ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Εφαρμογή IoT με χρήση Raspberry Pi

Παπαδόπουλος Σωτήριος
Μπράνκοβ Νικολάι

Εισηγητής: Πάρις Μαστοροκώστας, Καθηγητής

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΩΝ ΠΤΥΧΙΑΚΗΣ ΕΡΓΑΣΙΑΣ

Οι κάτωθι υπογεγραμμένοι **Παπαδόπουλος Σωτήριος** του **Σταύρου**, με αριθμό μητρώου **45404** και **Νικολάι Μπράνκοβ** του **Λατσεζάρ**, με αριθμό μητρώου **44971**, φοιτητές του Τμήματος Μηχανικών Πληροφορικής και Υπολογιστών του Πανεπιστημίου Δυτικής Αττικής, πριν αναλάβουμε την εκπόνηση της Πτυχιακής Εργασίας μας, δηλώνουμε ότι ενημερωθήκαμε για τα παρακάτω:

«Η Πτυχιακή Εργασία (Π.Ε.) αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο του συγγραφέα, όσο και του Ιδρύματος και θα πρέπει να έχει μοναδικό χαρακτήρα και πρωτότυπο περιεχόμενο.

Απαγορεύεται αυστηρά οποιοδήποτε κομμάτι κειμένου της να εμφανίζεται αυτούσιο ή μεταφρασμένο από κάποια άλλη δημοσιευμένη πηγή. Κάθε τέτοια πράξη αποτελεί προϊόν λογοκλοπής και εγείρει θέμα Ηθικής Τάξης για τα πνευματικά δικαιώματα του άλλου συγγραφέα. Αποκλειστικός υπεύθυνος είναι ο συγγραφέας της Π.Ε., ο οποίος φέρει και την ευθύνη των συνεπειών, ποινικών και άλλων, αυτής της πράξης.

Πέραν των όποιων ποινικών ευθυνών του συγγραφέα σε περίπτωση που το Ίδρυμα του έχει απονείμει Πτυχίο, αυτό ανακαλείται με απόφαση της Συνέλευσης του Τμήματος. Η Συνέλευση του Τμήματος με νέα απόφασή της, μετά από αίτηση του ενδιαφερόμενου, του αναθέτει εκ νέου την εκπόνηση της Π.Ε. με άλλο θέμα και διαφορετικό επιβλέποντα καθηγητή. Η εκπόνηση της εν λόγω Π.Ε. πρέπει να ολοκληρωθεί εντός τουλάχιστον ενός ημερολογιακού 6μήνου από την ημερομηνία ανάθεσής της. Κατά τα λοιπά εφαρμόζονται τα προβλεπόμενα στο άρθρο 18, παρ. 5 του ισχύοντος Εσωτερικού Κανονισμού.»

Ευχαριστίες

Για την διεκπεραίωση της παρούσας Πτυχιακής Εργασίας, θα θέλαμε να ευχαριστήσουμε τον επιβλέποντα καθηγητή μας, κ. Πάρι Μαστοροκώστα για την εποικοδομιτική συνεργασία και την πολύτιμη συμβολή του στην ολοκλήρωσή της.

Ευχαριστούμε, επίσης, όλους τους καθηγητές μας για τις γνώσεις και δεξιότητες με τις οποίες μας εφοδίασαν κατά την διάρκεια των σπουδών μας.

Ευχαριστούμε, τέλος, τις οικογένειές μας, που μας στήριξαν τόσο κατά τη διάρκεια των σπουδών, όσο και κατά την συγγραφή της παρούσας εργασίας μας.

Περίληψη

Σκοπός της πτυχιακής αυτής είναι αφενός η ολοκλήρωσή της, ως υποχρέωσή μας για την διεκδίκηση του τίτλου σπουδών, αφετέρου δε, η παρουσίαση ταυτοχρόνως μιας ολοκληρωμένης λύσης που συνδυάζει software και hardware, η οποία μπορεί να καταλήξει στην παροχή τελικού προϊόντος και υπηρεσιών.

Η θεματική περιοχή της πτυχιακής είναι το διαδίκτυο των πραγμάτων (internet of things, IoT), ενώ η εφαρμογή αφορά στο Raspberry Pi.

Σε hardware αναπτύσσουμε έναν μετεωρολογικό σταθμό ο οποίος πέρα από την παροχή δεδομένων τύπου θερμοκρασία, υγρασία, βαρομετρική πίεση κτλ, δεν θα έχει ανάγκη για εξωτερική τροφοδοσία και σύνδεση με το διαδίκτυο. Για να επιτευχθεί αυτό, χρησιμοποιούμε φωτοβολταϊκό πάνελ που κατά την διάρκεια της ημέρας τροφοδοτεί την συσκευή και φορτίζει μια μπαταρία για να λειτουργεί και την νύχτα, καθώς και 2G Module για την σύνδεσή με το διαδίκτυο. Συνεπώς απαιτείται μόνο η ασφαλής τοποθέτηση της συσκευής σε μέρος που υπάρχει φως για λίγες ώρες την μέρα, και κάλυψη από σήμα κινητής τηλεφωνίας, έστω και σε ταχύτητες GPRS. Επιπλέον, η συσκευή είναι εφοδιασμένη με GPS ώστε να μπορεί να εμφανίζει το στίγμα της στον χάρτη από την στιγμή που ενεργοποιείται, χωρίς να απαιτεί περεταίρω ρυθμίσεις απ' τον χρήστη.

Σε software αναπτύσσουμε έναν μετεωρολογικό σταθμό, έναν server και έναν client. Για να επιτευχθεί η επικοινωνία μεταξύ τον server και τον μετεωρολογικό σταθμό, θα χρησιμοποιήσουμε το πρωτόκολλο επικοινωνίας MQTT. Οι μετρήσεις του μετεωρολογικού σταθμού θα αποθηκεύονται μέσα σε σχεσιακή βάση δεδομένων MySQL . Ο server θα χρησιμοποιήσει Nginx για reverse proxy για τον MQTT broker και τον HTTP server που θα αναπτυχθεί με τη βοήθεια της πλατφόρμας ανάπτυξης λογισμικού Node.js και του framework Express.js. Ο client θα αναπτυχθεί χρησιμοποιώντας το framework Angular της Google.

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: μετεωρολογικός σταθμός, αυτόνομο, IoT, MQTT, real-time

Abstract

The purpose of this thesis, is to complete our studies and acquire our degree, but at the same time, to present a complete solution that combines both software and hardware, that can lead to final product and services.

The title of our thesis is defined as followed: “IoT Applications with the use of Raspberry Pi”

As far as hardware is concerned a weather station has been developed, that beyond the feed of information like temperature, humidity, barometric pressure etc, would also not be in need of external power supply and internet connection. To achieve this, we use a solar panel which throughout the day will supply the weather station and also charge the batteries, so that it can still work at night. The station also has a 2G Module for its internet connection. Therefore, the only thing required, is the safe placing of the station, at a place which sees light for a few hours a day and offers mobile network coverage, even with GPRS speeds. Furthermore, our device has a GPS antenna, which will provide its location data on map, from the time of its activation, without the need of further configuration by the user.

In terms of software, we develop a weather station, a server, and a client. To achieve the communication between the server and the weather station, we will use the communication protocol MQTT. Weather station’s measurements will be stored at MySQL’s relational database. Nginx has been employed to reverse proxy for the MQTT broker and for the HTTP server which will be developed with the help of software development platform Node.js and Express.js framework. Client will be developed using Google’s Angular framework.

KEY WORDS: weather station, autonomous, IoT, MQTT, real-time

Περιεχόμενα

Κεφάλαιο 1^{Ο}: Εισαγωγή

1.1 Γενικά

Το αποτέλεσμα της παρούσας πτυχιακής εργασίας προήλθε από τις γνώσεις που αποκτήθηκαν παρακολουθώντας τα μαθήματα Μηχατρονική, ανάπτυξη δικτυακών εφαρμογών, ασφάλεια και διαχείριση δικτυακών συστημάτων, λειτουργικά συστήματα, προγραμματισμός πυρήνα συστημάτων, τεχνολογίες ευρυζωνικών δικτύων, μικροελεγκτές, δίκτυα, μικροηλεκτρονική, θεωρία κυκλωμάτων, ηλεκτρονικά.

1.2 Σκοπός και στόχοι της εργασίας

Το κίνητρο για την εκπόνηση της παρούσας εργασίας δόθηκε από την ιδέα της δημιουργίας μιας μετεωρολογικής υπηρεσίας καθώς και της παροχής εξειδικευμένου υλικού εξοπλισμού, που από κοινού είχαμε οι δύο συγγραφείς. Ο κ. Πάρις Μαστοροκώστας δέχτηκε να μας κατευθύνει στην πραγματοποίηση της ιδέας μας, που ταυτοχρόνως είναι και η πτυχιακή μας εργασία.

Βασικός σκοπός της παρούσας εργασίας είναι (όπως αναφέρεται και στην περίληψη), η δημιουργία μιας μετεωρολογικής υπηρεσίας που θα παρέχει την δυνατότητα σε εταιρίες να αγοράζουν τους αυτόνομους μετεωρολογικούς σταθμούς, με μόνη απαίτηση, την ασφαλή και σταθερή τοποθέτησή τους, σε σημείο που υπάρχει τουλάχιστον GPRS ή 2G σήμα κινητής τηλεφωνίας.

1.3 Δομή της εργασίας

Στο δεύτερο κεφάλαιο θα παρουσιαστεί αναλυτικά ο εξοπλισμός (hardware) που χρησιμοποιήθηκε για την δημιουργία του αυτόνομου μετεωρολογικού σταθμού. Για την διευκόλυνσή μας (συγγραφέων και αναγνωστών), θα χρησιμοποιηθούν υποκεφάλαια για την ανάλυση του κάθε κομματιού του εξοπλισμού, που βρίσκεται μεμονωμένο στην αγορά.

Στο τρίτο κεφάλαιο θα παρουσιαστεί αναλυτικά η συνδεσμολογία και ο συνδυασμός των προαναφερθέντων κομματιών υλικού εξοπλισμού, για την δημιουργία του επιθυμητού αποτελέσματος.

Στο τέταρτο κεφάλαιο θα παρουσιαστούν αναλυτικά τα λογισμικά που χρησιμοποιήθηκαν για την δημιουργία της υπηρεσίας απεικόνισης των πληροφοριών που προκύπτουν απ' τους αισθητήρες του μετεωρολογικού σταθμού. Για την διευκόλυνσή μας, θα χρησιμοποιηθούν υποκεφάλαια για την ανάλυση του κάθε λογισμικού, που βρίσκεται μεμονωμένο προς διανομή στο διαδίκτυο.

Στο πέμπτο κεφάλαιο θα παρουσιαστεί ο συνδυασμός των λογισμικών καθώς και η χρήση των επιμέρους προγραμμάτων και υπηρεσιών που δημιουργήθηκαν με την βοήθεια αυτών, για την δημιουργία της υπηρεσίας απεικόνισης των πληροφοριών.

Στο έκτο κεφάλαιο θα περιγράψουμε το πώς πραγματοποιήσαμε κάποια βασικά tests ώστε να ξέρουμε ότι και το hardware και το software δουλεύουν ακριβώς όπως επιθυμούμε. Επίσης θα αναφέρουμε κάποια απ' τα προβλήματα που αντιμετωπίσαμε κατά την ανάπτυξη του πρακτικού μέρους της παρούσας πτυχιακής.

1.4 BOM (Bill of Materials)

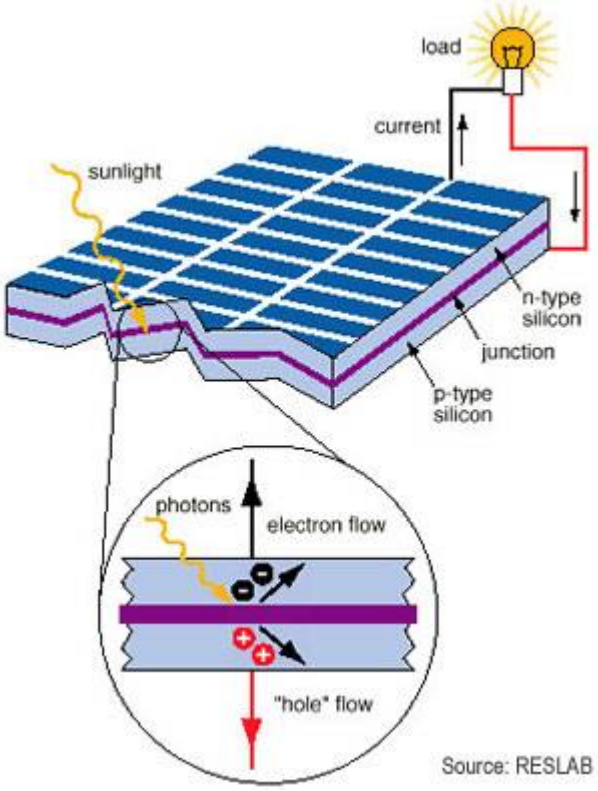
Parts	Price
OEM Solar Panel 20W	25,00 €
Raspberry Pi Zero W Basic Kit	18,20 €
Adafruit Fona 808 Breakout	65,00 €
2x Panasonic 18650 3400mAh NCR18650B	10,40 €
BME280 CJMCU-280E	4,89 €
2A Adjustable Step Up 18650 Lithium Battery Charging Discharge Integrated Module	1,70 €
Clear Acrylic Case For Raspberry Pi Zero & Zero W	1,36 €
Slim Sticker-type GSM/Cellular Quad-Band Antenna - 3dBi uFL	3,60 €
GPS Passive Antenna uFL	6,30 €
XL4005 Step-Down 5A Buck Converter	4,50 €
3x Black Rocker Switch 6A/250V ON/OFF	1,20 €
1 Slot 18650 Battery Holder With 2 Leads	2,40 €
18650 26650 Lithium Li-ion Battery Tester LCD Meter Voltage Current Capacity	5,04 €
Total	149,59 €

Κεφάλαιο 2^{Ο}: Εξοπλισμός

2.1 Φωτοβολταϊκό πάνελ

2.1.1 Τι είναι το φωτοβολταϊκό πάνελ

Τα φωτοβολταϊκά (ή Φ/Β) συστήματα αποτελούν μια από τις εφαρμογές των Ανανεώσιμων Πηγών Ενέργειας, με τεράστιο ενδιαφέρον για την Ελλάδα. Εκμεταλλευόμενο το φωτοβολταϊκό φαινόμενο, το φωτοβολταϊκό σύστημα παράγει ηλεκτρική ενέργεια από την ηλιακή ενέργεια.



2.1.2 Τεχνολογία

Ένα φωτοβολταϊκό σύστημα αποτελείται από ένα ή περισσότερα πάνελ (ή πλαίσια, ή όπως λέγονται συχνά στο εμπόριο, «κρύσταλλα») φωτοβολταϊκών στοιχείων (ή «κυψελών», ή «κυττάρων»), μαζί με τις απαραίτητες συσκευές και διατάξεις για τη μετατροπή της ηλεκτρικής ενέργειας που παράγεται στην επιθυμητή μορφή.

Το φωτοβολταϊκό στοιχείο είναι συνήθως τετράγωνο, με πλευρά 120-160mm. Δυο τύποι πυριτίου χρησιμοποιούνται για την δημιουργία φωτοβολταϊκών στοιχείων: το άμορφο και το κρυσταλλικό πυρίτιο, ενώ το κρυσταλλικό πυρίτιο διακρίνεται σε μονοκρυσταλλικό ή πολυκρυσταλλικό. Το άμορφο και το κρυσταλλικό πυρίτιο παρουσιάζουν τόσο πλεονεκτήματα, όσο και μειονεκτήματα, και κατά τη μελέτη του φωτοβολταϊκού συστήματος γίνεται η αξιολόγηση των ειδικών συνθηκών της εφαρμογής (κατεύθυνση και διάρκεια της ηλιοφάνειας, τυχόν σκιάσεις κλπ.) ώστε να επιλεγεί η κατάλληλη τεχνολογία.

Στο εμπόριο διατίθενται φωτοβολταϊκά πάνελ – τα οποία δεν είναι παρά πολλά φωτοβολταϊκά στοιχεία συνδεδεμένα μεταξύ τους, επικαλυμμένα με ειδικές μεμβράνες και εγκιβωτισμένα σε γυαλί με πλαίσιο από αλουμίνιο – σε διάφορες τιμές ονομαστικής ισχύος, ανάλογα με την τεχνολογία και τον αριθμό των φωτοβολταϊκών κυψελών που τα αποτελούν. Έτσι, ένα πάνελ

36 κυψελών μπορεί να έχει ονομαστική ισχύ 70-85 W, ενώ μεγαλύτερα πάνελ μπορεί να φτάσουν και τα 200 W ή και παραπάνω.

Η κατασκευή μιας γεννήτριας κρυσταλλικού πυριτίου μπορεί να γίνει και από ερασιτέχνες, μετά από την προμήθεια των στοιχείων. Το κόστος είναι απίθανο να είναι χαμηλότερο από την αγορά έτοιμης γεννήτριας, καθώς η προμήθεια ποιοτικών στοιχείων είναι πολύ δύσκολη. Εκτός από το πυρίτιο χρησιμοποιούνται και άλλα υλικά για την κατασκευή των φωτοβολταϊκών στοιχείων, όπως το Κάδμιο - Τελλούριο (CdTe) και ο ινδοδισεληνιούχος χαλκός. Σε αυτές τις κατασκευές, η μορφή του στοιχείου διαφέρει σημαντικά από αυτή του κρυσταλλικού πυριτίου, και έχει συνήθως τη μορφή λωρίδας πλάτους μερικών χιλιοστών και μήκους αρκετών εκατοστών. Τα πάνελ συνδέονται μεταξύ τους και δημιουργούν τη φωτοβολταϊκή συστοιχία, η οποία μπορεί να περιλαμβάνει από 2 έως και αρκετές εκατοντάδες φωτοβολταϊκές γεννήτριες.

Η ηλεκτρική ενέργεια που παράγεται από μια Φ/Β συστοιχία είναι συνεχούς ρεύματος (DC), και για το λόγο αυτό οι πρώτες χρήσεις των φωτοβολταϊκών αφορούσαν εφαρμογές DC τάσης: κλασικά παραδείγματα είναι ο υπολογιστής τσέπης («κομπιουτεράκι») και οι δορυφόροι. Με την προοδευτική αύξηση όμως του βαθμού απόδοσης, δημιουργήθηκαν ειδικές συσκευές – οι αναστροφείς (inverters) - που σκοπό έχουν να μετατρέψουν την έξοδο συνεχούς τάσης της Φ/Β συστοιχίας σε εναλλασσόμενη τάση. Με τον τρόπο αυτό, το Φ/Β σύστημα είναι σε θέση να τροφοδοτήσει μια σύγχρονη εγκατάσταση (κατοικία, θερμοκήπιο, μονάδα παραγωγής κλπ.) που χρησιμοποιεί κατά κανόνα συσκευές εναλλασσόμενου ρεύματος(AC).

2.1.3 Βαθμός απόδοσης

Ο βαθμός απόδοσης εκφράζει το ποσοστό της ηλιακής ακτινοβολίας που μετατρέπεται σε ηλεκτρική ενέργεια στο φωτοβολταϊκό στοιχείο. Τα πρώτα φωτοβολταϊκά στοιχεία, που σχεδιάστηκαν τον 19ο αιώνα, δεν είχαν παρά 1-2% απόδοση, ενώ το 1954 τα εργαστήρια Bell Laboratories δημιούργησαν τα πρώτα Φ/Β στοιχεία πυριτίου με απόδοση 6%. Στην πορεία του χρόνου όλο και αυξάνεται ο βαθμός απόδοσης: η αύξηση της απόδοσης, έστω και κατά μια ποσοστιαία μονάδα, θεωρείται επίτευγμα στην τεχνολογία των φωτοβολταϊκών. Στην σημερινή εποχή ο τυπικός βαθμός απόδοσης ενός φωτοβολταϊκού στοιχείου βρίσκεται στο 13 – 19%, ο οποίος, συγκρινόμενος με την απόδοση άλλου συστήματος (συμβατικού, αιολικού, υδροηλεκτρικού κλπ.), παραμένει ακόμη αρκετά χαμηλός. Αυτό σημαίνει ότι το φωτοβολταϊκό σύστημα καταλαμβάνει μεγάλη επιφάνεια προκειμένου να αποδώσει την επιθυμητή ηλεκτρική ισχύ. Ωστόσο, η απόδοση ενός δεδομένου συστήματος μπορεί να βελτιωθεί σημαντικά με την τοποθέτηση των φωτοβολταϊκών σε ηλιοστάτη. Οι προϋποθέσεις αξιοποίησης των Φ/Β συστημάτων στην Ελλάδα είναι από τις καλύτερες στην Ευρώπη, αφού η συνολική ενέργεια που δέχεται κάθε τετραγωνικό μέτρο επιφάνειας στην διάρκεια ενός έτους κυμαίνεται από 1400-1800 kWh.

2.1.4 Πλεονεκτήματα και μειονεκτήματα

Τα φωτοβολταϊκά συστήματα έχουν τα εξής πλεονεκτήματα:

- Τεχνολογία φιλική στο περιβάλλον: δεν προκαλούνται ρύποι από την παραγωγή ηλεκτρικής ενέργειας
- Η ηλιακή ενέργεια είναι ανεξάντλητη ενεργειακή πηγή, διατίθεται παντού και δεν στοιχίζει απολύτως τίποτα

- Με την κατάλληλη γεωγραφική κατανομή, κοντά στους αντίστοιχους καταναλωτές ενέργειας, τα Φ/Β συστήματα μπορούν να εγκατασταθούν χωρίς να απαιτείται ενίσχυση του δικτύου διανομής
- Η λειτουργία του συστήματος είναι ολοσχερώς αθόρυβη
- Έχουν σχεδόν μηδενικές απαιτήσεις συντήρησης
- Έχουν μεγάλη διάρκεια ζωής: οι κατασκευαστές εγγυώνται τα «κρύσταλλα» για 20-30 χρόνια λειτουργίας
- Υπάρχει πάντα η δυνατότητα μελλοντικής επέκτασης, ώστε να ανταποκρίνονται στις αυξανόμενες ανάγκες των χρηστών
- Μπορούν να εγκατασταθούν πάνω σε ήδη υπάρχουσες κατασκευές, όπως είναι π.χ. η στέγη ενός σπιτιού ή η πρόσοψη ενός κτιρίου,
- Διαθέτουν ευελιξία στις εφαρμογές: τα Φ/Β συστήματα λειτουργούν άριστα τόσο ως αυτόνομα συστήματα, όσο και ως αυτόνομα υβριδικά συστήματα όταν συνδυάζονται με άλλες πηγές ενέργειας (συμβατικές ή ανανεώσιμες) και συσσωρευτές για την αποθήκευση της παραγόμενης ενέργειας. Επιπλέον, ένα μεγάλο πλεονέκτημα του Φ/Β συστήματος είναι ότι μπορεί να διασυνδεθεί με το δίκτυο ηλεκτροδότησης (διασυνδεδεμένο σύστημα), καταργώντας με τον τρόπο αυτό την ανάγκη για εφεδρεία και δίνοντας επιπλέον τη δυνατότητα στον χρήστη να πωλήσει τυχόν πλεονάζουσα ενέργεια στον διαχειριστή του ηλεκτρικού δικτύου, όπως ήδη γίνεται στο Φράιμπουργκ της Γερμανίας.

Ως μειονέκτημα θα μπορούσε να καταλογίσει κανείς στα φωτοβολταϊκά συστήματα το κόστος τους, το οποίο, παρά τις τεχνολογικές εξελίξεις παραμένει ακόμη αρκετά υψηλό. Μια γενική ενδεικτική τιμή είναι 2700 ευρώ ανά εγκατεστημένο κιλοβάτ (kW) ηλεκτρικής ισχύος. Λαμβάνοντας υπόψη ότι μια τυπική οικιακή κατανάλωση απαιτεί από 1,5 έως 3,5 κιλοβάτ, το κόστος της εγκατάστασης δεν είναι αμελητέο. Το ποσό αυτό, ωστόσο, μπορεί να αποσβεστεί σε περίπου 5-6 χρόνια και το Φ/Β σύστημα θα συνεχίσει να παράγει δωρεάν ενέργεια για τουλάχιστον άλλα 25χρόνια. Ωστόσο, τα πλεονεκτήματα είναι πολλά, και το ευρύ κοινό έχει αρχίσει να στρέφεται όλο και πιο πολύ στις ανανεώσιμες πηγές ενέργειας και στα φωτοβολταϊκά ειδικότερα, για την κάλυψη ή την συμπλήρωση των ενεργειακών του αναγκών.

2.1.5 Το πάνελ της εργασίας

Το πάνελ που διαλέξαμε για την κατασκευή είναι 20 watt μονοκρυσταλλικού πυριτίου σειρά SE 20 υψηλής απόδοσης. Περιέχει κυψέλες με αντιανακλαστική επίστρωση και γυαλί για την παραγωγή περισσότερης ενέργειας (περισσότερες kWh ανά kWp). Η απόδοσή του είναι 16,9% σε σχέση με την προσπίπτουσα ηλιακή ενέργεια.

Χαρακτηριστικά:

Rating Power (Pm): 20W
(Vn): 12V
Tolerance: 0 +5%
Rated Voltage (Vm): 17.8V
Rated Current (Im): 1.25A
Open Circuit Voltage (Voc): 21.7 V
Short Circuit Voltage (Isc): 1.27 A
Size 55x35 cm
Weight 1.7 KG
Διαστάσεις 480x350x25mm

2.2 Μπαταρία Ιόντων Λιθίου (Li-Ion)

2.2.1 Τι είναι η μπαταρίες ιόντων λιθίου (Li-Ion)

Μπαταρία ιόντων λιθίου (lithium-ion battery ή Li-ion battery ή LIB) είναι ένας τύπος επαναφορτιζόμενης μπαταρίας στην οποία τα ιόντα λιθίου κινούνται από το αρνητικό ηλεκτρόδιο προς το θετικό ηλεκτρόδιο κατά τη διάρκεια της εκφόρτισης και αντίστροφα κατά τη φόρτιση. Οι μπαταρίες ιόντων λιθίου χρησιμοποιούν μια παρεμβαλλόμενη ένωση του λιθίου ως υλικό του ενός ηλεκτροδίου, συγκρινόμενες με το μεταλλικό λίθιο που χρησιμοποιείται σε μια μη επαναφορτιζόμενη μπαταρία λιθίου. Ο ηλεκτρολύτης, που επιτρέπει την ιονική μετακίνηση και τα δύο ηλεκτρόδια είναι τα συστατικά του στοιχείου μπαταρίας ιόντων λιθίου.

2.2.2 Εφαρμογές

Οι μπαταρίες ιόντων λιθίου παρέχουν ελαφριές πηγές ισχύος υψηλής ενεργειακής πυκνότητας για διάφορες συσκευές. Για να τροφοδοτήσουν μεγαλύτερες συσκευές, όπως ηλεκτρικά οχήματα, η σύνδεση πολλών μικρών μπαταριών σε παράλληλο κύκλωμα είναι πιο αποτελεσματική από τη σύνδεση μιας μοναδικής μεγάλης μπαταρίας. Τέτοιες συσκευές περιλαμβάνουν:

- Φορητές συσκευές: αυτές περιλαμβάνουν κινητά τηλέφωνα, έξυπνα τηλέφωνα, φορητούς υπολογιστές, υπολογιστές ταμπλέτες, ψηφιακές φωτογραφικές μηχανές, φορητές κάμερες, ηλεκτρονικά τσιγάρα, κονσόλες παιχνιδιών χειρός και φακούς.
- Ηλεκτρικά εργαλεία: Οι μπαταρίες ιόντων λιθίου χρησιμοποιούνται σε εργαλεία όπως ασύρματα δράπανα, τριβεία, πριόνια και σε ποικίλο εξοπλισμό κήπου συμπεριλαμβανόμενων χορτοκοπτικών και ψαλιδιών.
- Ηλεκτρικά οχήματα: συμπεριλαμβανομένων ηλεκτρικών αυτοκινήτων, υβριδικών οχημάτων, ηλεκτρικών δίτροχων, προσωπικούς μεταφορείς (personal transporters) και προχωρημένα ηλεκτρικά αμαξίδια. Επίσης τηλεχειριζόμενα μοντέλα, και ειδικά οχήματα.

2.2.3 Απόδοση

Ειδική ενεργειακή πυκνότητα: 100 έως 250 W·h/kg (360 έως 900 kJ/kg)

Ογκομετρική ενεργειακή πυκνότητα: 250 έως 620 W·h/L (900 έως 2230 J/cm³)

Specific power density: 300 έως 1500 W/kg (στα 20 δευτερόλεπτα και 285 W·h/L)

Επειδή οι μπαταρίες ιόντων λιθίου μπορούν να έχουν ποικιλία υλικών θετικού και αρνητικού ηλεκτροδίου, η ενεργειακή πυκνότητα και η τάση ποικίλλουν αντίστοιχα.

Η τάση ανοικτού κυκλώματος είναι υψηλότερη από τις υδατικές μπαταρίες (όπως μπαταρίες μολύβδου-οξέος, νικελίου-μετάλλου υδριδίου και νικελίου-καδμίου). Η εσωτερική αντίσταση αυξάνεται με την επανάληψη της φόρτισης και την ηλικία. Η αύξηση της εσωτερικής αντίστασης προκαλεί την πτώση υπό φορτίο της τάσης στους ακροδέκτες, που μειώνει το μέγιστο ρεύμα υπερέντασης. Ενδεχομένως, η αύξηση της αντίστασης σημαίνει ότι η μπαταρία δεν μπορεί να λειτουργήσει πια για αρκετό χρόνο.

Οι μπαταρίες με θετικό ηλεκτρόδιο φωσφορικού λιθίου-σιδήρου και αρνητικό ηλεκτρόδιο γραφίτη έχουν ονομαστική τάση ανοικτού κυκλώματος 3,2 V και τυπική τάση φόρτισης 3,6 V. Οι μπαταρίες με θετικό ηλεκτρόδιο οξειδίου λιθίου-νικελίου-μαγγανίου-κοβαλτίου(NMC) και αρνητικό ηλεκτρόδιο έχουν ονομαστική τάση 3,7 V με μέγιστη τάση κατά τη φόρτιση 4,2 V. Η διαδικασία φόρτισης εκτελείται σε σταθερή τάση με κύκλωμα περιορισμού ρεύματος (δηλαδή, φόρτιση με σταθερό ρεύμα μέχρι η τάση να φτάσει στα 4,2 V στο στοιχείο και συνέχιση με σταθερή εφαρμοζόμενη φόρτιση μέχρι το ρεύμα να πέσει κοντά στο μηδέν). Συνήθως, η φόρτιση τελειώνει στο 3% του αρχικού ρεύματος φόρτισης. Στο παρελθόν, οι μπαταρίες ιόντων λιθίου δεν μπορούσαν να φορτιστούν γρήγορα και χρειαζόταν τουλάχιστον δύο ώρες για πλήρη φόρτιση. Τα στοιχεία της τρέχουσας γενιάς μπορούν να φορτιστούν πλήρως σε 45 λεπτά ή λιγότερο. Το 2015 ερευνητές επέδειξαν μικρή μπαταρία χωρητικότητας 600 mAh που φορτιζόταν σε ποσοστό 68 τοις εκατό της χωρητικότητας σε δύο λεπτά και μπαταρία 3.000 mAh που φορτιζόταν κατά 48 τοις εκατό σε πέντε λεπτά. Η δεύτερη μπαταρία είχε ενεργειακή πυκνότητα 620 Wh/L. Η συσκευή χρησιμοποίησε ετεροάτομα δεσμευμένα σε μόρια γραφίτη στην άνοδο.

Η απόδοση των κατασκευαζόμενων μπαταριών έχει βελτιωθεί με την πάροδο του χρόνου. Παραδείγματος χάρη, από το 1991 έως το 2005 η ενεργειακή χωρητικότητα ανά τιμή μπαταριών ιόντων λιθίου έχει βελτιωθεί περισσότερο από δέκα φορές, από 0,3Wh ανά δολάριο σε πάνω από 3Wh ανά δολάριο.

2.2.4 Η μπαταρία της εργασίας

Για την εργασία μας επιλέξαμε μπαταρία ιόντων λιθίου τύπου 18650. Συγκεκριμένα την Panasonic NCR18650B 3400mah. Καταλήξαμε στην μπαταρία αυτή μετά από έρευνα αγοράς, για αρκετούς λόγους.

Καταρχάς, χρησιμοποιείται από την Tesla που φτιάχνει αμιγώς ηλεκτρικά αυτοκίνητα, απ’ το 2013 για τα μοντέλα Model S και X. Συνεπώς, κυκλοφορεί καιρό στο εμπόριο, χωρίς να έχει ακουστεί κάποιο «παράπονο». Απ’ το όνομά της προκύπτει ότι η μπαταρία έχει 18 mm διάμετρο και 65 mm ύψος, το οποίο είναι πολύ καλό μέγεθος για τα 3400 mah χωρητικότητας που προσφέρει. Η μπαταρία αυτή επιτρέπει μέγιστο ρεύμα εκφόρτισης 6,7A τα οποία είναι υπέρ αρκετά για την περίπτωσή μας, ενώ το ρεύμα φόρτισης ανέρχεται στα 1,6A, που θα μας δίνει την δυνατότητα να την φορτίζουμε με μόλις 2-3 ώρες ήλιο ημερησίως. Το βάρος της δεν ξεπερνάει τα 48,5gr. Τέλος, υπάρχει πληθώρα κυκλωμάτων φόρτισης/εκφόρτισης, και μετατροπής τάσης από τα 3 – 4,2V που δίνει η μπαταρία στα 5V που χρειαζόμαστε εμείς.

Χαρακτηριστικά:

- Κατασκευαστής: PANASONIC
- Τύπος συσσωρευτή: Li-Ion
- Μέγεθος μπαταρίας: MR18650
- Τάση ονομαστική: 3.6V
- Χωρητικότητα: 3350mAh
- Εξωτερικές διαστάσεις: Ø18.2 x 65mm
- Μέγιστο Ρεύμα: 6.7A
- Unprotected
- Flat top

2.3 Raspberry Pi Zero W

2.3.1 Τι είναι το Raspberry Pi;

Το Raspberry Pi είναι ένας μικρός υπολογιστής σε μέγεθος μιας πιστωτικής κάρτας. Δημιουργήθηκε στο Ηνωμένο Βασίλειο από το Raspberry Pi Foundation για την ευκολότερη εκμάθηση της επιστήμης των υπολογιστών στα σχολεία.

Η πρώτη γενιά (Raspberry Pi 1 model B) δόθηκε στο κοινό τον Φεβρουάριο του 2012. Το 2014 κυκλοφόρησε η νέα βελτιωμένη έκδοση του Raspberry Pi 1 η οποία ονομάστηκε RPi 1 model B+. Τον Απρίλιο του 2014 η εταιρία κυκλοφόρησε τον μικρότερο και οικονομικότερο υπολογιστή τσέπης. Το μοντέλο ονομάστηκε Raspberry Pi Zero και είχε κόστος μόλις 5 $. Τον Φεβρουάριο του 2015 κυκλοφόρησε το Raspberry Pi 2 το οποίο είχε την διπλάσια μνήμη RAM από τα προηγούμενα μοντέλα. Φτάνοντας στο σήμερα, τον Φεβρουάριο του 2016, κυκλοφόρησε στο Raspberry Pi 3 Model B το οποίο έχει ενσωματωμένο WiFi και Bluetooth και ακριβώς το ίδιο μέγεθος με το προηγούμενο.

Το δημοφιλέστερο λειτουργικό σύστημα ονομάζεται Raspbian και φυσικά βασίζεται στο Linux. Επίσης υπάρχουν και άλλα λειτουργικά συστήματα ειδικά σχεδιασμένα για το Raspberry Pi όπως Ubuntu, Windows 10 IOT Core, RISC OS καθώς και διάφορες άλλες εκδόσεις που προσομοιάζουν πλήρως ένα media center σύστημα.

2.3.2 Λόγοι επιλογής της έκδοσης Zero W

Επιλέξαμε το Raspberry Pi Zero W για 2 λόγους:

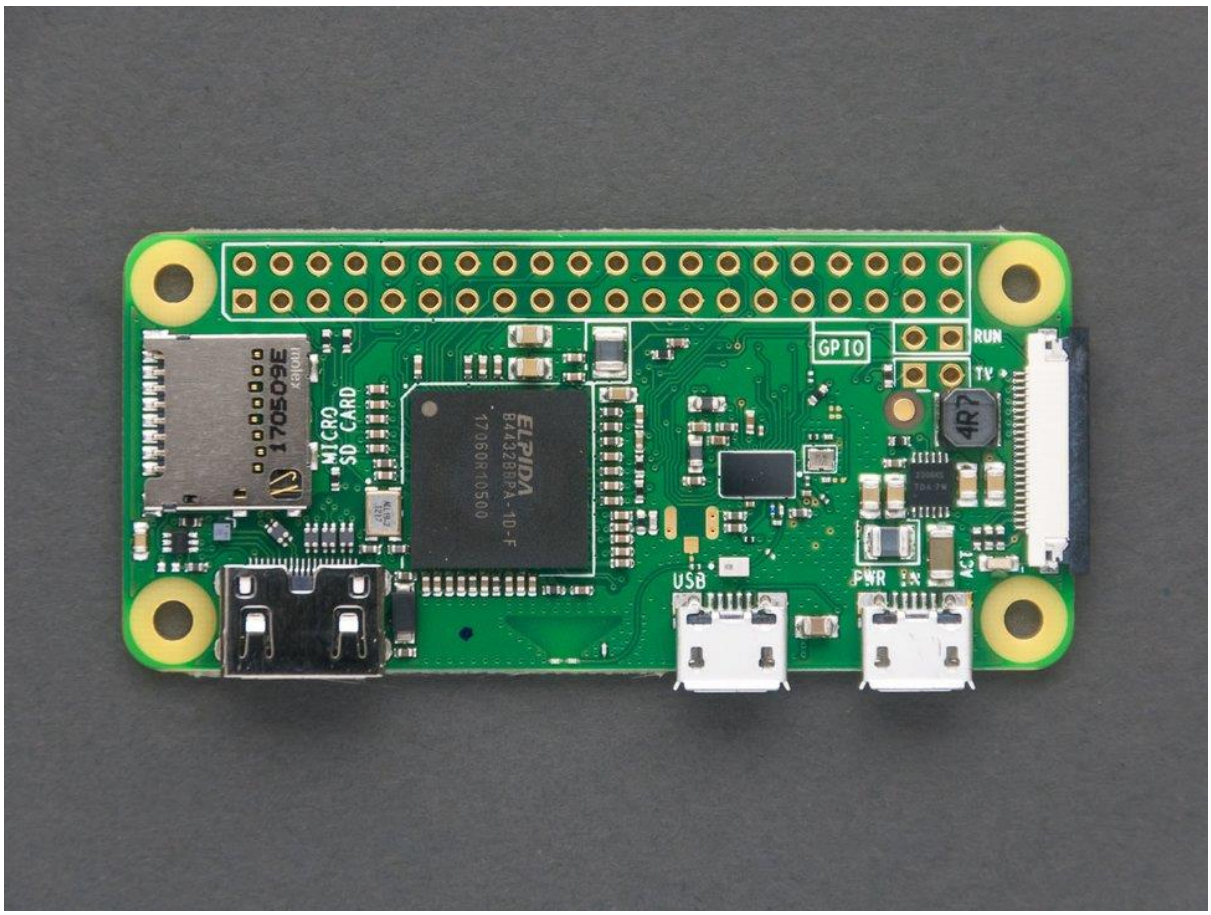
Πρώτος και προφανέστερος, το κόστος. Με μόλις 17€, έχουμε στα χέρια μας έναν υπολογιστή που μπορεί να τρέξει το δικό του UNIX λειτουργικό, το Raspbian, είτε στην πλήρη έκδοση με GUI, είτε στην lite (non-GUI) έκδοση. Η διαφορά του Zero με το Zero W είναι ότι το δεύτερο φέρει κάρτα δικτύου Wi-Fi καθώς και Bluetooth Low Energy έκδοση 4.1, εξού και το W από την λέξη Wireless. Η δυνατότητα αυτή μας επιτρέπει να κάνουμε γρήγορα κάποιες δοκιμές πάνω του, χωρίς να του έχουμε δική του οθόνη με hdmi, πληκτρολόγιο και ποντίκι, καθώς το χειριζόμαστε μέσω SSH.

Δεύτερος λόγος, είναι η κατανάλωση ρεύματος. Σε σχέση με τα υπόλοιπα μοντέλα, χρησιμοποιεί έναν πιο αργό επεξεργαστή, καθώς και λιγότερη ram, για την επίτευξη αυτού. Τα χαρακτηριστικά του όμως για την δική μας χρήση, είναι ήδη υπέρ αρκετά. Συγκεκριμένα, σύμφωνα με την ίδια την Raspberry μετά από μετρήσεις σε όλα τα μοντέλα της, προκύπτουν τα παρακάτω αποτελέσματα:

	Zero	Zero W	A+	A	B+	B	Pi2B	Pi3B
	/mA	/mA	/mA	/mA	/mA	/mA	/mA	/mA
Idling	100	120	100	140	200	360	230	230
Loading LXDE	140	160	130	190	230	400	310	310
Watch 1080p Video	140	170	140	200	240	420	290	290
Shoot 1080p Video	240	230	230	320	330	480	350	350

2.3.3 Χαρακτηριστικά έκδοσης Zero W

- 802.11 b/g/n wireless LAN
- Bluetooth 4.1
- Bluetooth Low Energy (BLE)
- 1GHz, single-core CPU
- 512MB RAM
- Mini HDMI and USB On-The-Go ports
- Micro USB power
- HAT-compatible 40-pin header
- Composite video and reset headers
- CSI camera connector



2.4 Adafruit FONA 808 - Mini Cellular GSM + GPS Breakout Board

2.4.1 Τι είναι το Adafruit FONA 808

Το Adafruit FONA 808 είναι ένα all-in-one κινητό τηλέφωνο σε κύκλωμα το οποίο μπορούμε να χειριστούμε με διάφορους τρόπους. Στην δική μας περίπτωση αυτό γίνεται μέσω σειριακής επικοινωνίας με το Raspberry. Χρησιμοποιεί το SIM808 chipset, το οποίο είναι SMT και μπορούμε να βρούμε ανεξάρτητα στην αγορά. Το chip αυτό κάνει όλες τις λειτουργίες ενός κανονικού κινητού τηλεφώνου (κλήσεις, SMS, δυνατότητα πρόσβασης στο internet κτλ), ενώ περιέχει και το κομμάτι του GPS το οποίο και θα χρησιμοποιήσουμε. Αξίζει να σημειωθεί ότι η πλακέτα της Adafruit FONA 808 - Mini Cellular GSM + GPS Breakout, περιέχει μόνο τα ηλεκτρονικά κομμάτια που απαιτούνται για τις λειτουργίες που προαναφέρθηκαν και για την σωστή λειτουργία του SIM808. Συνεπώς θα πρέπει να προσθέσουμε, κάρτα SIM, κεραία κινητής τηλεφωνίας, κεραία GPS, μπαταρία λιθίου, και τροφοδοσία, εξαρτήματα στα οποία θα αναφερθούμε παρακάτω ή έχουν ήδη αναφερθεί στην παρούσα πτυχιακή.

2.4.2 Χαρακτηριστικά Adafruit FONA 808

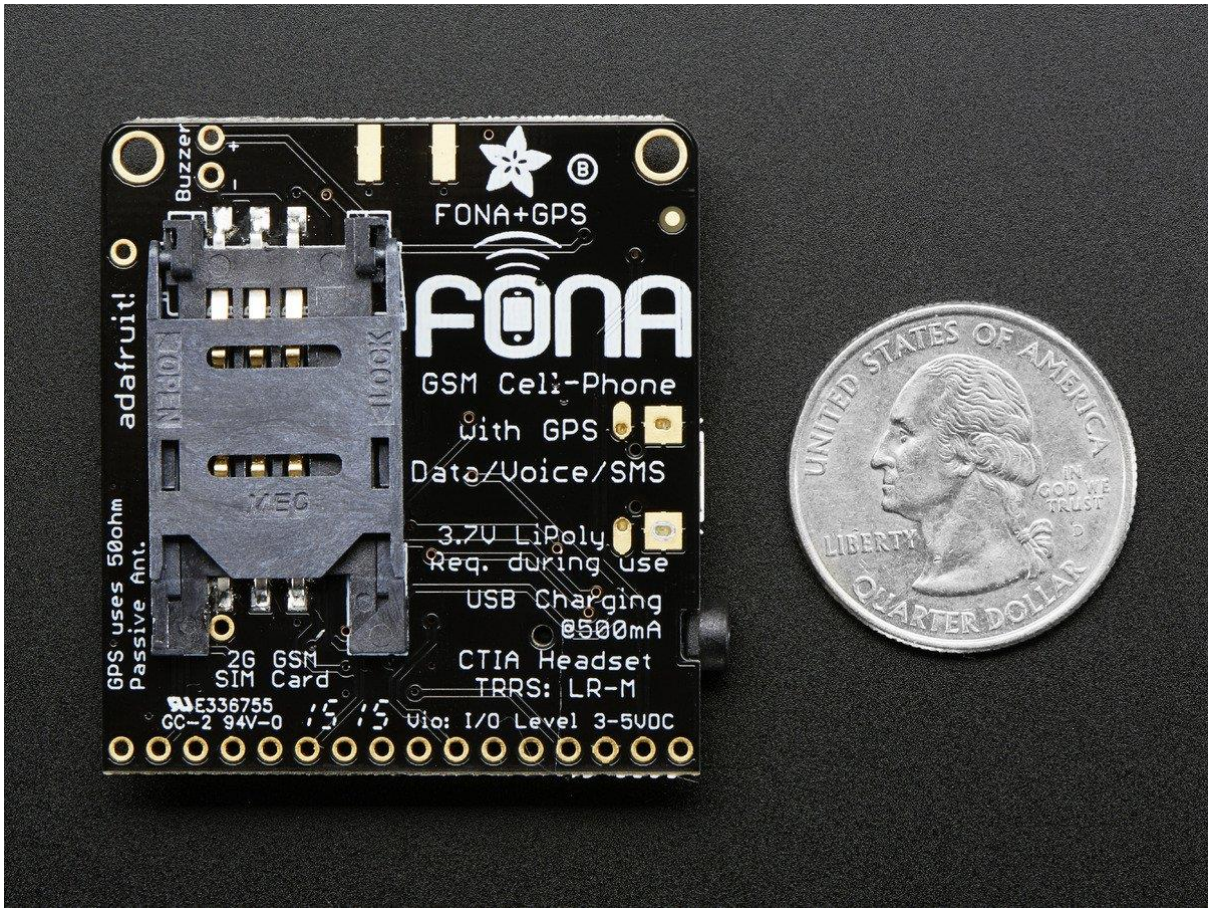
Χαρακτηριστικά που αφορούν το κομμάτι του κινητού:

- Quad-band 850/900/1800/1900MHz - connect onto any global GSM network with any 2G SIM (in the USA, T-Mobile is suggested)
- Fully-integrated GPS (MT3337 chipset with -165 dBm tracking sensitivity) that can be controlled and query over the same serial port
- Make and receive voice calls using a headset or an external 32Ω speaker + electret microphone
- Send and receive SMS messages
- Send and receive GPRS data (TCP/IP, HTTP, etc.)
- PWM/Buzzer vibrational motor control
- AT command interface with "auto baud" detection

Χαρακτηριστικά που αφορούν το κομμάτι του GPS:

- 22 tracking / 66 acquisition channels
- GPS L1 C/A code
- Sensitivity: Tracking: -165 dBm, Cold starts : -147 dBm
- Time-To-First-Fix: Cold starts: 32s (typ.), Hot starts: 1s (typ.), Warm starts: 5s (typ.)
- Accuracy: approx 2.5 meters


Key
GSM ANT
Net
PWR
PWM1
SIMCom
SIM808
S2-10606-Z1F01
SN: MP0615162512907
IMEI: 865067020356971
GPS ANT
ADC
5V
BatGND
Vio
Rst
RX
TX
RTS
Key
RI
PStat
NS
+Mic-
+SPKR-


Buzzer
FONA+GPS
adafruit!
FONA
GSM Cell-Phone
With GPS
Data/Voice/SMS
3.7V LiPoly
Req. during use
USB Charging
@500mA
GPS uses 50ohm
Passive Ant.
2G GSM
SIM Card
CTIA Headset
TRRS: LR-M
E336755
GC-2 94V-0
Vio: I/O Level 3-5VDC
UNITED STATES OF AMERICA
IN GOD WE TRUST
LIBERTY
QUARTER DOLLAR

2.5 Διάφορα εξαρτήματα (sensors, μικροηλεκτρονικά, κτλ)

2.5.1 CJMCU-280E BME280 High Precision Sensor

Το CJMCU-280E είναι μια πλακέτα που φέρει τον αισθητήρα της Bosch BME280 (SMT Device) και βγάζει pinout για I2C ή SPI interface απ' το οποίο θα επικοινωνεί με το Raspberry. Ο παραπάνω αισθητήρας μετράει θερμοκρασία, υγρασία, και βαρομετρική πίεση. Είναι εξαιρετικά μικρός, με αρκετά μεγάλη ακρίβεια και μικρό κόστος.

Χαρακτηριστικά:
Relative humidity range: 0 ~ 100%
The temperature range: -40°C ~ +85°C
Humidity measurement response time: 1S
Hysteresis: 2% relative humidity
Pressure range: 300 ~ 1100 hPa
Absolute pressure accuracy: absolute ± 1hPa (after soldering)
Absolute temperature accuracy: ±0.5°C (at 25°C)
Communication interface: I2C, SPI

2.5.2 XL4005 Step-Down 5A Buck Converter

Η τάση εξόδου του φωτοβολταϊκού, είναι μεταβλητή και ανάλογη του ποσού του φωτός που δέχεται απ' το περιβάλλον. Συνεπώς χρειαζόμαστε ένα regulator που θα κάνει τις τάσεις μεγαλύτερες των 5V ακριβώς 5V, ενώ θα μπορεί να δώσει έως και 3A για να είμαστε σίγουροι ότι μπορούμε τροφοδοτήσουμε τους φορτιστές των μπαταριών λιθίου καθώς και τα κυκλώματά μας. Το XL4005 που δέχεται είσοδο 4 – 38 V τάσης και το μέγιστο ρεύμα εξόδου είναι 5A. Τα χαρακτηριστικά του υπερκαλύπτουν τις ανάγκες μας, διότι το φωτοβολταϊκό μας βγάζει 20W (20V, 1A), στις καλύτερες δυνατές συνθήκες.

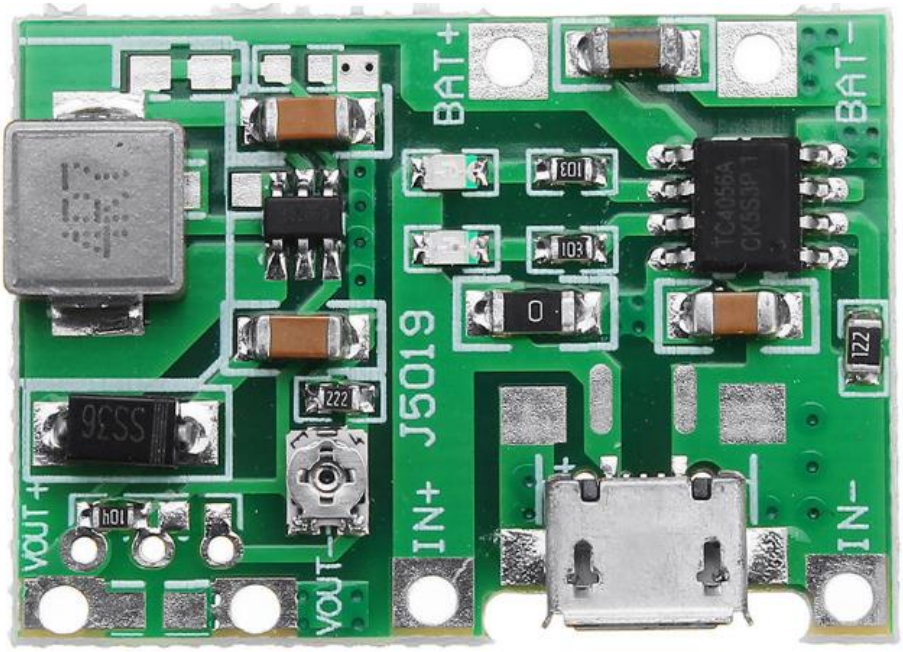
2.5.3 18650 Charger and Voltage Booster/Regulator

Το XL4005 στο οποίο προαναφερθήκαμε δίνει σταθερή τάση 5V. Η μπαταρία για να φορτίσει θέλει 4,2V και ρεύμα φόρτισης έως 1A. Το ρεύμα θα πρέπει επίσης να διακόπτεται αυτόματα όταν η μπαταρία φτάσει το 100%, δηλαδή όταν είναι πλήρως φορτισμένη. Όταν το φωτοβολταϊκό δεν έχει αρκετό φως ώστε να σχηματιστεί διαφορά δυναμικού στα άκρα του μεγαλύτερη των 5V, θα πρέπει αυτομάτως να τραβάει το κύκλωμα ρεύμα απ' την μπαταρία. Η μπαταρία δίνει τάση 4,2V όταν είναι γεμάτη και 3V όταν είναι άδεια. Συνεπώς θα πρέπει να υπάρξει κύκλωμα που θα κάνει τις παραπάνω τάσεις 5V ώστε να λειτουργούν το Raspberry και το Adafruit Fona 808. Όλα τα παραπάνω «προβλήματα» έρχεται να λύσει η παρακάτω πλακέτα φόρτισης/εκφόρτισης.

Στους ακροδέκτες IN+ και IN- απαιτεί τάση εισόδου 4,5 – 8 V, στους οποίους συνδέουμε την έξοδο του LM2576T. Στους BAT+ και BAT- συνδέεται η μπαταρία. Πάνω απ' το VOUT- βλέπουμε ένα σταυρό. Αυτό είναι το ποτενσιόμετρο για την τάση εξόδου στους ακροδέκτες VOUT+ και VOUT-. Η τάση αυξάνεται αριστερόστροφα και έχει προσαρμοστεί ώστε να βγάζει 5.1V που είναι η καλύτερη τάση λειτουργίας για το Raspberry. Το κύκλωμα αυτό τροφοδοτεί ταυτόχρονα το Raspberry και φορτίζει την μπαταρία, ενώ όταν σταματήσει η τάση εισόδου, συνεχίζει να δίνει ρεύμα στην έξοδο απ' την μπαταρία.

Χαρακτηριστικά κυκλώματος:

Input voltage: 4.5-8V
Output voltage: continuous adjustable 4.3-27V (Counter clockwise to boost voltage)
Charging voltage: 4.2V
Charging current: up to 1A
Discharge current: the maximum 2A
Output reference maximum current: 5V 1.4A, 9V 0.8A, 12V 0.6A
Quiescent current: about 0.5 mA
Overshoot protection: Yes
Over discharge protection: None (also can be said yes, because 2V cut-off boost but does not block the output, output has 2V)
Deadline boost:: 2V, depending on whether added the additional protection board
Size: 3.3 x 2.3 x 0.9cm



2.5.4 18650 Battery Tester/Monitor

Ως μηχανικοί σκεφτήκαμε ότι θα ήταν πολύ χρήσιμο να έχουμε την δυνατότητα να βλέπουμε αν η μπαταρία φορτίζει, καθώς και πόση απομένει την ώρα που δεν φορτίζει. Επίσης, η συσκευή που θα το έκανε αυτό θα ήταν εξίσου χρήσιμο να έχει φωτιζόμενη ένδειξη ή οθόνη καθώς η μπαταρία εκφορτίζει όταν δεν υπάρχει αρκετό φως. Οποιεσδήποτε παραπάνω δυνατότητες όπως μέτρηση κατανάλωσης ρεύματος, ωμική αντίσταση κυκλώματος κτλ, θα ήταν αδιαμφησβήτητα “nice to have”.

Καταλήξαμε στο παρακάτω 18650 Battery Tester/Monitor, με τα εξής χαρακτηριστικά και την συνδεσμολογία της τρίτης κατά σειρά εικόνας. Το παρακάτω module έχει ρυθμιστεί να δείχνει γεμάτη την μπαταρία όταν είναι στα 4,2V και άδεια όταν είναι στα 3V.

Χαρακτηριστικά:

Measure the battery voltage
Range: 2.8~3.0V
Resolution: 0.1V
Accuracy: 2%
Measure the battery capacity
Battery capacity is the battery grid form on the LCD display, a total of five girds.
Battery capacity is distinguished by the battery voltage
Displaying X0 grids stand for full capacity, X1 girds for 4/5 capacity, X2 grids for 3/5 capacity, X3 grids for 2/5 capacity, X4 grid for 1/5 capacity, X5 grid for 0/5 capacity. (default: 3.7V full capacity, 2.8V is exhausted)
Can be set the full capacity 2.9v-30.0V, the exhausted 2.8V-29.9V

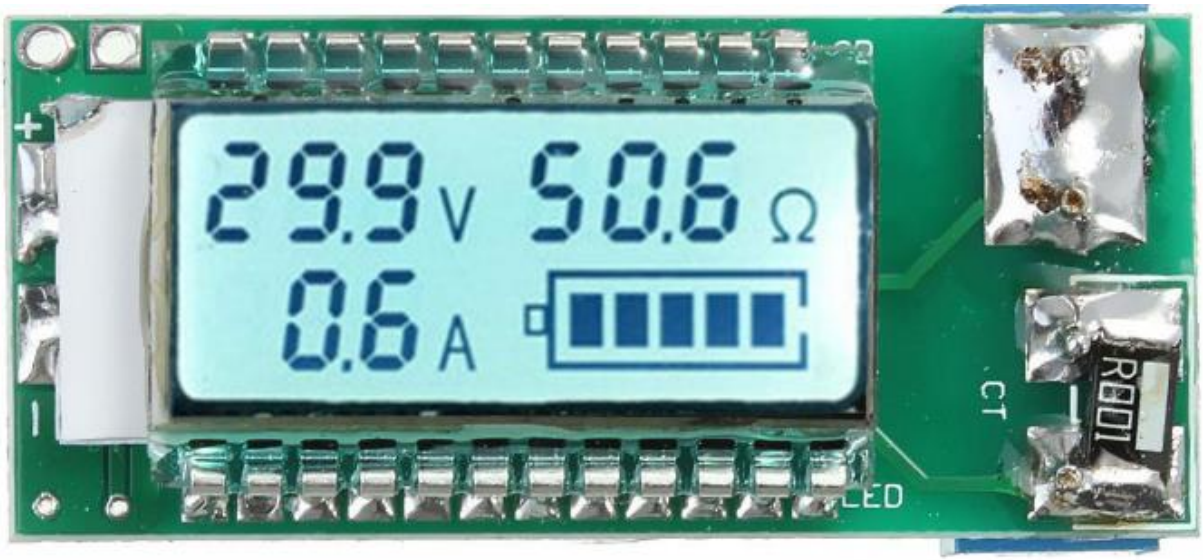
Measure the current:
Range: 0.0~10.0A
Resolution: 0.1A
Accuracy: 2%

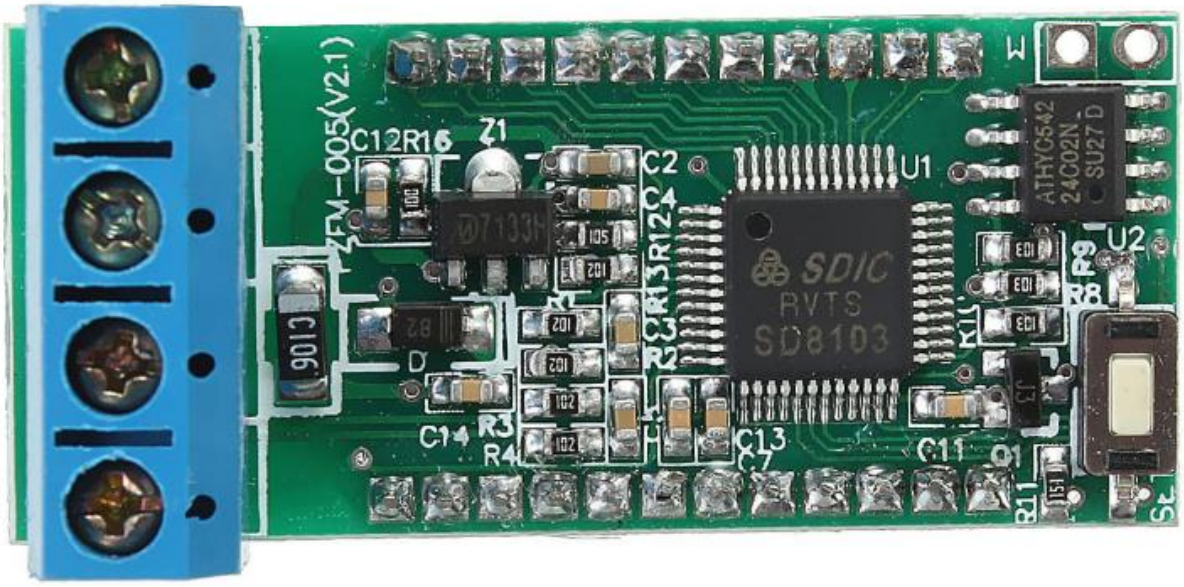
Measure the resistance of the current:
Range: 0.0~10.0A
Resolution: 0.1A
Accuracy: 2%

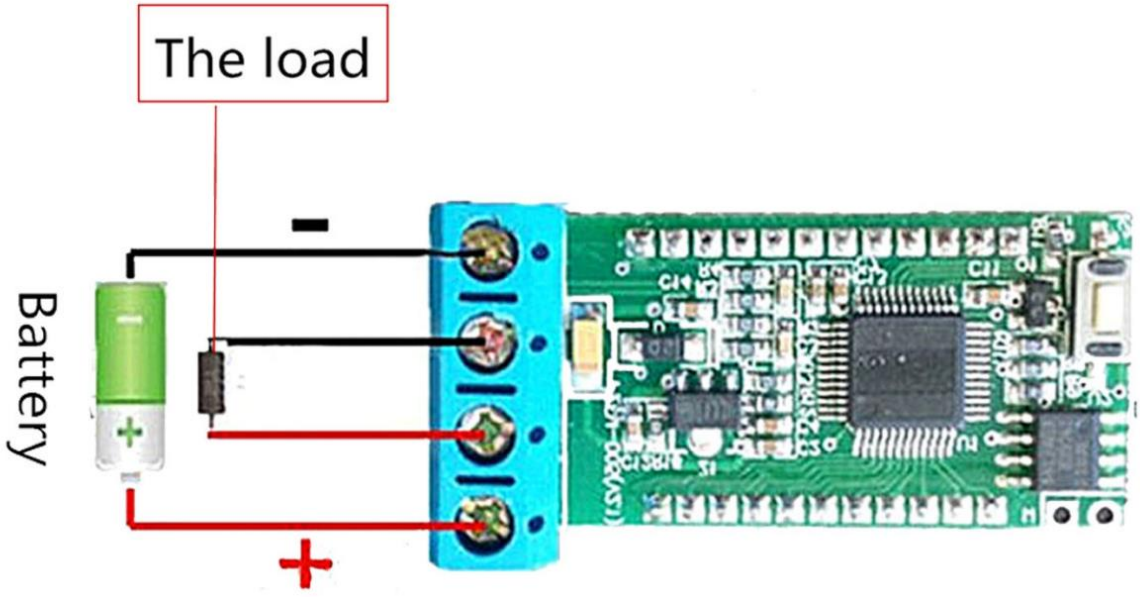
Measure the resistance of the load:
Range: 0~999Ω, when over 999Ω display “OL”
Resolution: when less than 10Ω, resolution is 0.01Ω
When less than 100Ω, resolution is 0.1 Ω
When less than 1000Ω, resolution is 1 Ω
Accuracy: 2%
With Backlight, default off-backlight, you can turn on/off the backlight by button

Specifications:

Size: 42 x 19 x 12mm
Display size: 25 x 12mm
Note: battery: ≥2.8V
Working temperature: -10℃ to 40℃

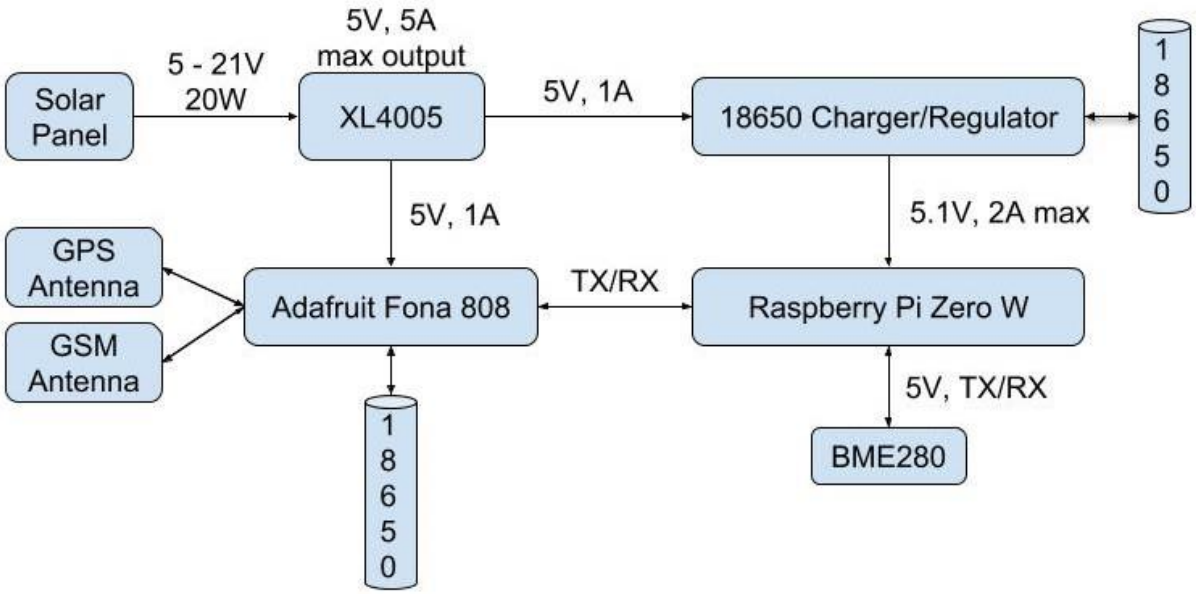

29.9 V
50.6 Ω
0.6 A
R001
CT
LED


ZFM-005(V2.1)
C12
R16
Z1
C2
U1
R13
R12
C4
R2
C3
R3
R4
C14
C13
C11
Q1
R11
U2
R9
R8
K1
SDIC
RVTS
SD8103
ATHYC542
24C02N
SU27 D


The load
Battery
-
+

Κεφάλαιο 3ο: Συνδεσμολογία υλικού εξοπλισμού

3.1 Λογικό διάγραμμα λειτουργίας του κυκλώματος



Στο παραπάνω διάγραμμα αποτυπώνονται όλα τα ενεργά και απαραίτητα στοιχεία του κυκλώματος για την δημιουργία ενός λειτουργικού κυκλώματος. Έχουν παραλειφθεί όργανα μετρήσεων και οθόνες ενδείξεων.

3.2 Επεξήγηση συνδεσμολογιών

Ξεκινώντας από πάνω αριστερά, το φωτοβολταϊκό πάνελ, παρέχει τροφοδοσία τάσης 5 – 21V και μέγιστης ισχύος 20W, ανάλογα με την ποσότητα του φωτός στον χώρο. Στην συνέχεια, χρησιμοποιούμε το XL4005 το οποίο είναι ρυθμισμένο να βγάζει για οποιαδήποτε είσοδο, ακριβώς 5V στην έξοδο, με μέγιστο ρεύμα τα 5A. Βάσει χαρακτηριστικών, ο φορτιστής και προαγωγός τάσης καταναλώνει 1A max για την φόρτιση της μπαταρίας, ενώ μπορεί να παρέχει ρεύμα 2A max εάν χρειαστεί. Προφανώς τα 2A παρέχονται μόνο αν η μπαταρία είναι φορτισμένη διότι το ρεύμα εισόδου δεν είναι μεγαλύτερο του 1A. Στην περίπτωσή μας δεν θα χρειαστεί διότι το Raspberry δεν ξεπερνάει τα 200mA κατανάλωση. Ο φορτιστής όταν δέχεται τάση στην είσοδο φορτίζει την μπαταρία με όσο ρεύμα περισσέψει αφού τροφοδοτήσει το Raspberry, ενώ όταν διακοπεί ή δεν επαρκεί το ρεύμα, το Raspberry λειτουργεί με την ενέργεια που είναι αποθηκευμένη στην μπαταρία. Η μπαταρία έχει τάση εξόδου 3 – 4.2 V από άδεια έως γεμάτη και όταν βρίσκεται σε χρήση ο φορτιστής λειτουργεί ως προαγωγός των τάσεων της μπαταρίας, σε 5V. Το XL4005 τροφοδοτεί επίσης το Adafruit Fona 808. Το Fona είναι φτιαγμένο έτσι ώστε να μην λειτουργεί χωρίς μπαταρία τύπου LiPo ή Li-ion συνδεδεμένη πάνω του, οπότε αναγκαστικά προσθέσαμε άλλη μια μπαταρία. Ως κινητό, περιλαμβάνει κύκλωμα φόρτισης, οπότε όταν δέχεται τάση στην είσοδο φορτίζει την μπαταρία, όταν δεν δέχεται, τροφοδοτείται απ’ την δική του μπαταρία. Στο Raspberry έχει συνδεθεί ο αισθητήρας BME280, τον οποίο και τροφοδοτεί. Η κατανάλωση του είναι πραγματικά αμελητέα. Σύμφωνα με το official datasheet, ανέρχεται στα 3.6μA @1Hz refresh rate. Το Fona στέλνει στο Raspberry τις πληροφορίες που συλλέγει απ’ το GPS. Το Raspberry συλλέγει αυτές του BME280 και του GPS μαζί. Τέλος το πακέτο αυτό, αποστέλλεται απ’ το Raspberry στο Fona, και απ’ το Fona μέσω της GSM κεραίας και του 2G δικτύου, στον server.

Κεφάλαιο 4ο: Λογισμικά

4.1 Node.js

4.1.1 Τι είναι το Node.js

Το Node.js είναι μια πλατφόρμα ανάπτυξης λογισμικού (κυρίως διακομιστών) χτισμένη σε περιβάλλον JavaScript. Στόχος του Node είναι να παρέχει ένα εύκολο τρόπο δημιουργίας κλιμακωτών διαδικτυακών εφαρμογών. Σε αντίθεση από τα περισσότερα σύγχρονα περιβάλλοντα ανάπτυξης εφαρμογών δικτύων μία διεργασία node δεν στηρίζεται στην πολυνηματικότητα αλλά σε ένα μοντέλο ασύγχρονης επικοινωνίας εισόδου/εξόδου.

4.1.2 Ιστορία

Το Node.js δημιουργήθηκε από τον Ryan Dahl το 2009. Η δημιουργία και η συντήρηση του έργου χορηγήθηκε από την εταιρία Joyent. Η ιδέα για την ανάπτυξη του node προήλθε από την ανάγκη του Ryan Dahl να βρει τον πιο αποδοτικό τρόπο να ενημερώνει τον χρήστη σε πραγματικό χρόνο για την κατάσταση ενός αρχείου που ανέβαζε στο διαδίκτυο. Επίσης επηρεάστηκε από το Mongrel του Zed Shaw. Επιπροσθέτως μετά από αποτυχημένα έργα σε C, Lua, Haskell η κυκλοφορία της μηχανής V8 (V8 JavaScript Engine) της Google τον ώθησε να ασχοληθεί με την JavaScript.

4.1.3 Χαρακτηριστικά

Το Node χαρακτηρίζεται από την έμφαση στην ασύγχρονη επικοινωνία μεταξύ των υπολογιστικών πόρων. Αυτό επιτυγχάνεται με την χρήση συμβάντων (events) που προσφέρει η Javascript και ονομάζονται callbacks. Για παράδειγμα όταν ένας περιηγητής ιστού φορτώσει πλήρως ένα αρχείο, ένας χρήστης πατάει κάποιο κουμπί, ολοκληρώνεται ένα αίτημα AJAX, τα συμβάντα αυτά πυροδοτούν ένα συγκεκριμένο callback. Αυτό με την σειρά του επιτρέπει την ροή του κώδικα χωρίς να αφήνει ανενεργό τον επεξεργαστή προκειμένου να εκτελεστεί μια λειτουργία, όπως μια επιτυχής ανάγνωση αρχείου από τον δίσκο.

4.1.4 Πού χρησιμοποιήθηκε η Node.js

Χρησιμοποιήσαμε Node.js στον server για την δημιουργία της ιστοσελίδας και του MQTT Broker, καθώς και στον μετεωρολογικό σταθμό για την συλλογή δεδομένων και την αποστολή τους στον server, μέσω του πρωτοκόλλου MQTT.

4.1.5 Γιατί Node.js αντί για Python

Η Python είναι η κύρια γλώσσα που χρησιμοποιείται στο IoT. Η JavaScript όμως, μας δίνει την δυνατότητα να χρησιμοποιήσουμε την ίδια γλώσσα στον server, client και στον μετεωρολογικό σταθμό. Επίσης, εξίσου σημαντικό είναι ότι η JavaScript προσφέρει πολύ καλύτερες επιδόσεις σε σχέση με την Python 3, σύμφωνα με τα παρακάτω benchmarks.

n-body

source	secs	mem	gz	cpu	cpu load
Node js	**26.31**	33,476	1297	26.30	0% 0% 1% 100%
Python 3	850.24	7,916	1242	850.06	0% 0% 0% 100%

mandelbrot

source	secs	mem	gz	cpu	cpu load
Node js	**18.21**	605,668	748	65.22	84% 82% 96% 97%
Python 3	263.04	51,780	688	1,050.80	100% 100% 100% 100%

spectral-norm

source	secs	mem	gz	cpu	cpu load
Node js	**15.81**	31,780	381	15.81	2% 100% 1% 2%
Python 3	182.12	52,752	443	705.87	96% 96% 96% 100%

fasta

source	secs	mem	gz	cpu	cpu load
Node js	**9.15**	37,052	1785	9.20	27% 0% 1% 74%
Python 3	63.19	680,688	1947	135.52	62% 70% 63% 22%

fannkuch-redux

source	secs	mem	gz	cpu	cpu load
Node js	**80.97**	30,924	473	80.95	0% 100% 1% 0%
Python 3	507.56	50,988	950	1,998.88	99% 99% 97% 99%

binary-trees

source	secs	mem	gz	cpu	cpu load
Node js	**23.87**	355,268	431	39.40	36% 30% 72% 35%
Python 3	83.95	451,732	589	290.57	88% 87% 87% 97%

regex-redux

source	secs	mem	gz	cpu	cpu load
Node js	**12.05**	854,584	408	12.87	29% 47% 21% 12%
Python 3	16.98	445,760	512	31.44	28% 74% 47% 38%

k-nucleotide

source	secs	mem	gz	cpu	cpu load
Node js	**64.56**	1,819,728	935	137.90	69% 74% 92% 76%
Python 3	72.80	189,720	1967	276.09	95% 96% 98% 94%

reverse-complement

source	secs	mem	gz	cpu	cpu load
Node js	16.61	708,876	1103	18.16	11% 54% 14% 31%
Python 3	16.03	1,007,016	814	19.29	18% 59% 44% 23%

pidigits

source	secs	mem	gz	cpu	cpu load
Node js		Bad Output			
Python 3	1.21	?	386	?	0% 3% 18% 48%

Node js	v11.9.0
Python 3	Python 3.7.1

4.2 MySQL

4.2.1 Τι είναι η MySQL

Η MySQL είναι ένα σύστημα διαχείρισης σχεσιακών βάσεων δεδομένων που μετρά περισσότερες από 11 εκατομμύρια εγκαταστάσεις. Έλαβε το όνομά της από την κόρη του Μόντυ Βιντένιους, τη Μάι (αγγλ. My). Το πρόγραμμα τρέχει έναν εξυπηρετητή (server) παρέχοντας πρόσβαση πολλών χρηστών σε ένα σύνολο βάσεων δεδομένων.

Ο κωδικός του εγχειρήματος είναι διαθέσιμος μέσω της GNU General Public License, καθώς και μέσω ορισμένων ιδιόκτητων συμφωνιών. Ανήκει και χρηματοδοτείται από μία και μοναδική κερδοσκοπική εταιρία, τη σουηδική MySQL AB, η οποία σήμερα ανήκει στην Oracle.

4.2.2 Χρήσεις

Η MySQL είναι δημοφιλής βάση δεδομένων για διαδικτυακά προγράμματα και ιστοσελίδες. Χρησιμοποιείται σε κάποιες από τις πιο διαδεδομένες διαδικτυακές υπηρεσίες, όπως το Flickr, το YouTube, η Wikipedia, το Google, το Facebook και το Twitter.

4.2.3 Λόγοι χρήσης MySQL

Χρησιμοποιούμε MySQL βάση δεδομένων, για να αποθηκεύσουμε τις μετρήσεις που παίρνουμε από το μετεωρολογικό σταθμό. Προτιμήσαμε την MySQL διότι είναι πλέον το πιο διαδεδομένο σύστημα διαχείρισης σχεσιακών βάσεων δεδομένων.

4.3 Frameworks και Modules

4.3.1 Στον Server

4.3.1.1 Express.js

Το Express.js ή απλά Express, είναι ένα framework που χρησιμοποιείται για κατασκευή διαδικτυακών εφαρμογών, βασισμένο στο Node.js, και στην γλώσσα JavaScript. Έχει αποκαλεστεί ως το “de facto” framework για κατασκευή διαδικτυακών εφαρμογών στο Node.js.

Το Express.js ιδρύθηκε από τον TJ Holowaychuk. Η πρώτη έκδοση, σύμφωνα με το GitHub repository, ήταν στις 22 Μαΐου 2010, έκδοση 0.12.0.

Τον Ιούνιο του 2014, τα δικαιώματα για την διαχείριση του project αποκτήθηκαν από την StrongLoop. Η StrongLoop αποκτήθηκε με την σειρά της, από την IBM τον Σεπτέμβριο του 2015.

4.3.1.2 Sequelize

Το Sequelize είναι promise-based ORM (Object-relational mapping) για την Node.js. Υποστηρίζει PostgreSQL, MySQL, SQLite και MSSQL.

4.3.1.3 Επιπλέον Modules

- Mosca – MQTT Broker
- MySQL2 – Driver για MySQL

4.3.2 Στον Client

4.3.2.1 Angular (web framework)

Το Angular είναι full-stack web application framework γραμμένο στην γλώσσα TypeScript (η οποία μεταγλωττίζεται σε JavaScript). Δημιουργήθηκε από την Angular Team στην Google και από μια κοινότητα ατόμων και οργανισμών.

4.3.2.2 Bootstrap

Το Bootstrap είναι μια συλλογή εργαλείων ανοιχτού κώδικα (Ελεύθερο λογισμικό) για τη δημιουργία ιστοσελίδων και διαδικτυακών εφαρμογών. Περιέχει HTML και CSS για τις μορφές τυπογραφίας, κουμπιά πλοήγησης και άλλων στοιχείων του περιβάλλοντος, καθώς και προαιρετικές επεκτάσεις JavaScript. Είναι το πιο δημοφιλές πρόγραμμα στο GitHub και έχει χρησιμοποιηθεί από τη NASA και το MSNBC, μεταξύ άλλων.

Το Bootstrap αναπτύχθηκε από τον Mark Otto και τον Jacob Thornton στο Twitter ως ένα πλαίσιο για την ενθάρρυνση της συνέπειας στα εσωτερικά εργαλεία. Πριν το Bootstrap, διάφορες βιβλιοθήκες χρησιμοποιήθηκαν για την ανάπτυξη της διεπαφής, η οποία οδήγησε σε αντιφάσεις και υψηλή φορολογική επιβάρυνση συντήρησης. Σύμφωνα με τον Twitter developer Mark Otto, για την αντιμετώπιση αυτών των προκλήσεων : «... πήρα μαζί μία σούπερ μικρή ομάδα προγραμματιστών για να σχεδιάσει και να κατασκευάσει ένα νέο εργαλείο εσωτερικής και είδα μια ευκαιρία να κάνουμε κάτι περισσότερο. Μέσω αυτής της διαδικασίας, είδαμε τους εαυτούς μας να χτίζουν κάτι πολύ πιο σημαντικό από οποιοδήποτε άλλο εσωτερικό εργαλείο. Μήνες αργότερα, καταλήξαμε σε μια πρώιμη έκδοση του Bootstrap ως έναν τρόπο για να καταγράψουμε και να μοιραστούμε κοινά πρότυπα σχεδιασμού και περιουσιακών στοιχείων εντός της εταιρείας».

Η πρώτη εγκατάσταση υπό πραγματικές συνθήκες συνέβη κατά τη διάρκεια της πρώτης Hackweek του Twitter. Ο Mark Otto έδειξε σε κάποιους συναδέλφους πώς να επιταχύνουν την ανάπτυξη του έργου τους με τη βοήθεια της εργαλειοθήκης. Ως αποτέλεσμα, δεκάδες ομάδες έχουν μετακινηθεί στο πλαίσιο.

Τον Αύγουστο του 2011 κυκλοφόρησε Twitter Bootstrap ως λογισμικό ανοιχτού κώδικα. Τον Φεβρουάριο του 2012, ήταν το πιο δημοφιλές έργο ανάπτυξης στο GitHub.

4.3.2.3 Επιπλέον Modules

- Chart.js – Για την δημιουργία διαγραμμάτων στην ιστοσελίδα
- Moment – Χρησιμοποιείται για να κάνουμε format τις ημερομηνίες
- MQTT – Client για σύνδεση με τον MQTT Broker (server)

4.3.3 Στον μετεωρολογικό σταθμό

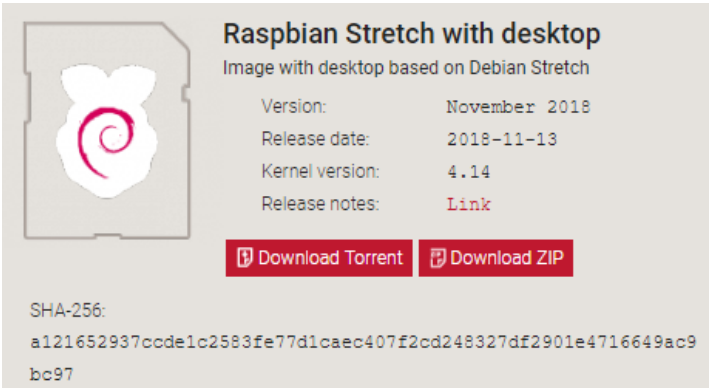
- @agilatech/bme280 – Driver για το bme280
- MQTT – Για σύνδεση με τον MQTT Broker
- serialport – Για την σειριακή επικοινωνία με το Fona 808

Κεφάλαιο 5ο: Χρήση λογισμικών για την δημιουργία του επιθυμητού αποτελέσματος

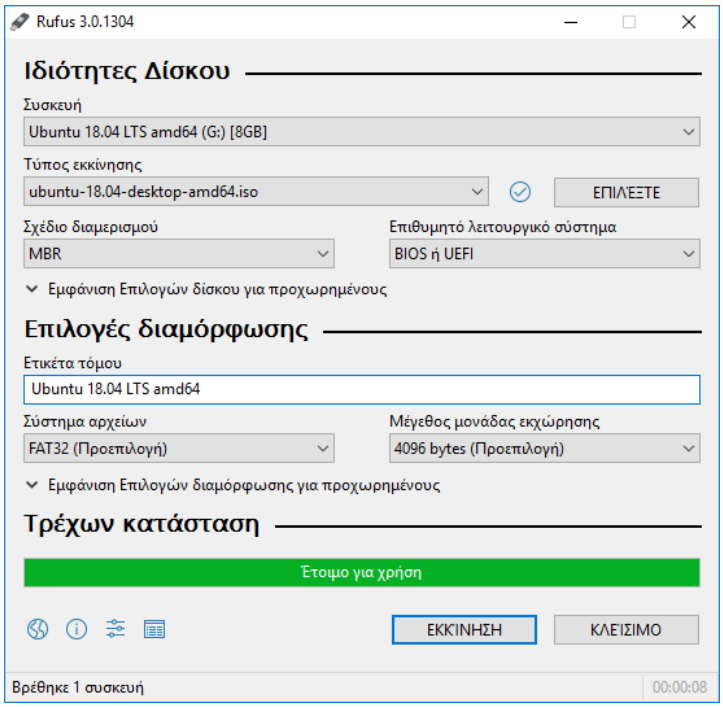
5.1 Εγκατάσταση λογισμικού στον μετεωρολογικό σταθμό

5.1.1 Raspbian Stretch with Desktop

Από την επίσημη ιστοσελίδα του Raspberry κατεβάσαμε το δωρεάν λειτουργικό σύστημα Raspbian Stretch with Desktop.



Στην συνέχεια με τη χρήση του Rufus, φορτώσαμε το λειτουργικό σύστημα σε μια microsd απ' την οποία θα κάνει boot το Raspberry.



Στην συνέχεια τοποθετήσαμε την microsd στο raspberry και ακολουθήσαμε τα βήματα για σύνδεση στο WiFi για τις βασικές ενημερώσεις, ρυθμίσεις και εγκαταστάσεις λογισμικών.

5.1.2 Εγκατάσταση λογισμικών

Τα λογισμικά που χρησιμοποιήσαμε εγκαταστάθηκαν μέσω του Terminal με τις παρακάτω εντολές:

- Ενημέρωση λογισμικού - sudo apt-get update
- Αναβάθμιση λογισμικού - sudo apt-get upgrade
- Node.js - sudo apt-get install nodejs
- PPP (Point-to-Point Protocol) - sudo apt-get install ppp
- Screen - sudo apt-get install screen
- pm2 - sudo npm install -g pm2

5.1.3 Προγραμματισμός

Δημιουργήσαμε καινούργιο φάκελο με όνομα station στα Documents με την εντολή ***mkdir station***.

Στην συνέχεια περιηγηθήκαμε στον φάκελο που δημιουργήσαμε με την εντολή ***cd ~/Documents/station***.

Με την εντολή ***npm init*** δημιουργήσαμε το αρχείο package.json και το Node.js module μας.

Εγκαταστήσαμε τα modules με τις παρακάτω εντολές:

- ***npm install --save @agilatech/bme280***
- ***npm install --save mqtt***
- ***npm install --save serialport***

Περιεχόμενα αρχείου **package.json**

```
{
  "name": "station",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "@agilatech/bme280": "^0.9.0",
    "mqtt": "^2.18.8",
    "serialport": "^7.1.4"
  }
}
```

Παρακάτω δημιουργούμε το αρχείο που θα είναι και το πρόγραμμά μας με την εντολή ***nano index.js***.

Και τέλος γράψαμε τον κώδικα για το πρόγραμμα.

```
const driver = require('@agilatech/bme280');
const hardware = new driver({elevation: 1750, mode: 'normal'});
const mqtt = require('mqtt');
const SerialPort = require('serialport');
const util = require('util');
const exec = util.promisify(require('child_process').exec);
const port = new SerialPort('/dev/serial0', {
  baudRate:115200,
  autoOpen: false
});

const readingsGPS = {fix:false, coordinates:[]};
const readings = {temperature:null,humidity:null,pressure:null};


let command = 'AT+CGNSPWR=1';
let client = null;
let gsm = false;
let attempts = 0;

(async ()=>{
	try{
		await exec('sudo poff fona');
	}catch(e){
		if(e.stdout != '/usr/bin/poff: No pppd is running.  None stopped.\n'){
			console.log(e);
			process.exit(1)
		}
	}

	port.open();

	setInterval(()=>{
		sensorRead();
		if(port.isOpen){
			if(readingsGPS.fix){
				port.close();
			}else{
				console.log('Quering GPS...');
				port.write(`${command}\r`);
			}
		}else{
			if(client === null){
				startClient();
			}else if(client.connected){
				console.log('Sending data!');
				client.publish('station',JSON.stringify({readings:readings,gps:readingsGPS}));
```

```
                }
            }
            console.log('The GPS and sensor readings.');
            console.log(readingsGPS);
            console.log(readings);

    },10000);

})();

function sensorRead(){
    hardware.reset();
  let waiting = setTimeout(function wait() {
    if (hardware.deviceActive()) {
      clearTimeout(waiting);
      for (let i = 0; i < hardware.deviceNumValues(); i++) {
                readings[hardware.nameAtIndex(i)] = hardware.valueAtIndexSync(i);
            }
    }else {
      waiting = setTimeout(wait, 100);
    }
  }, 100);
};

async function startGSM(){
    console.log('Starting GSM');
    try{
        await exec('sudo pon fona');
        gsm = true;
    }catch(e){
        console.log(e);
        process.exit(1);
    }
}

async function startClient(){
    console.log('Creating client...');

    await startGSM();

            if(!gsm){
        return;
    }

    client =
mqtt.connect('wss://station.thesis',{username:'station',password:'3!Tk1kj078a~e0lH',rejectUnautho
ri$

    client.on('connect',()=>{attempts=0;});

    client.on('close',()=>{
```

```
            attempts++;
            console.log('Client closed.');
            if(attempts > 2){
                  console.log('Too many attempts!');
                  process.exit(1);
            }
            client.end(true);
            client = null;
      });

      client.on('error',(e)=>{
            attempts++;
            console.log(e);
            if(attempts > 2){
                  console.log('Too many attempts!');
                  process.exit(1);
            }
            client.end(true);
            client = null;
                  });
}

port.on('close',()=>{
      console.log('The port is now closed.');
      startClient();
});

port.on('data',data=>{
      data = data.toString('UTF-8');
      console.log(data);
      if(command == 'AT+CGNSPWR=1'){
            if(data.match(/OK\r\n$/) !== null){
                  command = 'AT+CGNSINF';
            }else if(data.match(/NO CARRIER/) === null){
                  console.log(data);
                  console.log('The GPS could not be started!');
                  process.exit(1);
            }
      }else if(data.match(/\r\n\+CGNSINF:[^\r\n]+\r\n/)!== null){
            let i = 0, match = {};

            match = /,/.exec(data)

            if(match !== null){
                  data = data.substr(match.index+1,data.length-1);
                  readingsGPS.fix = Boolean(parseInt(data[0]));
            }

            readingsGPS.coordinates = [];

            while((match = /,/.exec(data)) !== null && (i < 3)){
```

```
                data = data.substr(match.index+1,data.length-1);
                if(i > 0){
                    let reading = data.match(/^[0-9.]+/);
                    if(reading !== null){
                        readingsGPS.coordinates.push(parseFloat(reading[0]));
                    }else{
                        readingsGPS.coordinates.push(null);
                    }
                }
                i++;
            }
        }
});

port.on('error',e=>{
    console.log(e);
    process.exit(1);
});
```

5.1.4 Επιπλέον ρυθμίσεις

Για να ξεκινάει το πρόγραμμα με την εκκίνηση του λειτουργικού, χρησιμοποιήσαμε το pm2. Για να ρυθμίσουμε το pm2 πληκτρολογούμε τις παρακάτω εντολές:

pm2 start ~/Documents/station/index.js - Εκκίνηση του προγράμματος

sudo pm2 save - Αποθήκευση των προγραμμάτων που εκκινήθηκαν

sudo pm2 startup systemd - Ρύθμιση ώστε να ξεκινάει με το λειτουργικό

5.2 Εγκατάσταση λογισμικού στον server

5.2.1 Εγκατάσταση λογισμικών

Τα λογισμικά που χρησιμοποιήσαμε εγκαταστάθηκαν μέσω του Terminal με τις παρακάτω εντολές:

- Ενημέρωση λογισμικού - sudo apt-get update
- Αναβάθμιση λογισμικού - sudo apt-get upgrade
- Node.js - sudo apt-get install nodejs
- pm2 - sudo npm install -g pm2
- nginx - sudo apt-get install nginx
- sudo npm install -g @angular/cli

5.2.2 Ρυθμίσεις

Με την βοήθεια του OpenSSL, δημιουργήσαμε SSL Certificate πληκτρολογώντας της παρακάτω εντολές στο Terminal:

- ***openssl req -newkey rsa:2048 -nodes -keyout key.pem -x509 -days 365 -out certificate.pem*** - Δημιουργία του SSL Certificate
- ***openssl x509 -outform der -in certificate.pem -out certificate.crt*** - Μετατροπή του pem σε crt
- ***openssl pkey -in certificate.pem -out certificate.key*** - Μετατροπή του pem σε key

Προσθέτουμε τον παρακάτω κώδικα στο αρχείο ssl.conf. Αυτό το βήμα απαιτείται για να ξέρει ο nginx (http server), που βρίσκεται το site μας.

sudo nano /etc/nginx/conf.d/ssl.conf

```
server {
   listen 443 ssl http2;
   listen [::]:443 ssl;

   ssl_certificate /home/station/station.cert;
   ssl_certificate_key /home/station/station.key;

   server_name station.thesis;
   root /home/station/public;

   location / {
      proxy_pass http://localhost:3042;
      proxy_http_version 1.1;
      proxy_set_header Upgrade $http_upgrade;
      proxy_set_header Connection "Upgrade";
   }

   location ~ \.(txt|png|js|css|ico) {}
}
```

Τέλος κάνουμε επανεκκίνηση του nginx με την εντολή ***sudo service nginx restart***.

5.2.3 Προγραμματισμός

Δημιουργήσαμε καινούργιο φάκελο με όνομα station στο home με την εντολή ***mkdir station***.

Στην συνέχεια περιηγηθήκαμε στον φάκελο που δημιουργήσαμε με την εντολή ***cd /home/station***.

5.2.3.1 Server side

Με την εντολή ***npm init*** δημιουργήσαμε το αρχείο package.json και το Node.js module μας.

Εγκαταστήσαμε τα modules με τις παρακάτω εντολές:

- ***npm install --save express***
- ***npm install --save mosca***
- ***npm install --save mysql2***
- ***npm install --save sequelize***

Περιεχόμενα αρχείου **package.json**

```
{
  "name": "station",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {},
  "author": "",
  "license": "ISC",
  "devDependencies": {},
  "dependencies": {
    "express": "^4.16.4",
    "mosca": "^2.8.3",
    "mysql2": "^1.6.4",
    "sequelize": "^4.42.0"
  }
}
```

Παρακάτω δημιουργούμε το αρχείο για την λειτουργία του server μας με την εντολή ***nano index.js***.

Και τέλος γράψαμε τον κώδικα για το πρόγραμμα.

```
const http = require('http');
const express = require('express');
const mosca = require('mosca');
const Sequelize = require('sequelize');
const path = require('path');

let last = null;

(async ()=>{

        const db = new Sequelize('akcommod_station','akcommod_station','2U3^iWV5Fq3-',{
                host: 'localhost',
                dialect: 'mysql',
                operatorsAliases: false,
                pool:{max: 5,min: 0,acquire: 30000,idle: 10000},
        });
```

```
	await db.authenticate();
	console.log('Database connection established.');
	const Reading = db.import(__dirname + '/models/reading');
	console.log('Readings model imported successfully');

	const app = express();
	const server = http.createServer(app);
	const mqttBroker = new mosca.Server();
	mqttBroker.attachHttpServer(server);

	app.get('/',(req,res)=>{
		res.sendFile(path.join(__dirname + '/public/index.html'));
	});

	app.get('/chart',(req,res)=>{
		res.sendFile(path.join(__dirname + '/public/index.html'));
	});

	app.get('/api/chart', async(req, res)=>{

		let readings = await Reading.findAll({
		  limit: 24,
		  order: [
				['createdAt', 'DESC'],
		  ]
		});

		readings = readings.reverse();
		res.json(readings);
	});

	server.listen(3042);

	mqttBroker.on('ready',()=>{
		mqttBroker.authenticate = (client, username, password, callback)=>{
			console.log(username);
			if(username === 'guest'){
				client.user=username;
				callback(null,true);
			}else if(username === 'station' && password.toString() ===
'3!Tk1kj078a~e0lH'){
		 client.user=username;
				callback(null,true);
			}
		};
		mqttBroker.authorizePublish = (client, topic, payload,
callback)=>callback(null,client.user === 'station');
	});

	mqttBroker.on('clientConnected', function(client) {
		console.log('client connected', client.id);
```

```
        });

        mqttBroker.on('published', function(packet, client) {
                try{
                        const data = JSON.parse(packet.payload.toString('utf8'));
                        if('readings' in data){
                                last = data;
                                const readings = last.readings;
                                Reading.create({
                                        temperature:parseFloat(readings.temperature),
                                        humidity:parseFloat(readings.humidity),
                                        pressure:parseFloat(readings.pressure),
                                        x:parseFloat(data.gps.coordinates[0]),
                                        y:parseFloat(data.gps.coordinates[1])
                                });
                        }
         }catch(e){
         }
        });
})();

function setLast(){
        let html = '';

        for(let k in data){
                html += `<p class="font-weight-bold">${k}</p>`;
                html += '<ul>';
                for(let c in data[k]){
                        html += `<li><span class="font-weight-bold">${c}</span>: ${data[k][c]}</li>`;
                }
                html += '</ul>';
   }
}
```

5.2.3.2 Client Side

Με την εντολή ***ng new front*** δημιουργήσαμε νέο Angular Project. Περιηγούμαστε στον φάκελο front που μόλις δημιουργήθηκε, με την εντολή ***cd front***.

Εγκαταστήσαμε τα modules με τις παρακάτω εντολές:

- ***npm install --save moment***
- ***npm install --save mqtt***
- ***npm install --save bootstrap***
- ***npm install --save chart.js***
- ***npm install --save @types/chart.js***

Στο αρχείο **angular.json** αλλάζουμε το outputPath σε "../public"

Περιεχόμενο αρχείου **package.json**

```
{
  "name": "front",
  "version": "0.0.0",
  "scripts": {
    "ng": "ng",
    "start": "ng serve",
    "build": "ng build",
    "test": "ng test",
    "lint": "ng lint",
    "e2e": "ng e2e"
  },
  "private": true,
  "dependencies": {
    "@angular/animations": "~7.2.0",
    "@angular/common": "~7.2.0",
    "@angular/compiler": "~7.2.0",
    "@angular/core": "~7.2.0",
    "@angular/forms": "~7.2.0",
    "@angular/platform-browser": "~7.2.0",
    "@angular/platform-browser-dynamic": "~7.2.0",
    "@angular/router": "~7.2.0",
    "@types/chart.js": "^2.7.42",
    "@types/leaflet": "^1.4.1",
    "bootstrap": "^4.3.1",
    "chart.js": "^2.7.3",
    "core-js": "^2.5.4",
    "moment": "^2.24.0",
    "mqtt": "^2.18.8",
    "rxjs": "~6.3.3",
    "tslib": "^1.9.0",
    "zone.js": "~0.8.26"
  },
  "devDependencies": {
    "@angular-devkit/build-angular": "~0.13.0",
    "@angular/cli": "~7.3.1",
    "@angular/compiler-cli": "~7.2.0",
    "@angular/language-service": "~7.2.0",
    "@types/node": "~8.9.4",
    "@types/jasmine": "~2.8.8",
    "@types/jasminewd2": "~2.0.3",
    "codelyzer": "~4.5.0",
    "jasmine-core": "~2.99.1",
    "jasmine-spec-reporter": "~4.2.1",
    "karma": "~3.1.1",
    "karma-chrome-launcher": "~2.2.0",
    "karma-coverage-istanbul-reporter": "~2.0.1",
    "karma-jasmine": "~1.1.2",
    "karma-jasmine-html-reporter": "^0.2.2",
    "protractor": "~5.4.0",
    "ts-node": "~7.0.0",
```

```
    "tslint": "~5.11.0",
    "typescript": "~3.2.2"
  }
}
```

Περιεχόμενο αρχείου **src/style.scss**

```
@import '../node_modules/bootstrap/scss/bootstrap';

// Typography
$font-family-sans-serif: "Nunito", sans-serif;
$font-size-base: 0.9rem;
$line-height-base: 1.6;

// Colors
$whie: rgb(255,255,255);
$blue: #3490dc;
$indigo: #6574cd;
$purple: #9561e2;
$pink: #f66D9b;
$red: #e3342f;
$orange: #f6993f;
$yellow: #ffed4a;
$green: #38c172;
$teal: #4dc0b5;
$cyan: #6cb2eb;
$theme-dark-blue: #008bd5 ;
$theme-light-blue: #385a8a;
$other-blue: #0066cc;
// Body
$body-bg: $whie;

//Font Awesome
$fa-font-path: "../fonts" !default;

.navbar-dark .navbar-nav .nav-link {
    color: $white;
  }

  .navbar-laravel {
    background-color: $theme-dark-blue;
    box-shadow: 0 2px 4px rgba(0, 0, 0, 0.04);
  }

  .card-header{
    background-color: $theme-dark-blue;
    color: $white;
  }

  .table{
```

```
  td{
    text-align: center;
  }
  th{
    text-align: center;
  }
}

.navbar-nav{
  .nav-item{
    .active{
      border-bottom: solid 3px;
    }
  }
}


.navbar-dark:hover .navbar-nav:hover .nav-link:hover{
  border-bottom: solid 3px  rgba(255,255,255,.5);
}

.page-item.active .page-link{
  background-color: $theme-dark-blue;
}

.navbar-brand{
  img{
    background-color: $white;
  }
}

#chart{
  height: 50% !important;
  width:  50% !important;;
  cursor:pointer;
  margin-left: auto;
  margin-right: auto;
  margin-top: 2.5%;
  margin-bottom: 2.5%;
}
```

Περιεχόμενο αρχείου **src/app/app.module.ts**

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { NavigationComponent } from './components/navigation/navigation.component';
import { IndexComponent } from './components/index/index.component';
import { ChartComponent } from './components/chart/chart.component';
import { HttpClientModule }    from '@angular/common/http';
```

```
@NgModule({
  declarations: [
    AppComponent,
    NavigationComponent,
    IndexComponent,
    ChartComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    HttpClientModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Περιεχόμενο αρχείου **src/app/app-routing.module.ts**

```
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';
import { IndexComponent } from './components/index/index.component';
import { ChartComponent } from './components/chart/chart.component';

const routes: Routes = [
  { path: '', component: IndexComponent },
  { path: 'chart', component: ChartComponent}
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

Περιεχόμενο αρχείου **src/app/app.component.ts**

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.scss']
})
export class AppComponent {
}
```

Περιεχόμενο αρχείου **src/app/app.component.html**

```
<app-navigation></app-navigation>
<main class="py-4">
   <router-outlet></router-outlet>
</main>
```

Περιεχόμενο αρχείου **src/app/components/nagivation/navigation.component.ts**

```
import { Component, OnInit } from '@angular/core';

@Component({
 selector: 'app-navigation',
 templateUrl: './navigation.component.html',
 styleUrls: ['./navigation.component.scss']
})
export class NavigationComponent implements OnInit {

 constructor() { }

 ngOnInit() {
 }

}
```

Περιεχόμενο αρχείου **src/app/components/nagivation/navigation.component.html**

```
<nav class="navbar navbar-expand-sm navbar-dark navbar-laravel">
 <a class="navbar-brand" routerLink="/">
    Weather Station
 </a>
 <button class="navbar-toggler" type="button" data-toggle="collapse" data-
target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-
expanded="false">
    <span class="navbar-toggler-icon"></span>
 </button>
 <div class="collapse navbar-collapse" id="navbarSupportedContent">
    <ul class="navbar-nav mr-auto">
       <li class="nav-item">
          <a routerLink="/"  class="nav-link" routerLinkActive="active"
[routerLinkActiveOptions]="{exact:
            true}">Home</a>
       </li>
       <li class="nav-item">
          <a routerLink="/chart"  class="nav-link" routerLinkActive="active">Chart</a>
       </li>
    </ul>
    <ul class="navbar-nav ml-auto">
    </ul>
 </div>
</nav>
```

Περιεχόμενο αρχείου **src/app/components/index/index.component.ts**

```
import { Component, OnInit, ViewChild, ElementRef } from '@angular/core';
import { MqttService } from 'src/app/services/mqtt.service';
import { map } from 'leaflet';

@Component({
 selector: 'app-index',
 templateUrl: './index.component.html',
 styleUrls: ['./index.component.scss']
})
export class IndexComponent implements OnInit {

 readings:any = {};
 gps:any = {};
 objectKeys = Object.keys;
 map: string = '#';

 constructor(private mqttService:MqttService) { }

 ngOnInit() {
  this.mqttService.getData().subscribe(data=>{
   data = JSON.parse(data);
   this.gps = data['gps'];
   this.readings = data['readings'];
   this.map=`https://maps.google.com/?q=${this.gps.coordinates[0]},${this.gps.coordinates[1]}`;
  });
 }

}
```

Περιεχόμενο αρχείου **src/app/components/index/index.component.html**

```
<div class="container-fluid">
  <div class="row justify-content-center">
    <div class="col-md-8">
      <div class="card">
       <div class="card-header text-center"><h3>Readings</h3></div>
       <div class="card-body">
          <h3 class="text-center" *ngIf="!objectKeys(readings).length">Loading...</h3>
          <table class="table table-stripped">
            <thead>
              <tr>
                <th *ngFor="let k of objectKeys(readings)">{{k}}</th>
              </tr>
            </thead>
            <tbody>
              <tr>
                <td *ngFor="let k of objectKeys(readings)">
                  {{readings[k]}}
```

```
                        <span class="font-weight-bold">{{k === 'temperature' ? '&#8451;': k ===
'humidity' ? '%' : 'mbr'}}</span>
                      </td>
                    </tr>
                  </tbody>
                </table>
              </div>
            </div>
            <div class="card">
              <div class="card-header text-center"><h3>GPS</h3></div>
              <div class="card-body">
                <h3 class="text-center" *ngIf="!objectKeys(gps).length">Loading...</h3>

                <table class="table table-stripped">
                  <thead>
                    <tr>
                      <th *ngFor="let k of objectKeys(gps)">{{k}}</th>
                    </tr>
                  </thead>
                  <tbody>
                    <tr>
                      <td *ngFor="let k of objectKeys(gps)">
                        <a *ngIf="k === 'coordinates'" [href]="map" target="_blank">{{gps[k]}}</a>
                        <span *ngIf="k !== 'coordinates'">{{gps[k]}}</span>
                      </td>
                    </tr>
                  </tbody>
                </table>
              </div>
            </div>
          </div>
        </div>
      </div>
```

Περιεχόμενο αρχείου **src/app/components/chart/chart.component.rs**

```
import { Component, OnInit, ViewChild, ElementRef } from '@angular/core';
import { Reading } from 'src/app/models/reading';
import { ReadingsService } from 'src/app/services/readings.service';
import { Chart, ChartConfiguration } from 'chart.js';
import * as moment from 'moment';
import {MqttService} from "../../services/mqtt.service";


@Component({
  selector: 'app-chart',
  templateUrl: './chart.component.html',
  styleUrls: ['./chart.component.scss']
})
export class ChartComponent implements OnInit {
```

```
readings:Reading[];
private chart:Chart;
private chartColors = {
  red: 'rgb(255, 99, 132)',
  orange: 'rgb(255, 159, 64)',
  yellow: 'rgb(255, 205, 86)',
  green: 'rgb(75, 192, 192)',
  blue: 'rgb(54, 162, 235)',
  purple: 'rgb(153, 102, 255)',
  grey: 'rgb(201, 203, 207)'
};
private config:ChartConfiguration = {
  type: 'line',
  data: {
    labels: [],
    datasets: [
      {
        label: 'Temperature(in Celsius)',
        fill: false,
        backgroundColor: this.chartColors.blue,
        borderColor: this.chartColors.blue,
        data: [],
      },
      {
        label: 'Humidity(in percentage)',
        fill: false,
        backgroundColor: this.chartColors.orange,
        borderColor: this.chartColors.orange,
        data: [],
      },
      {
        label: 'Atmospheric pressure(in mbr)',
        fill: false,
        backgroundColor: this.chartColors.purple,
        borderColor: this.chartColors.purple,
        data: [],
      }
    ]
  },
  options: {
    responsive: true,
    title: {
      display: true,
      text: ''
    },
    tooltips: {
      mode: 'index',
      intersect: false,
    },
    hover: {
      mode: 'nearest',
```

```
      intersect: true
    },
    scales: {
      xAxes: [{
        display: true,
        scaleLabel: {
          display: true,
          labelString: 'Hours'
        }
      }],
      yAxes: [{
        display: true,
        scaleLabel: {
          display: true,
          labelString: 'Measurements'
        }
      }]
    }
  }
};

@ViewChild('canvas') canvas:ElementRef;

constructor(private readingService:ReadingsService, private mqttService:MqttService) { }

ngOnInit() {
  this.chart = new Chart(this.canvas.nativeElement,this.config);
  this.readingService.getReadings().subscribe(data=>{
    this.readings=data;
    this.updateChart();
  });

  this.mqttService.getData().subscribe(data=>{
    /*(data = JSON.parse(data);
    this.gps = data['gps'];
    this.readings = data['readings'];
    this.map=`https://maps.google.com/?q=${this.gps.coordinates[0]},${this.gps.coordinates[1]}`;*/
    if(this.chart.data.datasets[0].data.length){
        data = JSON.parse(data);
        this.readings.shift();
        this.readings.push(<Reading>{
          temperature:data['readings'].temperature,
          humidity:data['readings'].humidity,
          pressure:data['readings'].pressure,
          x:data['gps'].coordinates[0],
          y:data['gps'].coordinates[1],
          createdAt:moment().toString()
        });
        this.updateChart();
    }
  });
```

```
  }

  private updateChart(){
    const temperatures = [];
    const humidities = [];
    const pressure = [];
    const labels = [];
    for(let i = 0; i < this.readings.length; i++){
      temperatures.push(this.readings[i].temperature);
      humidities.push(this.readings[i].humidity);
      pressure.push(this.readings[i].pressure);

      labels.push(`${ moment(this.readings[i].createdAt).format('DD/MM/YY HH:mm')} -
x:${this.readings[i].x.toFixed(2)}, y:${this.readings[i].y.toFixed(2)}`);
    }
    this.chart.data.datasets[0].data = temperatures;
    this.chart.data.datasets[1].data = humidities;
    this.chart.data.datasets[2].data = pressure;
    this.chart.data.labels = labels;

    this.chart.update();
  }

}
```

Περιεχόμενο αρχείου **src/app/components/chart/chart.component.html**

```
<div class="container-fluid">
    <div class="row justify-content-center">
        <div class="col-md-8">
            <div class="card">
                <div class="card-header text-center"><h3>Last 24 Readings</h3></div>
                <div class="card-body" id="data">
                    <canvas  #canvas></canvas>
                </div>
            </div>
        </div>
    </div>
</div>
```

Περιεχόμενο αρχείου **src/app/models/reading.ts**

```
export class Reading {
    id:number;
    pressure:number;
    humidity:number;
    temperature:number;
    x:number;
    y:number;
```

```
  createdAt:string;
}
```

Περιεχόμενο αρχείου **src/app/services/mqtt.service.ts**

```
import { Injectable } from '@angular/core';
import { Observable } from 'rxjs';
import { connect, MqttClient } from 'mqtt';
import {environment} from "../../environments/environment";


@Injectable({
 providedIn: 'root'
})
export class MqttService {

 private observable:Observable<string>;
 private client:MqttClient;

 constructor() {
  const client = connect(environment.mqtt,{username:'guest'});

  client.on('connect',()=>{
   console.log('Client connected!');
   client.subscribe('station');
  });

  client.on('error',(e)=>{
   console.log(e);
  });

  this.observable = new Observable((observer) => {
   client.on('message',(topic,payload)=>{
    const decoder = new TextDecoder('utf-8');
    observer.next(decoder.decode(payload));
   });
  });

  this.client = client;
 }

 getData():Observable<string>{
  return this.observable;
 }
}
```

Περιεχόμενο αρχείου **src/app/services/readings.service.ts**

```
import { Injectable } from '@angular/core';
import { HttpClient, HttpHeaders } from '@angular/common/http';
import { Observable } from 'rxjs';
```

```
import { Reading } from '../models/reading';
import {environment} from "../../environments/environment";


@Injectable({
  providedIn: 'root'
})
export class ReadingsService {

  constructor(private http: HttpClient) { }

  getReadings(): Observable<Reading[]> {
    const httpOptions = {
      headers: new HttpHeaders({ 'Content-Type': 'application/json' ,'Access-Control-Allow-Origin':'*'})
    };
    return this.http.get<Reading[]>(environment.api,httpOptions);
  }
}
```

Με την εντολή ***ng build --aot=true --buildOptimizer=true --optimization=true --prod=true*** μεταγλωττίσαμε το project.

Κεφάλαιο 6ο: Tests και επίλυση τεχνικών προβλημάτων

6.1 Software Tests

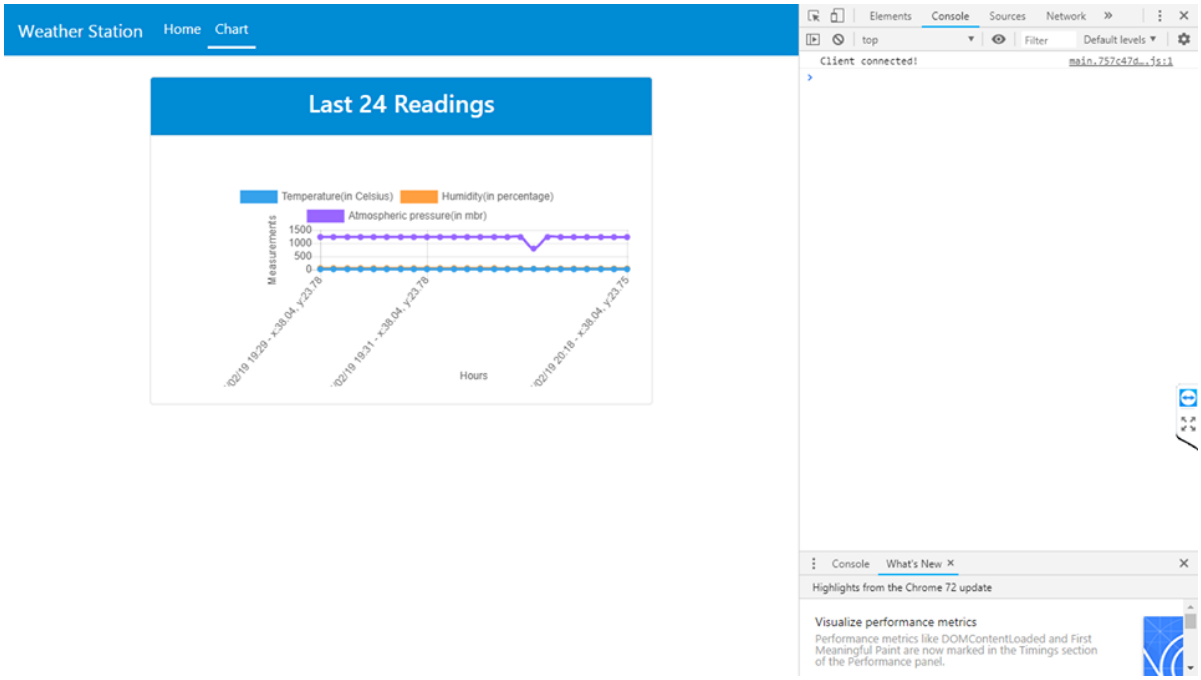
Με την βοήθεια του παρακάτω κώδικα εξομοιώσαμε τη σύνδεση του μετεωρολογικού σταθμού με τον MQTT Broker για να δοκιμάσουμε αν το ο σταθμός μπορεί να επικοινωνήσει με τον server.

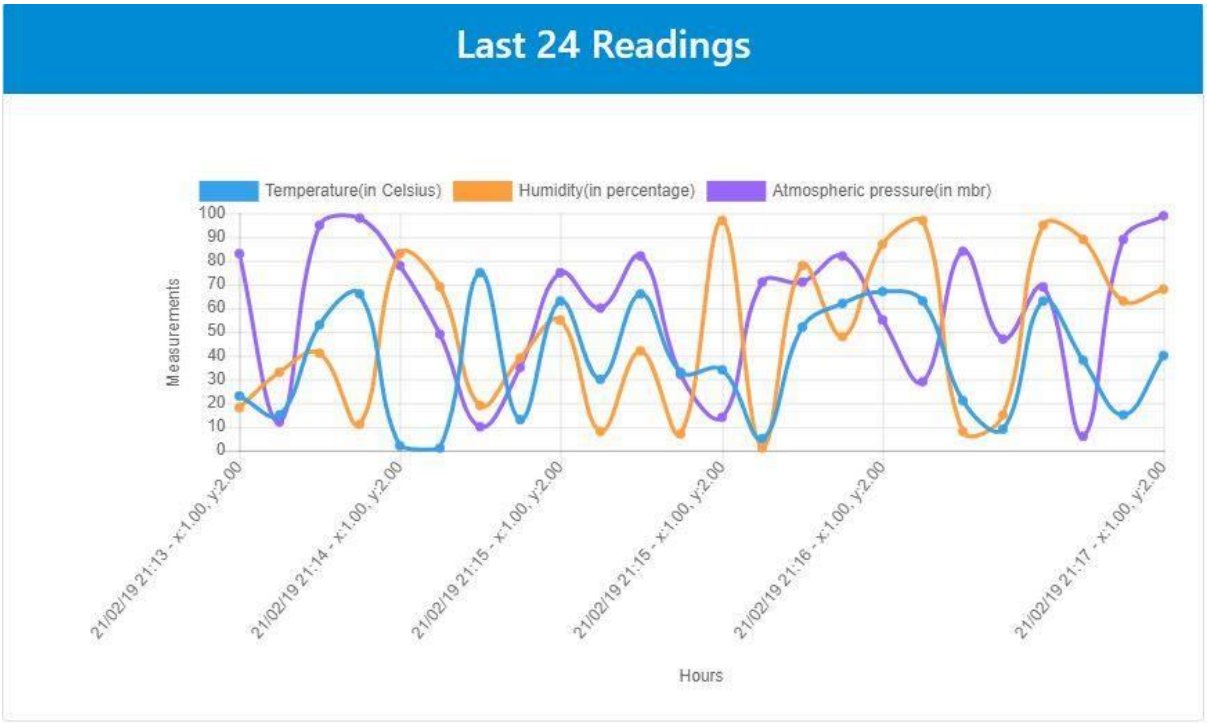
```
const mqtt = require('mqtt')
const client  =
mqtt.connect('wss://station.thesis',{username:'station',password:'3!Tk1kj078a~e0lH',rejectUnautho
rized:false});

client.on('connect', ()=>{
  setInterval(()=>{
    client.publish('station',JSON.stringify({
       readings:{
          temperature:Math.floor(Math.random() * 100) + 1,humidity:Math.floor(Math.random() *
100) + 1,pressure:Math.floor(Math.random() * 100) + 1
       },
       gps:{
          fix:true,
          coordinates:[
             1,2
          ]
```

```
        }
    }));
  },10000);
})

client.on('error',(e)=>{
    console.log(e);
});
```

Ο παραπάνω κώδικας παράγει τυχαίες τιμές.

Περιηγηθήκαμε στο https://station.thesis για να δοκιμάσουμε αν ο http server είναι προσβάσιμος και λειτουργεί σύμφωνα με τις προκαθορισμένες προδιαγραφές. Στην συνέχεια ανοίξαμε την κονσόλα του Chrome για να ελέγξουμε για τυχών λάθη και να επιβεβαιώσουμε ότι ο MQTT Client συνδέεται με το server.

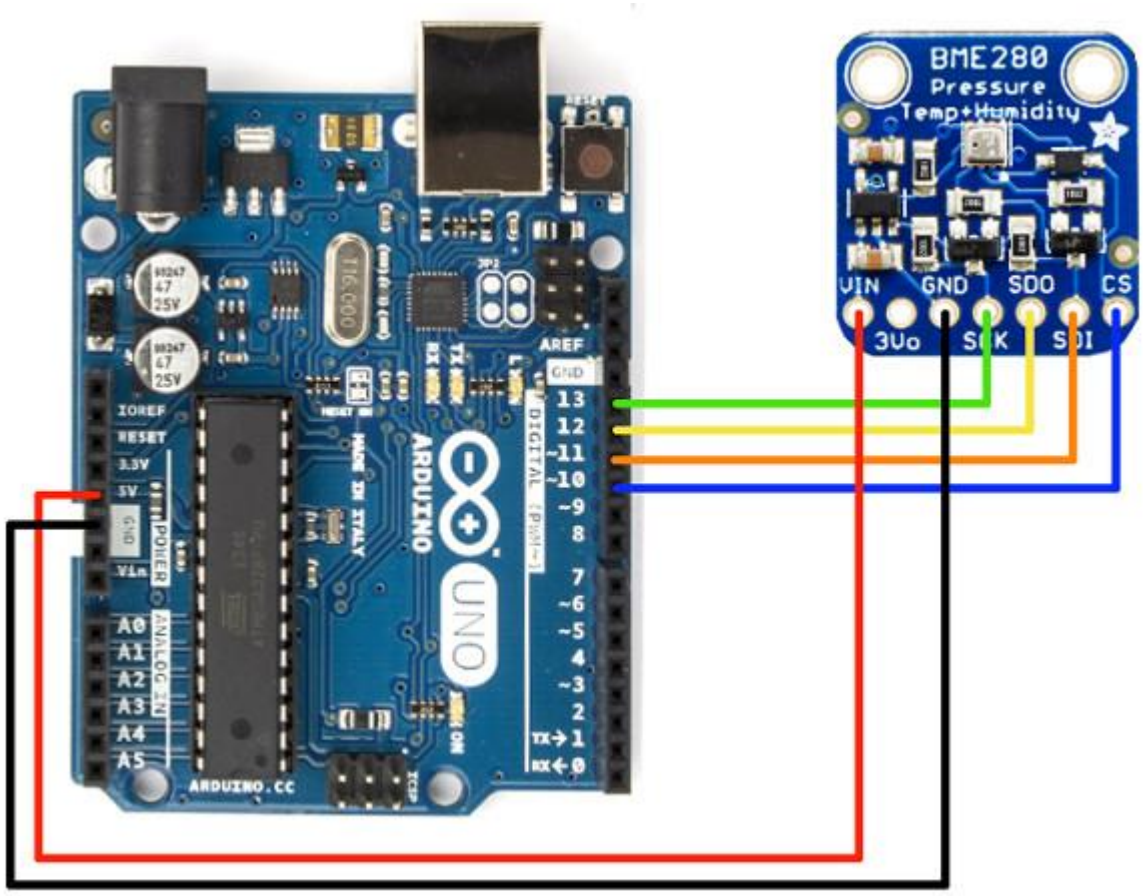



Last 24 Readings
Temperature(in Celsius)
Humidity(in percentage)
Atmospheric pressure(in mbr)
Measurements
100
90
80
70
60
50
40
30
20
10
0
21/02/19 21:13 - x:1.00, y:2.00
21/02/19 21:14 - x:1.00, y:2.00
21/02/19 21:15 - x:1.00, y:2.00
21/02/19 21:15 - x:1.00, y:2.00
21/02/19 21:16 - x:1.00, y:2.00
21/02/19 21:17 - x:1.00, y:2.00
Hours

6.2 Hardware Tests

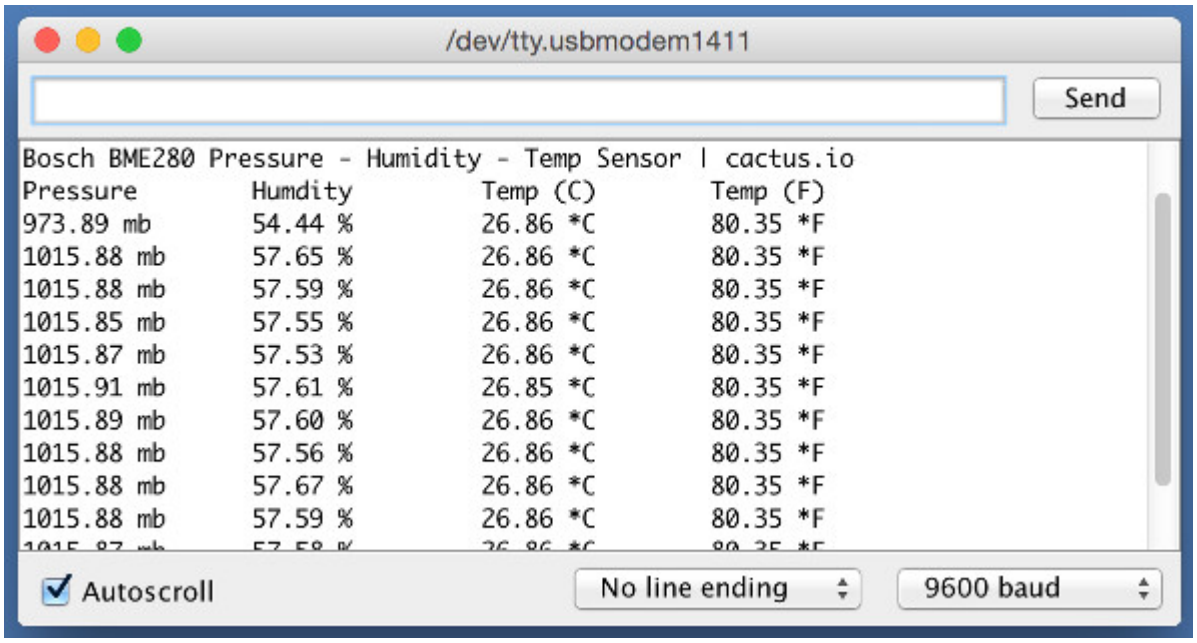
6.2.1 BME280

Για να τεστάρουμε τον BME280 αν δουλεύει όπως πρέπει, τον βάλαμε σε ένα Arduino, που είναι γρηγορότερο σαν διαδικασία, καθώς είναι controller και όχι υπολογιστής. Η συνδεσμολογία και ο κώδικας φαίνονται παρακάτω. Το συνδέσαμε με την κατάλληλη συνδεσμολογία για επικοινωνία μέσω SPI interface, το οποίο θα χρησιμοποιούσαμε και αργότερα στο Raspberry.



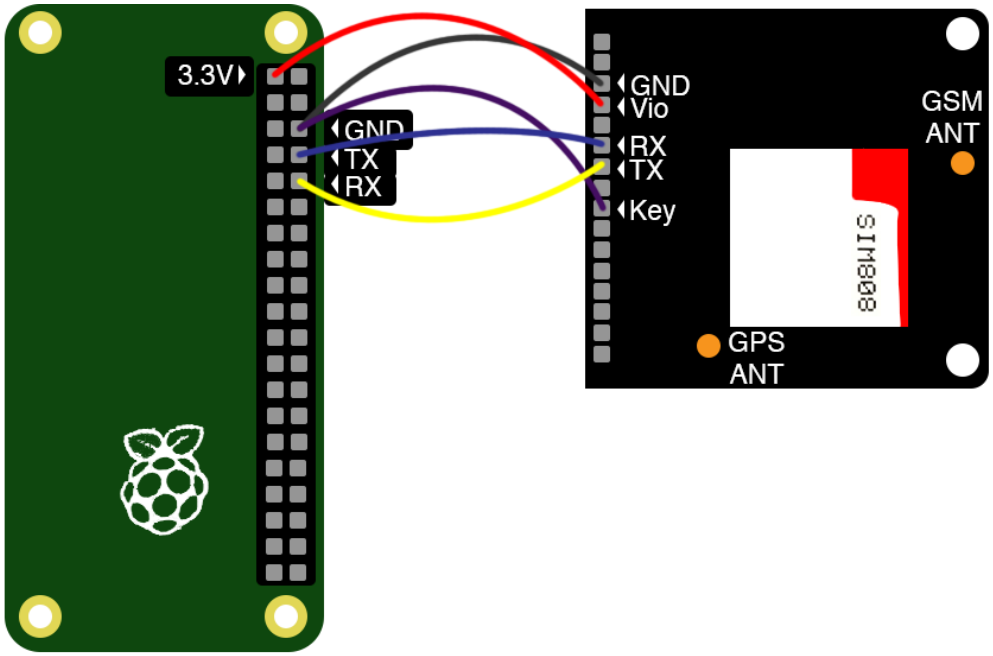

```
#include <SPI.h>
#include "cactus_io_BME280_SPI.h"

#define BME_SCK 13;// Serial Clock
#define BME_MISO 12;// Serial Data Out
#define BME_MOSI 11;// Serial Data In
#define BME_CS 10;// Chip Select

// Create BME280 object
// BME280_SPI bme(BME_CS); // Using Hardware SPI
BME280_SPI bme(BME_CS,BME_MOSI,BME_MISO,BME_SCK); // Using Software SPI

void setup() {

Serial.begin(9600);
Serial.println("Bosch BME280 Pressure - Humidity - Temp Sensor | cactus.io");

if (!bme.begin()) {
Serial.println("Could not find a valid BME280 sensor, check wiring!");
while (1);
}

bme.setTempCal(-1);// Sensor was reading high so offset by 1 degree C
```

```
Serial.println("Pressure\tHumdity\t\tTemp\ttTemp");
}

void loop() {

bme.readSensor();

Serial.print(bme.getPressure_MB()); Serial.print(" mb\t"); // Pressure in
millibars
Serial.print(bme.getHumidity()); Serial.print(" %\t\t");
Serial.print(bme.getTemperature_C()); Serial.print(" *C\t");
Serial.print(bme.getTemperature_F()); Serial.println(" *F");

// Add a 2 second delay.
delay(2000); //just here to slow down the output.
}
```



6.2.2 Adafruit Fona 808 Breakout

Για να τεστάρουμε το Fona 808 το συνδέσαμε με τον τρόπο που φαίνεται παρακάτω στο Raspberry. Ενεργοποιήσαμε στο κινητό την λειτουργία hotspot και συνδέσαμε το Raspberry σε αυτό, ώστε να μπορούμε να συνδεθούμε με ssh. Έπειτα βγήκαμε σε εξωτερικό χώρο για να είμαστε σίγουροι ότι θα πιάσει GPS. Στην συγκεκριμένη περίπτωση για τροφοδοσία χρησιμοποιήσαμε ένα powerbank με 2 εξόδους USB, μία για το Raspberry και μία για το Fona. Συνδεθήκαμε με ssh στο Raspberry και από εκεί με εντολές ελέγξαμε αν δουλεύει σωστά το Fona. Το Fona έπιασε GPS με λίγη δυσκολία (θα αναλυθεί στα προβλήματα περαιτέρω), ωστόσο, μπορούσαμε να κάνουμε τηλεφωνική κλήση και να συνδεθούμε στο διαδίκτυο κανονικά. Στο δεύτερο στιγμιότυπο απ' την οθόνη του κινητού φαίνεται απ' τα μηδενικά που δείχνει σε προηγούμενο χρόνο ότι το GPS δεν έπιανε άμεσα. Πήρε περίπου 10 λεπτά για να έχει fixed position, ενώ οι προδιαγραφές έλεγαν μέσος χρόνος από cold start-up ότι θα ήταν περίπου 32 δευτερόλεπτα.




92% 15:24
OK
at+cgnsinf
+CGNSINF: 1,0,20181208131946.000,,,,0.00,0.0,0,,,,,2,0,,,31,,
OK
at+cgnsinf
+CGNSINF: 1,0,20181208132010.000,,,,0.00,0.0,0,,,,,2,0,,,30,,
OK
at+cgnsinf
+CGNSINF: 1,0,20181208132101.000,,,,0.00,0.0,0,,,,,2,0,,,26,,
OK
at+ch
ERROR
at+cgnsinf
+CGNSINF: 1,0,20181208132148.000,,,,0.00,0.0,0,,,,,2,0,,,27,,
OK
at+cgns inf
+CGNSINF: 1,0,20181208132229.000,,,,0.00,0.0,0,,,,,3,0,,,25,,
OK
at+cgnsinf
+CGNSINF: 1,1,20181208132348.000,38.039702,23.779532,114.500,0.52
,57.4,1,,4.1,4.2,1.0,,6,3,,,24,,
OK
ESC / | - HOME ↑ END PGUP FN
TAB CTRL ALT ← ↓ → PGDN
1 2 3 4 5 6 7 8 9 0
q w e r t y u i o p
a s d f g h j k l
z x c v b n m
☺1# EN

6.3 Προβλήματα που αντιμετωπίσαμε

6.3.1 GPS Antenna

Η κεραία που πήραμε για το GPS ήταν active. Συνεπώς απαιτούσε μια μικρή ενίσχυση για να πιάνει καλά. Το καλώδιό της ήταν 3m με αποτέλεσμα να παρουσιάζει αρκετά μεγάλη (για το Fona) ωμική αντίσταση. Θεωρήσαμε ότι εφόσον οι προδιάγραφες δεν τονίζουν κάτι για την κεραία του GPS, και επειδή το κύκλωμα (Fona) απαιτεί εξωτερική τροφοδοσία, ότι θα είναι ικανό να την τροφοδοτήσει. Εξάλλου δεν είχε ιδιαίτερες απαιτήσεις σε ισχύ. Σύμφωνα με τις προδιαγραφές, ήθελε 3-5V, 0.1A δηλαδή 0,3W. Επειδή όμως ο χρόνος για fixed position ήταν αρκετά μεγαλύτερος απ' αυτόν που αναγραφόταν στις προδιαγραφές, μετά από αρκετό ψάξιμο αποφασίσαμε να πάρουμε μια passive κεραία με μήκος καλωδίου 20cm. Ωστόσο, αξίζει να σημειωθεί ότι στο GitHub χρησιμοποιούσαν active κεραία, και παρά τις προδιάγραφες που έλεγαν ότι πιάνει σε 32 sec fixed position, τους φαινόταν «λογική» η δεκάλεπτη αναμονή για GPS. Είμαστε ευχαριστημένοι που τελικά το κύκλωμα δουλεύει όπως αναγράφεται στις προδιαγραφές με την νέα κεραία. Φάγαμε αρκετό χρόνο περιμένοντας σε διάφορα tests να πιάσει σήμα GPS, αφού χαρακτηριστικά είχαμε διαβάσει στο GitHub το εξής: GPS takes forever to find satellites (and actually forever if you're indoors), so we probably don't have a fix yet, but we can still see some info.

6.3.2 Adafruit Fona 808 Battery

Το Fona 808 είναι στην ουσία μια μητρική ενός κινητού. Στα χαρακτηριστικά έλεγε ότι έχει δυνατότητα να φορτίζει και να εκφορτίζει κανονικά όπως και ένα κινητό, την δική του μπαταρία λιθίου. Δεν έλεγε πουθενά, ότι είναι απαραίτητη για την λειτουργία του. Θεωρήσαμε ότι εφόσον θέλει 3.3V για να λειτουργεί και 5V σε ξεχωριστό pin για να φορτίζει μπαταρία (εφόσον υπάρχει), ότι μπορεί να λειτουργήσει και χωρίς αυτή, ειδικά αν δώσουμε και τις 2 τάσεις. Παρόλα αυτά, τα 3.3V ήταν για να λειτουργεί το chipset του, ενώ τα 5V δεν ήταν μόνο για κύκλωμα φόρτισης. Όμως αποδείχτηκε ότι συνδεσμολογικά, απαιτεί τα 5V σε κάποια τμήματα του κυκλώματος εκτός του chipset. Μετά από αρκετό ψάξιμο στο FAQ της εταιρείας, διαβάσαμε ότι το κύκλωμα δεν κλείνει χωρίς της τοποθέτηση μπαταρίας στα αντίστοιχα pins. Αναγκαστικά, αγοράσαμε άλλη μία 18650 μπαταρία λιθίου.

6.3.3 Raspberry Pi Zero W SSH via Serial

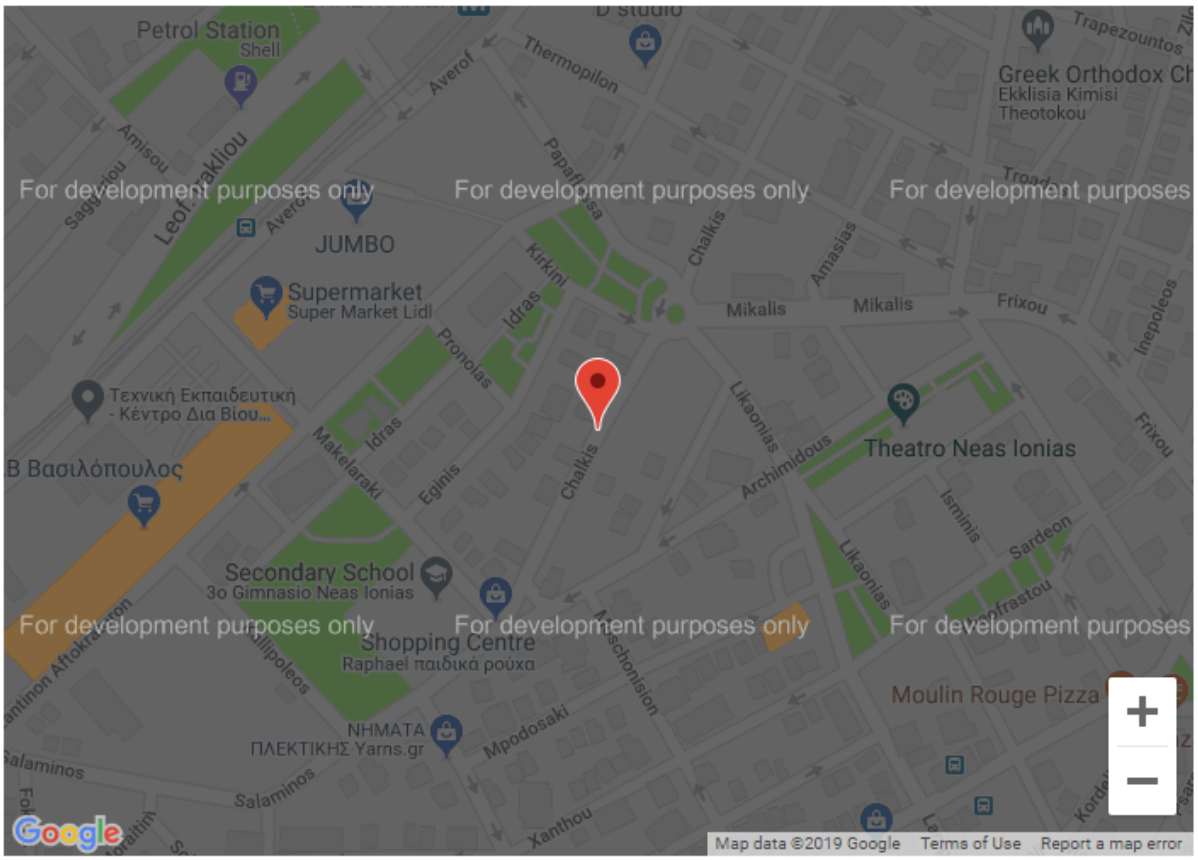
Το Raspbian εκ κατασκευής προσπαθεί να δώσει όσους περισσότερους τρόπους γίνεται για να επικοινωνήσουμε μέσω SSH. Εμείς χρειαστήκαμε μόνο το SSH μέσω WiFi. Η δυνατότητα SSH μέσω Serial έπρεπε να απενεργοποιηθεί διότι απέτρεπε την σειριακή επικοινωνία με το Adafruit Fona 808.

6.3.4 Adafruit Fona 808 GSM/GPS

Το Fona 808 έχει μόνο έναν δίαυλο επικοινωνίας τον οποίο μπορεί είτε να τον χρησιμοποιήσει για να παίρνει position data απ' το GPS, είτε για να στέλνει data μέσω GSM. Ευτυχώς, η αλλαγή από GPS σε GSM γίνεται σε περίπου 1 δευτερόλεπτο και ταυτόχρονα το GPS δεν αποσυνδέεται. Συνεχίζει να «δίνει» συντεταγμένες, απλά εφόσον εμείς έχουμε γυρίσει στο GSM, δεν τις διαβάζουμε. Συνεπώς στο πρόγραμμα που βρίσκεται στον μετεωρολογικό σταθμό, θεωρώντας ότι η συσκευή προορίζεται για σταθερή εγκατάσταση, περιμένουμε πρώτα να πιάσει GPS, και αφού πιάσει στέλνουμε μόνιμα πληροφορίες (υγρασία, θερμοκρασία, βαρομετρική πίεση, συντεταγμένες), μέσω GSM. Όλες οι πληροφορίες ανανεώνονται κάθε 10 δευτερόλεπτα, εκτός απ' τις συντεταγμένες που τις διαβάζουμε μια φορά όταν πιάσει στην αρχή.

6.3.5 Google Maps or alternative maps, developer purposes only error

Είχαμε σκοπό οι συντεταγμένες που παίρνουμε απ' το GPS, να τοποθετούν ένα σημάδι πάνω στον χάρτη (Google Maps, Bing Maps, ή άλλο). Δυστυχώς, αποδείχθηκε ότι τώρα πια αυτό κοστίζει. Σε προηγούμενο χρόνο γνωρίζαμε ότι κοστίζει ένας λογαριασμός Gmail να έχει μόνιμα πολλά στίγματα στον χάρτη, ή ένα στίγμα που κινείται. Η κοστολόγηση γινόταν με το πόσα στίγματα ανά μήνα στέλναμε. Συνεπώς συσχετιζόταν άμεσα με το ρυθμό ανανέωσης της πληροφορίας. Πρόσφατα όμως αυτό άλλαξε και ακόμα και το «δωρεάν» πακέτο απαιτεί στους περισσότερους γνωστούς χάρτες, να δώσει κάποιος τα στοιχεία μιας τραπεζικής κάρτας. Πράγμα με το οποίο δεν συμφωνούσαμε γιατί ακόμα και να μην ξεπερνούσαμε το όριο του δωρεάν πακέτου, θα είχαν τα στοιχεία της κάρτας, που μπορεί να κλαπούν σε μια διαδικτυακή επίθεση. Για το λόγο αυτό, αποφασίσαμε να τοποθετούμε μόνο της συντεταγμένες ως υπερσύνδεσμο στην σελίδα μας, και από εκεί αν κάνουμε κλικ να μας πηγαίνει στον Google Maps και να βάζει εκεί το point, σαν να κάναμε κλικ στον χάρτη.



Maps

APIs in Maps include: Maps SDK for Android; Maps SDK for iOS; Maps JavaScript API; Maps Static API; Street View API; Maps Embed API; and Maps URLs.

SKU	$200 MONTHLY CREDIT EQUIVALENT FREE USAGE	MONTHLY VOLUME RANGE (PRICE PER THOUSAND)		
		0—100,000	100,001-500,000	500,001+
Mobile Native Static Maps	Unlimited loads	$0.00	$0.00	
Mobile Native Dynamic Maps	Unlimited loads	$0.00	$0.00	
Embed	Unlimited loads	$0.00	$0.00	
Embed Advanced	Up to 14,000 loads	$14.00	$11.20	CONTACT SALES for volume discounts.
Static Maps	Up to 100,000 loads	$2.00	$1.60	
Dynamic Maps	Up to 28,000 loads	$7.00	$5.60	
Static Street View	Up to 28,000 panos	$7.00	$5.60	
Dynamic Street View	Up to 14,000 panos	$14.00	$11.20	

Κεφάλαιο 7ο: Μελλοντικές επεκτάσεις και βελτιώσεις

7.1 Επεκτάσεις

Οι παρακάτω αισθητήες θα μπορούσαν να προστεθούν στην παρούσα πτυχιακή. Δυστυχώς, αυτό δεν μπορούσε να γίνει κυρίως για οικονομικούς λόγους, αλλά και για την ιδιαιτερότητα των συγκεκριμένων αισθητήρων στην τοποθέτησή τους.

- Αισθητήρας ανέμου
- Αισθητήρας βροχής

7.2 Βελτιώσεις Hardware

7.2.1 Φωτοβολταϊκό

Το φωτοβολταϊκό που χρησιμοποιούμε, στην καλύτερη περίπτωση, φορτίζει πολύ αργά τις μπαταρίες. Ιδανικό θα ήταν να χρησιμοποιήσουμε ένα επόνυμο φωτοβολταϊκό, το οποίο θα μπορούσε να τις φορτίζει με ελάχιστο φως, σε λιγότερο από 2 ώρες. Αυτό θα βοηθούσε στο να τοποθετηθούν σταθμοί σαν τους δικούς μας, σε χώρες όπως Σουηδία, Νορβηγία, Φινλανδία. Ως παράδειγμα θα μπορούσαμε να χρησιμοποιήσουμε το LG Mono X Plus 300W.

7.2.2 All-in-one circuit

Αυτήν την στιγμή χρησιμοποιούμε πολλά έτοιμα κυκλώματα από διάφορες γνωστές εταιρίες. Τα ηλεκτρονικά τους μέρη μπορούν να βρεθούν εύκολα στο εμπόριο. Συνεπώς, θα μπορούσε να σχεδιαστεί ένα PCB, που να περιέχει όλα αυτά τα συστήματα σε μια πλακέτα. Αυτό θα απαιτούσε αρκετές ώρες δουλειάς και prototyping, μέχρι να φτάσουμε στο ιδανικό αποτέλεσμα.

7.2.3 IP68 και MIL-STD-810

Εφόσον η κατασκευή μας είναι φιταγμένη για να βρίσκεται σε εξωτερικό χώρο, θεωρούμε απαραίτητο τον σχεδιασμό και υλοποίηση ενός στεγανοποιημένου σασί, το οποίο θα μπορεί να λειτουργεί σε αντίξοες καιρικές συνθήκες. Επιπλέον, η κατασκευή αυτή θα μπορούσε να πληρεί τις

προϋποθέσεις του MIL-STD-810 (US Military standard). Έτσι θα διασφαλίζαμε την ασφάλεια της συσκευής από τυχών «βανδαλισμούς», από τα ζώα της περιοχής, αλλά και χτυπήματα, σεισμούς κτλ.

7.3 Βελτιώσεις Software

7.3.1 Registered Companies/Users and Devices

Κάθε πιθανός αγοραστής ή χρήστης της συσκευής, θα πρέπει να έχει την δυνατότητα, να βλέπει τους δικές του, μόνο, μετεωρολογικούς σταθμούς πάνω στον χάρτη. Συνεπώς, θα μπορούσαμε να υλοποιήσουμε ένα σύστημα, όπου κάθε χρήστης που έχει μια τουλάχιστον έναν μετεωρολογικό σταθμό, να έχει προσωπικό λογαριασμό για αυτόν/ους. Ο κάθε μετεωρολογικός σταθμός με την σειρά του, θα δηλώνεται (register) κάτω απ' τον λογαριασμό του χρήστη.

7.3.2 Πρόβλεψη καιρού

Η πρόβλεψη καιρού απαιτεί την τοποθέτηση αρκετών μετεωρολογικών σταθμών. Η δημιουργία ενός δικτύου απ' αυτούς, θα μας έδινε την δυνατότητα πρόβλεψης του καιρού, με την χρήση του κατάλληλου αλγορίθμου και της σωστής εκμετάλλευσης των δεδομένων της βάσης μας. Σε συνδυασμό με την προηγούμενη παράγραφο (7.3.1), ο κάθε χρήστης θα μπορεί να έχει πρόβλεψη καιρού απ' τους δικούς του μόνο σταθμούς.

Βιβλιογραφία

Πέρα απ' τους παρακάτω συνδέσμους αξίζει να σημειωθεί ότι χρησιμοποιήθηκαν γνώσεις από τα παρακάτω μαθήματα:

1. Εισαγωγή στον προγραμματισμό
2. Ηλεκτρονικά
3. Λογικά Κυκλώματα
4. Θεωρία Κυκλωμάτων
5. Μικροηλεκτρονική
6. Δομημένος προγραμματισμός
7. Αλγόριθμοι και δομές δεδομένων
8. Αντικειμενοστραφής προγραμματισμός
9. Βάσεις δεδομένων
10. Ψηφιακές Επικοινωνίες
11. Δίκτυα Η/Υ
12. Σ.Α.Μ. - Μικροελεγκτές
13. Ανάπτυξη Διαδικτυακών Εφαρμογών
14. Λειτουργικά Συστήματα
15. Ασφάλεια & Διαχείριση Διαδικτυακών Συστημάτων
16. Μηχανική Λογισμικού
17. Μηχατρονικά Συστήματα
18. Τεχνολογίες Ευρυζωνικών Δικτύων

Δικτυογραφία

1. https://el.wikipedia.org/wiki/%CE%9C%CF%80%CE%B1%CF%84%CE%B1%CF%81%CE%AF%CE%B1_%CE%B9%CF%8C%CE%BD%CF%84%CF%89%CE%BD_%CE%BB%CE%B9%CE%B8%CE%AF%CE%BF%CF%85
2. http://dynamicsolartech.com/about-us/how-solar-panels-work/
3. https://el.wikipedia.org/wiki/%CE%A6%CF%89%CF%84%CE%BF%CE%B2%CE%BF%CE%BB%CF%84%CE%B1%CF%8A%CE%BA%CF%8C_%CF%83%CF%8D%CF%83%CF%84%CE%B7%CE%BC%CE%B1
4. https://www.devobox.com/index.php?id_product=306&controller=product&id_lang=2
5. https://www.ideahellas.gr/product/%CF%86%CF%89%CF%84%CE%BF%CE%B2%CE%BF%CE%BB%CF%84%CE%B1%CF%8A%CE%BA%CE%B1-%CF%83%CF%85%CF%83%CF%84%CE%B7%CE%BC%CE%B1%CF%84%CE%B1/fotovoltaikos-syllektis-20watt-me-plaisio-alouminou-monokrystalliko-pyritiou/
6. http://blog.evandmore.com/lets-talk-about-the-panasonic-ncr18650b/
7. http://www.ardumotive.com/raspberrypigr.html
8. https://raspi.tv/2017/how-much-power-does-pi-zero-w-use
9. https://www.raspberrypi.org/products/raspberry-pi-zero-w/
10. https://www.adafruit.com/product/2542
11. https://cdn-shop.adafruit.com/datasheets/BST-BME280_DS001-10.pdf
12. https://www.banggood.com/CJMCU-280E-BME280-High-Precision-Atmospheric-Pressure-Sensor-For-Arduino-p-1103115.html?rmmds=myorder&cur_warehouse=CN

13. https://www.banggood.com/3_7V-9V-5V-2A-Adjustable-Step-Up-18650-Lithium-Battery-Charging-Discharge-Integrated-Module-p-1264852.html?rmmds=myorder&cur_warehouse=CN
14. https://www.banggood.com/18650-26650-Lithium-Li-ion-Battery-Tester-LCD-Meter-Voltage-Current-Capacity-p-1044589.html?rmmds=myorder&cur_warehouse=CN
15. https://grobotronics.com/voltage-regulator-switching-lm2576t-5v-3a.html
16. http://www.ti.com/lit/ds/symlink/lm2576.pdf
17. https://www.banggood.com/0_28-Inch-2_5V-30V-Mini-Digital-Voltmeter-p-974258.html?rmmds=myorder&ID=228&cur_warehouse=CN
18. https://www.cableworks.gr/ilektronika/arduino-and-microcontrollers/buck-and-boost-converters/buck-converter-step-down-dc-dc/xl4005-step-down-5a-buck-converter-438vdc/
19. https://el.wikipedia.org/wiki/Nodejs
20. https://benchmarksgame-team.pages.debian.net/benchmarksgame/faster/node-python3.html
21. https://el.wikipedia.org/wiki/MySQL
22. https://en.wikipedia.org/wiki/Express.js
23. http://docs.sequelizejs.com/
24. https://en.wikipedia.org/wiki/Angular_(web_framework)
25. https://el.wikipedia.org/wiki/Bootstrap
26. https://www.raspberrypi.org/downloads/raspbian/
27. https://github.com/initialstate/fona-pi-zero/wiki
28. https://rufus.ie/
29. https://www.npmjs.com/
30. http://pm2.keymetrics.io/docs/usage/startup/
31. https://www.nginx.com/blog/websocket-nginx/
32. https://www.ibm.com/support/knowledgecenter/en/SSWHYP_4.0.0/com.ibm.apimgmt.cmc.doc/task_apionprem_gernerate_self_signed_openSSL.html
33. https://learn.adafruit.com/adafruit-bme280-humidity-barometric-pressure-temperature-sensor-breakout/arduino-test
34. http://cactus.io/hookups/sensors/barometric/bme280/hookup-arduino-to-bme280-barometric-pressure-sensor-spi
35. https://cloud.google.com/maps-platform/pricing/sheet/