδία. Αφού λοιπόν γίνει αυτό, θα πρέπει να πατήσει το κουμπί *Channels* και έπειτα το *New Channel.* Στη συνέχεια ο χρήστης θα πρέπει να δώσει ένα όνομα στο Channel το οποίο ουσιαστικά θα είναι το όνομα της ζυγαριάς και να συμπληρώσει τα ονόματα των Field 1, Field 2, Field 3 και Field 4 τα οποία αντιστοιχούν στις εξής κατηγορίες : Βάρος, Θερμοκρασία, Υγρασία και Μπαταρία. Τέλος πατάει το κουμπί *Sava Channel* το ThingSpeak είναι έτοιμο για χρήση.
Η διαδικασία αυτή παρουσιάζεται στις επόμενες φωτογραφίες.



Εικόνα 41 : Ρύθμιση Channel (1)

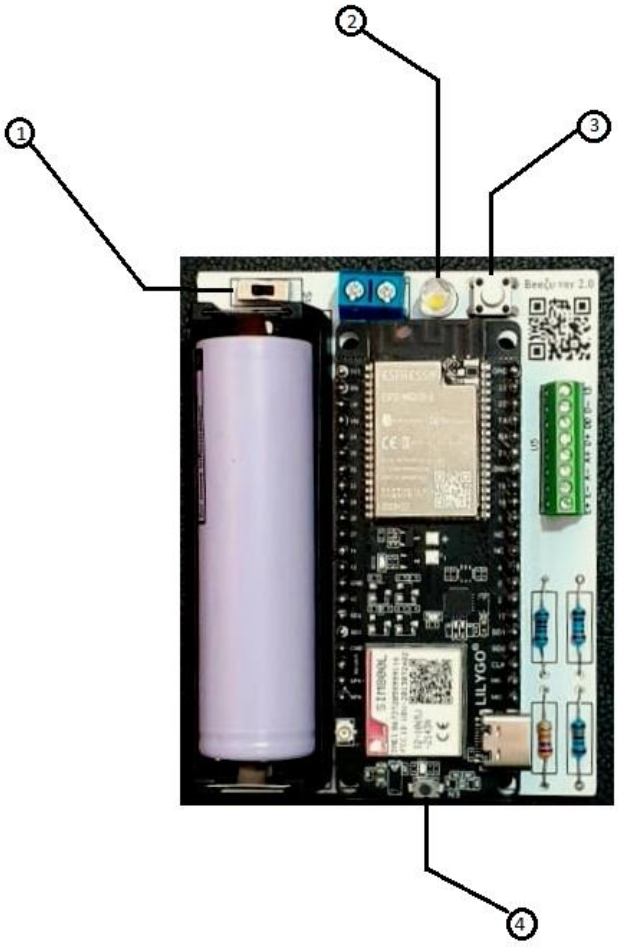


Εικόνα 42 : Ρύθμιση Channel (2)



Εικόνα 43 : Ρύθμιση Channel (3)

6.2 Ρύθμιση της μελισσοκομικής ζυγαριάς



Εικόνα 44: PCB μελισσοκομικής ζυγαριάς

1	Διακόπτης On/Off
2	Λευκό LED ειδοποίησης
3	Κουμπί Set up
4	Κουμπί Reset

Παρακάτω λοιπόν, θα γίνει η ανάλυση της διαδικασίας που απαιτείται για την ρύθμιση της μελισσοκομικής ζυγαριάς :

- Ξεκινώντας, στο PCB ο διακόπτης on/off πρέπει να γυρίσει προς την πλευρά που αναγράφεται το s1.
- Έπειτα, πρέπει να πατηθεί το κουμπί reset όπου ακολουθούν δύο πιθανά ενδεχόμενα.
    - Πρώτον, στα επόμενα δευτερόλεπτα να μην αναβοσβήσει το λευκό LED ειδοποίησης, όπου σε αυτήν την περίπτωση θα πρέπει να επαναληφθεί η παραπάνω διαδικασία.
    - Δεύτερον, το λευκό LED ειδοποίησης να αναβοσβήσει μια φορά. Σε αυτήν την περίπτωση, πρέπει πατηθεί το κουμπί set up και το λευκό LED ειδοποίησης θα αρχίσει να αναβοσβήνει συνεχώς. Τότε, η μελισσοκομική ζυγαριά θα βρίσκεται σε set up mode.
- Στην συνέχεια, αφού η ζυγαριά βρίσκεται σε set up mode, μια οποιαδήποτε ηλεκτρονική συσκευή θα πρέπει να συνδεθεί στο Wi-Fi που έχει ως ονομασία **Beezy**, με τον κωδικό **B3EZY!**.
- Αφού πραγματοποιηθεί η σύνδεση, ο χρήστης θα πρέπει να ανατρέξει στο browser και να κατευθυνθεί στην διεύθυνση http://192.168.4.1.
- Εκεί, ο χρήστης θα έχει την δυνατότητα να συμπληρώσει το όνομα της μελισσοκομικής ζυγαριάς που επιθυμεί, όπως επίσης να επιλέξει τον τρόπο αποστολής μηνύματος μεταξύ SMS, ThingSpeak ή και των δύο.
    - Εάν επιλεχθεί το SMS, ο χρήστης πρέπει να συμπληρώσει την ποσότητα των μηνυμάτων που θέλει να λαμβάνει καθημερινά (ανάλογα με την ποσότητα τα μηνύματα στέλνονται τις ανάλογες ώρες, οι οποίες αναφέρονται στον παρακάτω πίνακα), δηλαδή από ένα έως τέσσερα, καθώς επίσης και τον τηλεφωνικό αριθμό που θα χρησιμοποιεί για να διαβάζει τα μηνύματα.
    - Εάν επιλεχθεί το ThingSpeak, συμπληρώνεται ξανά η ιδανική ποσότητα μηνυμάτων, αλλά μαζί με αυτήν πρέπει να συμπληρωθεί το API Key και ο πάροχος της κάρτας sim που βρίσκεται μέσα στην μελισσοκομική ζυγαριά.
    - Εάν επιλεχθούν και οι δύο τρόποι μαζί, θα πρέπει να συμπληρωθούν όλα τα παραπάνω.

Τέλος, εφόσον όλα τα απαραίτητα πεδία είναι συμπληρωμένα, ο χρήστης θα πρέπει να πατήσει save, αποθηκεύοντας τα δεδομένα του στο σύστημα, ώστε με αυτόν τον τρόπο η ζυγαριά να έχει την δυνατότητα να λειτουργεί αυτόνομα.

1 μήνυμα/μέρα	**21:00**	**-**	**-**	**-**
2 μηνύματα/μέρα	**09:00**	**21:00**	**-**	**-**
3 μηνύματα/μέρα	**05:00**	**14:00**	**21:00**	**-**
4 μηνύματα/μέρα	**09:00**	**15:00**	**21:00**	**03:00**

ΚΕΦΑΛΑΙΟ 7
Συμπεράσματα & Μελλοντική εργασία

Με την περάτωση αυτής της πτυχιακής εργασίας, καταλήγουμε σε κάποια σημαντικά συμπεράσματα που αφορούν τις δυνατότητες της ζυγαριάς αλλά και τους περιορισμούς που προκύπτουν από την κατασκευή της. Ως επακόλουθο λοιπόν, είναι να αναφερθούν μελλοντικές βελτιώσεις για τους περιορισμούς αυτούς, με σκοπό η ζυγαριά να γίνεται όλο και καλύτερη με την πάροδο του χρόνου.

Πρώτα λοιπόν, λόγω του ότι το σύστημα θα λειτουργεί κυρίως σε δύσβατες περιοχές και σε συνδυασμό με το ότι υπάρχουν διαφορετικές εταιρίες κινητής τηλεφωνίας με διαφορετικό εύρος σήματος, είναι πιθανό να υπάρχουν φορές όπου η ζυγαριά δεν θα μπορεί να συνδεθεί στο δίκτυο κινητής. Αυτό είναι ένα πρόβλημα το οποίο μπορεί να περιοριστεί αισθητά με τη χρήση μιας μεγάλης εξωτερικής κεραίας SMA, αλλά και πάλι δεν εξαλείφεται πλήρως.

Κατόπιν, όσον αφορά την μπαταρία, η διάρκεια της μεταβάλλεται εξαιτίας πολλών παραγόντων. Για παράδειγμα, σε περιπτώσεις μεγάλης κακοκαιρίας το φωτοβολταϊκό πάνελ δε θα έχει τη δυνατότητα να την φορτίσει, με αποτέλεσμα, αν ακόμα και μετά από μερικές μέρες εξακολουθεί να υπάρχει κακοκαιρία, το σύστημα να μην λειτουργεί. Πρόκειται για ένα ζήτημα το οποίο μπορεί να μειωθεί σημαντικά με τη χρήση μπαταρίας μεγαλύτερης χωρητικότητας σε συνδυασμό με κάποιο πιο ισχυρό πάνελ, καθώς και με την υλοποίηση ενός MPPT (Maximum Power Point Tracking).

Έκτος από αυτό, σε κάποια μελλοντική έκδοση της ζυγαριάς θα μπορούσε να προστεθεί ένα ανεμόμετρο για την καταμέτρηση της ταχύτητας του αέρα, μια πολύ σημαντική πληροφορία καθώς οι μέλισσες δεν βγαίνουν έξω από την κυψέλη τους σε συνθήκες πολύ δυνατών ανέμων.

Στην συνέχεια, ένας ακόμη περιορισμός είναι εμφανής στην δωρεάν συνδρομή της εφαρμογή ThingSpeak, αφού επιτρέπει την δημιουργία μόνο τεσσάρων Channels, δηλαδή μόνο τέσσερείς ζυγαριές. Ένας ξεχωριστός τρόπος επίλυσης αυτού του προβλήματος, είναι με την δημιουργία μιας εφαρμογής IoT με, θεωρητικά, την δυνατότητα απεριορίστων συσκευών.

Τέλος, θα μπορούσε να γίνει η σχεδίαση της πλακέτας χωρίς την χρήση development boards με σκοπό να μειωθεί το κόστος και να μπορέσει η ζυγαριά να μπει σε μια εμπορική γραμμή παραγωγής.

Για να συνοψίσουμε λοιπόν, η έξυπνη μελισσοκομική ζυγαριά είναι ένα ενσωματωμένο σύστημα με πολλές δυνατότητες, προσφέροντας μεγάλη βοήθεια στον σύγχρονο μελισσοκόμο, αλλά βέβαια έχει ακόμα μεγάλες προοπτικές εξέλιξης, οι οποίες θα πρέπει να υλοποιηθούν.

Βιβλιογραφία

[1] https://el.wikipedia.org/wiki/Διαδίκτυο

[2] https://el.wikipedia.org/wiki/Διαδίκτυο_των_πραγμάτων

[3] https://el.wikipedia.org/wiki/Ενσωματωμένο_σύστημα

[4] https://el.wikipedia.org/wiki/Μικροελεγκτής

[5] https://glidemtech.com/hardware-design-service

[6] https://en.wikipedia.org/wiki/ESP32

[7] https://en.wikipedia.org/wiki/GSM

[8] https://datasheetspdf.com/pdf-file/989664/SIMCom/SIM800L/1

[9] https://en.wikipedia.org/wiki/Solar_panel

[10] https://en.wikipedia.org/wiki/Load_cell

[11] https://www.digikey.com/htmldatasheets/production/1836471/0/0/1/hx711.html

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

[13] https://grobotronics.com/images/companies/1/datasheets/TP4056%20Datasheet.pdf?1530184261538

[14] https://en.wikipedia.org/wiki/Diode

[15] https://en.wikipedia.org/wiki/Light-emitting_diode

[16] https://en.wikipedia.org/wiki/Resistor

[17] https://en.wikipedia.org/wiki/Voltage_divider

[18] https://en.wikipedia.org/wiki/SMS

[19] https://thingspeak.com/pages/learn_more

[20] https://en.wikipedia.org/wiki/ThingSpeak

[21] https://randomnerdtutorials.com/esp32-deep-sleep-arduino-ide-wake-up-sources

[22] https://en.wikipedia.org/wiki/Printed_circuit_board

[23] https://en.wikipedia.org/wiki/Through-hole_technology

[24] https://en.wikipedia.org/wiki/Surface-mount_technology

[25] https://en.wikipedia.org/wiki/Circuit_diagram

[26] https://www.electronicshub.org/

[27] https://en.wikipedia.org/wiki/Gerber_format

[28] https://en.wikipedia.org/wiki/Bill_of_materials

[29] https://docs.platformio.org/en/latest//what-is-platformio.html

[30] https://en.wikipedia.org/wiki/EasyEDA

[31] https://github.com/Xinyuan-LilyGO/LilyGo-T-Call-SIM800

[32] https://randomnerdtutorials.com/esp32-sim800l-publish-data-to-cloud/

[33] https://randomnerdtutorials.com/power-esp32-esp8266-solar-panels-battery-level-monitoring/

[34] https://www.raviyp.com/gsm-network-time-synchronization-for-rtc-using-at-clts-command/

[35] https://randomnerdtutorials.com/esp32-web-server-spiffs-spi-flash-file-system/

[36] https://www.teachmemicro.com/how-to-use-load-cell-hx711-arduino/