ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ ΗΛΕΚΤΡΟΝΙΚΗΣ
ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΜΕΛΕΤΗ ΚΑΙ ΑΝΑΠΤΥΞΗ ΠΛΑΤΦΟΡΜΑΣ ΕΠΕΞΕΡΓΑΣΤΗ ΑΝΟΙΚΤΟΥ ΛΟΓΙΣΜΙΚΟΥ ΓΙΑ ΤΗΝ ΥΛΟΠΟΙΗΣΗ ΜΕΤΕΩΡΟΛΟΓΙΚΟΥ ΔΙΚΤΥΟΥ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΣΤΑΜΑΤΟΠΟΥΛΟΣ ΣΤΑΜΑΤΙΟΣ

Επιβλέποντες: Η. ΚΟΥΚΟΥΤΣΗΣ
ΚΑΘΗΓΗΤΗΣ Ε.Μ.Π.

Δρ. – Μηχ. Χ. ΜΑΡΜΑΛΙΔΗΣ
Γραφείο Έρευνας και Τεχνολογικών Εξελίξεων Πολεμικού Ναυτικού

Αθήνα, Ιούνιος 2015

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ ΗΛΕΚΤΡΟΝΙΚΗΣ
ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΜΕΛΕΤΗ ΚΑΙ ΑΝΑΠΤΥΞΗ ΠΛΑΤΦΟΡΜΑΣ ΕΠΕΞΕΡΓΑΣΤΗ ΑΝΟΙΚΤΟΥ ΛΟΓΙΣΜΙΚΟΥ ΓΙΑ ΤΗΝ ΥΛΟΠΟΙΗΣΗ ΜΕΤΕΩΡΟΛΟΓΙΚΟΥ ΔΙΚΤΥΟΥ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΣΤΑΜΑΤΟΠΟΥΛΟΣ ΣΤΑΜΑΤΙΟΣ

Επιβλέποντες: Η. ΚΟΥΚΟΥΤΣΗΣ
ΚΑΘΗΓΗΤΗΣ Ε.Μ.Π.

Δρ. – Μηχ. Χ. ΜΑΡΜΑΛΙΔΗΣ
Γραφείο Έρευνας και Τεχνολογικών Εξελίξεων Πολεμικού Ναυτικού

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 29η Ιουνίου 2015.

Η. ΚΟΥΚΟΥΤΣΗΣ	Κ. ΠΑΠΑΟΔΥΣΣΕΥΣ	Η. ΚΑΜΠΟΥΡΑΚΗΣ
ΚΑΘΗΓΗΤΗΣ Ε.Μ.Π.	ΚΑΘΗΓΗΤΗΣ Ε.Μ.Π.	ΚΑΘΗΤΗΤΗΣ Ε.Μ.Π.

Αθήνα, Ιούνιος 2015

.....
ΣΤΑΜΑΤΟΠΟΥΛΟΣ ΣΤΑΜΑΤΙΟΣ
Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

ΠΕΡΙΛΗΨΗ

Τα τελευταία χρόνια έχει εκδηλωθεί τεράστιο ενδιαφέρον για τα αυτοματοποιημένα συστήματα συλλογής και επεξεργασίας δεδομένων. Το ενδιαφέρον αυτό εκφράζεται τόσο από την ακαδημαϊκή κοινότητα σε ερευνητικό επίπεδο, όσο και από τους εν δυνάμει χρήστες της τεχνολογίας. Συλλογή δεδομένων καλείται η μέθοδος μέσω της οποίας μεγέθη όπως η τάση , το ρεύμα η θερμοκρασία , κ . α ., συλλέγονται, απεικονίζονται, και καταγράφονται μέσω κάποιας υπολογιστικής μονάδας . Για το σκοπό αυτό ένα σύστημα συλλογής δεδομένων (data acquisition system) μετατρέπει το εισερχόμενο σήμα σε ψηφιακό δεδομένο. Υπεύθυνο για την επεξεργασία των αποκτημένων πληροφοριών είναι το λογισμικό που φέρει το σύστημα DAQ μέσω του οποίου εκτελούνται λειτουργίες όπως : μετρήσεις ηλεκτρικών μεγεθών, ανάλυση σήματος στο πεδίο της συχνότητας / χρόνου , στατιστική ανάλυση σήματος, και πλήθος άλλων επεξεργασιών . Με τη χρησιμοποίηση μονάδων υψηλής τεχνολογίας και λογισμικών επεξεργασίας σημάτων, μπορεί να επιτευχθεί υψηλή ακρίβεια μέτρησης και ελέγχου της πληροφορίας του σήματος για την αξιοποίησή του τόσο σε εργαστηριακή, όσο και σε βιομηχανική εφαρμογή .

Η παρούσα διπλωματική έχει διπλό στόχο. Αφενός μεν να αποτελέσει μια εισαγωγή στα αυτοματοποιημένα συστήματα συλλογής και επεξεργασίας δεδομένων μέσω αισθητηρίων με την χρήση μικροελεγκτών (arduino) και μικροϋπολογιστών (raspberry pi) και αφετέρου να παρουσιάσει αναλυτικά την υλοποίηση ενός συγκεκριμένου συστήματος λήψης και αποθήκευσης δεδομένων καθώς και την απομακρυσμένη πρόσβαση σε αυτά μέσω του διαδικτύου. Με αυτόν τον τρόπο επομένως μπορεί να καλυφθεί ένα ευρύ φάσμα εφαρμογών.

Πιο συγκεκριμένα στο *κεφάλαιο 1* πραγματοποιείται μια εισαγωγή στον μικροελεγκτή Arduino, στα διαθέσιμα αισθητήρια που μπορεί να υποστηρίξει για τις διάφορες εφαρμογές καθώς και τα διαθέσιμα shields/προεκτάσεις για την υλοποίηση σύνθετων εφαρμογών.
Στην συνέχεια στο *κεφάλαιο 2* παρουσιάζεται η διαδικασία ανάπτυξης και υλοποίησης προγραμμάτων σε γλώσσα wiring μέσω του προγραμματιστικού περιβάλλοντος IDE.

Ύστερα, στο *κεφάλαιο 3* πραγματοποιείται μια εισαγωγή στις βασικές δομές, στη θεωρία λειτουργίας και στις εφαρμογές των υπολογιστικών συστημάτων raspberry pi και την συμβολή τους στο internet of things.

Στο *κεφάλαιο 4* περιγράφεται συνολικά το υλικό που χρησιμοποιήθηκε για την υλοποίηση του αυτοματοποιημένου συστήματος συλλογής και επεξεργασίας δεδομένων και ειδικότερα ενός αυτόνομου μετεωρολογικού σταθμού που θα συλλέγει δεδομένα όπως η θερμοκρασία η υγρασία και η ατμοσφαιρική πίεση. Επιπροσθέτως γίνεται αναλυτική περιγραφή του τρόπου διασύνδεσης των επιμέρους συστημάτων που έχουν χρησιμοποιηθεί, παρουσιάζονται οι αισθητήρες που

χρησιμοποιήθηκαν, τα κριτήρια επιλογής τους, ο τρόπος συνδεσμολογίας καθώς και οι βιβλιοθήκες που αξιοποιήθηκαν για την διεπαφή τους με την πλακέτα Arduino.

Στο *κεφάλαιο 5* γίνεται μια εισαγωγή στον αυτόνομο μετεωρολογικό σταθμό, η ανάλυση των μετεωρολογικών φαινομένων και παρουσιάζεται η μεθοδολογία καταγραφής τους. Κατόπιν πραγματοποιείται η παρουσίαση του μετεωρολογικού μοντέλου Ζαμπρέτι (Zambretti) βάσει του οποίου πραγματοποιούνται οι προγνώσεις καιρού και επομένως η αξιοποίηση των δεδομένων που έχουν συλλεχθεί και αποθηκευτεί.

Επιπρόσθετα στο *κεφάλαιο 6* παρουσιάζεται η διαδικασία παραμετροποίησης του hardware πριν την υλοποίηση της εφαρμογής.

Εν τω μεταξύ στο *κεφάλαιο 7* περιγράφεται και σχολιάζεται αναλυτικά η διαδικασία κατασκευής του συστήματος DAQ – ΑΜΣ, καθώς και η δομή του κώδικα που απαιτείται για να υλοποιήσει το συγκεκριμένο αυτοματοποιημένο συστήματα συλλογής και επεξεργασίας δεδομένων της πλατφόρμας του Arduino και του raspberry pi με σκοπό την αξιοποίηση τους ως αυτόνομο μετεωρολογικό σταθμό, με δυνατότητες καταγραφής, επεξεργασίας και αποθήκευσης των μετεωρολογικών δεδομένων καθώς και την απομακρυσμένη πρόσβαση σε αυτά για την περεταίρω αξιοποίησή τους .

Τέλος στο *κεφάλαιο 8* παρουσιάζεται μια μελλοντική προέκταση της διπλωματικής και αξιοποίησής της σε ένα ευρύτερο φάσμα εφαρμογών.

Λέξεις-Κλειδιά

Μικροελεγκτής, μικροεπεξεργαστής, κώδικας wiring, αισθητήρια, DAQ, συλλογή και επεξεργασία δεδομένων, αυτοματοποιημένο σύστημα .

ABSTRACT

In recent years, a huge interest in automated data collection and processing systems is manifested. This interest is expressed both by the academic community at research level and the potential users of the technology.

Collection of data is the method by which items such as voltage, current, temperature e.t.c. are collected, displayed, and recorded via a computational unit. For this purpose, a data collection system (data acquisition system) converts the incoming signal to digital data. Responsible for the processing of acquired information is the software that bears the DAQ system through which operations such as: electrical quantity measurements, signal analysis in the frequency and/or time domain and statistical signal analysis are performed to name a few.

By using high-tech units and signal processing software, high measurement accuracy and information control of the signal can be achieved for utilization both in lab as well as in industrial application.

This thesis has a double objective. Firstly to provide an introduction to automated data collection and processing systems by sensors using microcontrollers (arduino) and micro (raspberry pi) and to present in detail the implementation of a particular system of data reception and storage as well as remote access via internet . In this way it can therefore cover a wide range of applications.

More particular, in *Chapter 1* an introduction to Arduino microcontroller is presented, together with the available sensors that can support the various applications and the available shields / extensions to implement complex applications.

In *Chapter 2* presents the process of developing and implementing programs in wiring language through the programming IDE environment.

In *Chapter 3* an introduction is made to the basic structures, the operating theory and applications of the raspberry pi systems and their contribution to the internet of things.

Chapter 4 describes overall the hardware used for the implementation of the automated data collection and processing system and in particular an autonomous weather station able to collect data such as temperature, humidity and atmospheric pressure. Additionally there is a detailed description of the interface of the individual systems that have been used, sensors used and their selection criteria, wiring and libraries were used for the Arduino interface board.

Chapter 5 includes an introduction to the autonomous weather station, an analysis of weather events and a presentation of their recording methodology. Following that, the Zambretti meteorological model is presented under which weather forecasts are made using the data collected and stored.

Chapter 6 shows the hardware configuration customisation before implementation of the application.

Chapter 7 describes and discuss in detail the system construction process DAQ - OMC and the structure of the code required to implement this automated data collection and processing system on the Arduino and raspberry pi platforms, in order to be used as autonomous weather station, with logging, processing and storage of meteorological data as well as remote access for further exploitation.

Finally, *Chapter 8* presents future extension of the thesis and its exploitation on a broader range of applications.

KEYWORDS

Microcontroller, microprocessor, wiring code, sensors, DAQ, data collection and processing, automated system .

Ευχαριστίες

Θα ήθελα να ευχαριστήσω θερμά τον καθηγητή μου κ. Ηλία Κουκούτση, για την άριστη πληροφόρηση που μας παρείχε σχετικά με τα συστήματα DAQ, τους μικροϋπολογιστές και τους μικροελεγκτές καθώς επίσης και για την βοήθειά του με τις υποδείξεις του σε κάθε δυσκολία που αντιμετωπίσαμε. Τέλος θα ήθελα να ευχαριστήσω τον Δρ. – Μηχ. Χρήστο Μαρμαλίδη, Ειδικό Επιστήμονα του Γραφείου Έρευνας και Τεχνολογικών Εξελίξεων Ναυτικού (Γ.Ε.Τ.Ε.Ν) για τις παρατηρήσεις και τις προτάσεις του που συντέλεσαν στην βελτίωση της διπλωματικής.

Αθήνα, 29 Ιουνίου 2015

Σταμάτιος Σταματόπουλος

Αφιερώνεται με ιδιαίτερη ευγνωμοσύνη στην οικογένεια μου.

Σταμάτιος Σταματόπουλος

Περιεχόμενα

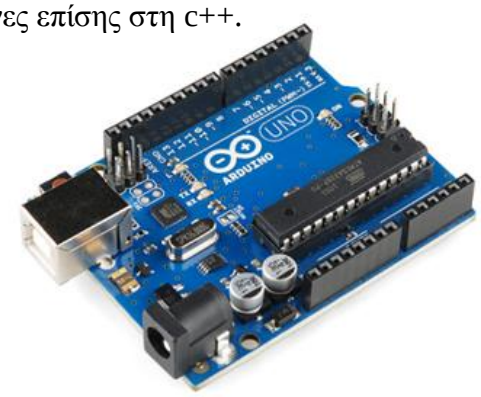
1. Ολοκληρωμένα μικροσυστήματα επεξεργασίας – Arduino

1.1. Η ιστορία του Arduino

Στην Ivrea της Ιταλίας το 2005 ξεκίνησε ένα σχέδιο προκειμένου να κατασκευαστεί μια συσκευή για τον έλεγχο προγραμμάτων και διαδραστικών σχεδίων από μαθητές, με χαμηλότερο κόστος από άλλα πρωτότυπα συστήματα που ήταν διαθέσιμα εκείνη την περίοδο. Εμπνευστής του σχεδίου αυτού ήταν ο καθηγητής Massimo Banzi, ο οποίος θέλησε να καταστήσει ευκολότερη τη μάθηση των ηλεκτρονικών για τους μαθητές του. Για το λόγο αυτό ζήτησε βοήθεια από τον David Cuatrielles, μηχανικό από το πανεπιστήμιο του Malmo. Οι ιδρυτές Massimo Banzi και David Cuatrielles ξεκίνησαν να παράγουν πλακέτες σε ένα μικρό εργοστάσιο στην Ivrea, στην ίδια περιοχή στην οποία στεγαζόταν η εταιρία υπολογιστών Olivetti. Ονόμασαν το έργο τους Arduin of Ivrea «Arduino» που μεταφράζεται ελεύθερα ως «γενναίος φίλος», με σκοπό να δημιουργήσουν έναν μικροελεγκτή προσιτό ως προς τη χρήση του. Την ανάπτυξη του λογισμικού για τον μικροελεγκτή την ανέθεσαν σε δύο φοιτητές του πανεπιστημίου Malmo. Η πρώτη παρτίδα που παράχθηκε αποτελούνταν από 200 μικροελεγκτές, υπό την εποπτεία του ηλεκτρολόγου μηχανικού Gianluca Martino. Οι μικροελεγκτές αυτοί ονομάστηκαν Serial Arduino και περιελάμβαναν μία ATmega8 με άμεση σύνδεση RS-232 με το μικροελεγκτή και όλα τα επιμέρους περιφερειακά του. Έκτοτε έχουν δημιουργηθεί δεκάδες πλακέτες Arduino παγκοσμίως, χάρη στο ευέλικτο και εύκολο στη χρήση hardware και software. Το Arduino μπορεί να χρησιμοποιηθεί από τον οποιοδήποτε με στοιχειώδεις γνώσεις ηλεκτρονικών που ενδιαφέρεται να δημιουργήσει διαδραστικά αντικείμενα ή περιβάλλοντα, με μόνο περιορισμό τη φαντασία του.

1.2. Τι είναι το Arduino

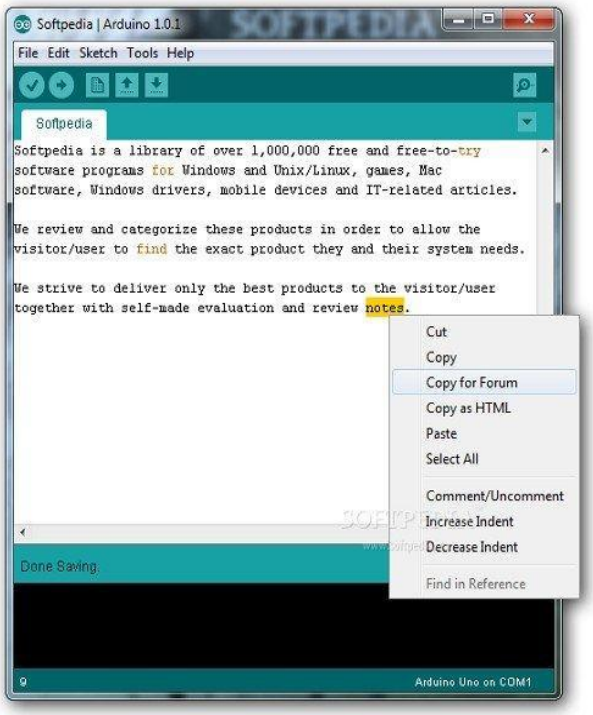
Το Arduino (Εικόνα 1) αποτελεί μια υπολογιστική πλατφόρμα βασιζόμενη σε μητρική πλακέτα ανοικτού κώδικα, που περιέχει έναν προγραμματιζόμενο μικροελεγκτή (MCU) και εισόδους/εξόδους (I/O) για σύνδεση με το φυσικό κόσμο. Ο μικροελεγκτής του προγραμματίζεται χρησιμοποιώντας τη γλώσσα προγραμματισμού wiring. Είναι βασισμένη στη c/c++, και περιλαμβάνει ένα σύνολο από βιβλιοθήκες, υλοποιημένες επίσης στη c++.



ARDUINO™

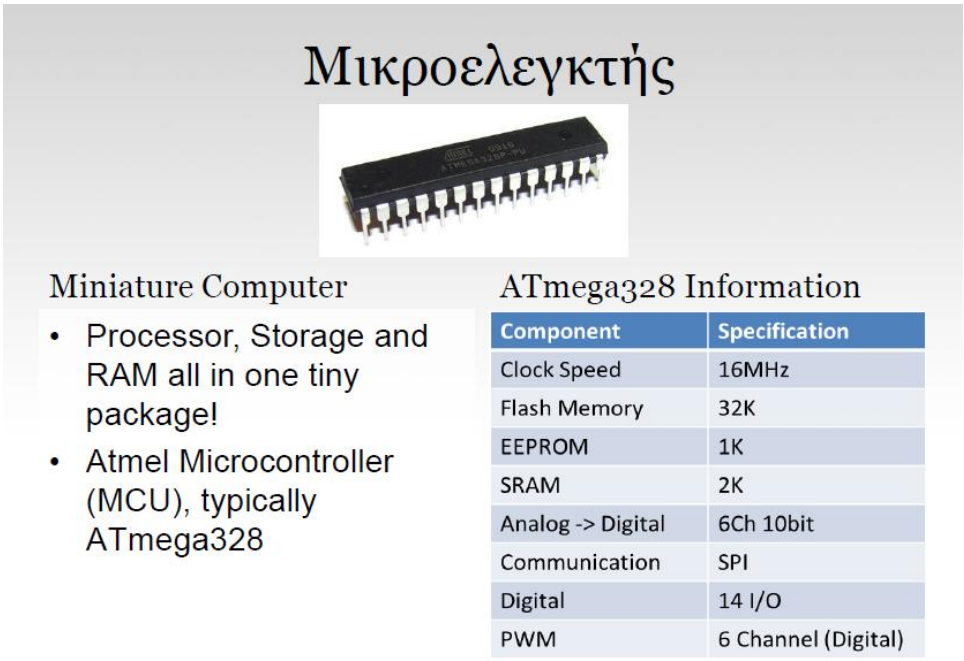
Εικόνα 1, Το arduino και το λογότυπό του

Μέσα από το περιβάλλον ανάπτυξης κώδικα IDE (ελεύθερο και δωρεάν, Εικόνα 2) η γλώσσα wiring μπορεί να συνταχθεί και να υλοποιηθεί σε οποιοδήποτε λειτουργικό σύστημα. Είναι ανοικτού κώδικα λογισμικό και μας επιτρέπει να αναπτύξουμε ένα υπολογιστικό σύστημα, το οποίο θα ελέγχει συσκευές του φυσικού κόσμου. Το Arduino μπορεί να χρησιμοποιηθεί για να αναπτύξουμε διαδραστικά αυτοματοποιημένες οντότητες, ικανές να δεχθούν ως εισόδους μια πληθώρα αισθητηρίων οργάνων, αλλά και διαφόρων συσκευών εξόδου, που είναι ικανές να ελέγχουν άλλες συσκευές του φυσικού κόσμου. Η υπολογιστική πλατφόρμα του Arduino μας παρέχει τη δυνατότητα να υλοποιούμε project αυτόνομα σε επίπεδο hardware ή αλληλοεξαρτώμενα σε επίπεδο software, που επικοινωνούν με άλλα ολοκληρωμένα μικροσυστήματα επεξεργασίας και υπολογιστές.



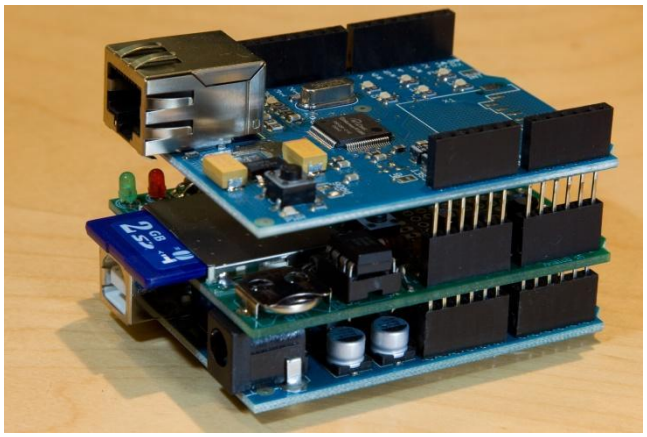
Εικόνα 2 , Περιβάλλον ανάπτυξης κώδικα IDE

Ειδικότερα, η υπολογιστική πλατφόρμα Arduino αποτελείται από μια πλακέτα με πυρήνα, έναν μικροελεγκτή Atmel AVR, όπως ο ATmega328 ή ο ATmega2560 (Εικόνα 3) ανάλογα με την έκδοσή του. Διαθέτει από 8 έως 16 σειριακές θύρες και έως 50 ψηφιακές (αναλόγως με τον τύπο και την έκδοση της πλακέτας) με τις οποίες αλληλεπιδρά με διάφορες συσκευές.



Εικόνα 3, Ο ATmega328

Επιπρόσθετα μέσω καρτών shields προσφέρεται η δυνατότητα πρόσθετης διασύνδεσης- επέκτασης του μικροελεγκτή με συσκευές που επικοινωνούν ασύρματα (Ethernet shield, Bluetooth shield κ.α., Εικόνα 4), καλύπτοντας με αυτό τον τρόπο μεγάλο εύρος απαιτήσεων.



Εικόνα 4, Shields

Όλες οι πλατφόρμες αποτελούνται από ένα γραμμικό ρυθμιστή τάσης 5V και έναν ταλαντωτή κρυστάλλου. Ο μικροελεγκτής είναι από κατασκευής του προγραμματισμένος με ένα bootloader (μικροκώδικας εκκίνησης υλικού), ώστε να μη χρειάζεται εξωτερικός προγραμματιστής όταν συνδέουμε νέο υλικό. Όλες οι πλακέτες είναι προγραμματισμένες μέσω μιας σειριακής σύνδεσης RS-232, και ο τρόπος με τον οποίο αυτό υλοποιείται διαφοροποιείται ανάλογα με την έκδοση. Συμπληρωματικά οι σειριακές πλακέτες Arduino περιέχουν ένα κύκλωμα αντιστροφής ανάμεσα στα σήματα των επιπέδων RS-232 και TTL. Ο προγραμματισμός της υπολογιστικής πλατφόρμας Arduino πραγματοποιείται μέσω USB, εφαρμόζοντας ένα chip προσαρμογέα usb to serial, όπως το FTDI FT232 ή εναλλακτικά χρησιμοποιούν ένα

αποσπώμενο USB σε πλακέτα σειριακού μετασχηματιστή ή Bluetooth ή άλλες μεθόδους. Η πλακέτα του Arduino έχει εκτεθειμένες τις περισσότερες επαφές I/O για άμεση χρήση με άλλα κυκλώματα ή add-on modules γνωστά και ως «ασπίδες». Τα add-on modules περιλαμβάνουν pins και από τις δύο πλευρές τους, ώστε να στοιβάζονται και να λειτουργούν παράλληλα, χρησιμοποιώντας τις ίδιες επαφές I/O. Η πλακέτα μπορεί να τροφοδοτηθεί είτε με τροφοδοτικό AC/DC Adapter των 7-12V, είτε απευθείας από τη USB θύρα του ηλεκτρονικού υπολογιστή, είτε μέσω μπαταρίας, όπου τα καλώδιά της θα τοποθετηθούν στα pin GND και Vin. Η επιλογή τροφοδοσίας USB ή εξωτερικής πηγής γίνεται αυτόματα. Τέλος, περιλαμβάνει ένα κουμπί επανεκκίνησης Reset σε περίπτωση που θέλουμε να το επαναφέρουμε στις εργοστασιακές ρυθμίσεις.

Εικόνα 5, Τροφοδοσία arduino από μπαταρία 9V

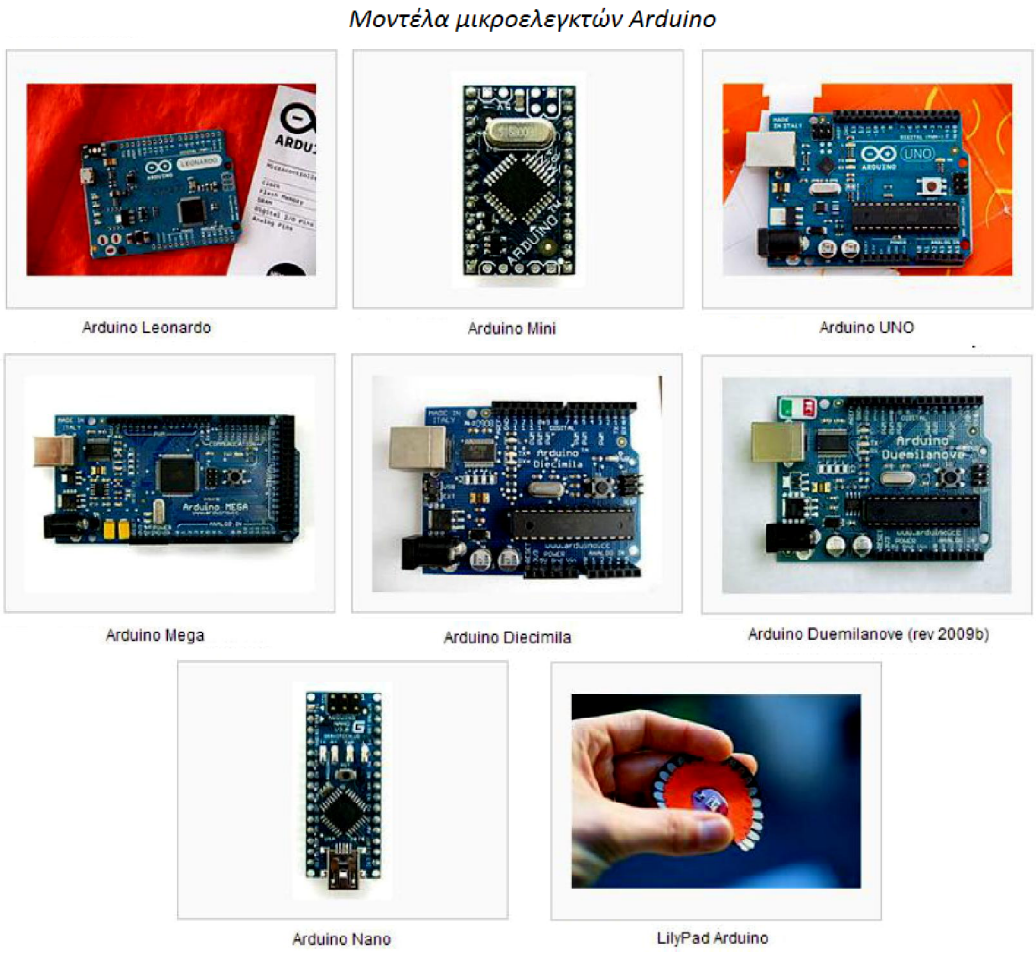
1.3. Μοντέλα Arduino

Τα μοντέλα Arduino (Εικόνα 6) που κυκλοφορούν στην αγορά μέχρι σήμερα είναι:

- Arduino UNO
- Arduino LilyPad Usb
- Arduino Ethernet
- Arduino Leonard
- Arduino Mega 2560
- Arduino Mega ADK
- Arduino Mega ADK REV3
- Arduino Frio
- Arduino Nano
- Arduino Pro
- Arduino Pro Mini
- Arduino Micro
- Arduino Due
- Arduino Esplora

Οι βασικές διαφορές τους εντοπίζονται στο πλήθος των pin που διαθέτουν, στα συμβατά shields, στην υπολογιστική ισχύ του μικροελεγκτή που χρησιμοποιούν, στην

υποστήριξη εξωτερικών interrupt, στο πλήθος των σειριακών interface και στο μέγεθος της μνήμης SRAM και EEPROM.



Εικόνα 6, Διάφοροι τύποι arduino

1.4. Arduino Shields

Τα shield είναι σχεδιασμένα ώστε να κουμπώνουν πάνω στο Arduino και να προωθούν τις υποδοχές τους, για να μπορούμε να συνδέουμε επιπλέον εξαρτήματα ή επόμενα shields. Αξίζει να τονιστεί ότι το κάθε shield χρησιμοποιεί ορισμένους από τους πόρους συνδεσιμότητας του Arduino, επομένως δε μπορούμε να συνδέσουμε απεριόριστα shield.

Το βασικό πλεονέκτημα των shield είναι ότι συνοδεύονται συνήθως από έτοιμες βιβλιοθήκες που επιτρέπουν να προγραμματίζουμε τα sketch σε high level. Κατά συνέπεια δεν χρειάζεται να διαβάζουμε datasheet ή να έχουμε εξειδικευμένες γνώσεις ηλεκτρονικής για να συνδέσουμε και να λειτουργήσουμε π.χ. ένα GPS module πάνω στο Arduino.

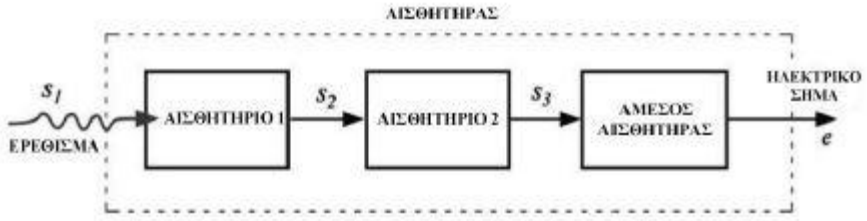
Πιο συγκεκριμένα κάθε φορά που συνδέεται ένα shield, εγκαθιστούμε τη βιβλιοθήκη που το συνοδεύει και χρησιμοποιούμε μια έτοιμη συνάρτηση, όπως για παράδειγμα την getLocation, για να πάρουμε το γεωγραφικό στίγμα και να το επεξεργαστούμε περαιτέρω στο sketch μας.

Από όλα τα παραπάνω γίνεται φανερό ότι δεν συνιστάται η αγορά κάποιας έκδοσης του Arduino που δεν είναι 100% συμβατή με τα shield της εφαρμογής που απαιτούνται για την υλοποίησή της.
Κάποιες από τις μονάδες επέκτασης της υπολογιστικής πλατφόρμας Arduino – shields είναι:

- Arduino GSM shield
- Arduino Ethernet shield
- Arduino WIFI shield
- Arduino WIFI SD shield
- Arduino Motor shield
- Arduino GPS shield
- Arduino PROTO shield

1.5. Arduino Sensors

Αισθητήρια ονομάζουμε τις ηλεκτρονικές διατάξεις οι οποίες είναι σε θέση να μετατρέπουν ενέργεια άλλης μορφής σε ηλεκτρική κατάλληλη για να μετρηθεί από μετρητικές διατάξεις. Το αισθητήριο έρχεται να γεφυρώσει ένα χάσμα μεταξύ του φυσικού κόσμου και των οργάνων που πρέπει να δημιουργήσει ο άνθρωπος, ώστε να έχει μια ολοκληρωμένη εικόνα των μεταβολών του. Με άλλα λόγια αποτελούν ηλεκτρονικές συσκευές οι οποίες παρακολουθούν τις μεταβολές του φυσικού κόσμου και παράγουν μια έξοδο ηλεκτρικού σήματος, ανάλογη των μεταβολών αυτών. Τα αισθητήρια δε λειτουργούν αυτόνομα, αλλά είναι πάντα μέρος ενός ευρύτερου συστήματος, που μπορεί να ενσωματώνει άλλους ανιχνευτές, επεξεργαστές σήματος, καταγραφείς δεδομένων κ. α.



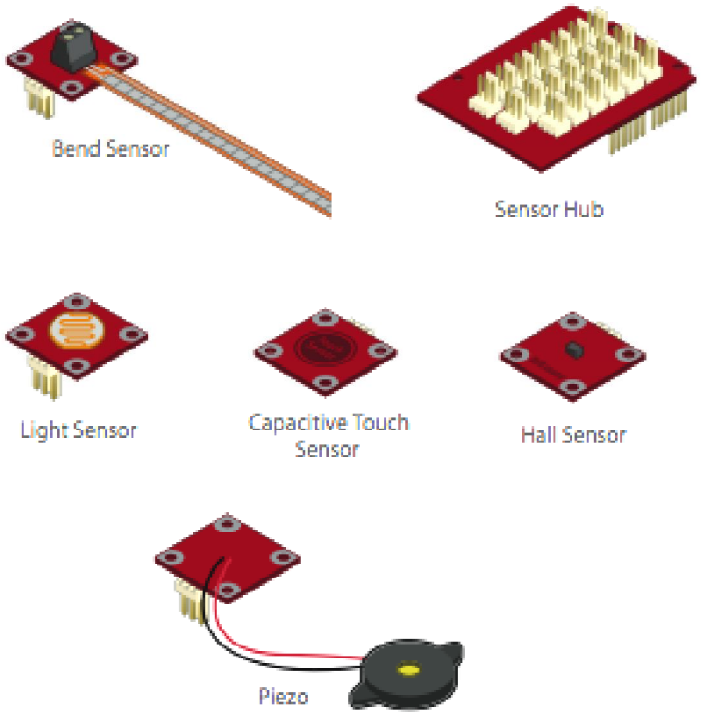
Εικόνα 7, Δομή αισθητήρα

Κάποια από τα πιο γνωστά αισθητήρια arduino που κυκλοφορούν στην αγορά παρουσιάζονται παρακάτω:

- Push buttons Sensors
- Touch pads Sensors
- Photoresistors
- Variable resistors
- Ultrasound -Proximity range finder Sensors
- Thermistors (temperature) Sensors
- Electro Magnetic Sensors

- Gas Sensors
- Hall Effect Sensors
- Magnetometer
- PIR Motion sensor
- Digital Humidity and Temperature Sensor
- Sensor Hub
- Bend Sensor
- Capacitive Touch Sensor
- Light Sensor
- Piezo Sensor

Οι αισθητήρες συνδέονται στις ψηφιακές και αναλογικές εισόδους του arduino, το οποίο στη συνέχεια, κάνοντας χρήση των κατάλληλων βιβλιοθηκών, διαβάζει τις τιμές τάσης στην είσοδό του και στη συνέχεια μετατρέπει την εισερχόμενη τάση σε κατάλληλες τιμές, τις οποίες μπορούμε να αξιοποιήσουμε.
Οι αισθητήρες που χρησιμοποιούνται από την υπολογιστική πλατφόρμα arduino, είναι κατά κανόνα φθηνοί και αξιόπιστοι, με μικρή κατανάλωση ρεύματος και μικροί σε μέγεθος. Λόγω της πληθώρας των αισθητηρίων που κυκλοφορούν στην αγορά (Εικόνα 8), μπορούμε εύκολα να αναζητήσουμε στο διαδίκτυο και σε διάφορες κοινότητες που ασχολούνται με το arduino, αφθονία βιβλιοθηκών.



Εικόνα 8, Διάφοροι τύποι αισθητηρίων

2. Πλατφόρμα ανάπτυξης λογισμικού Arduino IDE

2.1 Δομή γλώσσας Wiring

Όπως αναφέρθηκε σε προηγούμενη παράγραφο, η γλώσσα του Arduino βασίζεται στη γλώσσα Wiring, μια παραλλαγή C/C++ για μικροελεγκτές αρχιτεκτονικής AVR, όπως ο ATmega, και υποστηρίζει όλες τις βασικές δομές της C καθώς και μερικά χαρακτηριστικά της C++. Για compiler χρησιμοποιείται ο AVR gcc και ως βασική βιβλιοθήκη C χρησιμοποιείται η AVR libc.
Λόγω της καταγωγής της από την C, στη γλώσσα του Arduino μπορούμε να χρησιμοποιήσουμε τις ίδιες βασικές εντολές και συναρτήσεις, με την ίδια σύνταξη, τους ίδιους τύπων δεδομένων και τους ίδιους τελεστές όπως και στη γλώσσα C. Πέρα από αυτές όμως, υπάρχουν κάποιες ειδικές εντολές, συναρτήσεις και σταθερές που βοηθούν για την διαχείριση του ειδικού hardware του Arduino.
Για την σύνταξη ενός προγράμματος ξεκινάμε χωρίζοντας πάντα το πρόγραμμα σε τρία μέρη: τη δομή, τις μεταβλητές/σταθερές και τις συναρτήσεις.
Πρώτα εξετάζουμε το κομμάτι της δομής. Η δομή ενός προγράμματος Arduino (Σχήμα 1) χωρίζεται σε τρία μέρη με την ακόλουθη σειρά: τη δήλωση μεταβλητών, το κομμάτι κώδικα που περιέχει την αρχικοποίηση καταστάσεων και μεταβλητών καθώς και τον κώδικα που θέλουμε να τρέξει μόνο μια φορά στο Arduino και το κομμάτι του κώδικα loop() που περιέχει το κυρίως πρόγραμμα μας και θα τρέχει συνέχεια (μέχρι να βγάλουμε το Arduino από το ρεύμα).

Σχήμα 1, Βασική δομή προγράμματος σε γλώσσα Wiring

Οι μεταβλητές παίζουν πολύ σημαντικό ρόλο στην σύνταξη του κώδικα καθώς αλλάζοντας τις τιμές τους κατά την διάρκεια εκτέλεσης του προγράμματος μπορούμε να επιτύχουμε διάφορες λειτουργίες. Μπορούν να πάρουν διάφορες τιμές, όπως νούμερα, χαρακτήρες, και τα δύο ή να έχουν λογική τιμή True ή False (αληθής - ψευδής). Ανάλογα με την τιμή αυτή τις αρχικοποιούμε ή τις δηλώνουμε αντίστοιχα στο πρώτο τμήμα της δομής του προγράμματος μας.
Κάθε όνομα που δίνουμε σε μια μεταβλητή ή σταθερά θα πρέπει να υπάρχει μόνο μια φορά μέσα στον κώδικά μας. Στις σταθερές ισχύει ότι και παραπάνω, αλλά πριν τον τύπο βάζουμε το χαρακτηριστικό 'const' (const “τύπος” “όνομα σταθεράς”). Μια σταθερά χρησιμοποιείται σαν συντόμευση μέσα στο πρόγραμμα, αποφεύγοντας να γράφουμε τιμές τις οποίες μπορεί να ξεχάσουμε παρακάτω. Μια μεταβλητή μπορεί να μην έχει αρχική τιμή, και η τιμή της να υπολογίζεται μετά από εκτέλεση κάποιας εντολής μέσα στο πρόγραμμα.
Οι πιο σημαντικές από τις εντολές ενός προγράμματος επεξηγούνται στους πίνακες που ακολουθούν:

Βασικές δομές/εντολές	**Λειτουργία**
setup()	Η συνάρτηση setup() καλείται στην αρχή κάθε σχεδίου. Εκεί αρχικοποιούνται οι μεταβλητές, οι λειτουργίες των pins, οι βιβλιοθήκες κ.τ.λ. Εκτελείται μόνο μια φορά ή κάθε φορά που γίνεται reset.
loop()	Η λειτουργία loop() εκτελείται μετά τη setup(), και όπως δηλώνει και το όνομά της, εκτελείται συνέχεια και ουσιαστικά εκεί βρίσκεται το πρόγραμμα που θέλουμε να εκτελεστεί.
if…else	Δομή επιλογής.
switch case	Δομή επιλογής ανάλογα με την τιμή μιας μεταβλητής. Σε κάθε περίπτωση χρησιμοποιούμε την εντολή break για να τερματίσει η δομή.
for	Δομή επανάληψης με γνωστό αριθμό επαναλήψεων.
while	Δομή επανάληψης· εκτελείται ο βρόχος συνέχεια έως ότου η συνθήκη που ακολουθεί τη while να γίνει ψευδής (false).
do…while	Δομή επανάληψης, ο βρόγχος εκτελείται τουλάχιστον μία φορά γιατί η συνθήκη ελέγχεται στο τέλος του βρόγχου.
break	Χρησιμοποιείται για την έξοδο από ένα βρόχο (for, while, do..while) ή από

	μια switch…case παρακάμπτοντας την κανονική εκτέλεση των εντολών.
continue	Χρησιμοποιείται για την παράκαμψη μιας επανάληψης ενός βρόχου και μεταπήδηση στην επόμενη επανάληψη.
return	Τερματίζει μια συνάρτηση ή επιστρέφει μια τιμή σε μια συνάρτηση που έχει κληθεί (τερματίζοντας την).
goto	Μεταφέρει τη ροή του προγράμματος σε άλλο σημείο χρησιμοποιώντας ετικέτες.
;	Χρησιμοποιείται στο τέλος μιας εντολής.
{}	Περικλείουν ένα σύνολο εντολών. Ένα άγκιστρο όταν ανοίγει θα πρέπει κάπου να κλείνει.
//	Εισάγεται από τον προγραμματιστή για ευκολία ανάγνωσης του σχεδίου.
#define	Με τη #define δηλώνονται στην αρχή του σχεδίου οι σταθερές που δεν θα αλλάξουν τιμή σε όλη τη διάρκεια της εκτέλεσης του σχεδίου.
#include	Χρησιμοποιείται για να δηλώνονται στο σχέδιο διάφορες βιβλιοθήκες.
delay(x)	Σταματά προσωρινά την ροή του προγράμματος για *time* ms. Η παράμετρος *time* είναι unsigned long (από 0 ως 2^32). Σημειώστε ότι παρά την προσωρινή παύση, συναρτήσεις των οποίων η εκτέλεση ενεργοποιείται από interrupt θα εκτελεστούν κανονικά κατά την διάρκεια μιας delay.
millis	Μετρητής που επιστρέφει το χρονικό διάστημα σε ms από την στιγμή που άρχισε η εκτέλεση του προγράμματος. Λάβετε υπόψη ότι λόγω του τύπου μεταβλητής (unsigned long δηλ. 32bit) θα γίνει overflow σε 2^32ms δηλαδή περίπου σε 50 μέρες, οπότε ο μετρητής θα ξεκινήσει πάλι από το μηδέν.
delayMicroseconds(x)	Καθυστερεί την εκτέλεση του σχεδίου για x microseconds.

Πίνακας 1, Βασικές δομές-Εντολές

Μεταβλητές/Τύποι δεδομένων	**Λειτουργία**
HIGH/LOW	Όταν διαβάζεται ή γράφεται η τιμή ενός digital pin, υπάρχουν δύο δυνατές τιμές που μπορεί να πάρει:

	η HIGH και η LOW.
true/false	Λογικές σταθερές για να εκφραστεί η «αλήθεια» ή το «ψέμα».
INPUT	Έχει την τιμή 0 και είναι αντίστοιχη του λογικού false.
OUTPUT	Έχει την τιμή 1 και είναι αντίστοιχη του λογικού true.
void	Χρησιμοποιείται για τη δήλωση συναρτήσεων. Οι συναρτήσεις αυτές δεν επιστρέφουν καμιά τιμή.
boolean	Μία Boolean μεταβλητή κρατά μια από τις τιμές true ή false. Κάθε Boolean μεταβλητή καταλαμβάνει 1 byte μνήμης.
char	Μία μεταβλητή char κρατά έναν χαρακτήρα. Οι χαρακτήρες αποθηκεύονται σαν αριθμοί με πρόσημο από -128 έως 127 (σύμφωνα με τον κώδικα ASCII)
unsigned char	Τύπος char χωρίς πρόσημο (κωδικοποιεί αριθμούς από 0 έως 255).
byte	Μία μεταβλητή byte αποθηκεύει έναν αριθμό χωρίς πρόσημο 8bits από το 0 έως 255 (ίδιο με unsigned char).
int	Μία μεταβλητή int αποθηκεύει ακέραιους αριθμούς. Καταλαμβάνει 2 bytes στη μνήμη. Το εύρος των ακεραίων είναι από -32,768 έως 32,767.
unsigned int	Μια τέτοια μεταβλητή αποθηκεύει ακεραίους αριθμούς χωρίς πρόσημο. Καταλαμβάνει 2 bytes στη μνήμη. Το εύρος των ακεραίων είναι από 0 έως 65,535.
word	Μία μεταβλητή word αποθηκεύει έναν 16bit αριθμό χωρίς πρόσημο από 0 έως 65,535
long	Χρησιμοποιείται για την αποθήκευση αριθμών των 32bits
float	Μια μεταβλητή τύπου float αυτή αποθηκεύει αριθμούς με υποδιαστολή. Έχει 6-7 ψηφία ακρίβειας δεκαδικών.
double	Αποθηκεύει δεκαδικούς αριθμούς. Χρησιμοποιεί 32bits (4 bytes). Διαφέρει με τη float στην ακρίβεια των δεκαδικών.
string - char array	Χρησιμοποιείται για την

	αποθήκευση strings. Τα strings αποθηκεύονται σαν πίνακες (array). Ένας χαρακτήρας τοποθετείται σε μονά εισαγωγικά (‘...’), ενώ πολλοί χαρακτήρες σε διπλά εισαγωγικά (“..”).
String – object	Η κλάση String μας επιτρέπει να διαχειριζόμαστε τα strings με διαφορετικό και πιο εύκολο τρόπο από την char.
array	Πρόκειται για μια συλλογή από μεταβλητές στις οποίες έχουμε πρόσβαση χρησιμοποιώντας απλά ένα δείκτη.
static	Χρησιμοποιείται για να δηλωθούν μεταβλητές οι οποίες θα είναι ορατές μόνο από μια συνάρτηση. Μια static μεταβλητή διατηρεί τα δεδομένα της μεταξύ των κλήσεων της ίδιας συνάρτησης.
volatile	Χρησιμοποιείται για να κατευθύνει τον compiler να φορτώσει τα δεδομένα της συγκεκριμένης μεταβλητής από τη RAM και όχι από τους καταχωρητές που αποθηκεύεται το πρόγραμμα και οι άλλες μεταβλητές. Χρησιμοποιείται κυρίως για να διαχειριζόμαστε σωστά τα interrupts.
const	Χρησιμοποιείται για να δηλώσουμε μια μεταβλητή σαν σταθερά

Πίνακας 2, Μεταβλητές και τύποι δεδομένων

Ψηφιακές συναρτήσεις I/O	**Λειτουργία**
pinMode()	Καθορίζει αν το συγκεκριμένο ψηφιακό *pin* θα είναι pin εισόδου ή pin εξόδου ανάλογα με την τιμή που δίνεται στην παράμετρο *mode* (INPUT ή OUTPUT αντίστοιχα).
digitalWrite()	Θέτει την κατάσταση *pinstatus* (HIGH ή LOW) στο συγκεκριμένο ψηφιακό *pin*.
digitalRead()	Επιστρέφει την κατάσταση του συγκεκριμένου ψηφιακού *pin* (0 για LOW και 1 για HIGH) εφόσον αυτό είναι pin εισόδου.

Πίνακας 3, Ψηφιακές συναρτήσεις εισόδου/εξόδου

Αναλογικές I/O	Λειτουργία
analogReference(type)	Δέχεται τις τιμές DEFAULT, INTERNAL ή EXTERNAL στην παράμετρο *type* για να καθορίσει την τάση αναφοράς (V_{ref}) των αναλογικών εισόδων (5V, 1.1V ή η εξωτερική τάση με την οποία τροφοδοτείται το pin AREF αντίστοιχα)
analogRead()	Επιστρέφει έναν ακέραιο από 0 εώς 1023, ανάλογα με την τάση που τροφοδοτείται το συγκεκριμένο *pin* αναλογικής εισόδου στην κλίμακα 0 ως V_{ref}.
analogWrite()	Θέτει το συγκεκριμένο ψηφιακό *pin* σε κατάσταση ψευδοαναλογικής εξόδου (PWM). Η παράμετρος *value* καθορίζει το πλάτος του παλμού σε σχέση με την περίοδο του παραγόμενου σήματος στην κλίμακα από 0 ως 255

Πίνακας 4, Αναλογικές συναρτήσεις εισόδου/εξόδου

Συναρτήσεις σειριακής επικοινωνίας	Λειτουργία
serial.available()	Λαμβάνει τον αριθμό των bytes (χαρακτήρων) που είναι διαθέσιμοι για διάβασμα στη σειριακή θύρα και τους αποθηκεύει στον προσωρινό καταχωρητή σειριακής θύρας, ο οποίος έχει μέγεθος 64 bytes.
serial.read()	Διαβάζει δεδομένα από τη σειριακή θύρα.
serial.print()	Γράφει δεδομένα στη σειριακή θύρα σαν ASCII χαρακτήρες.
serial.println()	Διοχετεύει τα δεδομένα *data* για αποστολή μέσω του σειριακού interface. Η παράμετρος *data* μπορεί να είναι είτε αριθμός είτε αλφαριθμητικό.
Serial.begin	Θέτει τον ρυθμό μεταφοράς δεδομένων του σειριακού interface (σε baud)

Πίνακας 5, Βασικές συναρτήσεις σειριακής επικοινωνίας

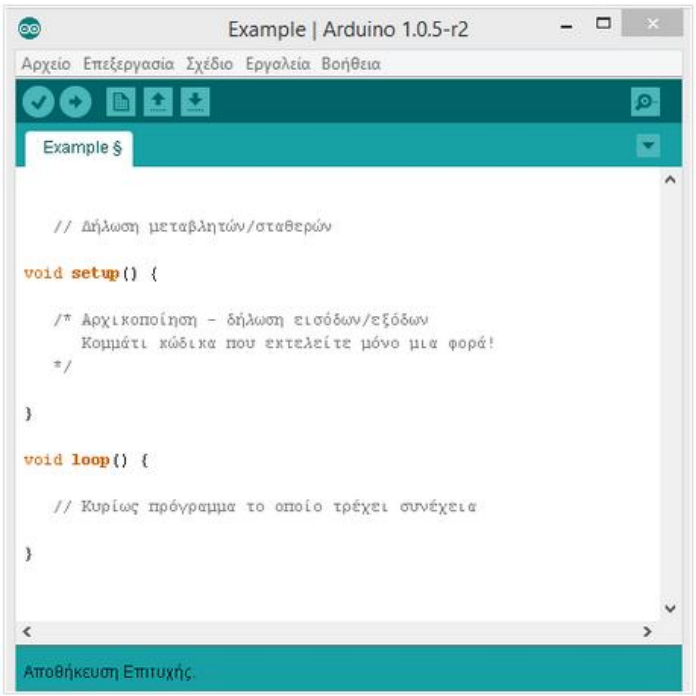
Εντολές Διακοπών	Λειτουργία
attachInterrupt	Θέτει σε λειτουργία το συγκεκριμένο *interrupt*, ώστε να ενεργοποιεί την συνάρτηση *function*, κάθε φορά που ικανοποιείται η συνθήκη που ορίζεται από την παράμετρο *triggermode*: ● LOW (ενεργοποίηση όταν η κατάσταση του pin που αντιστοιχεί στο συγκεκριμένο interrupt γίνει LOW) ● RISING (όταν από LOW γίνει HIGH) ● FALLING (όταν από HIGH γίνει LOW) ● CHANGE (όταν αλλάξει κατάσταση γενικά)
detachInterrupt	Απενεργοποιεί το συγκεκριμένο *interrupt*.
noInterrupts	Σταματά προσωρινά την λειτουργία όλων των interrupt

interrupts	Επαναφέρει την λειτουργία των interrupt που διακόπηκε προσωρινά από μια εντολή noInterrupts.

Πίνακας 6, Εντολές διακοπών

2.2 Λογισμικό IDE

Το Arduino IDE (freeware λογισμικό, Εικόνα 9) αποτελεί ένα ολοκληρωμένο περιβάλλον ανάπτυξης (integrated development environment) και είναι απλούστερο και πιο φιλικό προς το χρήστη, σε αντίθεση με παρόμοια περιβάλλοντα ανάπτυξης όπως το Eclipse , το Xcode και το Visual Studio.
Περιέχει έναν επεξεργαστή πηγαίου κώδικα («editor»), μεταγλωττιστή («compiler»), εργαλεία αυτόματης παραγωγής κώδικα, αποσφαλματωτή («debugging»), συνδέτη, σύστημα ελέγχου εκδόσεων και εργαλεία κατασκευής γραφικών διασυνδέσεων χρήστη για τις υπό ανάπτυξη εφαρμογές.
Μέσα στο μενού υπάρχουν οι επιλογές για την σειριακή θύρα («serial port») με την οποία θα επικοινωνεί το Arduino με τον Η/Υ καθώς και η καρτέλα επιλογής της υπολογιστικής πλατφόρμας Arduino που χρησιμοποιείται κάθε φορά («board»).



Εικόνα 9, Λογισμικό IDE

Με άλλα λόγια συνδέουμε το hardware μέρος του arduino με το λογισμικό IDE για να φορτώσουμε το πρόγραμμα που έχουμε συντάξει και να πραγματοποιηθεί η μεταξύ τους επικοινωνία.
Για την αποκατάσταση της επικοινωνίας αυτής θα πρέπει το Arduino να συνδεθεί σε μια από τις θύρες USB του υπολογιστή και λόγω του ελεγκτή Serial-over-USB, θα πρέπει να αναγνωριστεί από το λειτουργικό σύστημα ως εικονική σειριακή θύρα.

Για την σύνδεση απαιτείται ένα καλώδιο USB από Type A σε Type B, όπως αυτό των εκτυπωτών. Για την αναγνώριση από το λειτουργικό σύστημα είναι απαραίτητη η εγκατάσταση τον οδηγό του FTDI chip (δηλαδή του ελεγκτή Serial-over-USB) ο οποίος υπάρχει στον φάκελο drivers του Arduino IDE. Μετά από αυτό στο κεντρικό παράθυρο του Arduino IDE θα εμφανιστεί, στο μενού Tools → Serial Port , η εικονική σειριακή θύρα (συνήθως COM#) έτοιμη να δεχτεί τους κώδικες.
Ο κώδικας που έχει γραφεί για το Arduino ονομάζεται sketch.
Στον ακόλουθο πίνακα παρουσιάζονται τα εργαλεία του περιβάλλοντος ανάπτυξης, υπό μορφή κουμπιών.

Εργαλείο	Περιγραφή
 Verify	Ελέγχει για συντακτικά λάθη στον κώδικα.
 Upload	Μεταγλωττίζει τον κώδικα και τον φορτώνει στο Arduino. Αν δεν είναι συντακτικά σωστός δεν μπορεί να γίνει η φόρτωση.
 New	Δημιουργεί ένα νέο sketch.
 Open	Παραθέτει ένα menu με όλα τα sketch. Ενεργοποιώντας ένα από αυτά, θα ανοίξει αυτόματα στο τρέχον παράθυρο.
 Save	Αποθηκεύει ένα sketch.
 Serial Monitor	Ανοίγει την σειριακή οθόνη και μέσω αυτής παρακολουθείται η ανταλλαγή δεδομένων που γίνεται στην σειριακή θύρα.

Πίνακας 6, Εργαλεία του περιβάλλοντος IDE

Αξίζει να τονιστεί πως με την ολοκλήρωση του προγράμματος sketch, για να αναγνωριστεί από τον μεταγλωττιστή C++ ως έγκυρο πρόγραμμα, ο χρήστης θα πρέπει να κάνει κλικ στο κουμπί "Upload to I/O board" του IDE και ένα αντίγραφο του κώδικα να γραφτεί σε ένα προσωρινό αρχείο με ένα παραπάνω include στην κορυφή και μία πολύ απλή συνάρτηση main() στο τέλος, ώστε τελικά να φτιαχτεί ένα έγκυρο C++ πρόγραμμα.

Συγκεντρωτικά το Arduino IDE παρέχει:

- Ένα πρακτικό περιβάλλον για την συγγραφή των προγραμμάτων (τα οποία ονομάζονται sketch στην ορολογία του Arduino) .
- Μερικές έτοιμες βιβλιοθήκες για προέκταση της γλώσσας και για να χειριζόμαστε εύκολα μέσα από τον κώδικά μας τα εξαρτήματα που συνδέουμε στο Arduino.

- Τον compiler για την μεταγλώττιση των sketch.
- Ένα serial monitor που παρακολουθεί τις επικοινωνίες της σειριακής (USB), αναλαμβάνει να στείλει αλφαριθμητικά της επιλογής μας στο Arduino μέσω αυτής, και είναι ιδιαίτερα χρήσιμο για το debugging των sketch.
- Την επιλογή να ανεβάσουμε το μεταγλωττισμένο sketch στο Arduino.

3. Ολοκληρωμένα μικροσυστήματα επεξεργασίας – Raspberry Pi

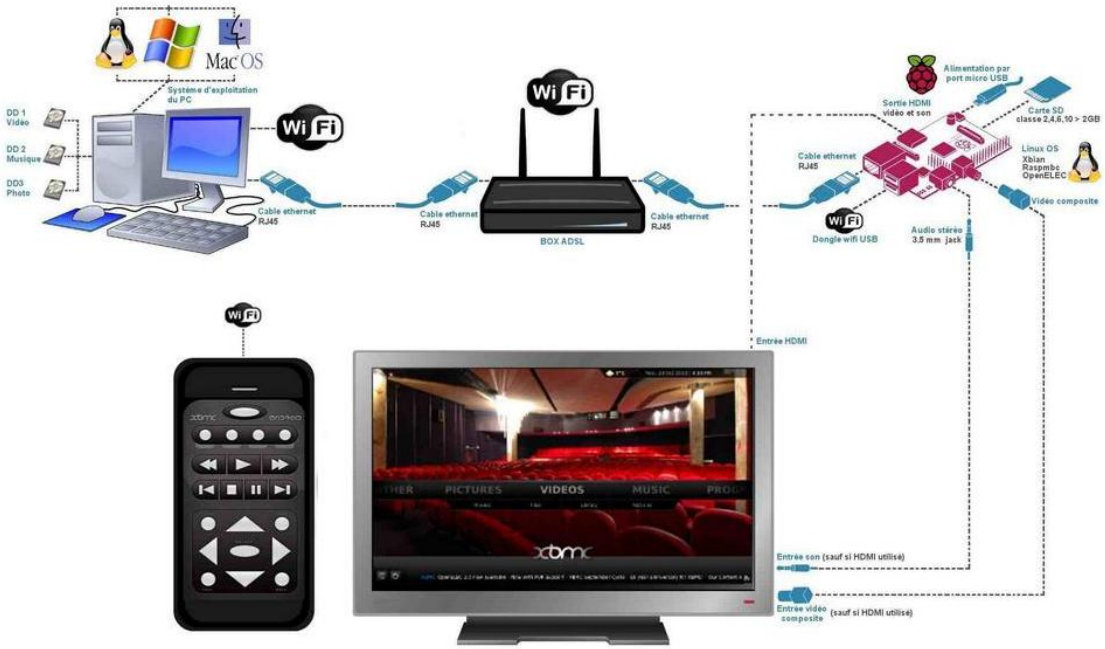
3.1 Το Raspberry Pi

Όσο η τεχνολογία προχωράει, οι συσκευές μικραίνουν. Αυτή είναι μια παραδοχή την οποία μπορεί να κάνει ο καθένας βλέποντας την εξέλιξη του υπολογιστή από το 1946 μέχρι και σήμερα. Μικρότερο μέγεθος σημαίνει μεγαλύτερη ευελιξία στην καθημερινή χρήση καθιστώντας τον υπολογιστή μια πρακτική συσκευή. Αλλά μέχρι που μπορεί να φτάσει η μείωση του μεγέθους;
Το Raspberry pi (Εικόνα 10) είναι ένας φθηνός υπολογιστής. Το μέγεθός του δεν ξεπερνά αυτό της πιστωτικής κάρτας.

Εικόνα 10, Raspberry Pi

Αναπτύχθηκε στο Ηνωμένο Βασίλειο από την εταιρεία “Raspberry Pi Foundation” με σκοπό την προώθηση της διδασκαλίας της επιστήμης των υπολογιστών στα σχολεία.
Η Element είναι η εταιρεία πίσω από την ιδέα του Raspberry Pi Board B, ενός ολόκληρου υπολογιστή σε μια πλακέτα με αρκετά εντυπωσιακά χαρακτηριστικά και δυνατότητες ανάπτυξης εφαρμογών και υπηρεσιών για τον καθένα.

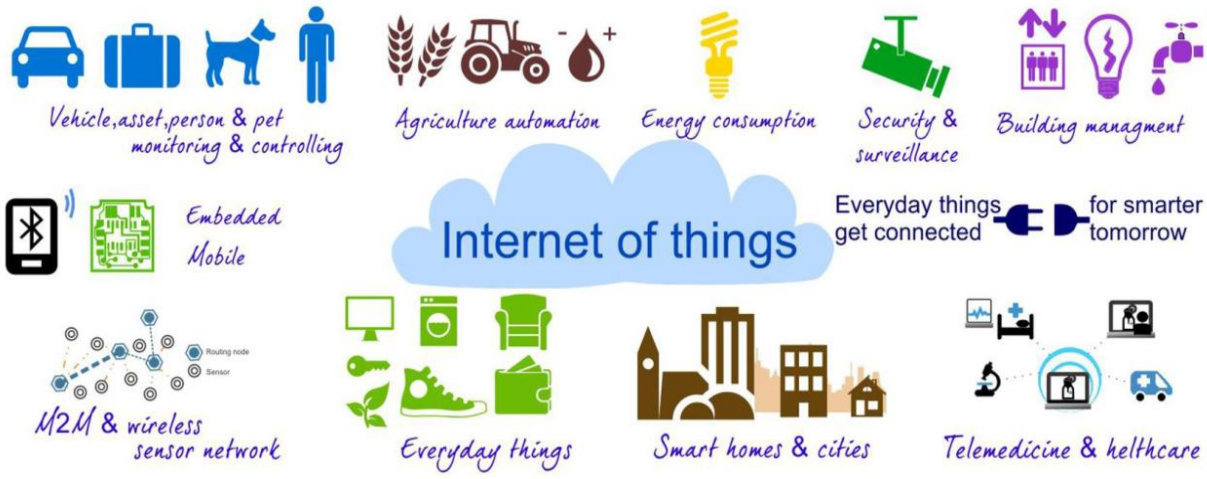
Σκεφτείτε ότι έχετε αγοράσει τα ηλεκτρονικά ενός κινητού χωρίς την μπαταρία ή την οθόνη αλλά έχετε την δυνατότητα να συνδέσετε πάνω την δική σας οθόνη, πληκτρολόγιο, ποντίκι και μνήμη για να τρέξετε Linux (Εικόνα 11). Αυτό ακριβώς προσφέρει το Raspberry Pi.



Εικόνα 11, Σύνδεση Raspberry ως πλατφόρμα πολυμέσων

Από τη μια φθηνό και δυνατό hardware από την άλλη σύγχρονο SDK. Το development πλέον γίνεται παιχνιδάκι και μέσω του Qt (λογισμικό) και το Raspberry Pi μπορεί να κάνει οτιδήποτε φαντάζεται ο προγραμματιστικός νους. Η φαντασία είναι ο μόνος παράγοντας που περιορίζει τον συνδυασμό Raspberry Pi και Qt για ανάπτυξη εφαρμογών, low power concepts και οποιασδήποτε άλλη χρήση.
Η τελευταία έκδοση του Raspberry Pi, το Raspberry Pi 2, είναι ένας προσιτός mini υπολογιστής,που είναι ικανός να τρέχει τόσο λειτουργικό σύστημα Windows 10 (Windows developer), όσο και άλλα λειτουργικά συστήματα βασισμένα στο Linux, όπως και το Snappy Ubuntu Core, το οποίο είναι ένα ελαφρύ Ubuntu Linux που έχει αναπτύξει η Canonical.

Το Raspberry Pi 2 μοιάζει περισσότερο με κινητό ή με tablet ως προς την υπολογιστική του ισχύ, καθώς είναι εφοδιασμένο με έναν τετραπύρηνο επεξεργαστή με πυρήνες ARM Cortex-A7. Το Raspberry Pi 2 θα είναι πιο γρήγορο και ισχυρό από τον προκάτοχό του και επεκτείνεται, προκειμένου να μπορέσει να μπει το νέο λειτουργικό σύστημα στην νέα εποχή του Διαδικτύου των Πραγμάτων (Εικόνα 12).



Εικόνα 12, internet of things

3.2 Τεχνικά χαρακτηριστικά του Raspberry Pi

Το Raspberry pi έρχεται σε δυο εκδόσεις, Model A και Model B. Η διαφορά τους βρίσκεται λίγο πολύ στην δυνατότητα σύνδεσής τους στο Ιντερνετ.
Τα τεχνικά χαρακτηριστικά του Raspberry Pi 2 είναι:

- Broadcom BCM2836 Arm 7 Quad Core Processor powered Single Board Computer, 900MHz
- 1GB RAM
- 40 pin extended GPIO
- 4 θύρες USB 2
- 4 pole Stereo output/Composite video port
- Full size HDMI
- Υποδοχή CSI camera για σύνδεση της Raspberry Pi camera
- Υποδοχή DSI display port για σύνδεση της οθόνης αφής Raspberry Pi
- Υποδοχή Micro SD για την φόρτωση του λειτουργικού συστήματος και την αποθήκευση δεδομένων
- Micro USB power source

3.3 Λογισμικό του Raspberry Pi και έλεγχος συσκευών

Η βασική έκδοση στην SD κάρτα είναι η διανομή του Linux Raspbian που στηρίζεται στη Debian. Μια ενδεικτική έκδοση που περιλαμβάνει προεγκατεστημένα προγράμματα κυρίως για τον έλεγχο των θυρών (ports) και κάποια άλλα όπως τα:
• Scratch
• Python 2.7

• Python IDLE
• Midori Browser
• Libre Office
• Xpdf
• Pidgin
• Synaptic
• VLC Player

Ο έλεγχος των I/O θυρών γίνεται με πολύ απλό κώδικα στην Python μέσω των κατάλληλων βιβλιοθηκών και παράλληλα, επειδή το Scratch (Scratch GPIO) έχει ενταχθεί ενεργά στην εκπαιδευτική διαδικασία, έχουν δημιουργηθεί πολλές βιβλιοθήκες σε Python που επιτρέπουν τη διεπαφή Scratch-Python και επομένως τον εύκολο έλεγχο των I/O του Raspberry pi.

Γενικά, το Raspberry Pi είναι καλό όταν χρειάζεται απεικόνιση ή σύνδεση στο διαδίκτυο. Πλεονεκτήματα:

- Το HDMI του δίνει τη δυνατότητα να συνδεθεί με τηλεόραση και στις δυο USB μπορεί να συνδεθεί πληκτρολόγιο και ποντίκι.
- Μέσω της θύρας Ethernet μπορεί να συνδεθεί στο διαδίκτυο.
- Το λειτουργικό σύστημα τρέχει μέσω SD κάρτας και επομένως μπορούμε να αλλάξουμε λειτουργικά, αλλάζοντας απλά την κάρτα.
- Για την τιμή του, είναι αρκετά δυνατό, αλλά και εύκολο στην χρήση του ακόμα και από αρχάριους χρήστες.

Μειονέκτημα:

- Δεν έχει τόσες δυνατότητες στη σύνδεση με εξωτερικούς αισθητήρες ή διακόπτες (όπως το Arduino ή το Beaglebone), άρα δεν είναι αρκετά καλή επιλογή για τον έλεγχο άλλων συσκευών.

4. Περιγραφή του hardware που χρησιμοποιείται για την υλοποίηση του συστήματος.

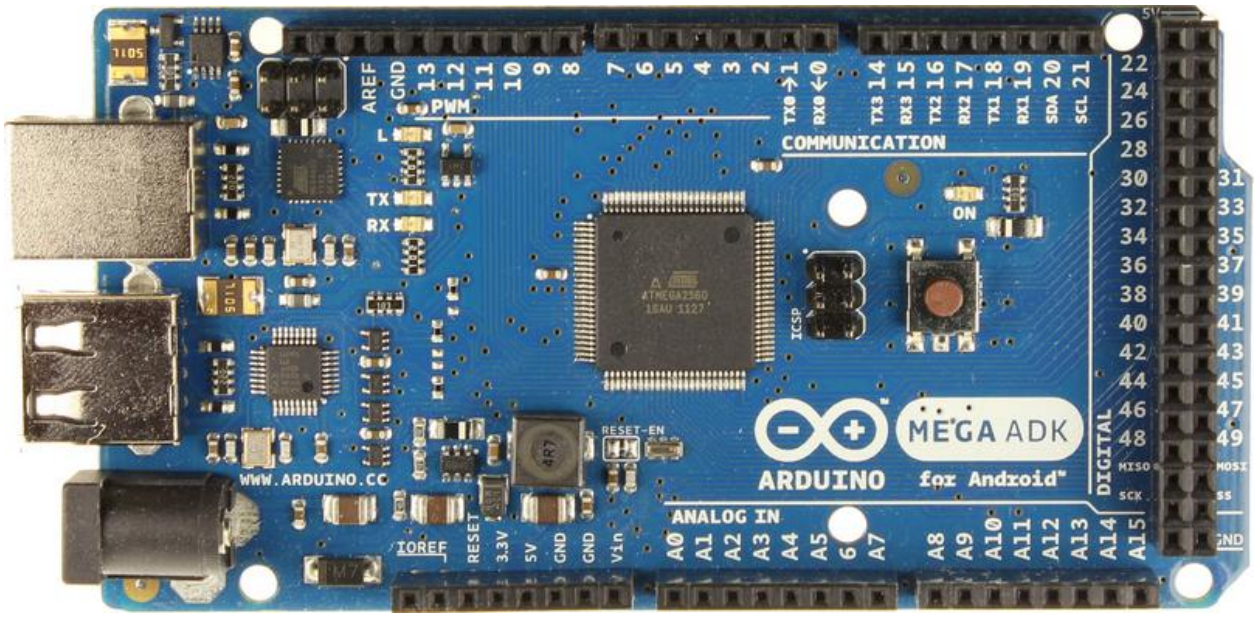
4.1. Arduino Mega ADK Rev 3

Το Arduino Mega ADK Rev3 είναι η πιο ισχυρή και πιο πρόσφατη υπολογιστική πλατφόρμα Arduino με πάρα πολλές δυνατότητες, και είναι η αναβάθμιση του Arduino Mega. Χρησιμοποιεί τον Atmega2560, μικροελεγκτή της ATMEL. Έχει 54 ψηφιακές θύρες εισόδου και εξόδου (I/O) εκ των οποίων οι 15 μπορούν να παράγουν «8-bit» P.W.M και άλλες 2 θύρες μπορούν να χρησιμοποιηθούν για «I2C» επικοινωνία καθώς και 6 εξωτερικά interrupt, 16 αναλογικές θύρες, 16MHz ταλαντωτή κρυστάλλου, κουμπί «reset», 4 σειριακές θύρες για «hardware» κεφαλή, και 4 θύρες «SPI» για την σύνδεση με άλλα περιφερειακά ή πλακέτες.

Στην τελευταία έκδοση του Arduino Mega Rev 3 έγιναν οι παρακάτω τροποποιήσεις:

- Αντικαταστάθηκε ο ATmega8U2 με τον ATmega16U2 για το USB-to-Serial Converter.
- Προστέθηκαν δύο επιπλέον pins, τα SDA και SCL, για TWI επικοινωνία και τοποθέτηση δίπλα στο AREF pin.
- Προστέθηκε το IOREF, το οποίο επιτρέπει στα onboard shields να προσαρμόζονται στην τάση που παρέχει το βασικό board.
- Έγινε βελτίωση του κυκλώματος στο κουμπί του RESET.

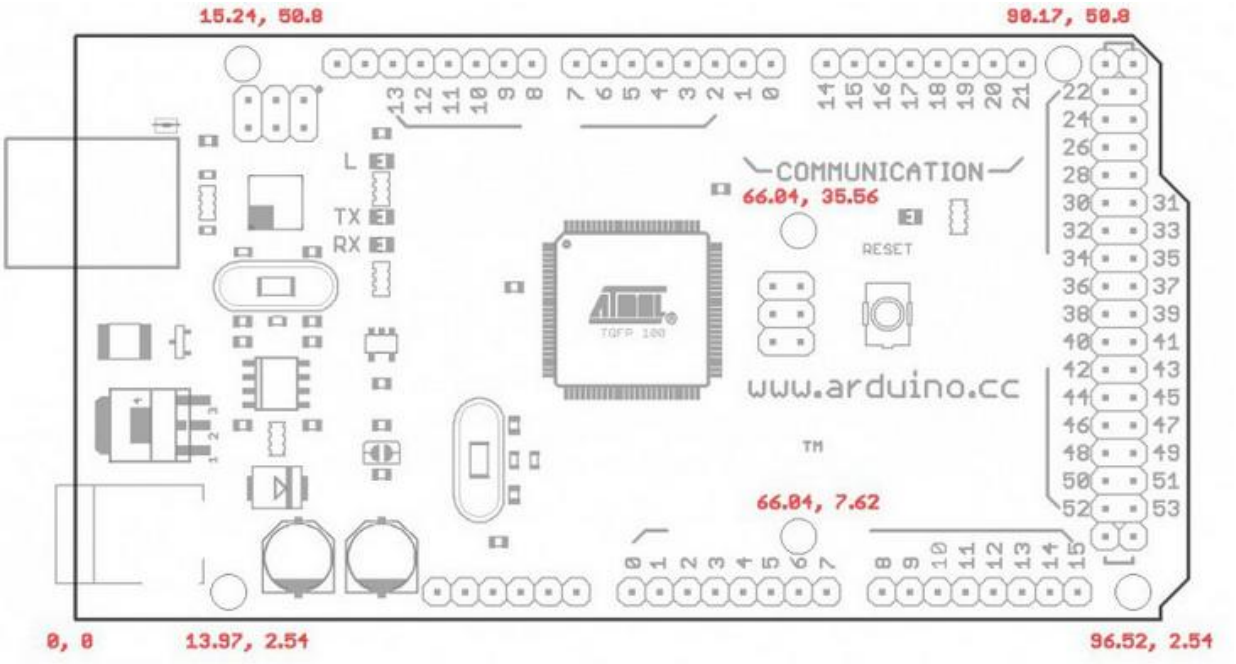
Στις εικόνες 13α & 13β απεικονίζονται τα Schematics του Arduino Mega ADK rev3.



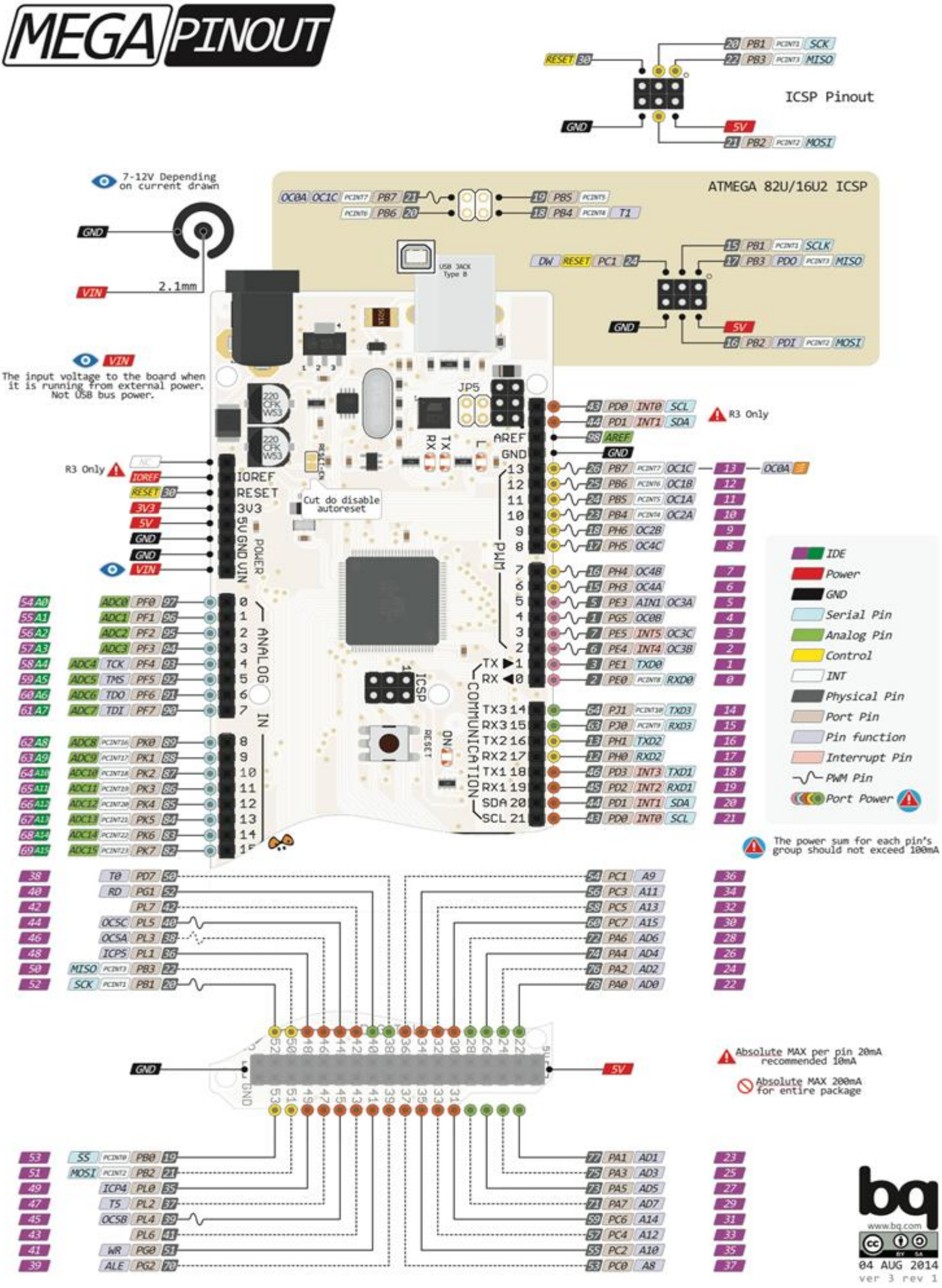
Εικόνα 13α**:** Front Schematic of Arduino Mega ADK rev3

Εικόνα 13β: Back Schematic of Arduino Mega ADK rev3

Έπειτα στα σχήματα 2α και 2β απεικονίζεται το Reference Design του Arduino Mega rev3.



Σχήμα 2α: Reference Design του Arduino Mega rev3.

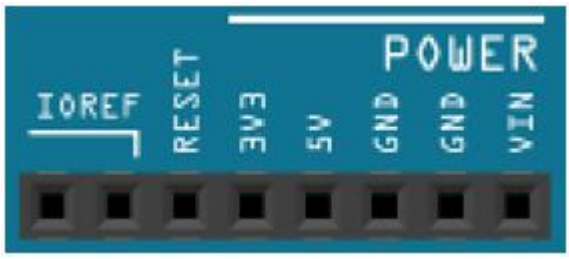


Σχήμα 2β: Reference Design του Arduino Mega rev3.

4.1.1. Ακροδέκτες POWER του Arduino Mega ADK rev3

Το Arduino Mega ADK Rev 3 μπορεί να τροφοδοτηθεί μέσω της θύρας USB του υπολογιστή, ή εναλλακτικά χρησιμοποιώντας εξωτερική τροφοδοσία μέσω μιας υποδοχής φις των 2.1mm, που βρίσκεται στην κάτω-αριστερή γωνία της πλακέτας του Arduino.
Στην περίπτωση που θα χρησιμοποιηθεί μπαταρία για την τροφοδοσία, τότε αυτή συνδέεται στα pin headers Vin και GND του POWER. Αν το Arduino Mega χρησιμοποιηθεί ως USB Host για σύνδεση κινητών τηλεφώνων android, τότε απαιτείται εξωτερικό τροφοδοτικό κατ' ελάχιστον 1.5A .
Χρειάζεται επίσης να σημειωθεί ότι η τιμή της εξωτερικής τροφοδοσίας του Arduino Mega ADK Rev 3 πρέπει να κυμαίνεται από 7 ως 12V, για να μην δημιουργηθούν προβλήματα, όπως αστάθεια ή υπερθέρμανση του board.

Στην Εικόνα 15 παρουσιάζονται οι είσοδοι / έξοδοι των ακροδεκτών POWER του Arduino Mega.



Εικόνα 15: Power pins of Arduino Mega ADK rev3

Οι ακροδέκτες POWER είναι οι ακόλουθοι:

- VIN Pin: Το Vin pin έχει διπλό ρόλο. Σε συνδυασμό με το pin γείωσης δίπλα του, μπορεί να λειτουργήσει ως μέθοδος εξωτερικής τροφοδοσίας του Arduino ή μπορεί να χρησιμοποιηθεί για να τροφοδοτήσει εξαρτήματα και συσκευές με την πλήρη τάση της εξωτερικής τροφοδοσίας (7~12V), πριν αυτή περάσει από τον ρυθμιστή τάσης όπως γίνεται με το pin των 5V.

- GND Pin: Είσοδοι γείωσης.

- 5V Pin: Η τάση του Pin αυτού μπορεί να προέρχεται από τη θύρα USB (5V), από την εξωτερική τροφοδοσία του φις των 2.1mm (7-12V) ή από το Vin pin του board (7-12V), αφού περάσει από ένα ρυθμιστή τάσης για να προσαρμοστεί στα 5V.
    - Είναι απαραίτητο να επισημάνουμε ότι αν το board τροφοδοτηθεί απευθείας μέσω των
    - pin 5V ή 3.3V pins γίνεται προσπέραση του regulator και μπορεί να καταστραφεί.

- 3.3V Pin: Η τάση αυτή δεν προέρχεται από την εξωτερική τροφοδοσία αλλά παράγεται από το ολοκληρωμένο FTDI μέσω του on-board regulator, με

μέγιστο ρεύμα που μπορεί να αντληθεί τα 50mA για την τροφοδοσία διατάξεων, συσκευών ή αισθητηρίων με τάση 3.3V.

- Reset Pin: όταν γειωθεί (με οποιοδήποτε από τα 3 pin με την ένδειξη «GND» που υπάρχουν στο Arduino) έχει ως αποτέλεσμα την επανεκκίνηση της υπολογιστικής πλατφόρμας Arduino.

- IOREF Pin: Το pin αυτό παρέχει στο board το κατάλληλο voltage reference για τη λειτουργία του microcontroller.

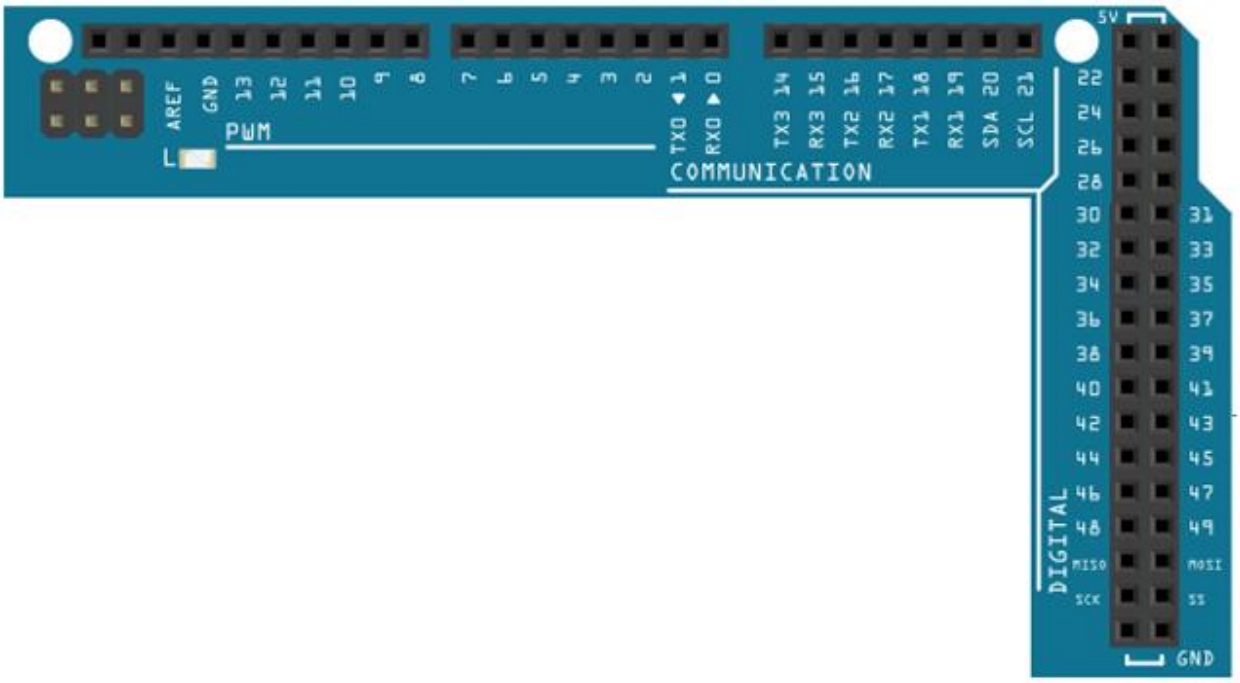
4.1.2. Ακροδέκτες ANALOG IN του Arduino Mega ADK rev3

Δίπλα από τους ακροδέκτες Power του Arduino, υπάρχουν οι ακροδέκτες A0 έως A15 με τη σήμανση ANALOG IN, όπως φαίνεται στην Εικόνα 16. Η τάση αναφοράς (Reference Voltage) για τις αναλογικές εισόδους μπορεί να ρυθμιστεί με τη χρήση εντολών, αφού πρώτα τροφοδοτήσουμε εξωτερικά με τάση το pin AREF, που βρίσκεται στην απέναντι πλευρά της πλακέτας.

Εικόνα 16: Analog In pins of Arduino Mega ADK rev3

4.1.3.Ακροδέκτες DIGITAL IN/OUT του Arduino Mega ADK rev3

Στην πάνω και στη δεξιά πλευρά του Arduino υπάρχουν 50 digital pin (θηλυκά), αριθμημένα από 0-49. Τα Pin αυτά μπορούν να λειτουργήσουν είτε ως ψηφιακές είσοδοι είτε ως ψηφιακοί έξοδοι, χρησιμοποιώντας τις εντολές pinMode(), digitalWrite() και digitalRead(). Η τάση λειτουργίας τους είναι 5V με μέγιστο ρεύμα που μπορούν να παρέχουν ή να δεχθούν τα 40mA.
Το κάθε ψηφιακό pin μπορεί να βρεθεί σε δύο λογικές καταστάσεις εξόδου (HIGH ή LOW) για να ορίσουμε μία κατάσταση της εξωτερικής συσκευής ή σε δύο λογικές καταστάσεις εισόδου, για να διαβάσουμε την κατάσταση της εξωτερικής συσκευής, χρησιμοποιώντας κάθε φορά τις κατάλληλες εντολές προγραμματισμού .Στην Εικόνα 17 παρουσιάζονται τα ψηφιακά pins των ακροδεκτών DIGITAL IN/OUT του Arduino Mega ADK rev3.



Εικόνα 17: Digital In/Out pins of Arduino Mega ADK rev3

Μερικά από αυτά τα 50 pin (0-49), εκτός από ψηφιακές είσοδοι/έξοδοι έχουν την δυνατότητα να χρησιμοποιηθούν και για άλλες λειτουργίες όπως:

- Τα digital pin 0 (Rx) και 1 (Tx) χρησιμοποιούνται επίσης και για να λαμβάνουν (RX) και να μεταδίδουν (TX) TTL σειριακά δεδομένα από και προς τον μικροεπεξεργαστή ATmega16U2 με την βοήθεια του USB-to-TTL Serial chip. Κατά συνέπεια όταν το πρόγραμμα στέλνει δεδομένα σειριακά, τότε αυτά προωθούνται στην θύρα USB μέσω του ελεγκτή «Serial-Over-Usb» όπως επίσης και στο pin 0 για να τα διαβάσει ενδεχομένως μία άλλη συσκευή (π. χ. ένα δεύτερο Arduino στο δικό του pin1). Αυτό πρακτικά σημαίνει ότι αν στο πρόγραμμα ενεργοποιηθεί το σειριακό «interface», καταλαμβάνονται δύο ψηφιακές θύρες εισόδου/εξόδου με άλλα λόγια χάνουμε 2 ψηφιακές εισόδους/εξόδους. Αξίζει να τονίσουμε πως TTL σειριακά δεδομένα έχουν τη δυνατότητα να στέλνουν ή να λαμβάνουν τα digital pin 19 (Rx) και 18 (Tx) ως Serial 1, τα digital pin 17 (Rx) και 16 (Tx) ως Serial 2, τα digital pin 15 (Rx) και 14 (Tx) ως Serial 3.

- Τα digital pin 2 και 3 μπορούν να ρυθμιστούν και ως εξωτερικά interrupt (interrupt 0 και 1 αντίστοιχα). Με άλλα λόγια, μπορούν να ρυθμιστούν μέσα από το πρόγραμμα ώστε να λειτουργούν αποκλειστικά ως ψηφιακές είσοδοι στις οποίες, όταν συμβαίνουν συγκεκριμένες αλλαγές τάσης, η κανονική ροή του προγράμματος θα σταματάει άμεσα και θα εκτελείται μία συγκεκριμένη συνάρτηση. Τα εξωτερικά interrupt είναι ιδιαίτερα χρήσιμα σε εφαρμογές που απαιτούν συγχρονισμό μεγάλης ακρίβειας και μπορεί να ρυθμιστούν για να προκαλέσουν την διακοπή μιας χαμηλής τιμής, μια άνοδο ή την πτώση ακμής. Την ίδια χρήση έχουν και τα digital pin 18 (interrupt 5), digital pin 19 (interrupt 4), digital pin 20 (interrupt 3) και digital pin 21 (interrupt 2).

- Τα digital pin 2 έως 13 και 44 έως 46 μπορούν να λειτουργήσουν και ως ψευδο-αναλογικές έξοδοι με το σύστημα PWM (Pulse Width Modulation). Έτσι έχουμε 8bit PWM output μέσω της analogWrite() function και μπορούμε αντί να έχουμε απλά την δυνατότητα HIGH-LOW, που παρέχουν οι υπόλοιπές ψηφιακές έξοδοι, να έχουμε ενδιάμεσες τιμές, μεταξύ 0 και 5 volt. Αυτό επιτυγχάνεται μεσω του PWM, το οποίο παρέχει ένα παλμό που εναλλάσσεται με κάποια συχνότητα και για ίσους χρόνους μεταξύ των τιμών 0 και 5V.

- Τα digital pin 50 (MISO), 51 (MOSI), 52 (SCK), 53 (SS) παρέχουν SPI communication, με τη χρήση της SPI library.

- Στο digital pin 13 υπάρχει ένα ενσωματωμένο LED, το οποίο μας δείχνει την κατάσταση του pin, δηλαδή όταν αυτό βρίσκεται σε τιμή HIGH το ενσωματωμένο LED είναι αναμμένο, ενώ όταν το pin είναι σε LOW το LED είναι σβηστό.

- AREF pin: Το Arduino Mega έχει 16 αναλογικές εισόδους A0 έως A15 όπου είναι προεπιλεγμένες να μετράνε από 0 έως 5 Volts. Μέσω του pin αυτού, έχουμε τη δυνατότητα να αλλάζουμε το Reference Voltage των αναλογικών εισόδων, μέσω της εντολής analogReference().

- I2C: Τα digital pin 20 (SDA) και 21 (SCL) έχουν προστεθεί για υποστήριξη TWI (Two-Wire Interface) communication χρησιμοποιώντας την Wire library. Στο Arduino Mega ADK rev3 board η βιβλιοθήκη αυτή επιτρέπει την επικοινωνία με I2C / TWI devices μέσω των pins SDA (data line), SCL (clock line) και GND.

- USB Host MAX3421E επικοινωνεί με το Arduino μέσω του SPI bus χρησιμοποιώντας τα digital pins 7 (RST), 50 (MISO), 51 (MOSI), 52 (SCK). Επειδή το digital pin 7 χρησιμοποιείται για την επικοινωνία με το MAX3421E, δεν πρέπει να προγραμματιστεί ως είσοδος ή έξοδος.

4.1.4. Τεχνικά χαρακτηριστικά του Arduino Mega ADK Rev 3

Όπως σε όλες τις υπολογιστικές πλατφόρμες, έτσι και σε αυτή του Arduino Mega rev3 χρησιμοποιούνται διάφορες μνήμες για την προσπέλαση των δεδομένων. Ο ATmega2560 που χρησιμοποιείται στον arduino Mega ADK Rev 3 διαθέτει ενσωματωμένη μνήμη τριών τύπων. Τη flash memory στην οποία αποθηκεύονται τα Arduino sketch, την SRAM (static random access memory) στην οποία δημιουργείται το sketch και χρησιμοποιεί τις μεταβλητές όταν τρέχει, και την EPPROM η οποία χρησιμοποιείται από τους προγραμματιστές για την αποθήκευση μακροχρόνιων πληροφοριών.

Πιο συγκεκριμένα:

- 256KB μνήμης Flash: 8KB χρησιμοποιούνται από το firmware των πλατφόρμων που έχει εγκαταστήσει ήδη ο κατασκευαστής Το firmware αυτό που στην ορολογία του Arduino ονομάζεται bootloader είναι αναγκαίο για την εγκατάσταση των προγραμμάτων στον μικροελεγκτή μέσω της θύρας USB, χωρίς δηλαδή να χρειάζεται εξωτερικός hardware programmer. Τα υπόλοιπα 248KB της μνήμης Flash χρησιμοποιούνται για την αποθήκευση αυτών ακριβώς των προγραμμάτων, αφού πρώτα μεταγλωττιστούν στον υπολογιστή. Η μνήμη Flash, δε χάνει τα περιεχόμενά της με απώλεια τροφοδοσίας ή επανεκκίνησης.

- 8KB μνήμης SRAM: Η SRAM(«static random access memory») είναι η ωφέλιμη μνήμη που μπορούν να χρησιμοποιήσουν τα προγράμματα για να αποθηκεύουν μεταβλητές, πίνακες κ.λ.π. κατά το «runtime». Όπως και σε έναν υπολογιστή, αυτή η μνήμη χάνει τα δεδομένα της όταν η παροχή ρεύματος σταματήσει ή αν γίνει reset. Κατά την διάρκεια μίας κανονικής λειτουργίας όλες οι μεταβλητές φορτώνονται σε αυτή καθ' όλη την διάρκεια της λειτουργίας του μικροελεγκτή.

- 4KB μνήμης EEPROM: Μπορεί να χρησιμοποιηθεί για εγγραφή ή ανάγνωση δεδομένων από τα προγράμματα. Σε αντίθεση με την SRAM, δε χάνει τα περιεχόμενά της με απώλεια τροφοδοσίας ή επανεκκίνησης και απαιτείται ειδική βιβλιοθήκη ώστε να μπορέσει κάποιος να έχει πρόσβαση σε αυτή.

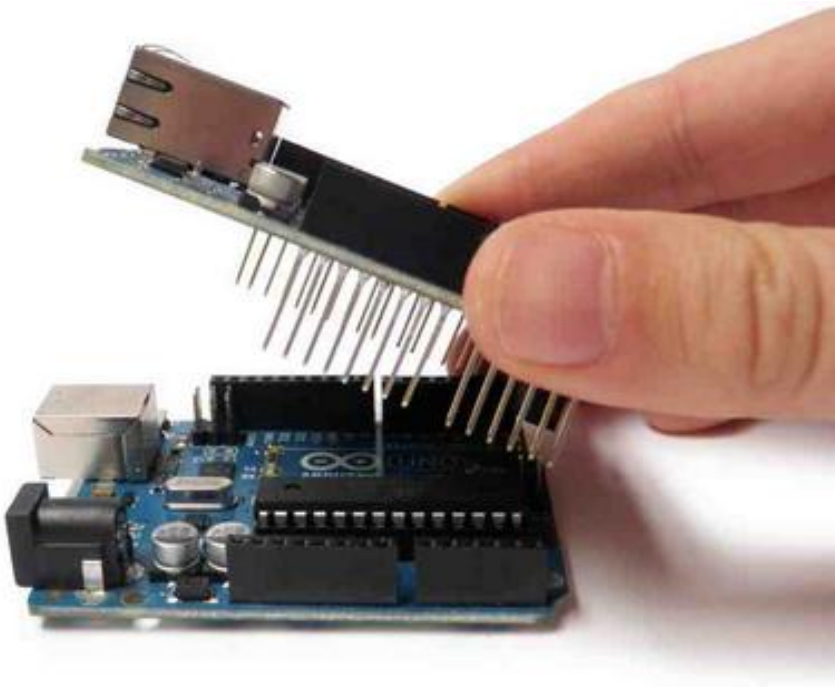
Ο πίνακας 7 παρουσιάζει αναλυτικότερα τα τεχνικά χαρακτηριστικά του Arduino Mega ADK rev3.

Τεχνικά χαρακτηριστικά Arduino Mega ADK Rev3	
Microcontroller:	ATmega2560
Operating Voltage:	5V
Input Voltage (recommended):	9V
Input Voltage (limits):	7-18V
Digital I/O Pins:	54 (14 provide PWM output)
Analog Input Pins:	16
DC Current per I/O Pin:	40 mA
DC Current for 3.3V Pin:	50 mA
Flash Memory:	256 KB (8kb used by bootloader)
SRAM:	8 KB
EEPROM:	4 KB
Clock Speed:	16 MHz

Πίνακας 7, Τεχνικά χαρακτηριστικά Arduino Mega ADK rev3.

4.2. Arduino Ethernet Shield rev 3

Τα shields στην αρχιτεκτονική Arduino ουσιαστικά αποτελούν ενσωμάτωση επιπλέον υλικού (hardware) στον μικροελεγκτή, με σκοπό την προσθήκη επιπλέον ιδιοτήτων, κυρίως όσον αφορά θέματα επικοινωνίας. Με την προσαρμογή του υλικού αυτού η επικοινωνία από σειριακή (μέσω usb) μετατρέπεται σε αυτή που παρέχει το νέο shield. Το Arduino Ethernet Shield (Εικόνα 19) συνδέει το Arduino στο internet με ευκολία, ενώ πάνω από το Ethernet Shield μπορεί να εφαρμόσει, αντίστοιχα, ένα άλλο shield.



Εικόνα 19, Arduino Ethernet Shield

Τα περισσότερα Ethernet shield που υπάρχουν για Arduino είναι βασισμένα στο W5100 της WIZnet και το ENC28J60 της Microchip.
Το Ethernet chip παρέχει ένα ip stack κατάλληλο τόσο για την μεταφορά TCP, όσο και για την μεταφρά UDP πακέτων, ενώ παρέχει την δυνατότητα έως τεσσάρων συνδέσεων socket ταυτόχρονα, χρησιμοποιώντας την Ethernet library (Ethershield-full tcp/ip stack, www server, dhcp)η οποία και χρησιμοποιείται για τη συγγραφή sketch στο Arduino. Το Ethernet Shield περιλαμβάνει ένα clip RJ45 με ενσωματωμένο line transformer και δυνατότητα για power over Ethernet τροφοδοσία. Στην περίπτωση που το Ethernet Shield υποστηρίζει PoE (Power over Ethernet), δεν χρειάζεται να συνδέσουμε ούτε το arduino ούτε το Ethernet Shield για τροφοδοσία, καθώς τροφοδοτούνται και τα δύο μέσω του καλωδίου Ethernet κατηγορίας 5 (CAT 5-RJ45), που χρησιμοποιείται για τη δικτύωσή τους.
Επάνω στην πλακέτα του Ethernet shield υπάρχει μια onboard micro-SD card slot, η οποία μπορεί να χρησιμοποιηθεί για την αποθήκευση αρχείων με σκοπό το διαμοιρασμό μέσω δικτύου (SD βιβλιοθήκη). Πάνω στο shield υπάρχει ένα reset

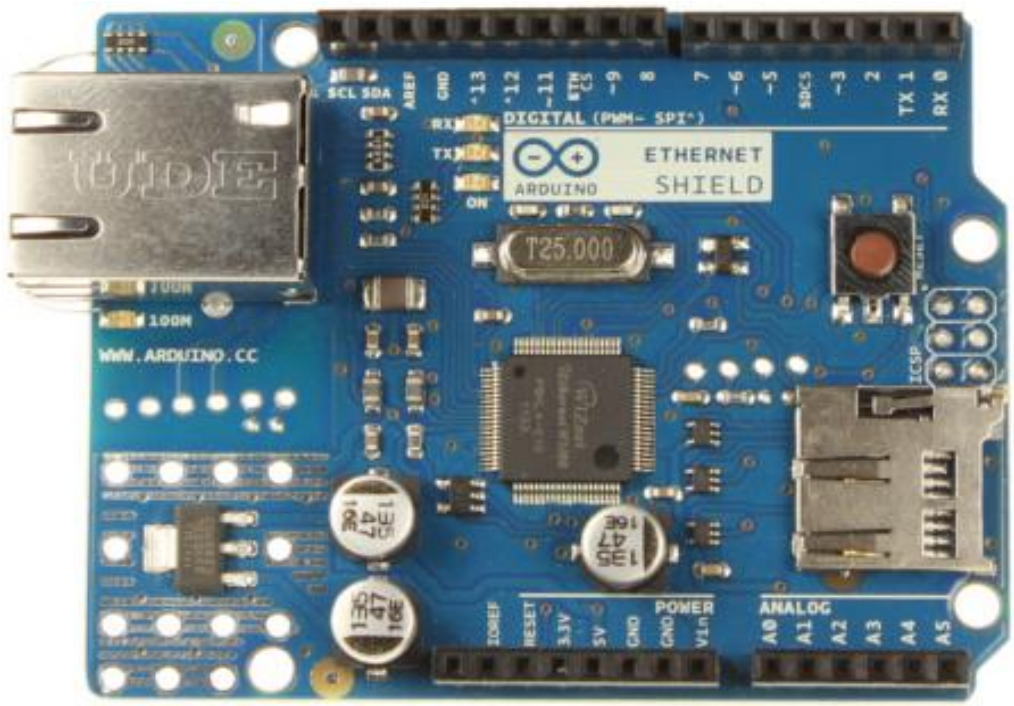
controller για να επιβεβαιώσει ότι το W5100 Ethernet module έχει γίνει κατάλληλα reset στο power-up.

Για τη σύνδεση με το Arduino χρειάζονται τα εξής pin:

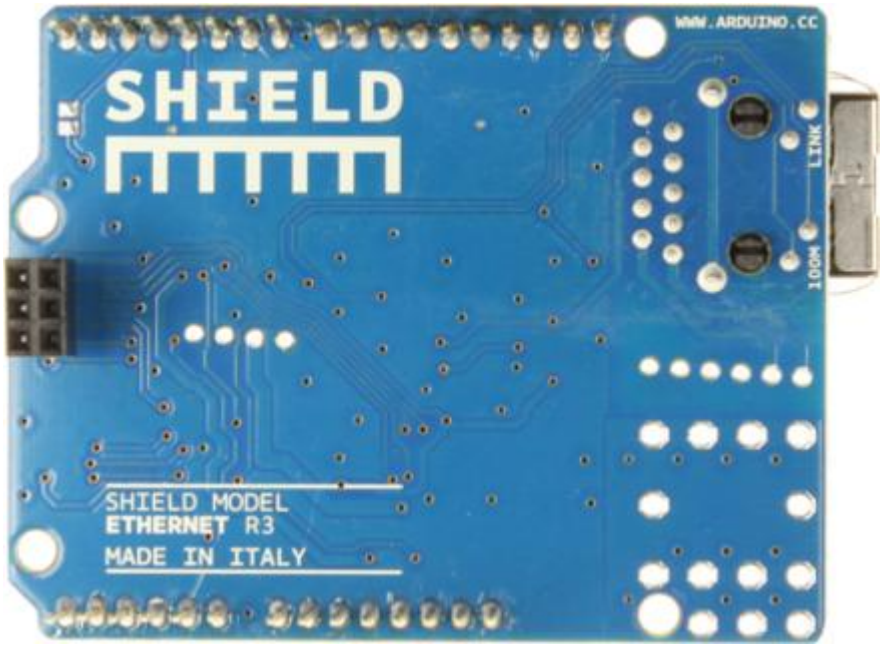
- GND
- VCC
- SO (Serial data out)
- SI (Serial data in)
- SCLK (Serial data clock)
- CS (Chip select)
- INT (Interrupt)

Το Arduino Mega Rev 3 επικοινωνεί με το W5100 και την SD card χρησιμοποιώντας το SPI bus μέσω του ICSP header στα pins 50, 51, 52. Επιπλέον τα pin 4 και 10 χρησιμοποιούνται για την επιλογή του W5100 / SD card αντίστοιχα και δεν μπορούν να χρησιμοποιηθούν για διαφορετική I/O χρήση. Είναι απαραίτητο να επισημανθεί πως το pin 53 του arduino πρέπει πάντα να δηλώνεται ως OUTPUT γιατί σε διαφορετική περίπτωση δεν θα υπάρξει επικοινωνία μέσω του SPI bus.

.



Εικόνα 20α, Arduino Ethernet Shield REV3 Front



Εικόνα 20β, Arduino Ethernet Shield REV3 Back

Το shield περιέχει μια σειρά από Led για την πληροφόρηση της λειτουργίας του όπως παρουσιάζεται στον παρακάτω πίνακα (πίνακας 8).

Περιγραφή λειτουργίας φωτεινών ενδείξεων του Ethernet Shield	
Ονομασία:	Περιγραφή Λειτουργίας:
PWR	Μας πληροφορεί για την ύπαρξη τροφοδοσίας στο board και στο shield
LINK	Επιβεβαιώνει την παρουσία δικτύωσης και αναβοσβήνει κατά την αποστολή ή λήψη δεδομένων
FULLD	Επιβεβαιώνει την υποστήριξη αμφίδρομης επικοινωνίας εντός του δικτύου (Full Duplex)
1OOM	Επιβεβαιώνει την ύπαρξη σύνδεσης 100 Mb/s από το δίκτυο προς το shield και 10Mb/s από το shield προς το δίκτυο
RX	Αναβοσβήνει κατά τη λήψη δεδομένων.
TX	Αναβοσβήνει κατά την αποστολή δεδομένων.
COLL	Αναβοσβήνει κατά τη διαπίστωση σύγκρουσης (collision) δεδομένων.

Πίνακας 8, Περιγραφή λειτουργίας φωτεινών ενδείξεων Ethernet Shield

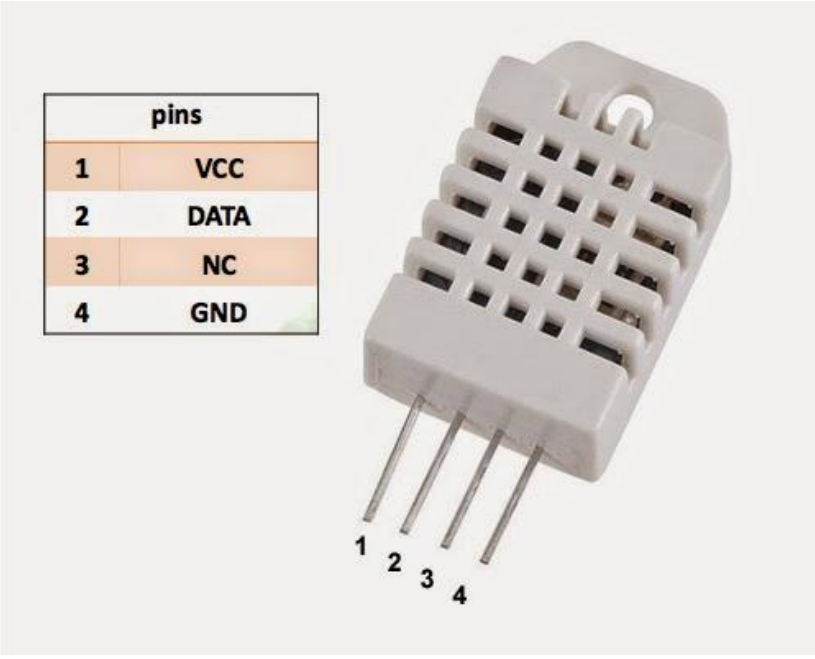
4.3.Αισθητήρας Θερμοκρασίας / Υγρασίας RHT-03

Ο αισθητήρας RHT-03 είναι ένας φθηνός και αξιόπιστος αισθητήρας. Έχει μικρό μέγεθος,μικρή κατανάλωση ρεύματος και μπορεί να 'διαβάσει' την θερμοκρασία και την υγρασία του χώρου σε απόσταση 20m. Η θερμοκρασία που μπορεί να μετρήσει είναι απο -40°C έως 80°C με ακρίβεια ενός δεκαδικού ψηφίου και απόκλιση ±0.5°C. Την υγρασία τη μετράμε σε ποσοστό απ' το 0% έως 100%, ενώ η απόκλισή της είναι συνήθως ±2% και μεταδίδει δεδομένα στην πλακέτα κάθε 2 sec. Λόγω αυτών των ιδιαιτεροτήτων του έχει χρησιμοποιηθεί σε πολλές εφαρμογές και υπάρχει πληθώρα βιβλιοθηκών στο διαδίκτυο. Τα τεχνικά χαρακτηριστικά του είναι:

	Υγρασία	Θερμοκρασία
Τάση	3.3-6V	
Έξοδος	Ψηφιακή	
Αισθητήρας	Πολυμερής Πυκνωτής	DS18B20
Εύρος	0-100% Rh	-40 - 80 C°
Ακρίβεια	2 %	±0,5 C°
Δειγματοληψία	0.5 Hz	

Πίνακας 9, Τεχνικά Χαρακτηριστικά RHT-03

Τα pins που διαθέτει είναι 4, και περιγράφονται στην Εικόνα 21



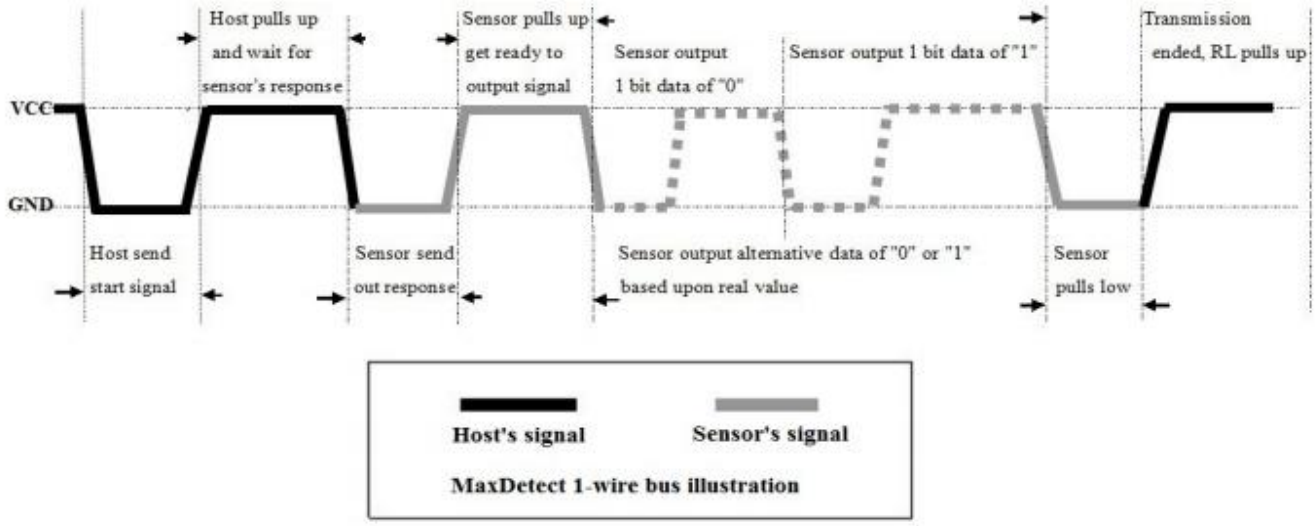
Εικόνα 21, τα pins του RHT-03 και η λειτουργία τους

Σύμφωνα με τον World Meteorological Organization (WMO), η ευαισθησία που απαιτείται για έναν αξιόπιστο αισθητήρα θερμοκρασίας καταγραφής της περιβαλλοντολογικής θερμοκρασίας πρέπει να κυμαίνεται από -30 C° έως 50 C°, ενώ

για την υγρασία από 5% έως 100%, επομένως η χρήση του RHT-03 αισθητήρα είναι η ιδανική, αν αναλογιστούμε το κόστος του.
Η λειτουργία του αισθητήρα, ως αισθητήρα θερμοκρασίας και υγρασίας, με τη χρήση Arduino, περιγράφεται ως εξής: Αρχικά η ψηφιακή έξοδος του Arduino στέλνει ένα σήμα έναρξης της τάξης των 500μs περίπου και αναμένει απάντηση από τον αισθητήρα. Στη συνέχεια, ο αισθητήρας μειώνει το σήμα στα 80μs και στέλνει εκ νέου ένα παρόμοιο σήμα. Συγχρόνως, στέλνει και ένα σήμα της τάξης των 50μs (διαδικασία χειραψίας). Έπειτα ακολουθούν σήματα κάθε 28μs για λογικό 0, και κάθε 70μs για λογικό 1. Κάθε δύο δευτερόλεπτα, ο αισθητήρας εκπέμπει 40 bit πληροφορίας τα οποία αντιστοιχούν σε:

- 8 bit για την τιμή της υγρασίας
- 8 bit για την ακέραια τιμή τις υγρασίας
- 8 bit για την τιμή της θερμοκρασίας
- 8 bit για την ακέραια τιμή τις θερμοκρασίας
- 8 bit για το bit ελέγχου ισοτιμίας των δεδομένων.

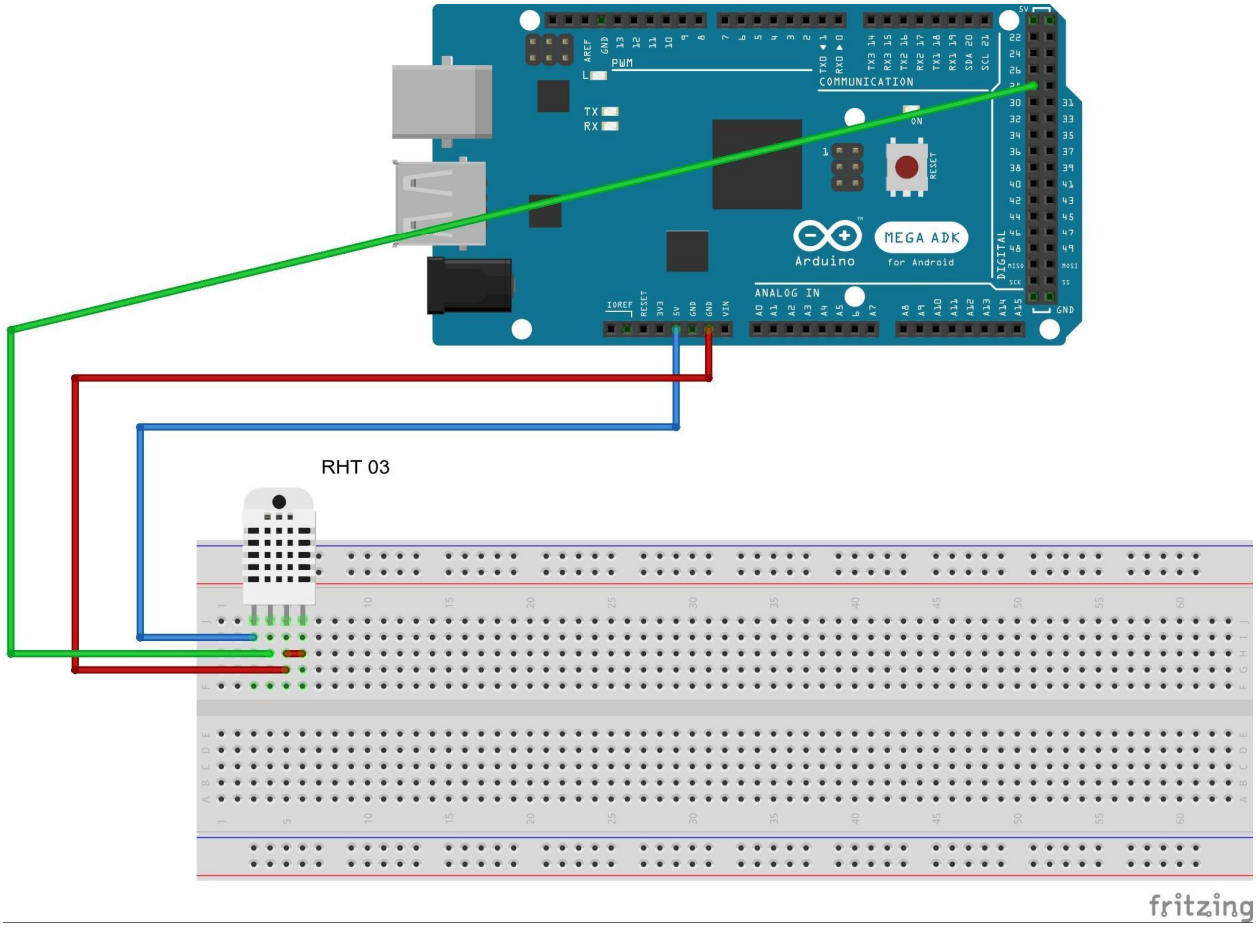
Η παραπάνω διαδικασία περιγράφεται συνοπτικά στο Σχήμα 3, που ακολουθεί.



Σχήμα 3, Λειτουργία λήψης και αποστολής θερμοκρασίας και υγρασίας

4.3.1. Συνδεσμολογία με την πλακέτα Arduino.

Η συνδεσμολογία μεταξύ του αισθητήρα και της πλακέτας Arduino είναι αρκετά απλή, όπως φαίνεται στην Εικόνα 22. Στον ακροδέκτη #1 του αισθητήρα συνδέεται η γείωση (GND), στον ακροδέκτη # 3-4 η τάση (+5v ή 3.3v) και στον ακροδέκτη #2 μία από τις ψηφιακές εισόδους του Arduino (επιλέξαμε την θύρα #28).



Εικόνα 22, Σύνδεση αισθητήρα RHTO3 με το Arduino Mega RDK Rev3

Η διασύνδεση του αισθητήρα με την πλακέτα Arduino πραγματοποιείται χρησιμοποιώντας τη βιβλιοθήκη DHTlib. Όπως αναφέραμε και προηγουμένως, η σύνδεση του αισθητήρα με το Arduino γίνεται μέσω μιας ψηφιακής εισόδου. Στην είσοδο αυτή εισέρχεται τάση η οποία έχει σταθμιστεί ανάλογα με την τιμή θερμοκρασίας / υγρασίας του περιβάλλοντος και μετατρέπεται σε τιμή η οποία αντιστοιχεί στην θερμοκρασία/υγρασία. Τη μετατροπή αυτή αναλαμβάνει η βιβλιοθήκη DHTlib.

Σύμφωνα με το datasheet της εταιρίας ο αισθητήρας αναγνωρίζει κάποιες εντολές:

1) readData(): επιστρέφει την κατάσταση του αισθητήρα.

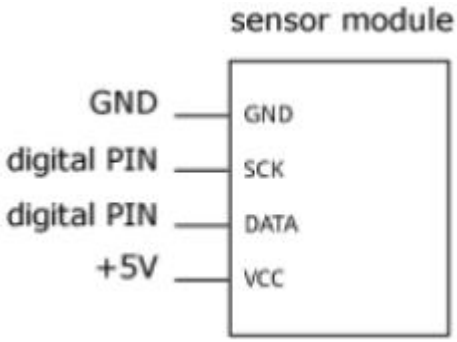
   Επιστρέφονται οι τιμές:

- DHT_ERROR_NONE: δεν υπάρχουν σφάλματα
- DHT_ERROR_CHECKSUM: σφάλμα κατά τον έλεγχο με το bit ισοτιμίας
- DHT_BUS_HUNG: σφάλμα
- DHT_ERROR_NOT_PRESENT: δεν εντοπίστηκε ο αισθητήρας
- DHT_ERROR_ACK_TOO_LONG: το σήμα ACK έληξε
- DHT_ERROR_SYNC_TIMEOUT: σφάλμα κατά τον συγχρονισμό
- DHT_ERROR_DATA_TIMEOUT: σφάλμα δεδομένων
- DHT_ERROR_TOOQUICK: ζητήθηκε γρηγορότερα από τον αναμενόμενο χρόνο (<2sec) αίτημα για νέα δεδομένα

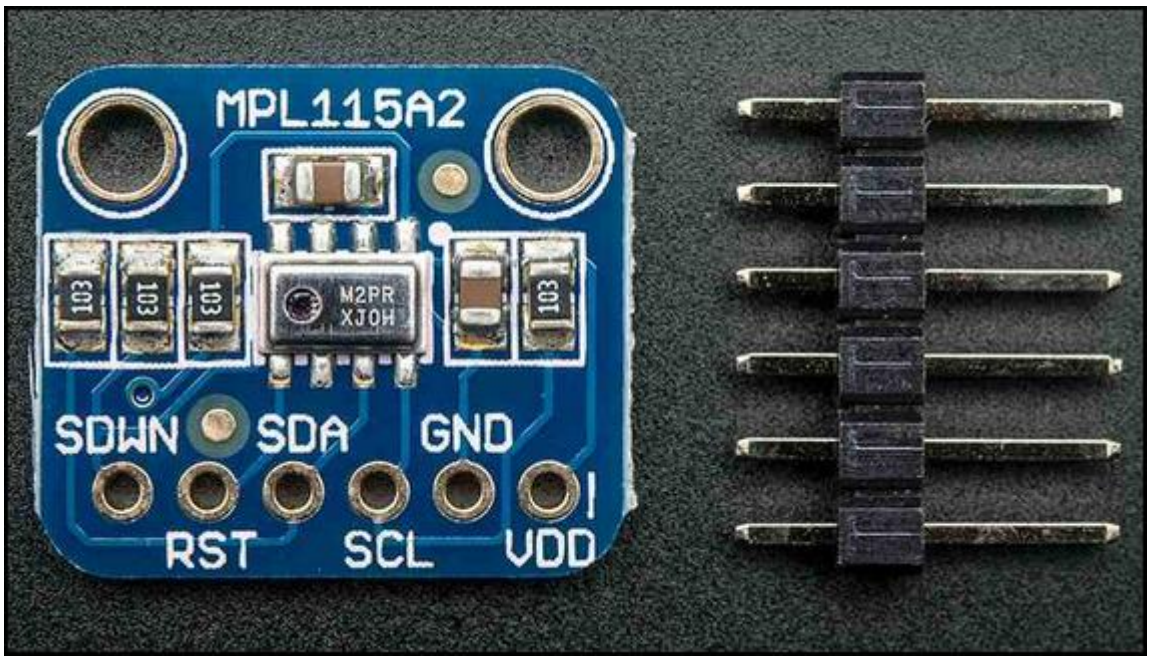
2)getHumidity(): επιστρέφει την τρέχουσα τιμή του αισθητήρα για την υγρασία σε 3)getTemperatureC(): επιστρέφει την τρέχουσα τιμή της θερμοκρασίας σε βαθμούς κελσίου.

4.4. Αισθητήρας Ατμοσφαιρικής Πίεσης MPL 11542

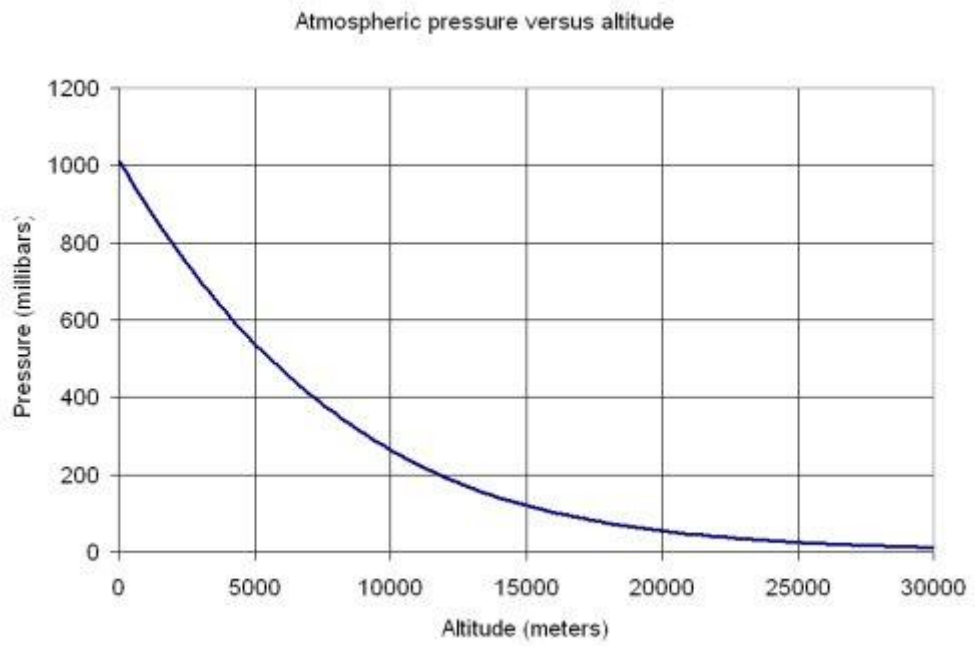
Ο αισθητήρας MPL 11542 (Εικόνα 23 και Σχήμα 4) έχει σχεδιαστεί για την μέτρηση ειδικά της ατμοσφαιρικής πίεσης αλλά και της θερμοκρασίας της ατμόσφαιρας. Επίσης λόγω της συσχέτισης της ατμοσφαιρικής πίεσης και του ύψους στην οποία γίνεται η μέτρηση, η βιβλιοθήκη παρέχει και την επιπλέον λειτουργία, της ένδειξης του τρέχοντος υψομέτρου της περιοχής του αισθητήρα. Η γραφική παράσταση της συνάρτησης μεταξύ του ύψους και της ατμοσφαιρικής πίεσης εμφανίζεται στο Σχήμα 5.



Σχήμα 4, ο αισθητήρας MPL 11542



Εικόνα 23, ο αισθητήρας MPL 11542



Σχήμα 5, γραφική παράσταση της συνάρτησης μεταξύ του ύψους και της ατμοσφαιρικής πίεσης

Οι τεχνικές προδιαγραφές του αισθητήρα είναι :

- Εύρος ευαισθησίας αισθητήρα: 50-115 kPa
- Ευαισθησία: ±1 kPa / 0.25m
- Θερμοκρασία περιβάλλοντος: -40 C έως +105 C
- Τροφοδοσία: 2,375 V – 5,5V

4.4.1 Συνδεσμολογία του αισθητήρα με την πλακέτα Arduino

Καρφίτσα Arduino	**ADXL345 καρφίτσα**
A4	SDA
A5	SCL
3V3	VCC
Gnd	GND

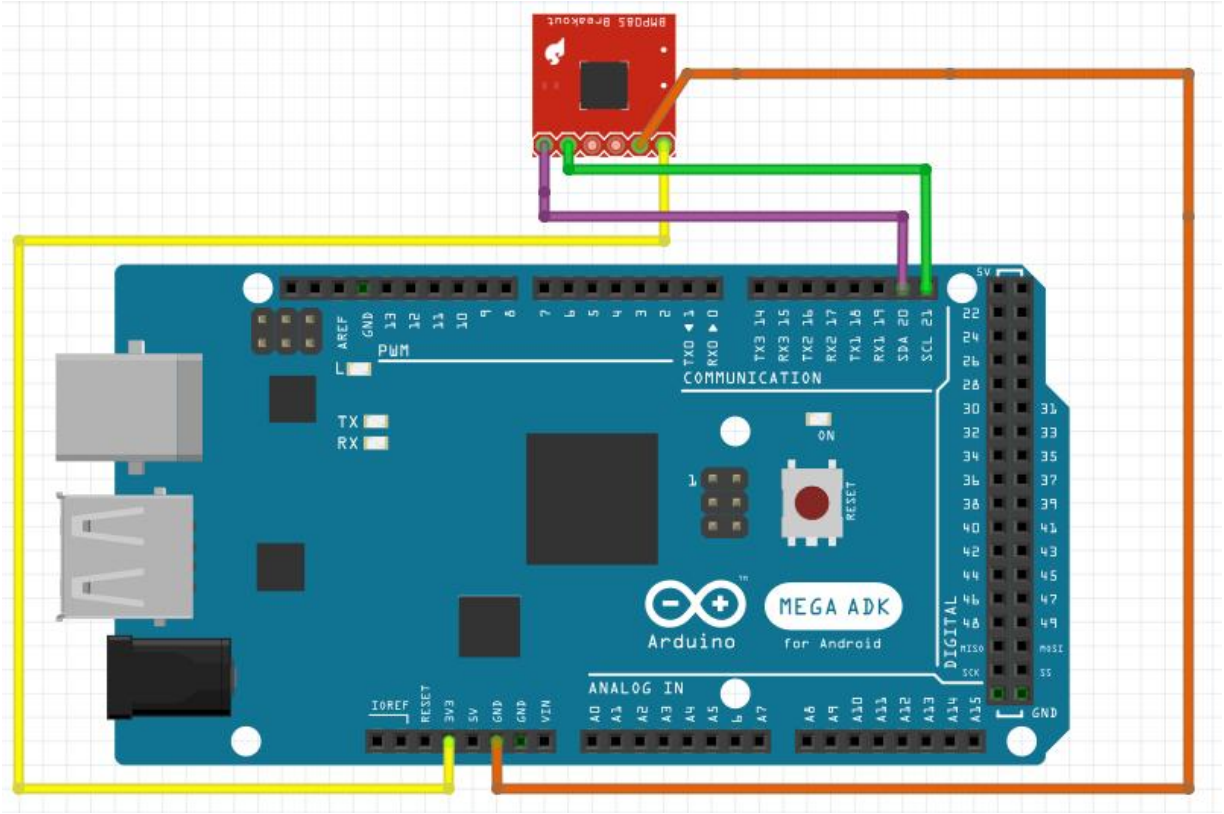
Πίνακας 10, Συνδεσμολογία του
MPL 115A42 με το Arduino Mega ADK Rev 3

Στον παραπάνω πίνακα (Πίνακας 10), καθώς και στην Εικόνα 24, βλέπουμε την συνδεσμολογία του αισθητήρα με την πλακέτα. Στον ακροδέκτη VCC του αισθητηρίου συνδέεται η τάση 3.3V του Arduino και στο GND η γείωσή του. Επιπλέον ο ακροδέκτης SCL συνδέεται με το ρολόι του arduino (θύρα 21) για την επίτευξη του συγχρονισμού του αισθητηρίου. Ο ακροδέκτης SDA συνδέεται με το πρωτόκολλο επικοινωνίας i2c, μέσω της θύρας 20 του Arduino, και είναι υπεύθυνο για την ανάγνωση των δεδομένων του αισθητήρα θερμοκρασίας και ατμοσφαιρικής

πίεσης. Να αναφέρουμε ότι σύμφωνα με το documentation του συγκεκριμένου αισθητήρα δεν πρέπει να γίνονται μετρήσεις συχνότερα από 3 δευτερόλεπτα.

Για την διασύνδεση με το Arduino θα γίνει χρήση της βιβλιοθήκης Adafruit_MPL115A2.h την οποία έχει αναπτύξει η εταιρία Adafruit και είναι ανοιχτού κώδικα. Για να χρησιμοποιήσουμε την βιβλιοθήκη αρκεί η αναφορά της στην αρχικοποίηση του προγράμματος.
Σύμφωνα με το datasheet της εταιρίας ο αισθητήρας αναγνωρίζει κάποιες εντολές:

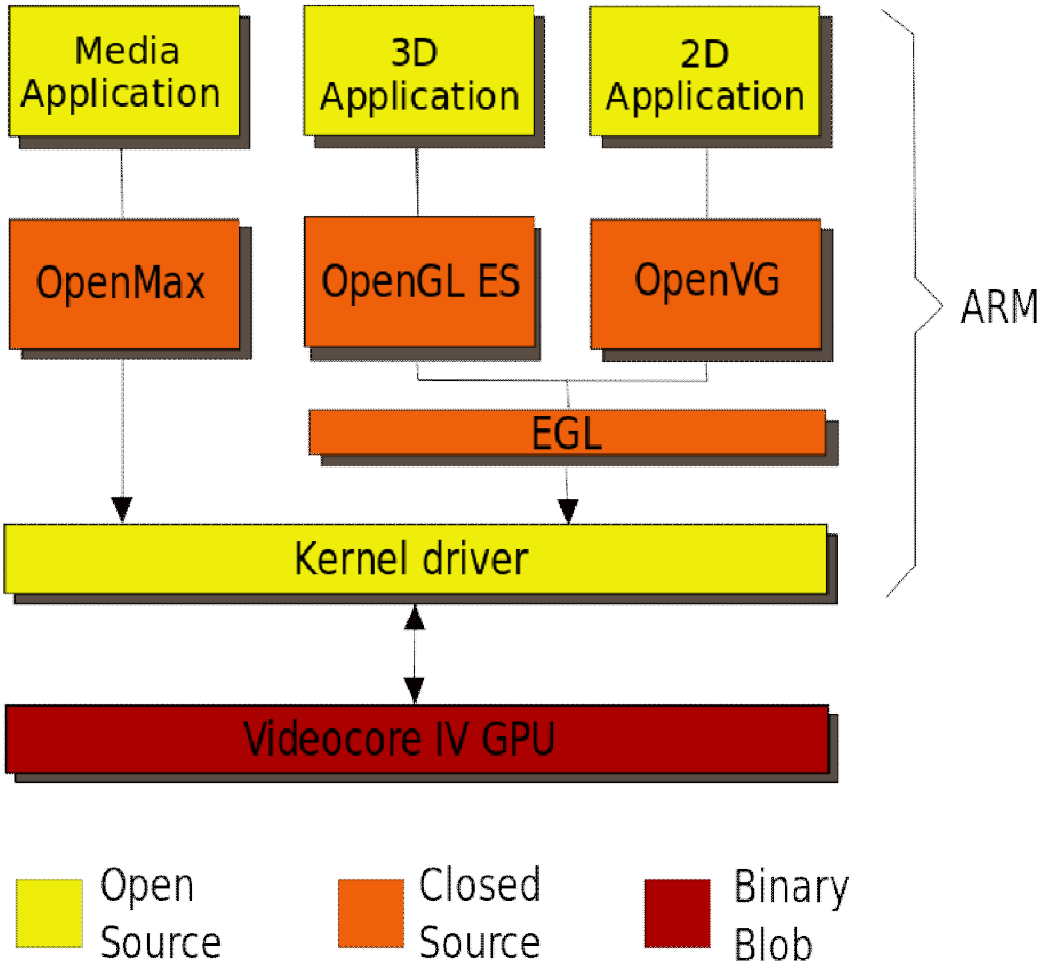
- begin(): γίνεται η έναρξη της ανάγνωσης από τον αισθητήρα.
- readTemperature(): επιστρέφει σε float την τιμή της τρέχουσας θερμοκρασίας.
- readPressure(): επιστρέφει σε float την τιμή της τρέχουσας ατμοσφαιρικής πίεσης.\
- readAltitute(int): δέχεται ως όρισμα την τιμή της ατμοσφαιρικής πίεσης και επιστρέφει το υψόμετρο.
- seaLevelPressure (ύψος, πίεση, ατμόσφαιρα): δέχεται ως ορίσματα την τιμή της ατμοσφαιρικής πίεσης σε (hPa), του ύψους (σε μέτρα) και της θερμοκρασίας (σε βαθμούς Celsius) και επιστρέφει την τιμή της ατμοσφαιρικής πίεσης της επιφάνειας της θάλασσας
- setup() γίνεται η σύνδεση του αισθητήρα με το Arduino



Εικόνα 24, Σύνδεση αισθητήρα MPL115A2 με το Arduino Mega RDK Rev3

4.5. Το Raspberry Pi Model B

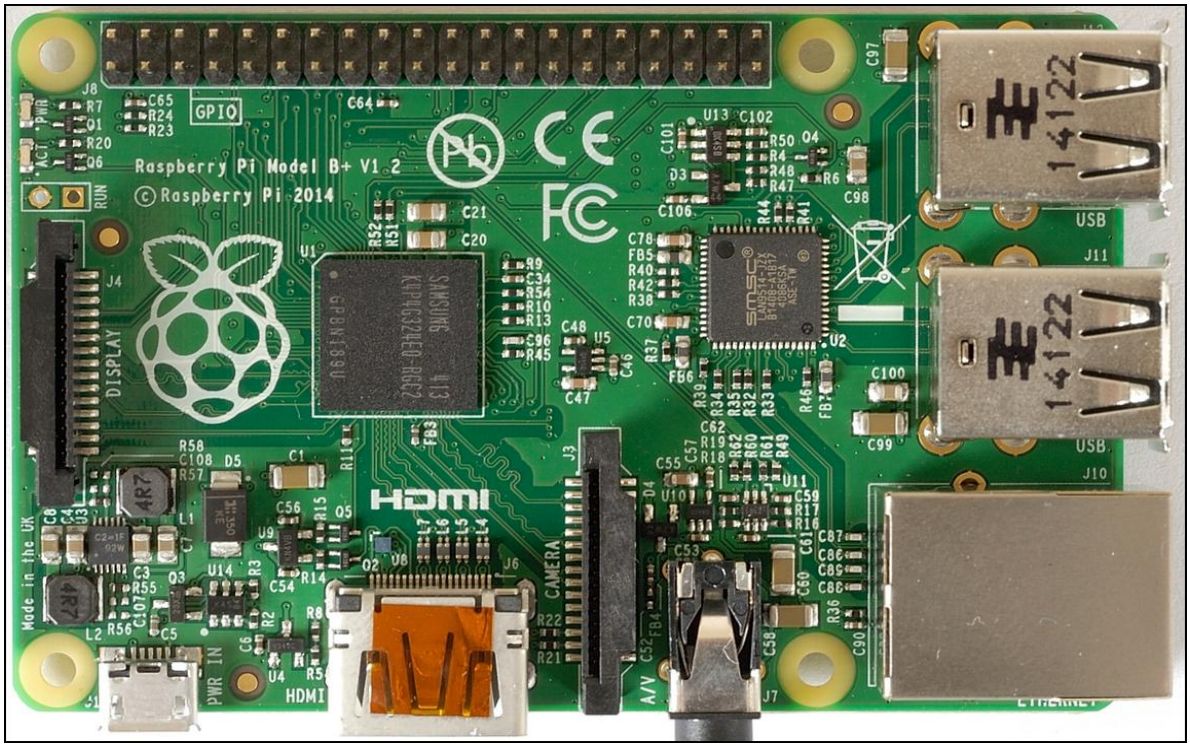
Το Raspberry Pi Model B υποστηρίζεται από το BCM2836 σύστημα της Broadcom, που περιέχει 900 MHz επεξεργαστή, ARM 7 Quad Core Processor, κάρτα γραφικών Video Core 4 και μνήμη RAM χωρητικότητας 1GB. Στο εσωτερικό του δεν περιλαμβάνεται κάποιος σκληρός δίσκος ή SSD disc, αλλά χρησιμοποιείται κάρτα μνήμης SD. Το ίδρυμα Raspberry Pi foundation παρέχει διανομές debian και arch linux, τροποποιημένες για τον ARM επεξεργαστή του. Κάποιες από αυτές είναι οι Rasbian OS, slackware ARM και Android.



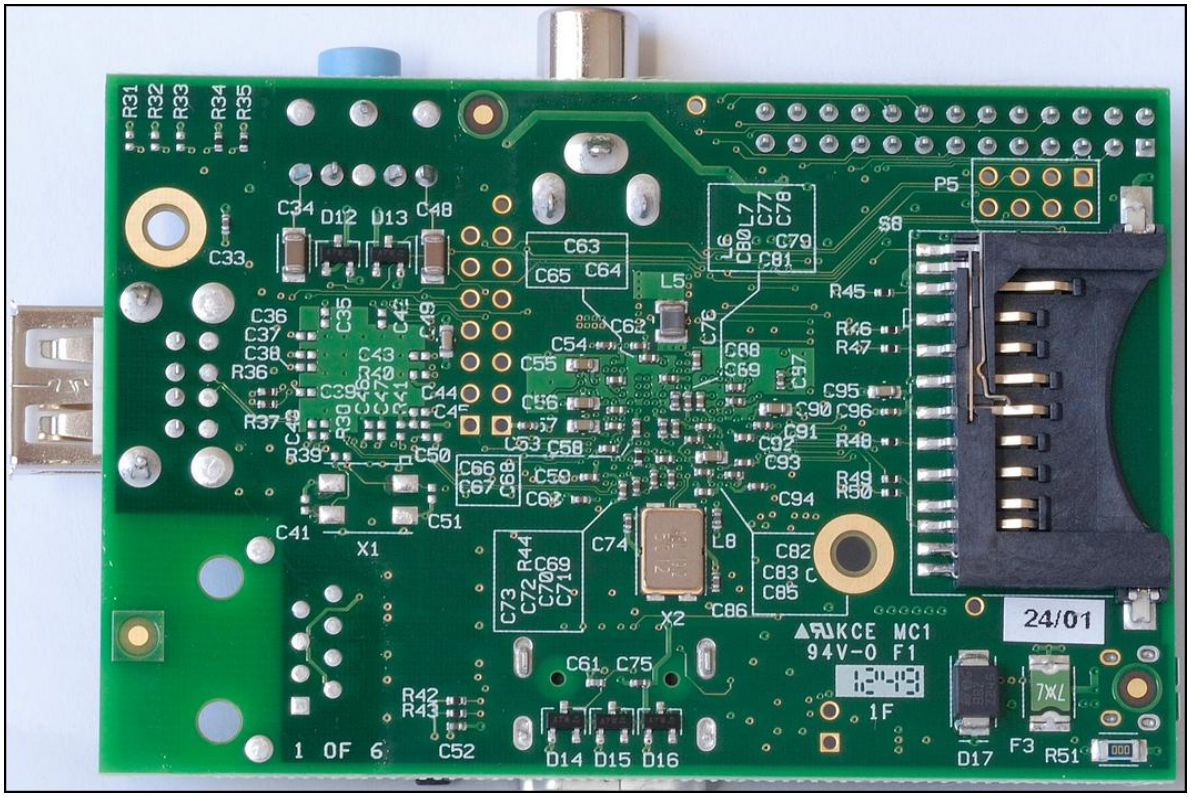
Σχήμα 6, Δομή του επεξεργαστή ARM

4.5.1. Schematics & Reference Design του Raspberry Pi Model B

Στις παρακάτω Εικόνες (25α, 25β) απεικονίζονται τα Schematics του Raspberry Pi Model B

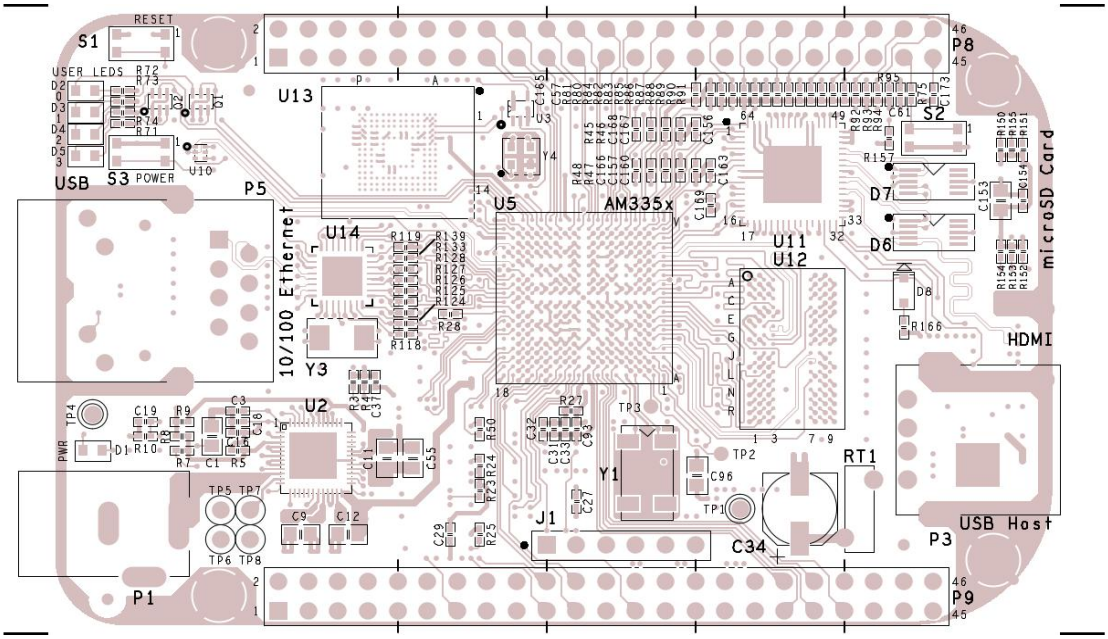


Εικόνα 25α, Front Schematic of Raspberry Pi Model B

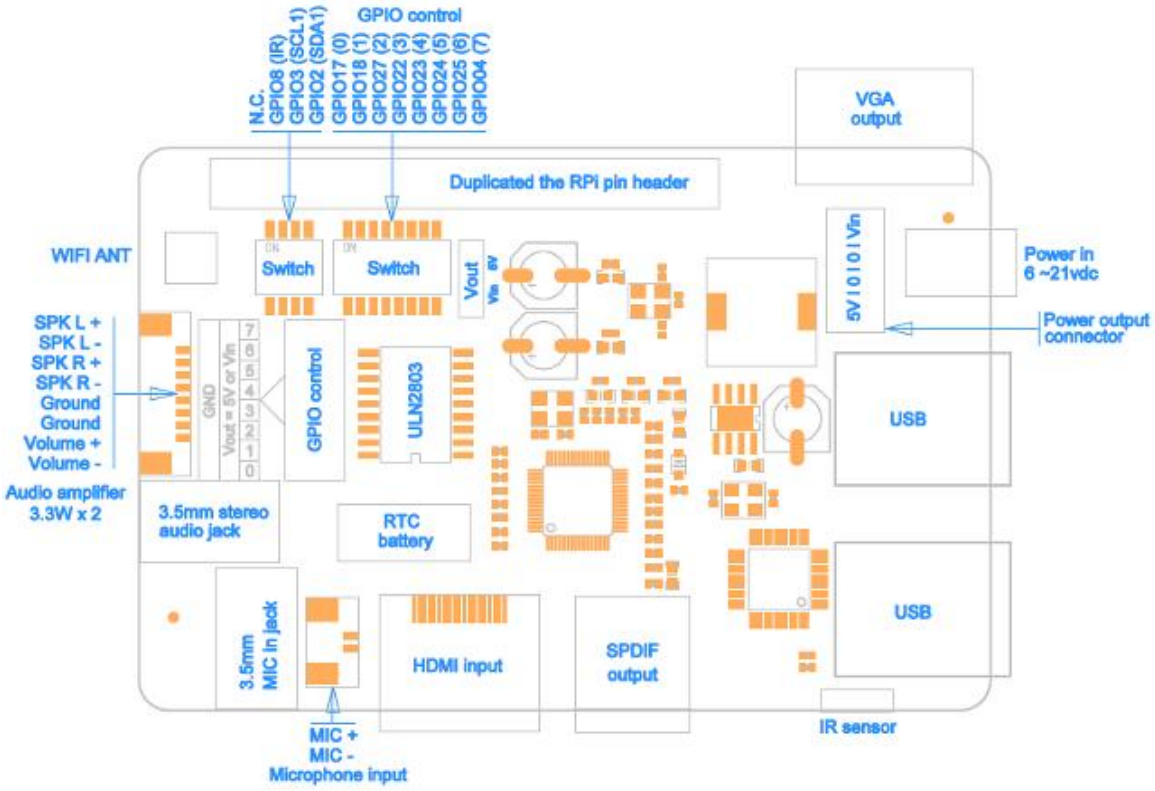


Εικόνα 25β, Back Schematic of Raspberry Pi Model B

Έπειτα στο Σχήμα 7α και 7β απεικονίζεται το Reference Design του Raspberry Pi Model B



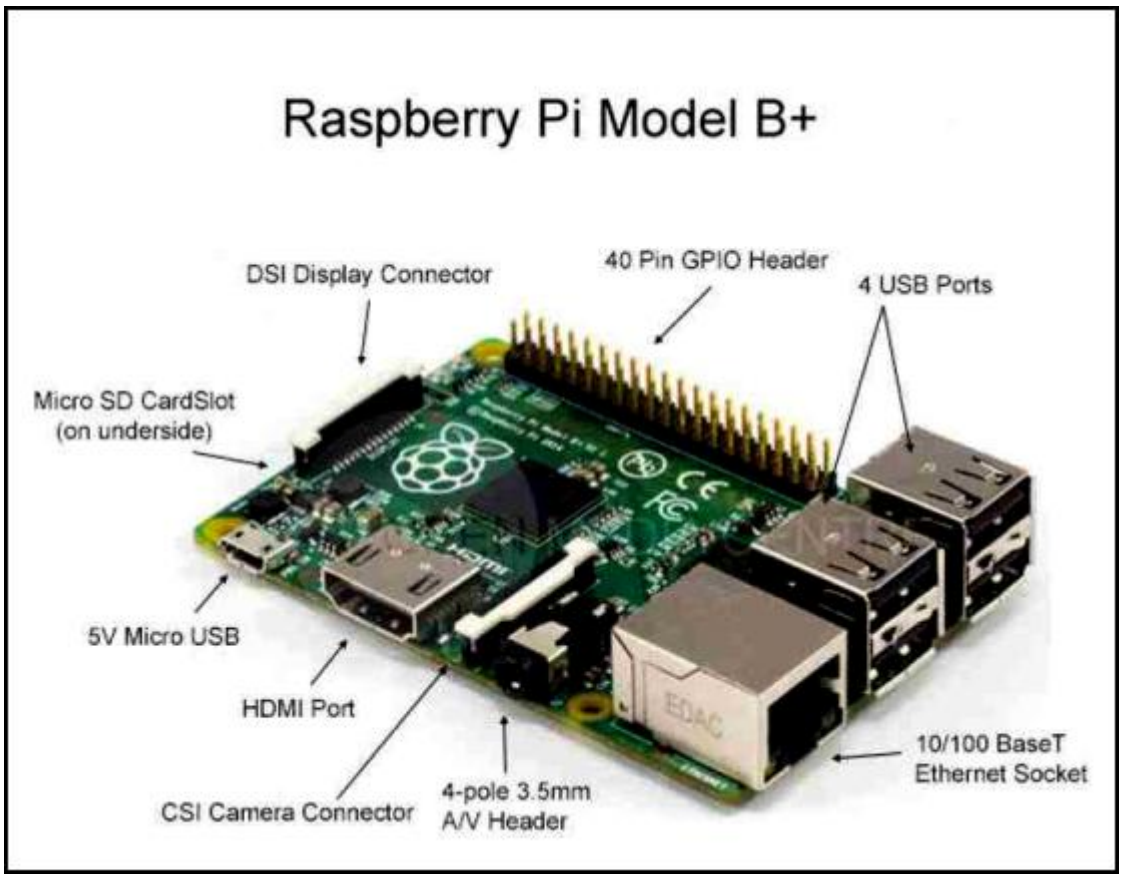
Σχήμα 7α, Reference Design του Raspberry Pi Model B



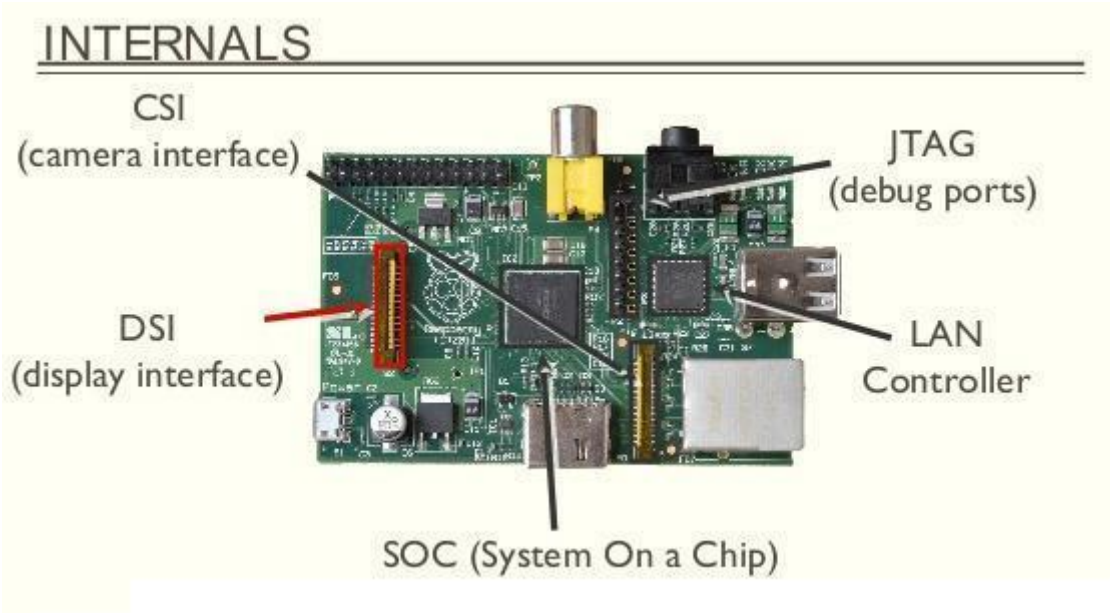
Σχήμα 7β, Reference Design του Raspberry Pi Model B

4.5.2. Ακροδέκτες του Raspberry Pi Model B

Το Raspberry Pi Model B συνδέεται (Εικόνα 27α και 27β) με οθόνη για την εμφάνιση διάφορων πληροφοριών μέσω θύρας HDMI. Επιπλέον δεν έχει άμεση σύνδεση σε δίκτυα που υποστηρίζουν το πρότυπο RS485, και γι' αυτό το λόγο απαιτείται μετατροπέας της σειριακής επικοινωνίας UART του Rasprberry Pi B σε RS485. Ακόμα περιλαμβάνει μια θύρα για την τροφοδοσία του (5V με ισχύ 2 Watt). Για το χειρισμό του λειτουργικού του υποστηρίζει 4 θύρες USB (τύπου 2.0), όπου και οι τέσσερις μαζί μπορούν να δώσουν μέχρι 1,2A ρεύματος, σε διαφορετική περίπτωση απαιτείται εξωτερική τροφοδοσία. Για την σύνδεση στο διαδίκτυο διαθέτει μία υποδοχή Ethernet. Εκτός από αυτά, διαθέτει δύο σειριακές θύρες για καλωδιοταινία, που εξυπηρετούν ανάγκες display και κάμερας, αλλά και 40 επιπλέον pin (GPIO θύρα) όπου συνδέονται λοιπές συσκευές μέσω ενός socket. Η υποδοχή microSD είναι τύπου Push - Push για εύκολη εισαγωγή και εξαγωγή της κάρτας SD. Τέλος, μέσω του jack 3,5" audio/video out έχει τη δυνατότητα να διανείμει εικόνα και ήχο.

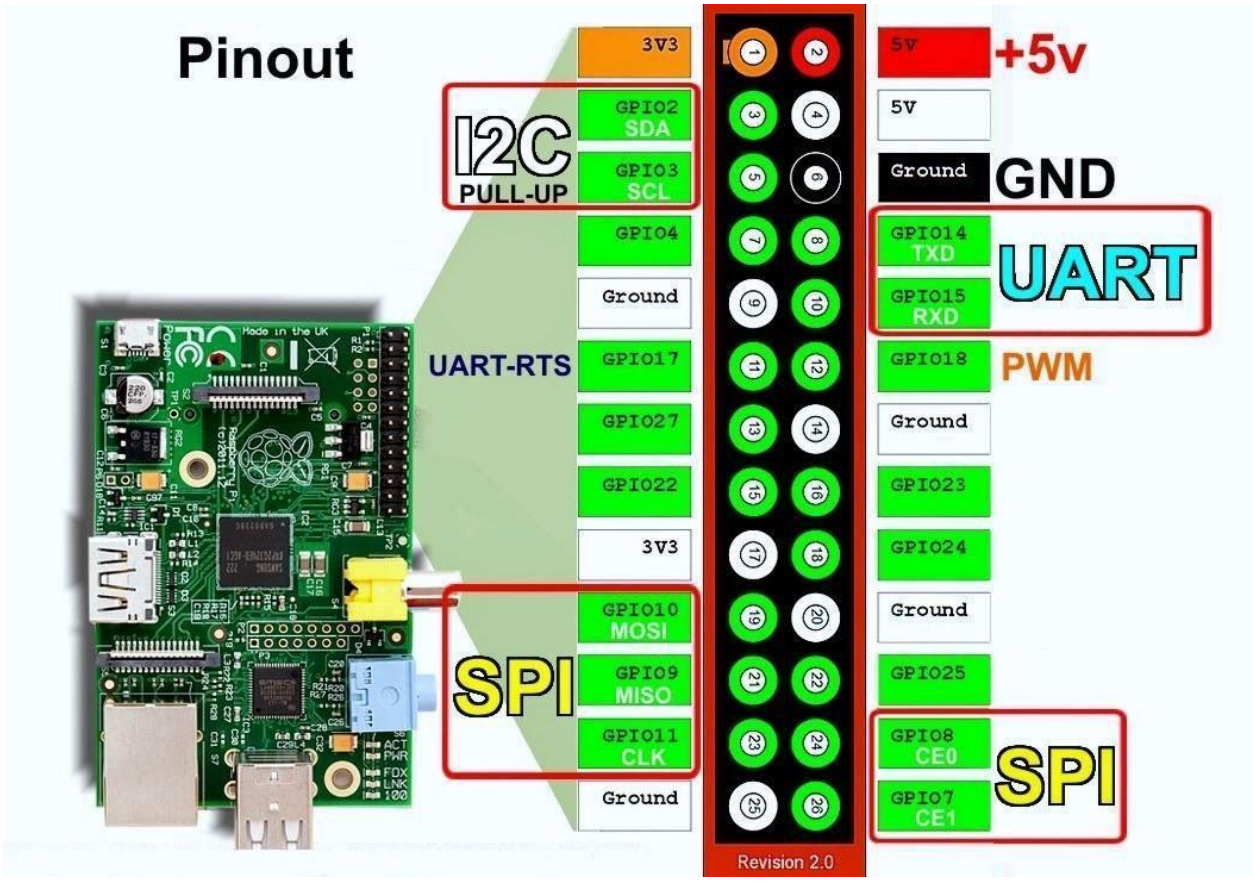


Εικόνα 27α, Θύρες/Ακροδέκτες σύνδεσης του Raspberry Pi Model B



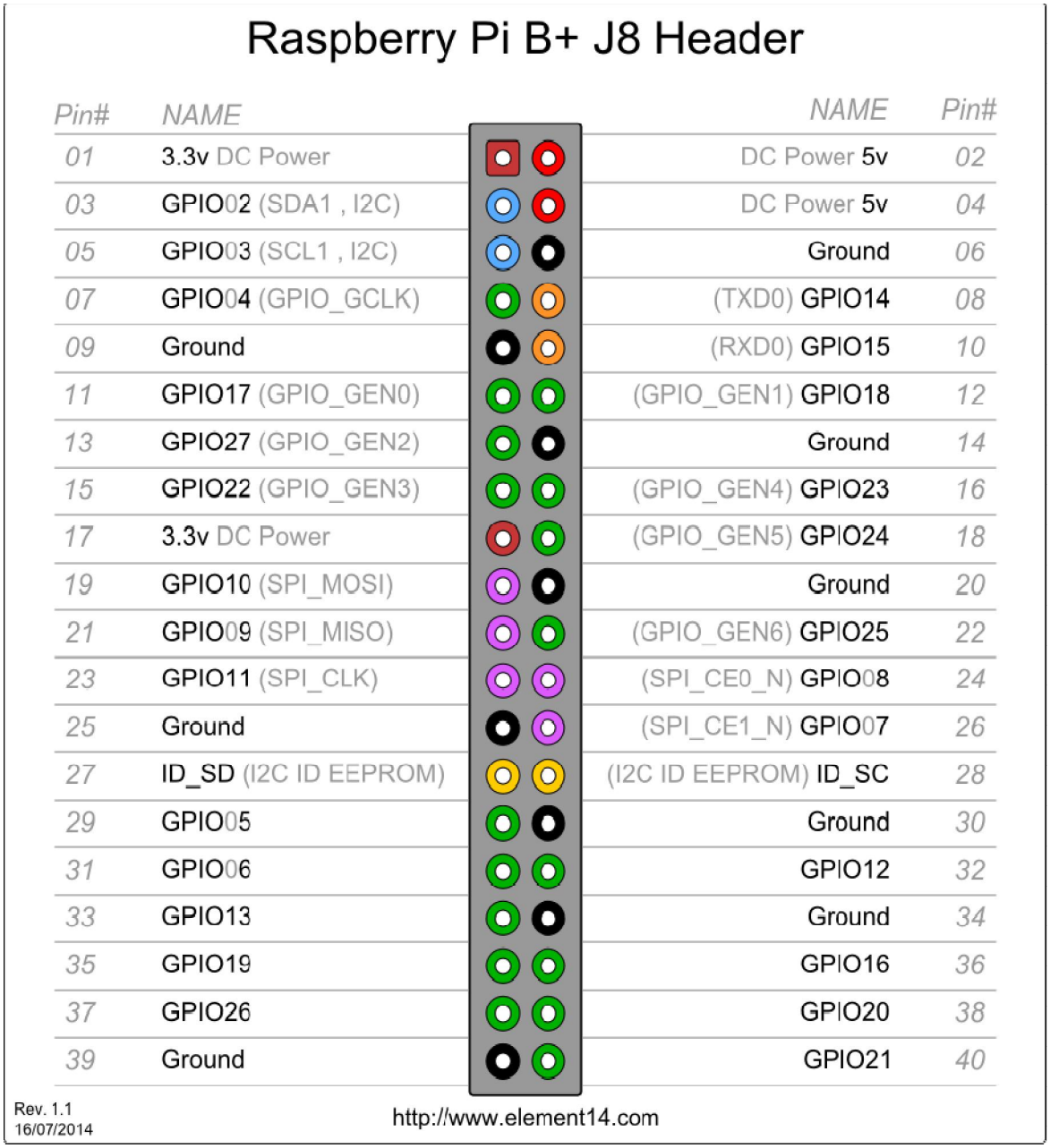
Εικόνα 27β, Θύρες/Ακροδέκτες σύνδεσης του Raspberry Pi Model B

Τα pins που βρίσκονται πάνω στην πλακέτα και παρουσιάζονται στο σχήμα 8 αποτελούν έναν ολόκληρο ξεχωριστό κόσμο. Είναι 2 σειρές από pins που το καθένα από αυτά έχει το δικό του ρόλο και το οποίο μπορεί να αλλάξει τη λειτουργία του μέσω λογισμικού, οπότε και ονομάζονται είσοδοι – έξοδοι γενικού σκοπού. Όλα μπορούν να αλλάξουν και να υποστηρίξουν διαφορετική λειτουργία όπως για παράδειγμα SPI,PWM ,I2C.

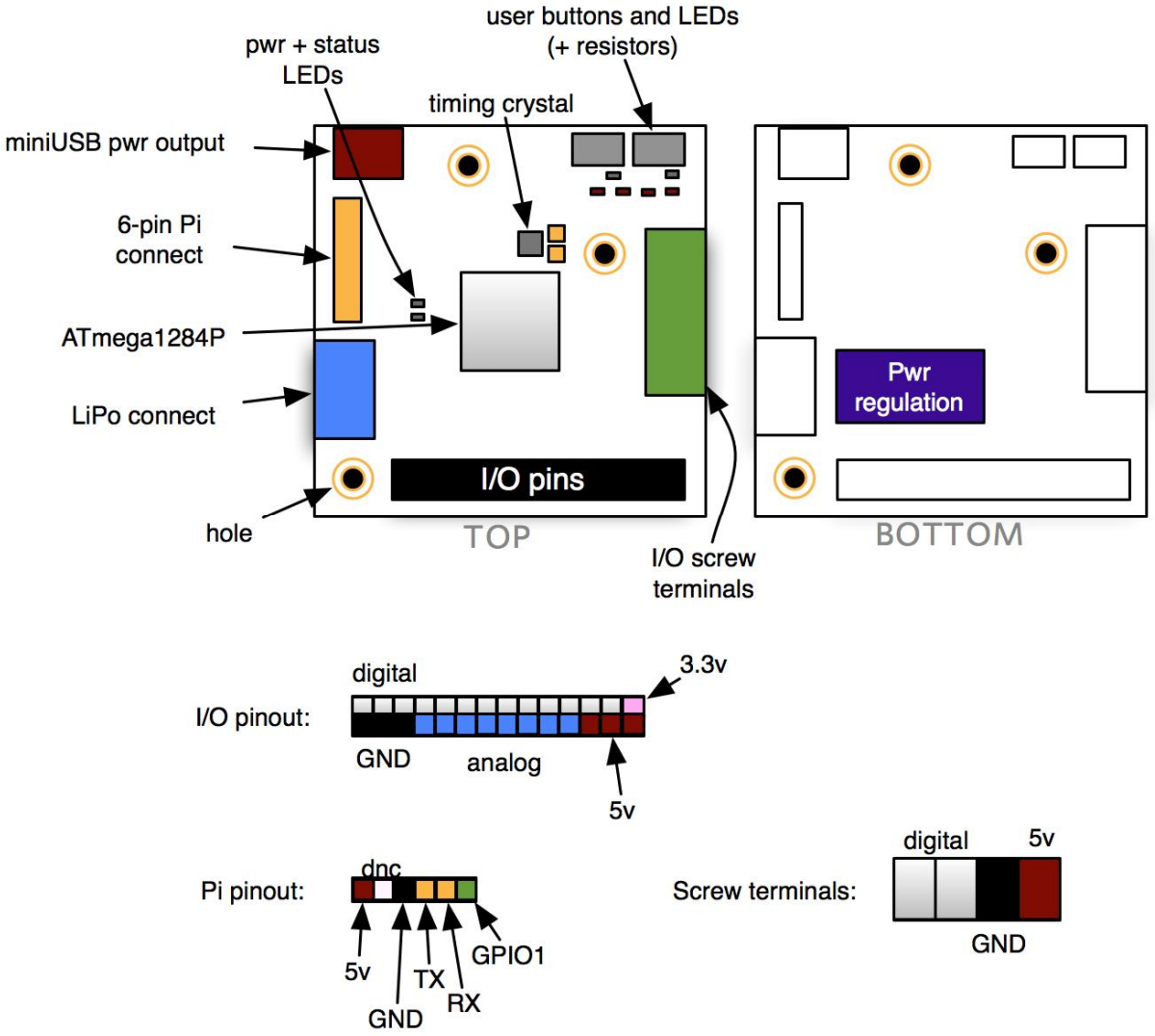


Σχήμα 8, Pins του Raspberry Pi Model B

Ας δούμε τα παρακάτω σχήματα (Σχήμα 9α και 9β) μέσα από τα οποία πραγματοποιείται η περιγραφή τους:



Σχήμα 9α, Περιγραφή των Pins του Raspberry Pi Model B



Σχήμα 9β, Περιγραφή των Pins του Raspberry Pi Model B

4.6. Βασικά χαρακτηριστικά που καθιστούν προτιμότερα το Arduino και το Raspberry Pi, σε σχέση με άλλους μικροελεγκτές:

I. Μικρότερο μέγεθος.
II. Επίτευξη ελέγχου ή μετρήσεων σε πραγματικό χρόνο.
III. Περισσότεροι διαθέσιμοι ακροδέκτες για ψηφιακές εισόδους / εξόδους.
IV. Η βασική αρχιτεκτονική των μικροεπεξεργαστών τους δε διαφέρει από αυτή των κοινών επεξεργαστών, καλύπτοντας έτσι μεγάλο εύρος απαιτήσεων
V. Ευκολότερη υλοποίηση εφαρμογών, λόγω της ταχύτερης διάδοσής τους και των μεγάλων κοινοτήτων και forum που παρέχουν πληθώρα project και πληροφοριών.
VI. Αυτονομία που μπορεί να επιτευχθεί από την τροφοδότηση των μικροελεγκτών μέσω του καλωδίου επικοινωνίας, όπως της θύρας Ethernet (POE).
VII. Έχουν αναπτυχθεί πολλά πολλά συμβατά shields και αισθητήρια, για την ανάπτυξη εφαρμογών, τόσο ερασιτεχνικών όσο και επαγγελματικών.

5. Τα μετεωρολογικά φαινόμενα και η μέτρησή τους

5.1. Αυτόνομος μετεωρολογικός σταθμός

Ένας μετεωρολογικός σταθμός αποτελεί μια εγκατάσταση, τοποθετημένη στην ξηρά ή στη θάλασσα, που διαθέτει τα απαραίτητα μέσα και τον εξοπλισμό για τη μέτρηση των ατμοσφαιρικών συνθηκών και την παροχή πληροφοριών για την πρόγνωση του καιρού και τη μελέτη του κλίματος. Οι μετρήσεις που λαμβάνονται περιλαμβάνουν τη θερμοκρασία, τη βαρομετρική πίεση, την υγρασία, την ταχύτητα και διεύθυνση του ανέμου, καθώς και τα επίπεδα της βροχόπτωσης. Οι παρατηρήσεις λαμβάνονται τουλάχιστον μία φορά την ημέρα, ενώ οι αυτοματοποιημένες μετρήσεις λαμβάνονται τουλάχιστον μία φορά κάθε ώρα.
Με τον όρο "Αυτόνομος μετεωρολογικός σταθμός" (ΑΜΣ) αναφερόμαστε σε ένα μετεωρολογικό σταθμό, στον οποίο γίνονται λήψεις μετεωρολογικών δεδομένων με αυτοματοποιημένο τρόπο και στην συνέχεια τα δεδομένα αυτά αποστέλλονται σε τοπικές ή κεντρικές μονάδες για την αποθήκευσή τους ή και την περαιτέρω επεξεργασία τους, από κεντρικές μονάδες επεξεργασίας (πχ Arduino). Ένας αυτόνομος σταθμός καιρού μπορεί να αποτελεί μέρος ενός δικτύου συνολικής καταγραφής, και ονομάζεται «Αυτοματοποιημένο Σύστημα Παρατήρησης Καιρικών Συνθηκών».

5.2. Καιρός

Καιρός είναι το σύνολο των μετεωρολογικών παραμέτρων σε μία συγκεκριμένη τοποθεσία και κατά μία συγκεκριμένη χρονική στιγμή. Με άλλα λόγια, καιρός ονομάζεται το σύνολο των μετεωρολογικών φαινομένων που παρατηρούνται στην ατμόσφαιρα της Γης από την άποψη της θερμοκρασίας, της πίεσης της υγρασίας και του υφισταμένου ανέμου (ένταση και διεύθυνση), με ότι άλλο φαινόμενο συνοδεύει αυτά. Ο καιρός προσδιορίζεται για μια συγκεκριμένη χρονική στιγμή του ίδιου πάντα τόπου και η πρόγνωση του βασίζεται στα αποτελέσματα των παρατηρήσεων του. Στην πραγματικότητα είναι μια συνάρτηση στην οποία εισάγονται μετεωρολογικά δεδομένα, ο τόπος, ο χρόνος κλπ. και με βάση τις καταγραφές και παρατηρήσεις που έχουν γίνει, εξάγεται ένα αποτέλεσμα. Το αποτέλεσμα αυτό αποτελεί την πρόγνωση του καιρού και η ακρίβεια του εξαρτάται από το μετεωρολογικό μοντέλο που θα χρησιμοποιηθεί.

5.3 Θερμοκρασία

Η θερμοκρασία είναι η φυσική ιδιότητα που προσδιορίζει τη θερμική κατάσταση διαφόρων σωμάτων και συνδέεται με την κινητική ενέργεια των σωματιδίων ενός συστατικού, το οποίο χαρακτηρίζεται θερμό ή ψυχρό ανάλογα με το αν απορροφάται ή εκλύεται θερμότητα από αυτό.
Θερμοκρασία ατμόσφαιρας ονομάζεται η θερμοκρασία την οποία έχει ο ατμοσφαιρικός αέρας πάνω από μια περιοχή. Η πρόγνωση του καιρού σε μια περιοχή βασίζεται κυρίως στη γνώση της εκάστοτε ατμοσφαιρικής πίεσης και της θερμοκρασίας της ατμόσφαιρας της υπόψιν περιοχής και των γύρω αυτής εκτάσεων.

Η θερμοκρασία της ατμόσφαιρας μετριέται με θερμόμετρα ή αισθητήρια και υπάρχουν διάφορες κλίμακες μέτρησης, με συνηθισμένες κλίμακες τις Κελσίου (Celsius, σύμβολο Cº) , Κέλβιν (Kelvin, σύμβολο Kº) και Φαρενάιτ (Fahrenheit, σύμβολο Fº).
Κάθε θερμοκρασία που μετριέται, αρχίζοντας από το 0 της κλίμακας Κελσίου ή της κλίμακας Φαρενάιτ, ονομάζεται σχετική θερμοκρασία, και καλείται θετική ή αρνητική όταν είναι πάνω ή κάτω του μηδενός αντίστοιχα. Η σχετική θερμοκρασία έχει ιδιαίτερα ευρύτατη χρήση τόσο στην καθημερινή ζωή, όσο και στις διάφορες τεχνικές και μηχανολογικές εφαρμογές.

5.4. Η ατμοσφαιρική πίεση

Η γη περιβάλλεται από ατμόσφαιρα. Η ατμόσφαιρα αποτελείται από ένα μείγμα αερίων που ονομάζεται ατμοσφαιρικός αέρας. Ο αέρας είναι διαφανής. Έχει μάζα και από τη γη ασκείται σε αυτόν η δύναμη του βάρους. Επομένως, όπως συμβαίνει με όλα τα ρευστά σώματα, ασκεί πίεση σε κάθε επιφάνεια που βρίσκεται μέσα σ' αυτόν. Η πίεση αυτή ονομάζεται ατμοσφαιρική πίεση ή «Βαρομετρική πίεση» και μειώνεται ανάλογα με υψόμετρο. Όπως ακριβώς η υδροστατική πίεση μιας κατακόρυφης στήλης νερού οφείλεται στο βάρος της, έτσι και η ατμοσφαιρική πίεση οφείλεται στο βάρος του αέρα.
Η ατμοσφαιρική πίεση είναι ένα από τα πιο σημαντικά μετεωρολογικά στοιχεία διότι οι καιρικές καταστάσεις και οι μεταβολές τους συνδέονται άμεσα μαζί της. Επομένως, το μοντέλο που θα χρησιμοποιηθεί για την πρόγνωση του ΑΜΣ βασίζεται κυρίως στις μεταβολές της ατμοσφαιρικής πίεσης. Οι μετρήσεις της ατμοσφαιρικής πίεσης που γίνονται σε ύψος μεγαλύτερο από αυτό της επιφάνειας της θάλασσας, πρέπει να τροποποιηθούν ώστε να αντιστοιχούν σε μηδενικό ύψος. Η διαδικασία αυτή ονομάζεται ως αναγωγή στην επιφάνεια της θάλασσας.
Η σχέση με την οποία εκφράζεται η αναγωγή στην επιφάνεια της θάλασσας ορίζεται ως

$$P = P_0 \left(1 - \frac{0{,}0065h}{°\mathrm{C} + 0{,}0065h + 213{,}75}\right)^{-5{,}257}$$

Όπου:
P0 = Η τρέχουσα τιμή της ατμοσφαιρικής πίεσης σε hPa.
h = Το υψόμετρο του σταθμού από την επιφάνεια της θάλασσας σε μέτρα (m).
^{o}C = η τρέχουσα θερμοκρασία σε βαθμούς Κελσίου.

Επιπρόσθετα, βαρομετρική τάση ονομάζεται η μεταβολή της τιμής της ατμοσφαιρικής πίεσης μέσα στην χρονική περίοδο παρατήρησης των 3 ωρών .

Στους επόμενους πίνακες (πίνακας 11α και 11β) παρατηρούμε τον τρόπο υπολογισμού της βαρομετρικής τάσης με συμβολισμούς:

- $p_0 = t - 3$ ώρες (όπου t η τρέχουσα ώρα),
- $p_1 = t - 2$ ώρες,
- $p_2 = t - 1$ ώρα και
- $p_3 = t$.

Πίνακας 11α,Τρόπος Υπολογισμού Βαρομετρικής Τάσης

Code digit	Descriptions		Graphical representation	$p_0 - p_3$	$p_0 + p_3 - p_1 - p_2$	$p_0 - p_1$
0	air pressure is the same as or higher than three hours earlier	rising, then falling		+	–	–
				0	–	+, 0, –
1	air pressure is higher than three hours earlier	rising, then steady		+	–	+, 0
2	air pressure is higher than three hours earlier	rising (steadily or irregularly)		+	0	+, 0, –
3	air pressure is higher than three hours earlier	falling or steady, then rising; or rising, then rising more rapidly		+	+	+, 0, –
4	air pressure is the same as three hours earlier	steady		0	0	+, 0, –
5	air pressure is the same as or lower than three hours earlier	falling, then rising		0	+	+, 0, –
				–	+	+
6	air pressure is lower than three hours earlier	falling, then steady; or falling, then falling more slowly		–	+	0, –
7	air pressure is lower than three hours earlier	falling (steadily or irregularly)		–	0	+, 0, –
8	air pressure is lower than three hours earlier	rising or steady, then falling		–	–	+, 0, –

+: result > 0, -: result < 0.

Πίνακας 11β, Τρόπος Υπολογισμού Βαρομετρικής Τάσης

5.4.1. Η βαρομετρική τάση

Η βαρομετρική τάση είναι μία απαραίτητη μεταβλητή για τον υπολογισμό της πρόγνωσης του καιρού, και χρησιμοποιείται από διάφορους αλγόριθμους, όπως ο Zambretti. Για τον υπολογισμό της βαρομετρικής τάσης αφαιρούμε τη μέση ωριαία τιμή της ατμοσφαιρικής πίεσης που μετρήθηκε πριν από τρεις ώρες από την τρέχουσα τιμή της ατμοσφαιρικής πίεσης.

$$Trend = P_t - P_{t-3}$$

Όπου:
Pt : η τρέχουσα μέση τιμή της ατμοσφαιρικής πίεσης
Pt-3: η προ τρίωρου μέση τιμή της ατμοσφαιρικής πίεσης.

Η συνάρτηση ορίζεται σε ένα διάστημα τεσσάρων διακριτών τιμών οι οποίες είναι:

- → 0: όταν το αποτέλεσμα της αφαίρεσης είναι 0 δηλαδή δεν υπάρχει μεταβολή (steady).
- → 1: όταν το αποτέλεσμα είναι θετικός αριθμός σημαίνει ότι η τάση είναι αυξανόμενη (rising)
- → 2: όταν το αποτέλεσμα είναι αρνητικός αριθμός τότε η τάση είναι μειούμενη (falling)
- → -1: όταν δεν έχουν συμπληρωθεί 3 ώρες μετρήσεων, οπότε δεν γίνεται να υπολογιστεί η βαρομετρική τάση.

5.5. Η υγρασία

Με τον όρο υγρασία αναφερόμαστε στους υδρατμούς, οι οποίοι αποτελούν νερό σε αέρια μορφή. Η υγρασία είναι παρούσα παντού στην ατμόσφαιρα, αλλά λόγω της εξαιρετικά δυναμικής περιοχής τιμών είναι δύσκολη η μέτρησή της. Η σχετική υγρασία του αέρα αποτελεί την πιο κοινή έκφραση της ατμοσφαιρικής υγρασίας. Χαρακτηρίζει το λόγο των υδρατμών που υπάρχουν στην ατμόσφαιρα σε μια δεδομένη τιμή θερμοκρασίας και πίεσης, σε σχέση με τη μέγιστη ποσότητα των υδρατμών την οποία ο αέρας είναι ικανός να κρατήσει στις ίδιες συνθήκες πίεσης και θερμοκρασίας, μέχρις ότου αυτός κορεστεί.
Όταν ο αέρας περιέχει τη μέγιστη τέτοια ποσότητα ονομάζεται κορεσμένος.
Όσο ψυχρότερος είναι ο αέρας τόσο μικρότερη ποσότητα υδρατμών μπορεί να συγκρατήσει. Αν λοιπόν μια μάζα υγρού και θερμού αέρα ψυχθεί θα φτάσει σε μια θερμοκρασία όπου δεν είναι δυνατόν πλέον να συγκρατήσει άλλους τους υδρατμούς από τους οποίους περιέχει. Οι υδρατμοί που περισσεύουν θα συμπυκνωθούν ως σταγονίδια πάνω στα αιωρούμενα μικροσωματίδια και θα δημιουργήσουν το νέφος. Αν δε, συμπυκνωθούν πάνω σε ψυχρά αντικείμενα θα δημιουργήσουν τη δρόσο (Εικόνα 28). Η θερμοκρασία στην οποία ο ακόρεστος αέρας καθώς ψύχεται φτάνει στο κορεσμό, ονομάζεται σημείο δρόσου.

Σημείο δρόσου χαρακτηρίζεται το σημείο εκείνο της θερμοκρασίας που όταν οι υδρατμοί ψυχθούν δημιουργούν το φαινόμενο της δρόσου, δηλαδή τις σταγόνες δρόσου.

Εικόνα 28, Δημιουργία δρόσου

Στη θερμοκρασία αυτή εξυπακούεται πως όταν ο αέρας είναι κορεσμένος και δεν μπορεί να συγκρατήσει άλλους υδρατμούς η σχετική υγρασία να είναι 100%. Σημειώνεται όμως ότι η θερμοκρασία κορεσμού της ατμόσφαιρας ή του "σημείου δρόσου" μπορεί να είναι οποιαδήποτε θερμοκρασία, πάνω από τους 0°C.
Η θερμοκρασία αυτή εξαρτάται μόνο από την ποσότητα των υδρατμών που περιέχει 1 κυβικό μέτρο αέρος, συνεπώς εξαρτάται από την απόλυτη υγρασία.
Η θερμοκρασία του σημείου δρόσου αποτελεί σπουδαίο μετεωρολογικό στοιχείο για ένα τόπο και γι' αυτό πάντοτε αναφέρεται στους μετεωρολογικούς χάρτες με τα σύμβολα D.P. από τα αρχικά του αγγλικού όρου Dew Point (Σημείο Δρόσου).
Και με βάση τις τιμές της θερμοκρασίας (T) και της υγρασίας (RH) μπορούμε να υπολογίσουμε και το σημείο δρόσου, σύμφωνα με τον παρακάτω τύπο:

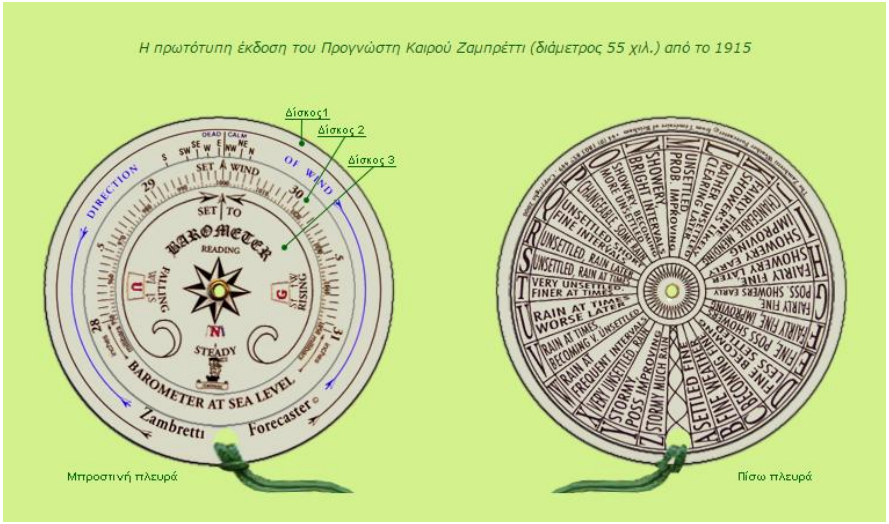
H = ((log10(RH) - 2.0) / 0.4343) + (17.62 * T / (T + 243.12))
DewPoint = 243.12 * H / (17.62 - H)

Η μεταβολή της θερμοκρασίας βάση του σημείου δρόσου παρουσιάζεται στο παρακάτω γράφημα (Γράφημα 1).

Γράφημα 1, Η μεταβολή της θερμοκρασίας βάση του σημείου δρόσου

5.6. Απλοποιημένα μετεωρολογικά μοντέλα

Στις αρχές του 20ου αιώνα όπου δεν υπήρχαν ηλεκτρονικοί υπολογιστές οι οποίοι θα μπορούσαν να πραγματοποιούν επίλυση διαφορικών εξισώσεων με μεγάλη ταχύτητα, η μετεωρολογία βασίζονταν κυρίως στις παρατηρήσεις της μεταβολής της βαρομετρικής τάσης π.χ. πως όταν η πίεση πέφτει γρήγορα σε διάστημα 3 ωρών τότε θα ακολουθήσει συννεφιά. Με βάση αυτές τις παρατηρήσεις κατασκευάστηκαν μηχανές στις οποίες ο χειριστής εισήγαγε μετεωρολογικά δεδομένα και εξήγαγε μια υποτυπώδη πρόγνωση. Δύο από τις πιο διαδεδομένες συσκευές πρόγνωσης καιρού, μέχρι την δεκαετία του 60, είναι ο προγνώστης «Σάγκερ» (Sager) και ο προγνώστης «Ζαμπρέττι» (Zambretti, Εικόνα 29).



Εικόνα 29, Μηχανή Zambretti

Το μοντέλο που θα χρησιμοποιηθεί, προκύπτει από την μελέτη του τρόπου λειτουργίας του προγνώστη Ζαμπρέττι, με μεταβλητές:

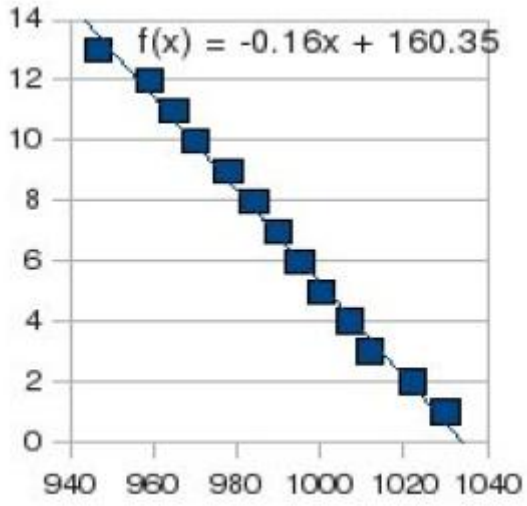
- την τρέχουσα τιμή της ατμοσφαιρικής πίεσης,
- την μέγιστη και ελάχιστη καταγραφείσα τιμή της ατμοσφαιρικής πίεσης
- η εποχή (καλοκαίρι ή χειμώνας)

Ο αλγόριθμος σχεδιάστηκε από τους Νεργέτι (Nergetii) και Ζάμπρα (Zambra) δύο Άγγλους κατασκευαστές οργάνων το 1915.
Υπάρχουν τρεις διαφορετικές συναρτήσεις, μία για κάθε κατάσταση της βαρομετρικής τάσης, από όπου εξάγεται το αποτέλεσμα πρόγνωσης:

α) Όταν η βαρομετρική τάση ανεβαίνει τότε η συνάρτηση που αντιστοιχεί είναι

$$Z_{rizing} = 130 - \left(\frac{p}{81}\right)$$

όπου p η τρέχουσα ατμοσφαιρική πίεση σε hPa, η γραφική παράσταση της οποίας διαγράφεται στο Γράφημα 2.

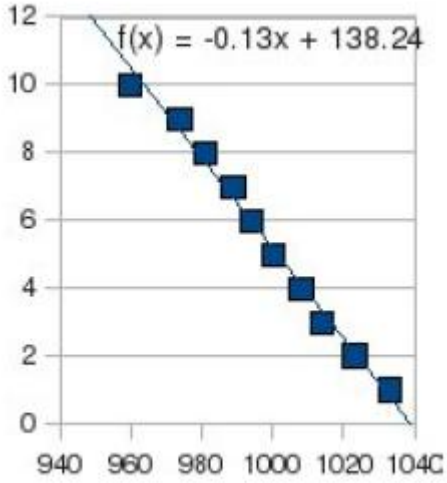


Γράφημα 2, γραφική παράσταση συνάρτησης αυξανόμενης βαρομετρικής τάσης

β) όταν η βαρομετρική τάση παραμένει σταθερή, τότε ισχύει:

$$Z_{steady} = 147 - \left(\frac{5 \cdot p}{376}\right)$$

η γραφική παράσταση της οποίας παρουσιάζεται στο Γράφημα 3.

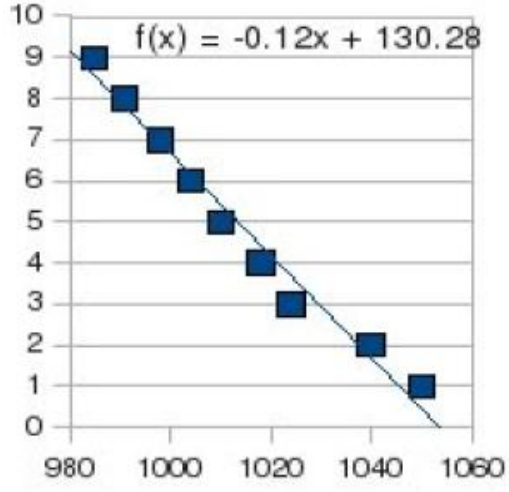


Γράφημα 3, γραφική παράσταση συνάρτησης σταθερής βαρομετρικής τάσης

γ) όταν η βαρομετρική τάση πέφτει τότε ισχύει:

$$Z_{falling} = 179 - \left(\frac{2 \cdot p}{129}\right)$$

η γραφική παράσταση της οποίας φαίνεται στο Γράφημα 4



Γράφημα 4, γραφική παράσταση φθίνουσας βαρομετρικής τάσης.

Το Z που προκύπτει είναι ένας ακέραιος αριθμός στον οποίο αντιστοιχεί η παρακάτω πρόβλεψη (Πίνακας 12):

Αποτελέσματα εξίσωσης και αντιστοιχία τους με τον δίσκο Zambretti

Αποτέλεσμα εξίσωσης Z	Γράμμα δίσκου Ζαμπρετι	Πρόγνωση	Ατμ. Πίεση	Αποτέλεσμα εξίσωσης Z	Γράμμα δίσκου Ζαμπρετι	Πρόγνωση	Ατμ. Πίεση
1;	A;	Αίθριος αμετάβλητος	1050	16;	S;	Άστατος, βροχοπτώσεις κατά διαστήματα.	989
2;	B;	Αίθριος	1040				
3;	D;	Αίθριος μεταβαλλόμενος (Αίθριος, λιγότερο αμετάβλητος)	1024	17;	W;	Συχνές βροχοπτώσεις.	981
				18;	X;	Έντονα άστατος, βροχοπτώσεις.	974
4;	H;	Σχεδόν αίθριος, αργότερα πιθανώς ελαφρές βροχοπτώσεις.	1018	19;	Z;	Θυελλώδης, έντονες βροχοπτώσεις.	960
				20;	A;	Αίθριος αμετάβλητος	1030
5;	O;	Ελαφρές βροχοπτώσεις εν συνεχεία άστατος.	1010	21;	B;	Αίθριος	1022
				22;	C;	Μεταβολή σε αίθριο	1012
6;	R;	Άστατος, αργότερα βροχοπτώσεις.	1004	23;	F;	Σχεδόν αίθριος, σε βελτίωση	1007
7;	U;	Βροχοπτώσεις κατά διαστήματα, αργότερα επιδείνωση.	998	24;	G;	Σχεδόν αίθριος, γρήγορα πιθανώς ελαφρές βροχοπτώσεις.	1000
8;	V;	Βροχοπτώσεις κατά διαστήματα, εν συνέχεια έντονα άστατος.	991	25;	I;	Αρχικά ελαφρές βροχοπτώσεις, βελτίωση.	995
9;	X;	Έντονα άστατος, βροχοπτώσεις.	985	26;	J;	Μεταβλητός σε βελτίωση.	990
10;	A;	Αίθριος αμετάβλητος	1033	27;	L;	Σχεδόν άστατος, βελτίωση αργά στην ημέρα	984
11;	B;	Αίθριος	1023				
12;	E;	Αίθριος, πιθανές ελαφρές βροχοπτώσεις	1014	28;	M;	Άστατος, πιθανή βελτίωση	978
				29;	Q;	Άστατος, αίθριος κατά διαστήματα.	970
13;	K;	Σχεδόν αίθριος, πολύ πιθανόν ελαφρές βροχοπτώσεις	1008	30;	T;	Έντονα άστατος, αίθριος κατά διαστήματα.	965
14;	N;	Ελαφρές βροχοπτώσεις, με διαστήματα ηλιοφάνειας	1000	31;	Y;	Θυελλώδης, πιθανή βελτίωση.	959
				32;	Z;	Θυελλώδης, έντονες βροχοπτώσεις.	947
15;	P;	Μεταβλητός με μερικές βροχοπτώσεις.	994				

Πίνακας 12, Πρόγνωση του καιρού βάσει του αποτελέσματος της εξίσωσης Z

Για να κάνουμε χρήση του αλγορίθμου Zambretti, θα πρέπει να ακολουθήσουμε την εξής διαδικασία:

1. Υπολογισμός της βαρομετρικής τάσης.
2. Με βάση το αποτέλεσμα του προηγούμενου υπολογισμού, χρησιμοποιούμε την κατάλληλη συνάρτηση Zr, Zs, Zf.
3. Μέτρηση της τρέχουσας ατμοσφαιρικής πίεσης.
4. Υπολογισμός της εξίσωσης Zambretti.

6. Παραμετροποίηση Hardware και Software πριν την υλοποίηση

6.1 Παραμετροποίηση Arduino Mega ADK Rev3

Όπως έχουμε αναφέρει και προηγουμένως το Arduino αποτελεί έναν μικροελεγκτή ο οποίος απαρτίζεται από κάποιο αριθμό πυλών οι οποίες μπορούν να λειτουργούν είτε ως είσοδοι είτε ως έξοδοι ανάλογα με τις απαιτήσεις της εφαρμογής μας. Οι είσοδοι ή έξοδοι αυτοί διαχειρίζονται μέσω του κώδικα που αναπτύσσουμε σε γλώσσα c/c++ με την χρήση του προγραμματιστικού περιβάλλοντος IDE. Με άλλα λόγια δεν απαιτείται η εγκατάσταση κάποιου λογισμικού στο arduino για να καθορίσουμε την λειτουργία του, παρά μόνο η φόρτωση του κώδικα της προς υλοποίηση εφαρμογής καθώς και των κατάλληλων βιβλιοθηκών του hardware που θα χρησιμοποιήσουμε.

6.2 Παραμετροποίηση Raspberry Pi

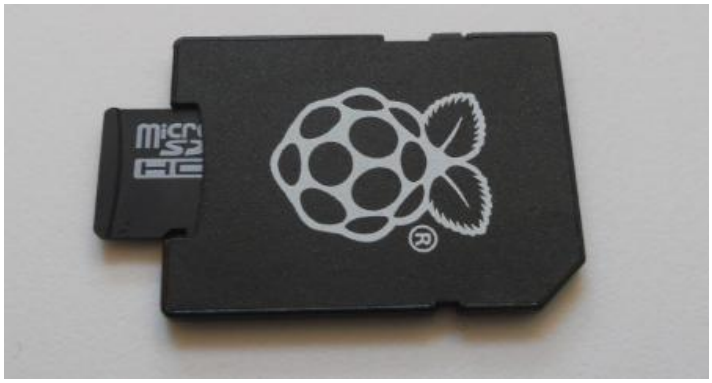
Από την άλλη πλευρά το Raspberry Pi αποτελεί έναν πλήρη υπολογιστή με επεξεργαστή τύπου ARM και επομένως για την υλοποίηση οποιασδήποτε εφαρμογής απαιτείται η εγκατάσταση κάποιας διανομής λειτουργικού συστήματος (εγκατάσταση

με το NOOBS) κατάλληλο για αυτό. Παρακάτω περιγράφεται η διαδικασία εγκατάστασης του λειτουργικού συστήματος rasbian.

→ Στην Εικόνα 30 φαίνονται συγκεντρωμένα όλα τα εξαρτήματα που χρειαζόμαστε για την εγκατάσταση του Raspberry Pi.

Εικόνα 30, απαραίτητα εξαρτήματα για την εγκατάσταση του Raspberry Pi

→ Για την εγκατάσταση του λειτουργικού θα χρησιμοποιήσουμε μια κάρτα Micro SD (Εικόνα31), η οποία θα πρέπει να έχει μέγεθος τουλάχιστον 4GB και να είναι κατ'ελάχιστο Class 4. Αυτές οι κάρτες, μαζί με τον αντάπτορα, είναι πολύ οικονομικές, και εφόσον οι προδιαγραφές είναι σωστές δεν αναμένονται συνήθως προβλήματα συμβατότητας.



Εικόνα 31, Κάρτα microSD

→ Η θύρα τροφοδοσίας για το Raspberry Pi είναι τύπου Micro USB, ίδια δηλαδή με των κινητών και επομένως θα χρησιμοποιήσουμε ένα φορτιστή κινητού τηλεφώνου (Εικόνα 32). Προσοχή όμως, αν χρειαστεί να συνδέσουμε στο Raspberry Pi κάποιον εξωτερικό σκληρό δίσκο με USB, ή γενικά οποιαδήποτε συσκευή έχει αυξημένες ανάγκες σε ρεύμα, θα πρέπει η τροφοδοσία να μπορεί να δώσει επαρκή Ampere. Το Raspberry Pi Model B+ και το Raspberry Pi 2 μπορούν να δώσουν maximum μέχρι 2 Ampere με την κατάλληλη τροφοδοσία. Συσκευές USB που χρειάζονται περισσότερα Ampere θα χρειαστούν επιπλέον εξωτερική τροφοδοσία για να λειτουργήσουν στο Raspberry Pi.

Εικόνα 32, Τροφοδοτικό για το Raspberry Pi

→ Ένα ακόμα απαραίτητο εξάρτημα είναι ένας Card Reader για την διαμόρφωση της κάρτας SD μέσω του υπολογιστή μας .

→ Τέλος, θα χρειαστούμε ένα καλώδιο τύπου HDMI-HDMI ή HDMI-DVI, καλώδιο Ethernet, καθώς και USB πληκτρολόγιο και ποντίκι (οι ενσύρματες λύσεις είναι συχνά οι καλύτερες από θέμα συμβατότητας).

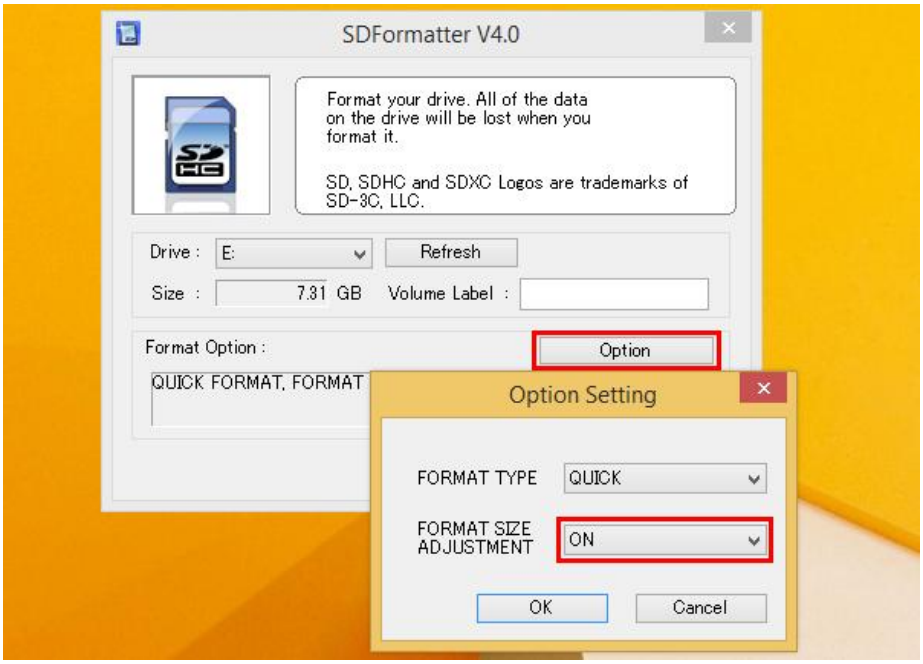
Εικόνα 33, σύνδεση Raspberry Pi με τροφοδοτικό και κάρτα microSD

6.2.1 Εγκατάσταση NOOBS στην κάρτα SD

Το NOOBS, από τα αρχικά της φράσης New Out Of Box Software - "καινούριο software, του κουτιού", σε ελεύθερη μετάφραση , είναι ένα σύστημα που διευκολύνει σημαντικά την αρχική εγκατάσταση Raspberry Pi.

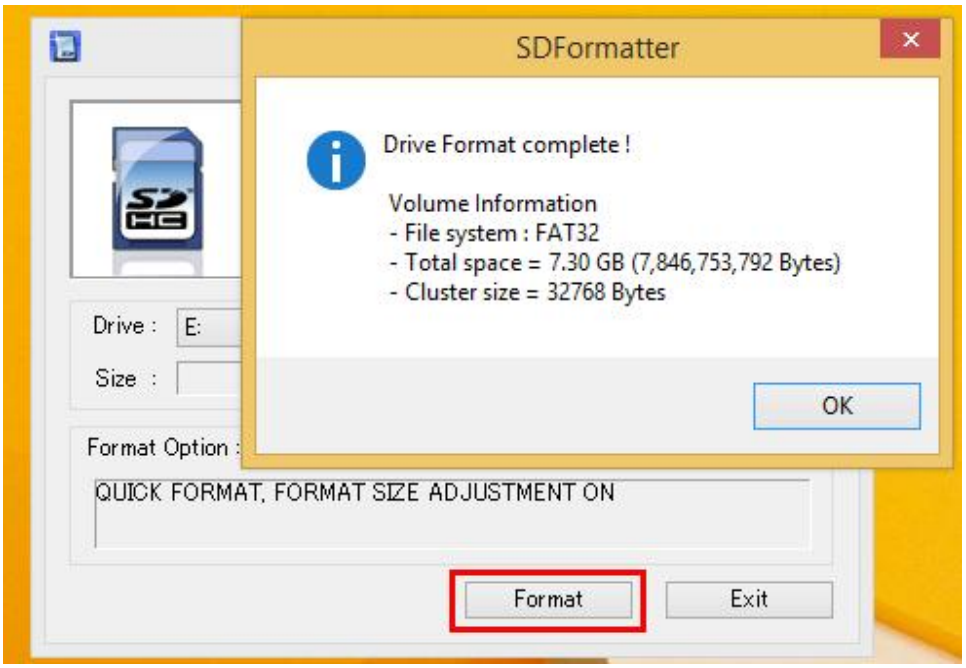
- Το πρώτο που χρειάζεται είναι να κάνουμε είναι να συνδέσουμε την κάρτα Micro SD με τον αντάπτορα στον υπολογιστή μας, και να τη διαμορφώσουμε κατάλληλα (Σύστημα FAT32).
- Έπειτα, κατεβάζουμε την εφαρμογή (από τον ιστότοπο www.raspberrypi.org/downloads) κάνοντας κλικ στο Accept (Εικόνα 34) στο κάτω μέρος της σελίδας, και στη συνέχεια αποσυμπιέζουμε το .zip. Στην συνέχεια, εγκαθιστούμε και τρέχουμε την εφαρμογή, και επιλέγουμε το "Option", στο οποίο αλλάζουμε το Format Size Adjustment σε "ON" (Εικόνα 35).

Εικόνα 34, Κατέβασμα εφαρμογής Raspberry Pi



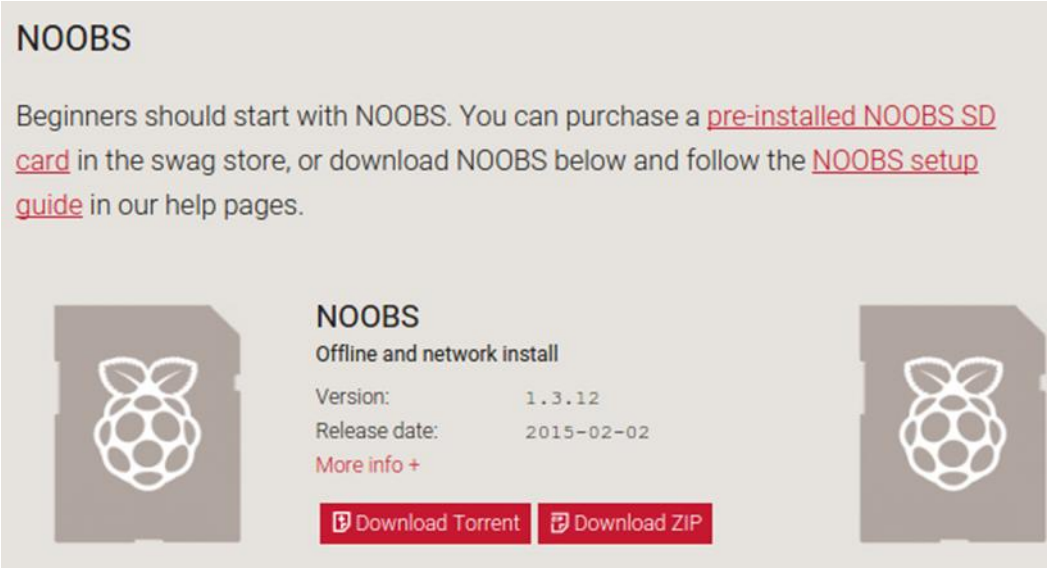
Εικόνα 35, Εγκατάσταση εφαρμογής Raspberry Pi

- Κάνοντας κλικ στο Format, σε λίγα δευτερόλεπτα έχει ολοκληρωθεί (Εικόνα 36).

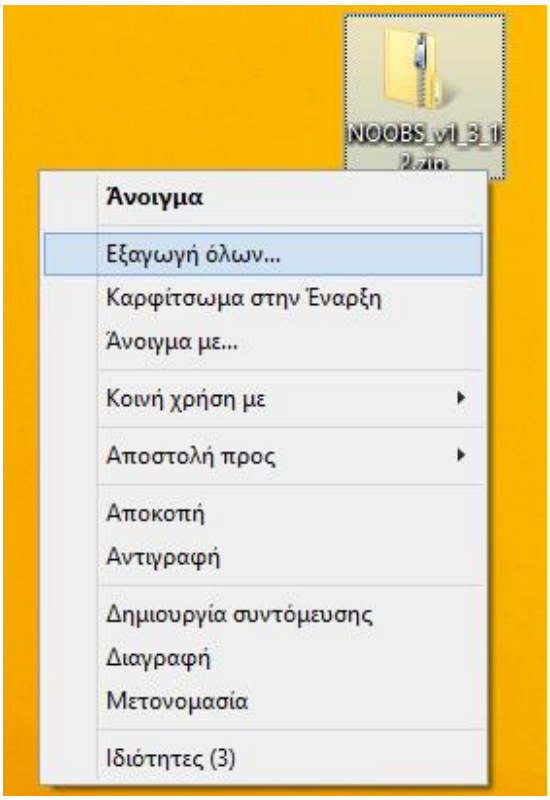


Εικόνα 36, Ολοκλήρωση εγκατάστασης του Raspberry Pi

- Με την κάρτα SD διαμορφωμένη, κατεβάζουμε την τελευταία έκδοση του NOOBS (Εικόνα 37), είτε σε .zip είτε μέσω Torrent. Αφού κατεβάσουμε το συμπιεσμένο φάκελο .zip, κάνουμε αποσυμπίεση σε όλα τα περιεχόμενά του (Εικόνα 38).

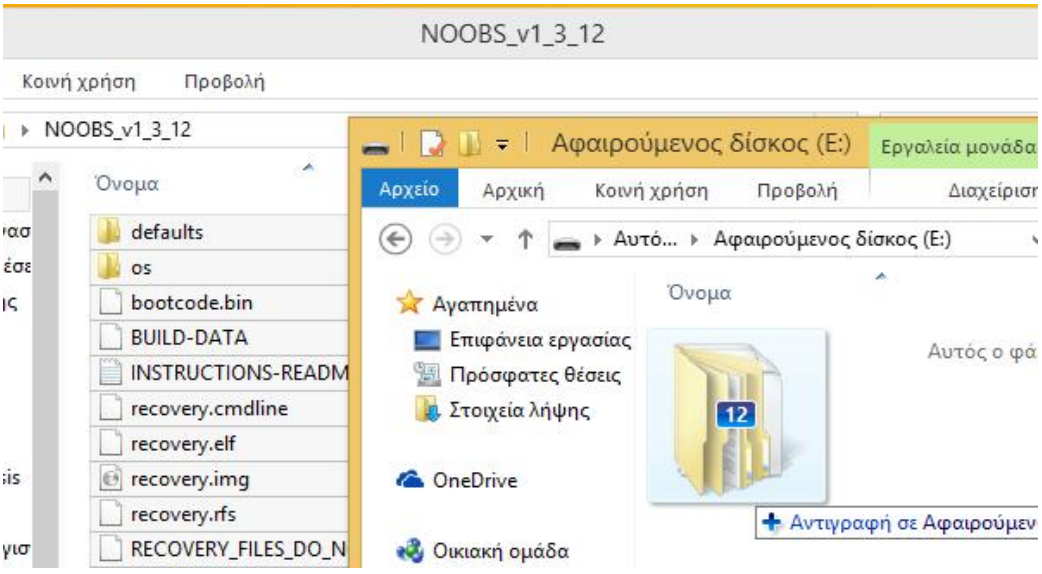


Εικόνα 37, Κατέβασμα εφαρμογής NOOBS

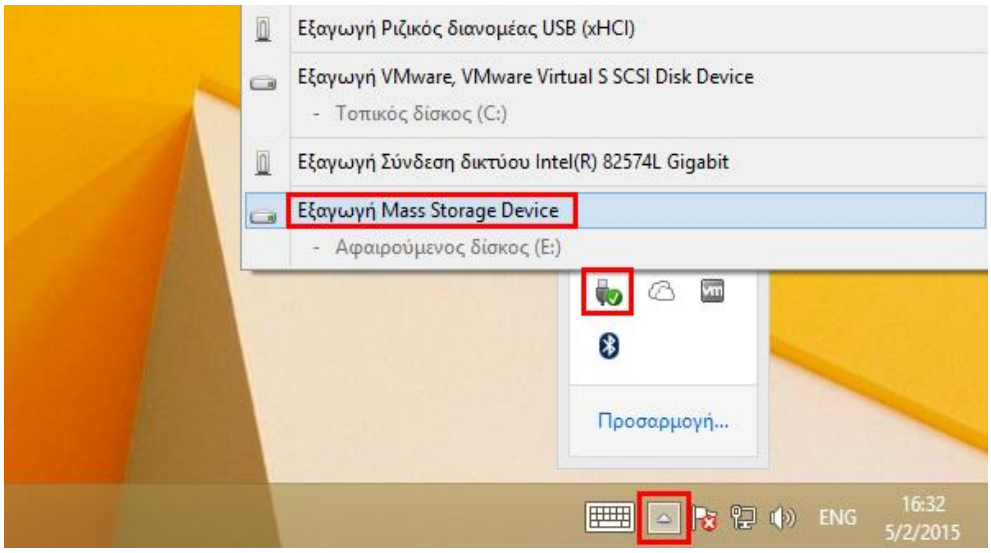


Εικόνα 38, Αποσυμπίεση περιεχομένων του NOOBS

- Ανοίγουμε το φάκελο στον οποίο έγινε η αποσυμπίεση, και τα αντιγράφουμε στην κάρτα SD (Εικόνα 39), και αφού ολοκληρωθεί η αντιγραφή, κάνουμε ασφαλή αφαίρεση της κάρτας SD (Εικόνα 40) .



Εικόνα 39, Αντιγραφή περιεχομένων στην κάρτα SD



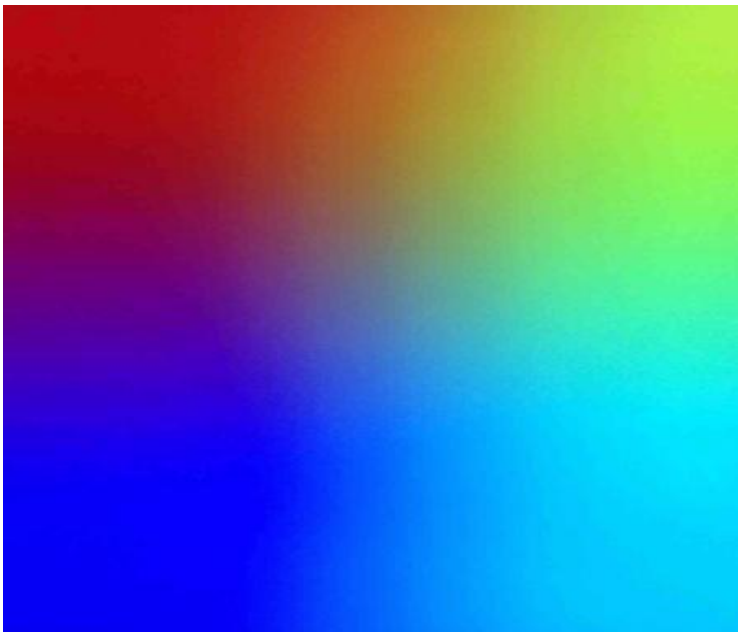
Εικόνα 40, Ασφαλής αφαίρεση της κάρτας SD

6.2.2 Εγκατάσταση Rasbperry Pi

- Αρχικά τοποθετούμε την microSD card στο Rasbperry Pi .Υπάρχει ένας μόνο τρόπος για να μπει, και μπαίνοντας θα "κλειδώσει". Για να αφαιρέσουμε την SD, την πιέζουμε προς τα μέσα, για να ξεκλειδώσει.
  Το Raspberry Pi δεν έχει διακόπτη On/Off. Αυτό σημαίνει πως μόλις το βάλουμε στην πρίζα, θα ξεκινήσει η λειτουργία του κατευθείαν. Γι' αυτό το λόγο τοποθετούμε πριν την τροφοδοσία το καλώδιο HDMI, ύστερα το Ethernet, και τέλος το πληκτρολόγιο και το ποντίκι (Εικόνα 41).

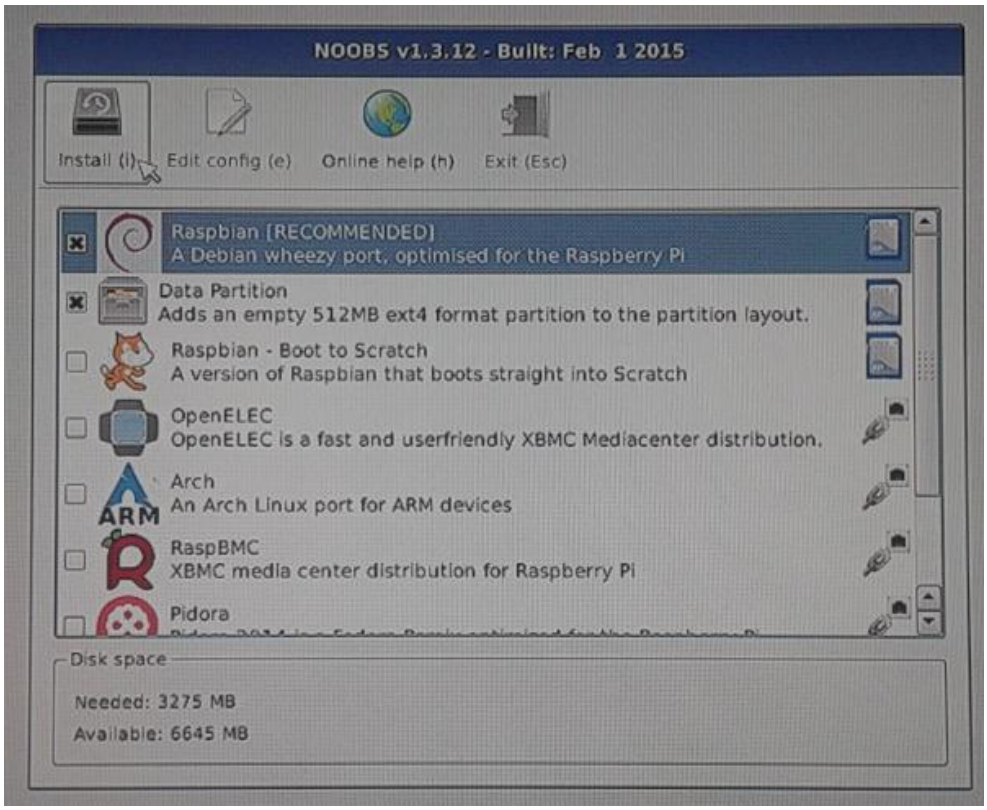
Εικόνα 41, Σύνδεση περιφερειακών στο Raspberry Pi

- Μόλις συνδέσουμε το Raspberry Pi με το τροφοδοτικό τότε αυτό εκκινεί και ξεκινώντας, θα μας δείξει μια οθόνη με διάφορα χρώματα, γνωστή και σαν rainbow screen (Εικόνα 42).



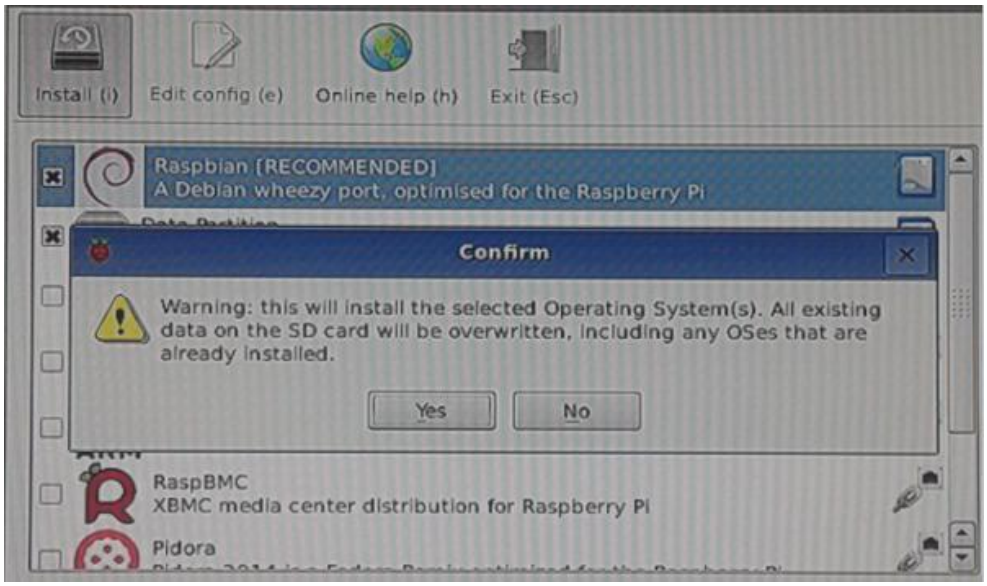
Εικόνα 42, Rainbow screen

- Σύντομα θα φορτώσει το NOOBS, το οποίο θα μας εμφανίσει μια λίστα με τα λειτουργικά συστήματα (Εικόνα 43).



Εικόνα 43, Τα λειτουργικά συστήματα του NOOBS

- Επιλέγουμε να εγκατασταθεί το Raspbian (Εικόνα 44), μια παραλλαγή του Debian για το Raspberry Pi, η οποία μπορεί δεν προϋποθέτει σύνδεση στο Internet. Οι περισσότερες από τις υπόλοιπες επιλογές (OpenELEC, Arch, RaspBMC κλπ) χρειάζονται σύνδεση στο ίντερνετ.



Εικόνα 44, Εγκατάσταση του Raspbian

- Έχοντας τσεκάρει το Raspbian και κάνοντας κλικ στο Install, το σύστημα μας προειδοποιεί πως θα διαγραφεί όλο το περιεχόμενο της SD. Επιλέγοντας "Yes", ξεκινάει η εγκατάσταση (Εικόνα 45).

Εικόνα 45, Εκκίνηση εγκατάστασης του Raspbian

- Εφόσον ολοκληρωθεί η διαδικασία της εγκατάστασης το σύστημα θα μας εμφανίσει το μήνυμα πως το λειτουργικό σύστημα (ή τα λειτουργικά συστήματα, αν επιλέξαμε πολλαπλά) εγκαταστάθηκαν επιτυχώς (Εικόνα 46).

Εικόνα 46, Επιτυχής εγκατάσταση των προγραμμάτων

- Τέλος κάνοντας κλικ στο OK, το Raspberry Pi θα κάνει επανεκκίνηση και θα είναι έτοιμο προς χρήση.

7. Ανάλυση και υλοποίηση του συστήματος DAQ - Αυτόνομου μετεωρολογικού σταθμού (ΑΜΣ)

Στην παρούσα διπλωματική εργασία επιλέξαμε να υλοποιήσουμε δύο εκδοχές ενός συστήματος DAQ - Αυτόνομου μετεωρολογικού σταθμού (ΑΜΣ). Στην πρώτη εκδοχή, το σύστημα DAQ - ΑΜΣ υλοποιείται μόνο με τη χρήση μικροελεγκτή (του Arduino Mega ADK Rev3), ενώ στη δεύτερη εκδοχή γίνεται επέκταση του συστήματος με την προσθήκη ενός μικροϋπολογιστή (του Raspberry Pi Model B).

7.1 Ανάλυση και υλοποίηση του συστήματος DAQ - ΑΜΣ με τη χρήση Arduino Mega ADK Rev3

7.1.1 Ανάλυση του συστήματος DAQ - ΑΜΣ

Το σύστημα DAQ - ΑΜΣ αποτελείται από:

- Arduino Mega ADK Rev3
- Arduino Ethernet Shield
- Αισθητήρας θερμοκρασίας / Υγρασίας (RHT 03)
- Αισθητήρας ατμοσφαιρικής πίεσης / Θερμοκρασίας (MPL 115A2)
- Breadboard και καλώδια (για τη σύνδεση των παραπάνω επιμέρους στοιχείων)

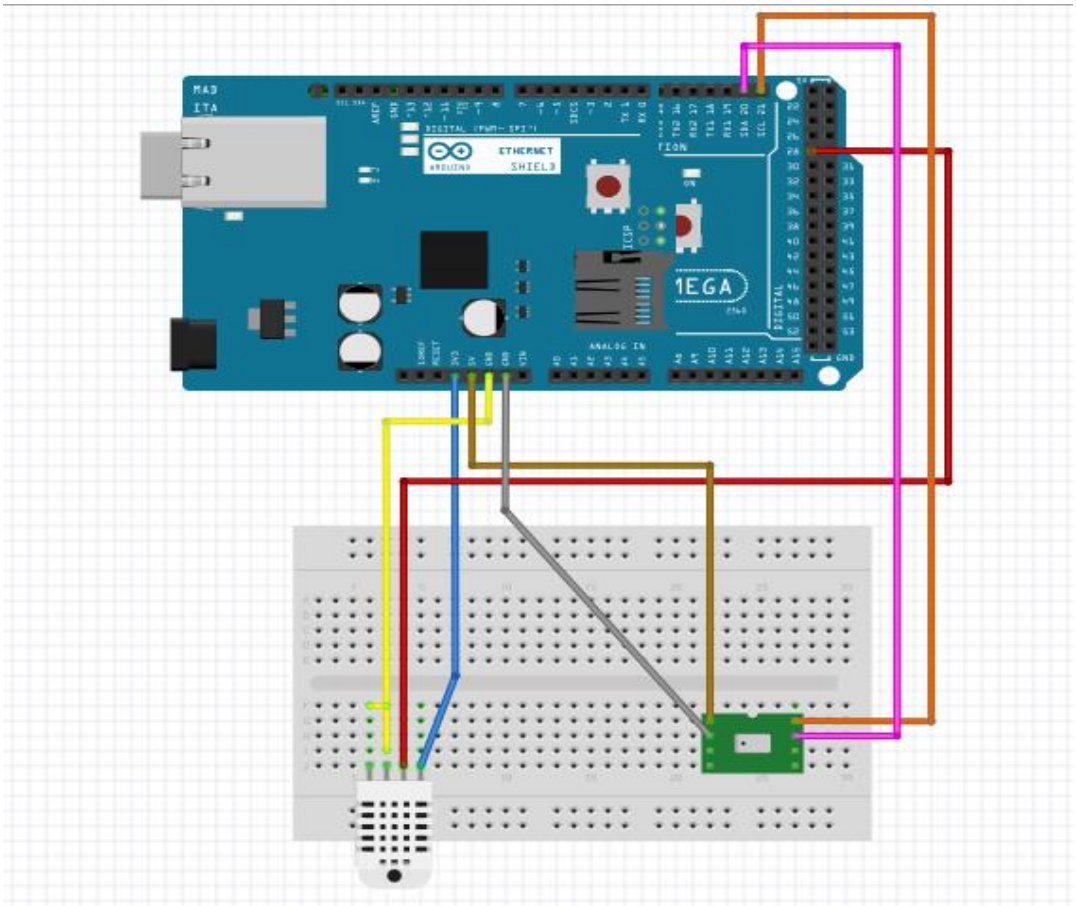
Σκοπός ανάπτυξης του συστήματος DAQ - ΑΜΣ είναι η παρακολούθηση και η καταγραφή της θερμοκρασίας, της υγρασίας και της ατμοσφαιρικής πίεσης, καθώς και η αποθήκευση για τη μετέπειτα επεξεργασία τους. Πιο συγκεκριμένα η

λειτουργία του συστήματος DAQ - ΑΜΣ με τη χρήση του Arduino Mega ADK Rev3 είναι η παρακάτω:

Αρχικά θα συλλέγονται οι μετρήσεις της θερμοκρασίας, της υγρασίας και της ατμοσφαιρικής πίεσης, μέσω των αισθητηρίων RHT 03 και MPL 115A2. Στη συνέχεια οι μετρήσεις θα αποθηκεύονται στην κάρτα SD του Ethernet Shield, αναγράφοντας ταυτόχρονα την ημερομηνία και την ώρα που ελήφθησαν. Επιπρόσθετα το Ethernet Shield θα χρησιμοποιείται ως Web Server, μέσω του οποίου θα είναι διαθέσιμα τα δεδομένα από απομακρυσμένη περιοχή, με μόνη προϋπόθεση τη σύνδεση στο Internet. Συμπεραίνουμε λοιπόν ότι υπάρχουν δύο τρόποι επεξεργασίας/εκμετάλευσης των δεδομένων, είτε απομακρυσμένα μέσω του διαδικτύου, είτε τοπικά μέσω της κάρτας SD. Η λειτουργία του Αυτόνομου Μετεωρολογικού Σταθμού ολοκληρώνεται χρησιμοποιώντας τον κώδικα Zambretti, για την εξαγωγή των προβλέψεων των αντίστοιχων τιμών θερμοκρασίας, υγρασίας και ατμοσφαιρικής πίεσης.

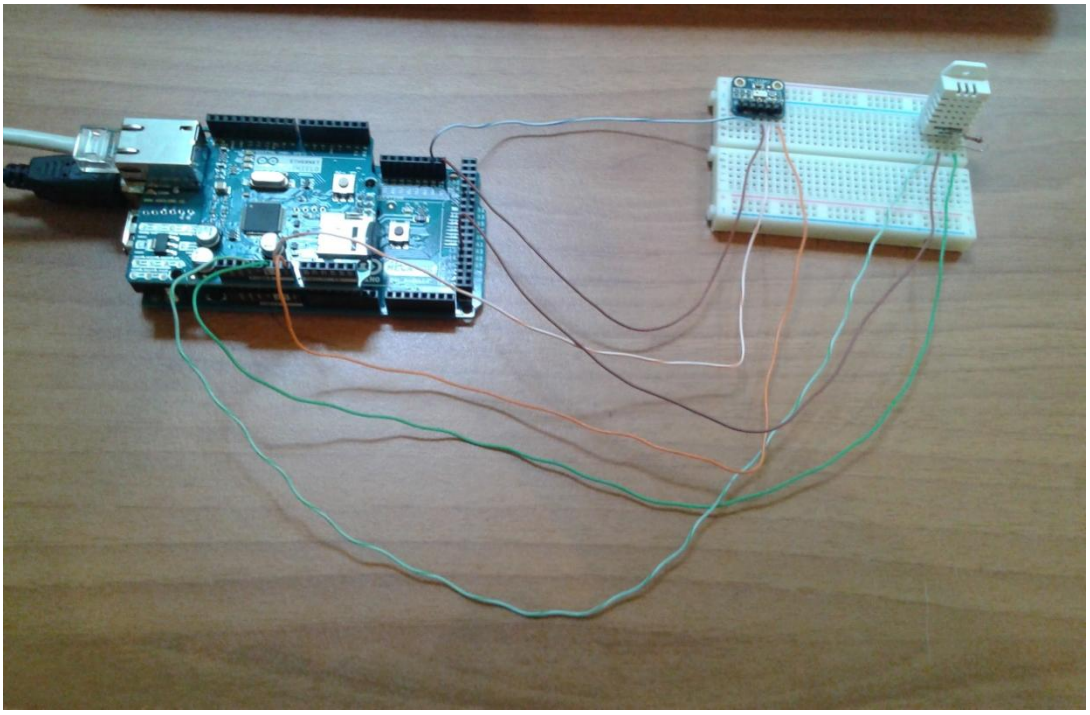
7.1.2 Υλοποίηση του συστήματος DAQ - ΑΜΣ

Η σύνδεση και υλοποίηση του συστήματος DAQ - ΑΜΣ φαίνεται στην εικόνα 47 , και αποτελεί μια προσομοίωση, χρησιμοποιώντας το πρόγραμμα ανοιχτού κώδικα Fritzing.

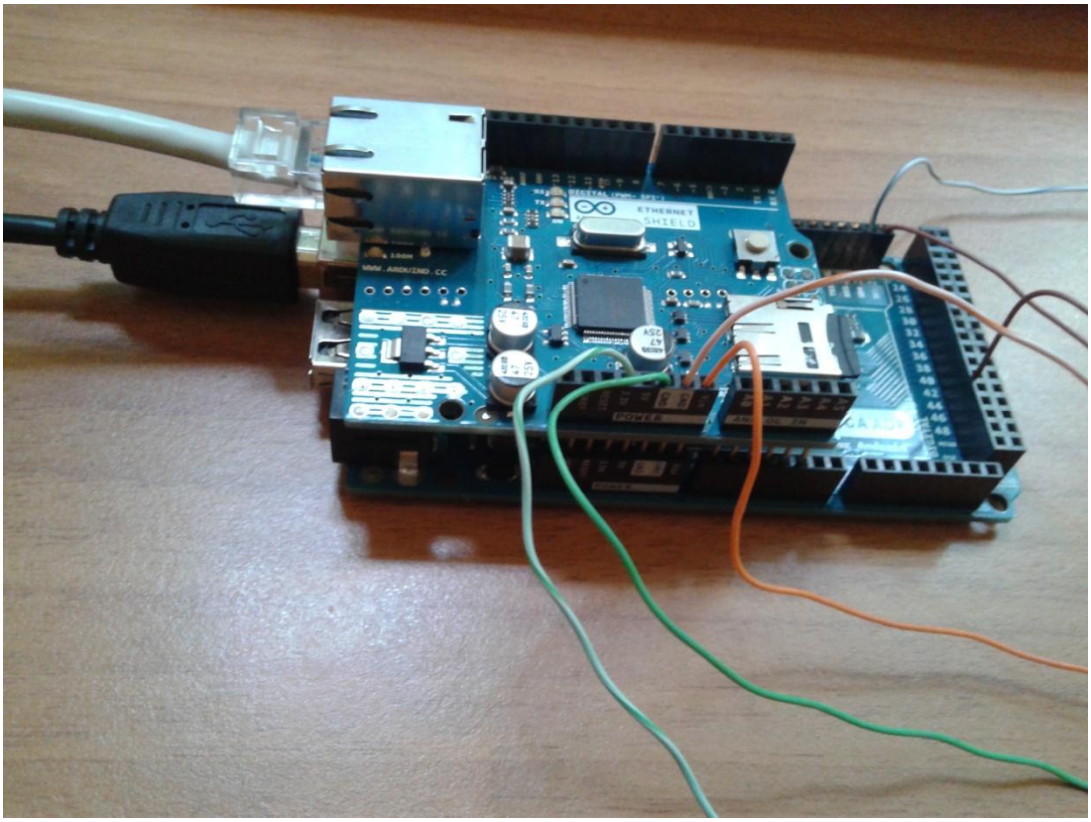


Εικόνα 47, Σύστημα DAQ - ΑΜΣ

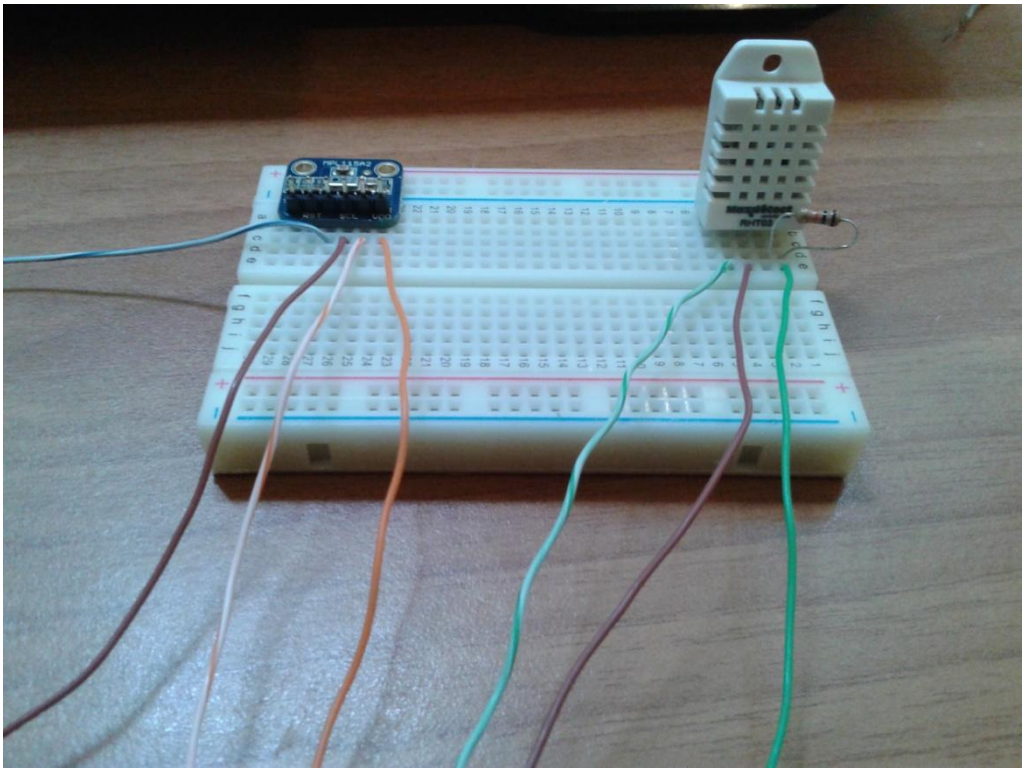
Στη συνέχεια παρουσιάζεται στις εικόνες 48α, 48β και 48γ η συνδεσμολογία που πραγματοποιήσαμε για την υλοποίηση του Αυτόνομου Μετεωρολογικού Σταθμού (Α.Μ.Σ.)



Εικόνα 48α, Η συνδεσμολογία του ΑΜΣ



Εικόνα 48β, Η συνδεσμολογία του ΑΜΣ-Arduino και Ethernet shield



Εικόνα 48γ, Η συνδεσμολογία του ΑΜΣ- Τα αισθητήρια

7.1.3 Δομή και υλοποίηση του κώδικα ΑΜΣ – DAQ

7.1.3.1 Δομή του κώδικα ΑΜΣ – DAQ

Παρακάτω παρουσιάζονται οι κώδικες που αναπτύχθηκαν για την υλοποίηση του συστήματος ΑΜΣ – DAQ. Οι κώδικες αυτοί φορτώνονται, μέσω του προγράμματος IDE, στο Arduino, χρησιμοποιώντας τη σειριακή επικοινωνία μεταξύ του ίδιου του Arduino και του υπολογιστή. Στο παράρτημα 1 γίνεται αναλυτική περιγραφή και επεξήγηση για τους κώδικες που χρησιμοποιήθηκαν.

<u>Κώδικας για τον αισθητήρα RHT 03</u>

```
#include <SPI.h>
#include <SD.h>
#include <Ethernet.h>
#include <Time.h>
#include <EthernetUdp.h>
#include <pin.h>
#include <dht.h>

dht DHT;
byte mac[] = {0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED};
IPAddress ip(192, 168, 2, 4);
```

```
EthernetServer server(80);
IPAddress timeServer(132, 163, 4, 101);
const int chipSelect = 4;
const int dhtPin = 42;
int chk;
float hum;
float temp;
const int timeZone = 3;
EthernetUDP Udp;
unsigned int localPort = 8888;

void setup()
{
  Serial.begin(9600);
if (Ethernet.begin(mac) == 0) {
while (1) {
 Serial.println("Failed to configure Ethernet using DHCP");
 delay(10000);
  }
}
 Ethernet.begin(mac, ip);
 server.begin();
 Serial.print("server is at ");
 Serial.println(Ethernet.localIP());
 Serial.print("IP number assigned by DHCP is ");
 Serial.println(Ethernet.localIP());
 Udp.begin(localPort);
 Serial.println("waiting for sync");
 setSyncProvider(getNtpTime);
 Serial.print("Initializing SD card...");
 pinMode(53, OUTPUT);

  if (!SD.begin(chipSelect)) {
    Serial.println("Card failed, or not present");
    return;
  }
 Serial.println("card initialized.");
}
time_t prevDisplay = 0;
void loop()
{
if (timeStatus() != timeNotSet) {
if (now() != prevDisplay) {
  prevDisplay = now();
```

```
  digitalClockDisplay();
  }
 }

chk = DHT.read22(dhtPin);
hum = DHT.humidity;
temp= DHT.temperature;
Serial.print("Humidity: ");
Serial.print(hum);
Serial.print(" %, Temp: ");
Serial.print(temp);
Serial.println(" Celsius");
delay(6000);
 String dataString = "";
 String dataString_2 = "";

 for (int digitalPin = 42; digitalPin < 43; digitalPin++) {
int sensor = digitalRead(digitalPin);
dataString += String(temp);
dataString_2 += String(hum);
if (digitalPin < 44) {
  dataString += ",";
  }
  EthernetClient client = server.available();
 if (client) {
Serial.println("new client");
boolean currentLineIsBlank = true;
while (client.connected()) {
if (client.available()) {
  char c = client.read();
  Serial.write(c);

 if (c == '\n' && currentLineIsBlank) {
    client.println("HTTP/1.1 200 OK");
    client.println("Content-Type: text/html");
    client.println("Connection: close");
    client.println("Refresh: 5");
    client.println();
    client.println("<!DOCTYPE HTML>");
    client.println("<html>");

    for (int digitalPin = 42; digitalPin < 43; digitalPin++) {
      int sensor = digitalRead(digitalPin);
      client.print("temperature: ");
```

```
      client.print(temp);
      client.println(" Celsius");
      client.print(" ");
      client.print("humidity: ");
      client.print(hum);
      client.print(" % ");
      client.print("  ");
      client.print(" ");
      client.print(" ");
      client.print("time:");
      client.print(hour());
      client.print(":");
      client.print(minute());
      client.print(":");
      client.print(second());
      client.print(" ");
      client.print(" ");
      client.print("date:");
      client.print(" ");
      client.print(day());
      client.print("/");
      client.print(" ");
      client.print(month());
      client.print("/");
      client.print(" ");
      client.print(year());
      client.println();

     client.println("<br />");
    }
    client.println("</html>");
    break;
  }
 if (c == '\n') {
    currentLineIsBlank = true;
  }
 else if (c != '\r') {
  currentLineIsBlank = false;
  }
}
}
delay(100);
client.stop();
Serial.println("client disconnected");
```

```
 }
 }
 File dataFile = SD.open("datalog.txt", FILE_WRITE);

 if (dataFile) {
dataFile.print("temp:"),dataFile.print(dataString),
dataFile.print("hum:"), dataFile.print(dataString_2);
dataFile.print(" ");
dataFile.print("time: ");
 dataFile.print(hour());
  dataFile.print(":");
 dataFile.print(minute());
  dataFile.print(":");
 dataFile.print(second());
 dataFile.print(" ");
  dataFile.print("date:");
 dataFile.print(day());
 dataFile.print("/");
 dataFile.print(month());
 dataFile.print("/");
 dataFile.print(year());
 dataFile.println();
dataFile.close();
Serial.println("temp:"),Serial.println(dataString);
Serial.println("hum:"), Serial.println(dataString_2);
 }
 else {
Serial.println("error opening datalog.txt");
 }
}
void digitalClockDisplay(){
 Serial.print(hour());
 printDigits(minute());
 printDigits(second());
 Serial.print(" ");
 Serial.print(day());
 Serial.print(" ");
 Serial.print(month());
 Serial.print(" ");
 Serial.print(year());
 Serial.println();
}
void printDigits(int digits){
 Serial.print(":");
```

```
 if(digits < 10)
Serial.print('0');
 Serial.print(digits);
}

const int NTP_PACKET_SIZE = 48;
byte packetBuffer[NTP_PACKET_SIZE];

time_t getNtpTime()
{
 while (Udp.parsePacket() > 0) ;
 Serial.println("Transmit NTP Request");
 sendNTPpacket(timeServer);
 uint32_t beginWait = millis();
 while (millis() - beginWait < 1500) {
int size = Udp.parsePacket();
if (size >= NTP_PACKET_SIZE) {
Serial.println("Receive NTP Response");
Udp.read(packetBuffer, NTP_PACKET_SIZE);
 secsSince1900 =  (unsigned long)packetBuffer[40] << 24;
 secsSince1900 |= (unsigned long)packetBuffer[41] << 16;
 secsSince1900 |= (unsigned long)packetBuffer[42] << 8;
 secsSince1900 |= (unsigned long)packetBuffer[43];
 return secsSince1900 - 2208988800UL + timeZone * SECS_PER_HOUR;
                                    }
 }
 Serial.println("No NTP Response ");
 return 0; }

void sendNTPpacket(IPAddress &address)
{
 memset(packetBuffer, 0, NTP_PACKET_SIZE);
 packetBuffer[0] = 0b11100011;
 packetBuffer[1] = 0;
 packetBuffer[2] = 6;
 packetBuffer[3] = 0xEC;
 packetBuffer[12]  = 49;
 packetBuffer[13]  = 0x4E;
 packetBuffer[14]  = 49;
 packetBuffer[15]  = 52;
 Udp.beginPacket(address, 123);
 Udp.write(packetBuffer, NTP_PACKET_SIZE);
 Udp.endPacket();
}
```

Κώδικας για τον αισθητήρα MPL 115A2

```
#include <SPI.h>
#include <SD.h>
#include <Ethernet.h>
#include <Wire.h>
#include <Time.h>
#include <EthernetUdp.h>
#include <pin.h>
#include <Adafruit_MPL115A2.h>
Adafruit_MPL115A2 mpl115a2;


byte mac[] = {0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED};
IPAddress ip(192, 168, 2, 4);
EthernetServer server(80);
IPAddress timeServer(132, 163, 4, 101); const int chipSelect = 4;

int chk;

const int timeZone = 3;
EthernetUDP Udp;
unsigned int localPort = 8888;
void setup()
{
  Serial.begin(9600);
 Serial.println("Getting barometric pressure ...");
 mpl115a2.begin();
if (Ethernet.begin(mac) == 0) {
    while (1) { Serial.println("Failed to configure Ethernet using DHCP");
    delay(10000);
  }
 }
Ethernet.begin(mac, ip);
server.begin();
 Serial.print("server is at ");
 Serial.println(Ethernet.localIP());
 Serial.print("IP number assigned by DHCP is ");
 Serial.println(Ethernet.localIP());
 Udp.begin(localPort);
 Serial.println("waiting for sync");
 setSyncProvider(getNtpTime);
 Serial.print("Initializing SD card...");
 pinMode(53, OUTPUT);
 if (!SD.begin(chipSelect)) {
        Serial.println("Card failed, or not present");
        return;
 }
 Serial.println("card initialized.");
}
```

```
time_t prevDisplay = 0;
void loop()
{
 float pressureKPA = 0, temperatureC = 0;
if (timeStatus() != timeNotSet) {
  if (now() != prevDisplay) {
     prevDisplay = now();
     digitalClockDisplay();
   }
 }

mpl115a2.getPT(&pressureKPA,&temperatureC);
Serial.print("Pressure (kPa): "); Serial.print(pressureKPA, 4); Serial.print(" kPa  ");
Serial.print("Temp (*C): "); Serial.print(temperatureC, 1); Serial.println(" *C both
measured together");
pressureKPA = mpl115a2.getPressure();
Serial.print("Pressure (kPa): "); Serial.print(pressureKPA, 4); Serial.println(" kPa");

temperatureC = mpl115a2.getTemperature();
Serial.print("Temp (*C): "); Serial.print(temperatureC, 1); Serial.println(" *C");


Serial.print("Pressure: ");
Serial.print(pressureKPA);
Serial.print(" kPa , Temp: ");
Serial.print(temperatureC);
Serial.println(" Celsius");
delay(6000); //Delay 2 sec.
EthernetClient client = server.available();
 if (client) {
          Serial.println("new client");
          boolean currentLineIsBlank = true;
          while (client.connected()) {
          if (client.available()) {
          char c = client.read();
          Serial.write(c);

if (c == '\n' && currentLineIsBlank) {
client.println("HTTP/1.1 200 OK");
client.println("Content-Type: text/html");
client.println("Connection: close");
client.println("Refresh: 5");
client.println();
client.println("<!DOCTYPE HTML>");
client.println("<html>");
client.print("temperature: ");
client.print(temperatureC);
client.println(" Celsius");
client.print(" ");
client.print("pressure: ");
```

```
client.print(pressureKPA);
client.print(" kPa ");
client.print("  ");
client.print(" ");
client.print(" ");
client.print("time:");
client.print(hour());
client.print(":");
client.print(minute());
client.print(":");
client.print(second());
client.print(" ");
client.print(" ");
client.print("date:");
client.print(" ");
client.print(day());
client.print("/");
client.print(" ");
client.print(month());
client.print("/");
client.print(" ");
client.print(year());
client.println();
client.println("<br />");
client.println("</html>");
    break;
 }
if (c == '\n') { currentLineIsBlank = true;}
  else if (c != '\r') {
    currentLineIsBlank = false;
  }
}
}
delay(100);
client.stop();
Serial.println("client disconnected");
 }
File dataFile = SD.open("datalog.txt", FILE_WRITE);
if (dataFile) {
dataFile.print("temp:"),dataFile.print(temperatureC),
dataFile.print("press:"), dataFile.print(pressureKPA);
dataFile.print(" ");
dataFile.print("time: ");
dataFile.print(hour());
dataFile.print(":");
dataFile.print(minute());
dataFile.print(":");
dataFile.print(second());
dataFile.print(" ");
dataFile.print("date:");
```

```
dataFile.print(day());
dataFile.print("/");
dataFile.print(month());
dataFile.print("/");
dataFile.print(year());
dataFile.println();
dataFile.close();
Serial.println("temp:"),Serial.println(temperatureC);
Serial.println("press:"), Serial.println(pressureKPA);
 }
 else {
   Serial.println("error opening datalog.txt");
 }
}
void digitalClockDisplay(){
Serial.print(hour());
printDigits(minute());
printDigits(second());
Serial.print(" ");
Serial.print(day());
Serial.print(" ");
Serial.print(month());
Serial.print(" ");
Serial.print(year());
Serial.println();
}

void printDigits(int digits){
   Serial.print(":");
 if(digits < 10)
  Serial.print('0');
   Serial.print(digits);
}
const int NTP_PACKET_SIZE = 48;
byte packetBuffer[NTP_PACKET_SIZE];
time_t getNtpTime()
{
 while (Udp.parsePacket() > 0) ;
 Serial.println("Transmit NTP Request");
 sendNTPpacket(timeServer);
 uint32_t beginWait = millis();
 while (millis() - beginWait < 1500) {
int size = Udp.parsePacket();
if (size >= NTP_PACKET_SIZE) {
Serial.println("Receive NTP Response");
Udp.read(packetBuffer, NTP_PACKET_SIZE);
unsigned long secsSince1900;
secsSince1900 =  (unsigned long)packetBuffer[40] << 24;
secsSince1900 |= (unsigned long)packetBuffer[41] << 16;
secsSince1900 |= (unsigned long)packetBuffer[42] << 8;
```

```
 secsSince1900 |= (unsigned long)packetBuffer[43];
 return secsSince1900 - 2208988800UL + timeZone * SECS_PER_HOUR;
        }
 }
 Serial.println("No NTP Response :-(");
 return 0;
}
void sendNTPpacket(IPAddress &address)
{
 memset(packetBuffer, 0, NTP_PACKET_SIZE);
 packetBuffer[0] = 0b11100011;
 packetBuffer[1] = 0;
 packetBuffer[2] = 6;
 packetBuffer[3] = 0xEC;
 packetBuffer[12]  = 49;
 packetBuffer[13]  = 0x4E;
 packetBuffer[14]  = 49;
 packetBuffer[15]  = 52;
 Udp.beginPacket(address, 123);
 Udp.write(packetBuffer, NTP_PACKET_SIZE);
 Udp.endPacket();
}
```

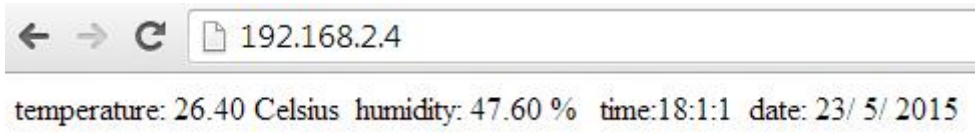
7.1.3.2 Υλοποίηση του κώδικα ΑΜΣ – DAQ

Αφού προγραμματίσαμε το Arduino Mega ADK Rev3, συνδέσαμε τους αισθητήρες και στη συνέχεια πραγματοποιήσαμε λήψη των μετρήσεων.

Λήψη μετρήσεων για το αισθητήριο RHT 03

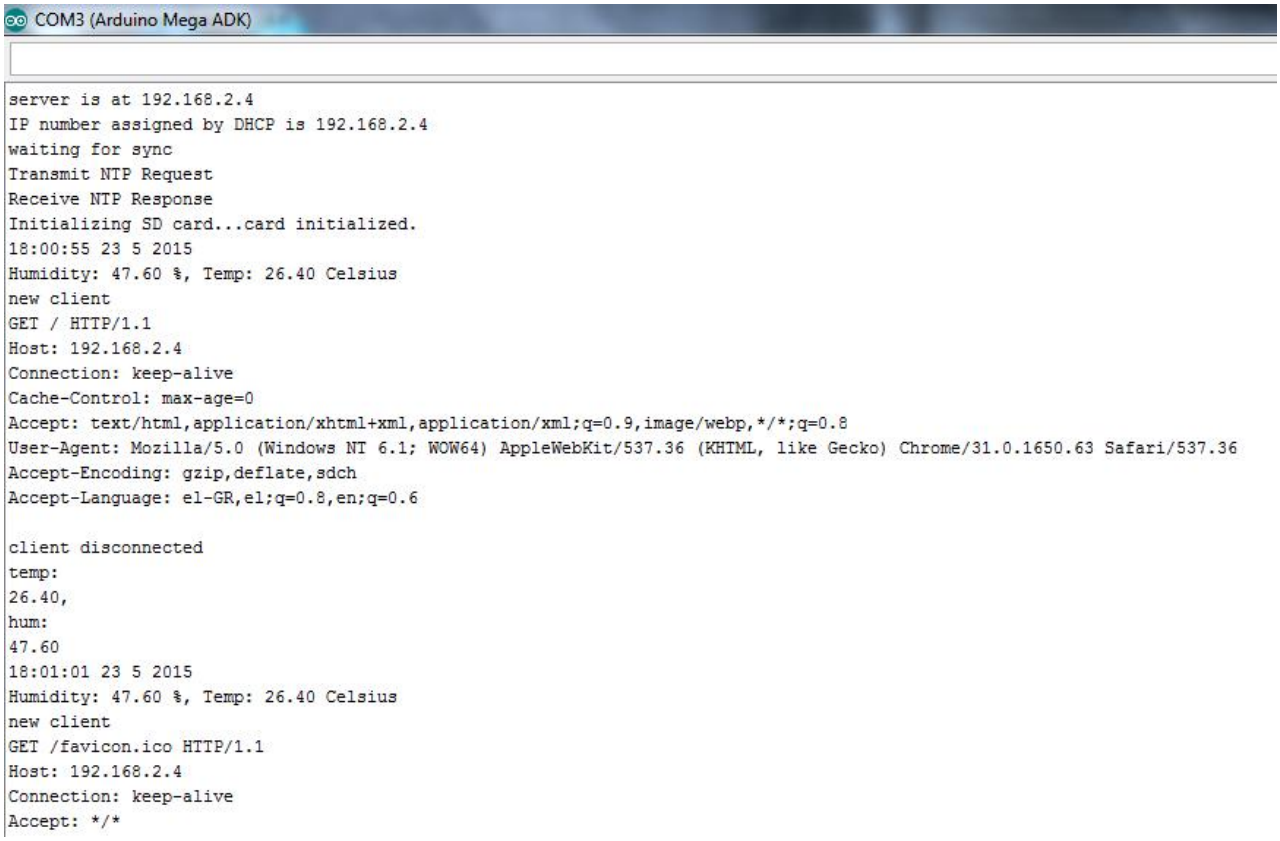
Στις εικόνες 49α, 49β και 49γ παρουσιάζονται τα αποτελέσματα της λήψης των μετρήσεων για το αισθητήριο μέτρησης θερμοκρασίας και υγρασίας.

Στην εικόνα 49α παρουσιάζεται το στιγμιότυπο από την απομακρυσμένη πρόσβαση μέσω του διαδικτύου (με πρόγραμμα περιήγησης) στις μετρήσεις.



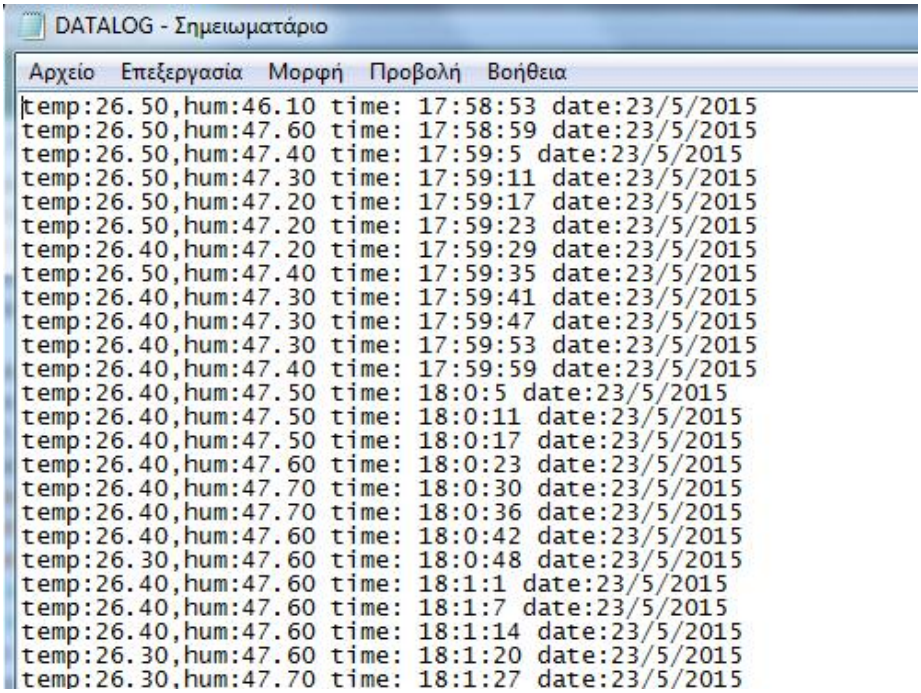
Εικόνα 49α, Τα αποτελέσματα της λήψης των μετρήσεων στην σελίδα του περιηγητή διαδικτύου

Στην εικόνα 49β παρουσιάζεται το στιγμιότυπο από τη σειριακή οθόνη του προγραμματιστικού περιβάλλοντος IDE του Arduino.



Εικόνα 49β, Τα αποτελέσματα της λήψης των μετρήσεων για το αισθητήριο μέτρησης θερμοκρασίας και υγρασίας στην σειριακή οθόνη IDE

Στην εικόνα 49γ παρουσιάζεται το στιγμιότυπο από την αποθήκευση των μετρήσεων στην κάρτα SD του Ethernet Shield.

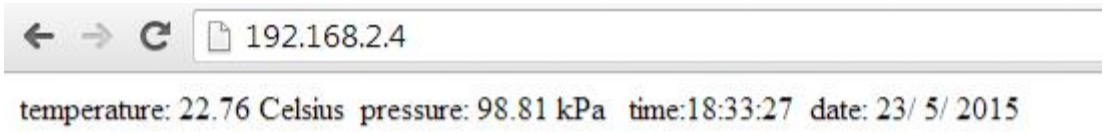


Εικόνα 49γ, Τα αποτελέσματα της αποθήκευσης των μετρήσεων στην κάρτα SD

Λήψη μετρήσεων για το αισθητήριο MPL 115A2

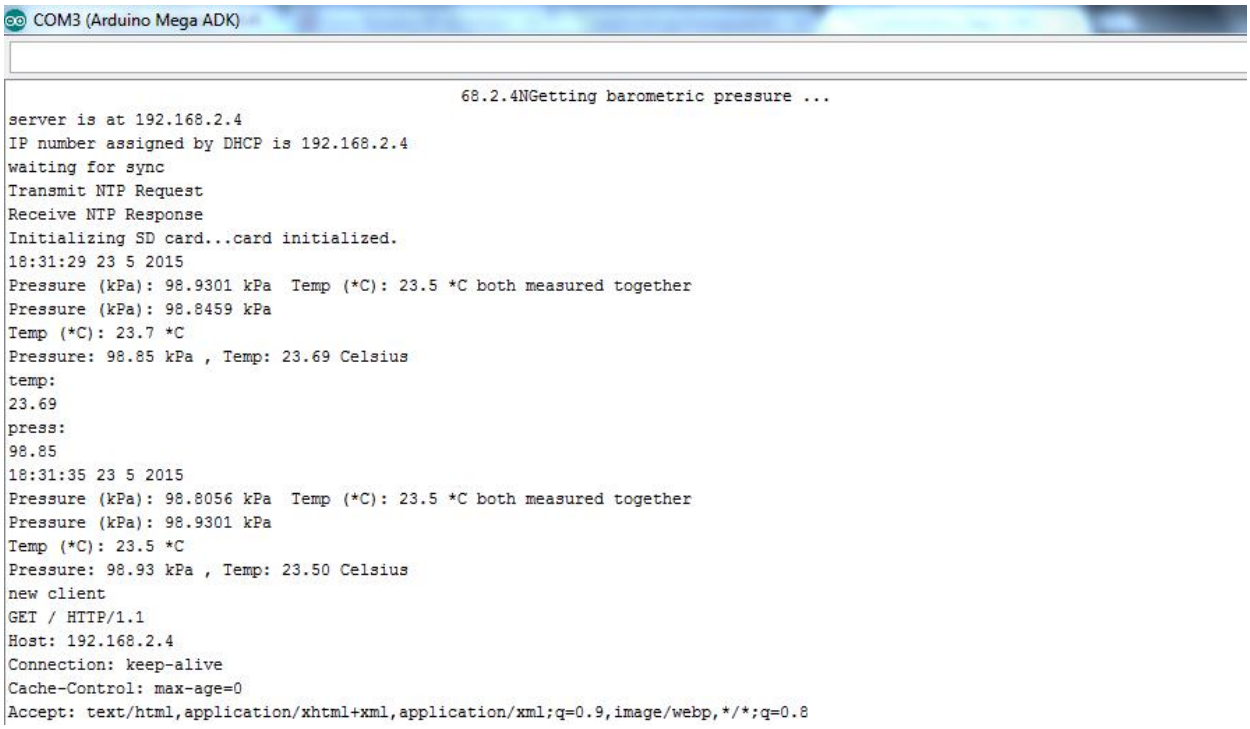
Στις εικόνες 50α, 50β και 50γ παρουσιάζονται τα αποτελέσματα της λήψης των μετρήσεων για το αισθητήριο μέτρησης θερμοκρασίας και ατμοσφαιρικής πίεσης.

Στην εικόνα 50α παρουσιάζεται το στιγμιότυπο από την απομακρυσμένη πρόσβαση μέσω του διαδικτύου (με πρόγραμμα περιήγησης) στις μετρήσεις.



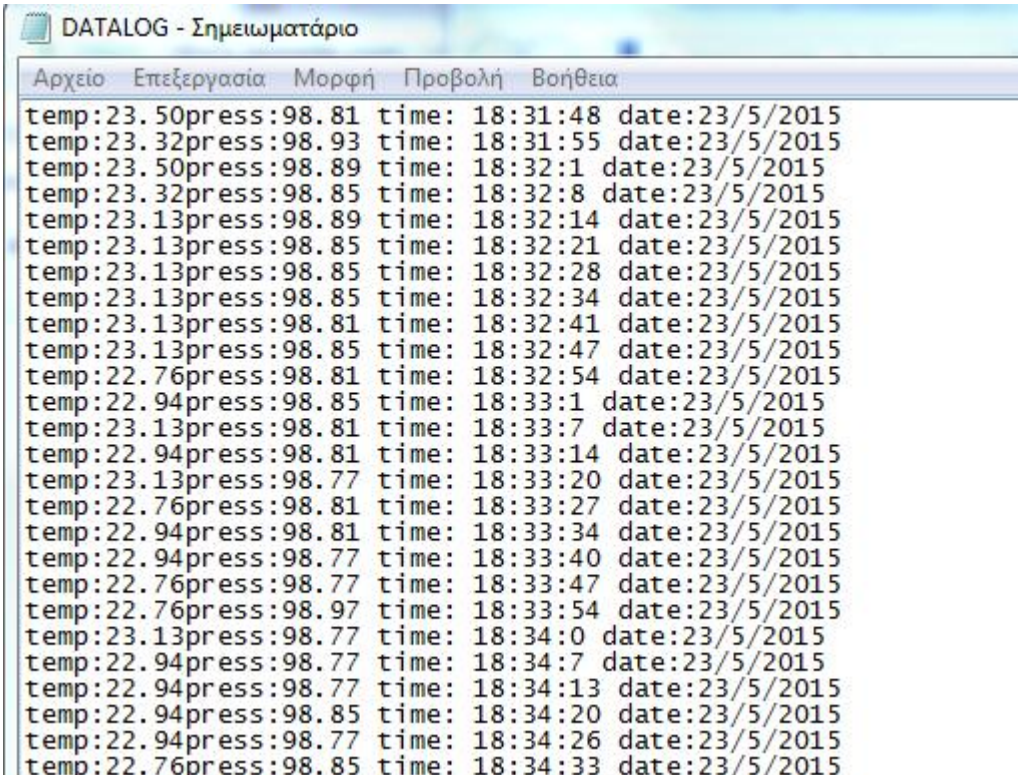
Εικόνα 50α, Τα αποτελέσματα της λήψης των μετρήσεων στην σελίδα του περιηγητή διαδικτύου

Στην εικόνα 50β παρουσιάζεται το στιγμιότυπο από τη σειριακή οθόνη του προγραμματιστικού περιβάλλοντος IDE του Arduino.



Εικόνα 50β, Τα αποτελέσματα της λήψης των μετρήσεων στην σειριακή οθόνη του προγράμματος IDE

Στην εικόνα 50γ παρουσιάζεται το στιγμιότυπο από την αποθήκευση των μετρήσεων στην κάρτα SD του Ethernet Shield.



Εικόνα 50γ, Τα αποτελέσματα της αποθήκευσης των μετρήσεων στην κάρτα SD

7.1.4 Συλλογή και επεξεργασία δεδομένων

Στην παρούσα ενότητα θα γίνει η παρουσίαση των αποτελεσμάτων των μετρήσεων, αλλά και η επεξεργασία αυτών. Αφού συλλέξαμε και ταξινομήσαμε σε πίνακες τα δεδομένα από τα αισθητήρια, στη συνέχεια δημιουργήσαμε και παρουσιάζουμε τις γραφικές παραστάσεις αυτών, για κάθε ημέρα μετρήσεων.

7.1.4.1 Συλλογή δεδομένων από τις μετρήσεις

Στους πίνακες 13α και 13β, που ακολουθούν, παρουσιάζονται τα αποτελέσματα των μετρήσεων από τους αισθητήρες, για κάθε μια από τις 20 μέρες λήψης αυτών.

Ημερομηνία	Μέγιστη τιμή θερμοκρασίας (ºC)	Μέγιστη τιμή υγρασίας (%)	Μέγιστη τιμή ατμοσφαιρικής πίεσης (hPa)
1/4/2015	21,9	30	1008,6
2/4/2015	20,3	48	1008,4
3/4/2015	20,4	66	1009,5
4/4/2015	15,5	62	1010,7
5/4/2015	16,7	34	1010,4
6/4/2015	20,6	77	1009,2
7/4/2015	18,4	86	1009
8/4/2015	12,9	64	1009,7
9/4/2015	13,1	85	1008,7
10/4/2015	16,7	68	1006,5
11/4/2015	21,7	34	1007,4
12/4/2015	22,9	81	1007,1
13/4/2015	23,8	80	1008,2
14/4/2015	23,9	73	1007,9
15/4/2015	22,3	72	1008,7
16/4/2015	21,6	83	1007,1
17/4/2015	22,1	74	1010,4
18/4/2015	24,6	80	1009,2
19/4/2015	25,1	69	1007,9
20/4/2015	20,7	57	1006,6

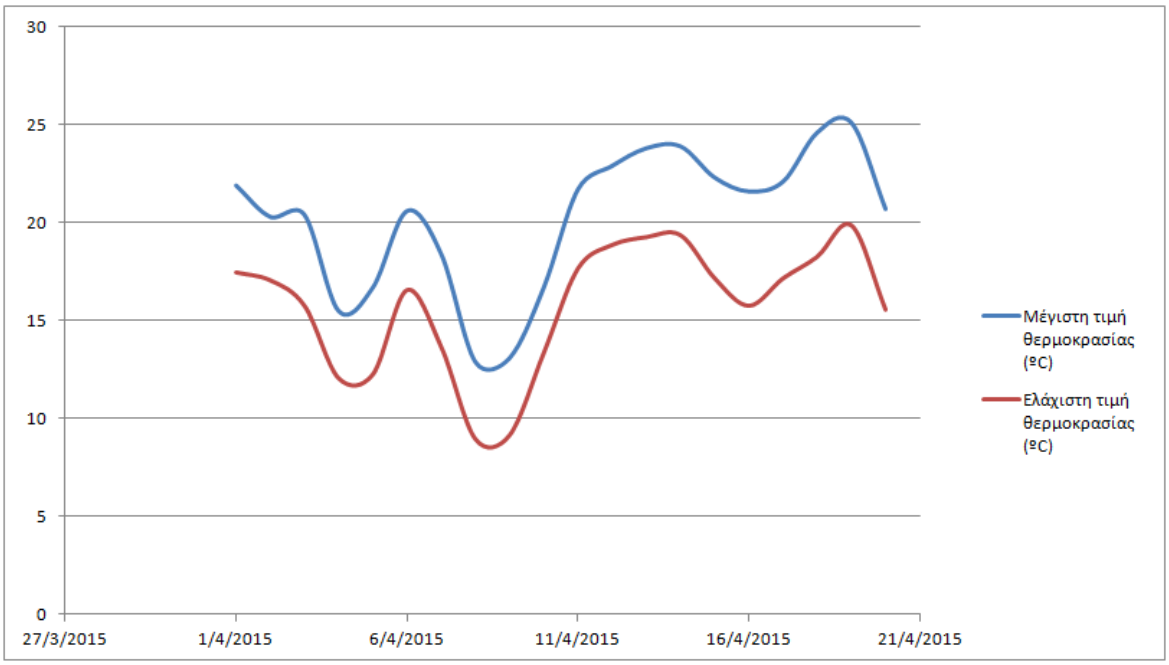
Πίνακας 13α, Μέγιστες τιμές των μετρήσεων

Ημερομηνία	Ελάχιστη τιμή θερμοκρασίας (ºC)	Ελάχιστη τιμή υγρασίας (%)	Ελάχιστη τιμή ατμοσφαιρικής πίεσης (hPa)
1/4/2015	17,5	27	1002,3
2/4/2015	17,1	38	1002,1
3/4/2015	15,8	60	1004,6
4/4/2015	12,1	58	1004,5
5/4/2015	12,3	28	1004,1
6/4/2015	16,6	70	1004,5
7/4/2015	13,7	75	1003,9
8/4/2015	9	29	1004,4
9/4/2015	9,2	52	1001,2
10/4/2015	13,4	33	1000,7
11/4/2015	17,7	27	1001,5
12/4/2015	18,9	49	1001,3
13/4/2015	19,3	51	1003,2
14/4/2015	19,4	46	1002,1
15/4/2015	17,2	43	1003
16/4/2015	15,8	55	1000,5
17/4/2015	17,2	39	1004,3
18/4/2015	18,3	47	1003,9
19/4/2015	19,9	31	1002,4
20/4/2015	15,6	24	1000,4

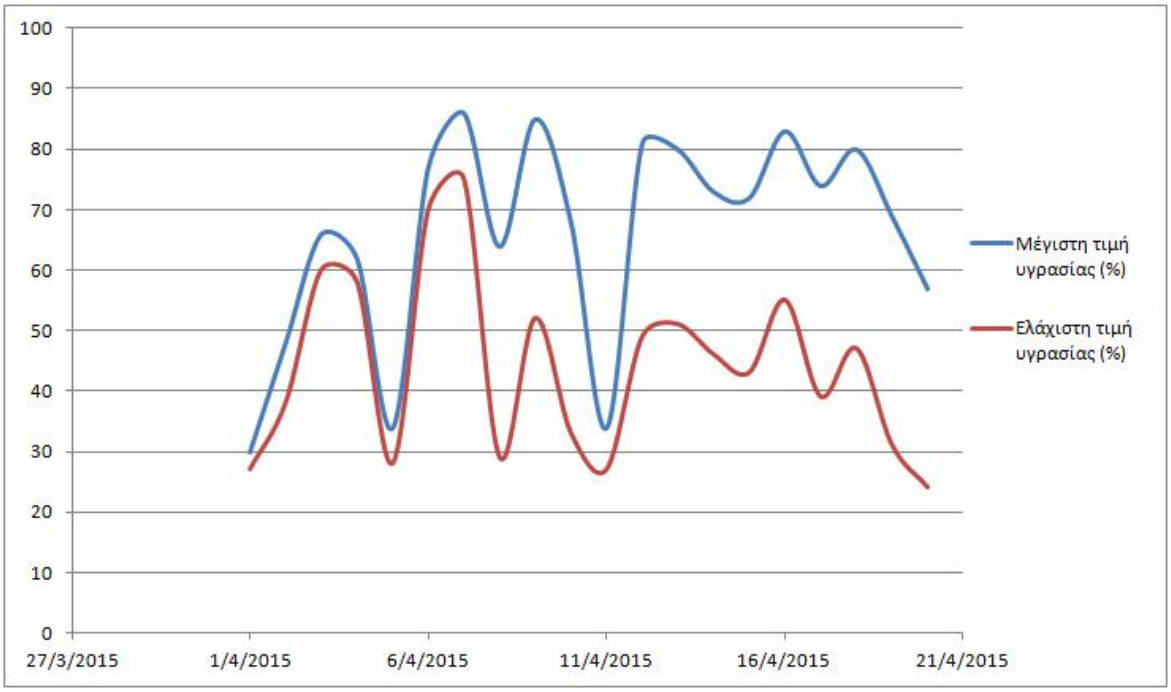
Πίνακας 13β , Ελάχιστες τιμές των μετρήσεων

7.1.4.2. Επεξεργασία των ληφθέντων δεδομένων από τις μετρήσεις

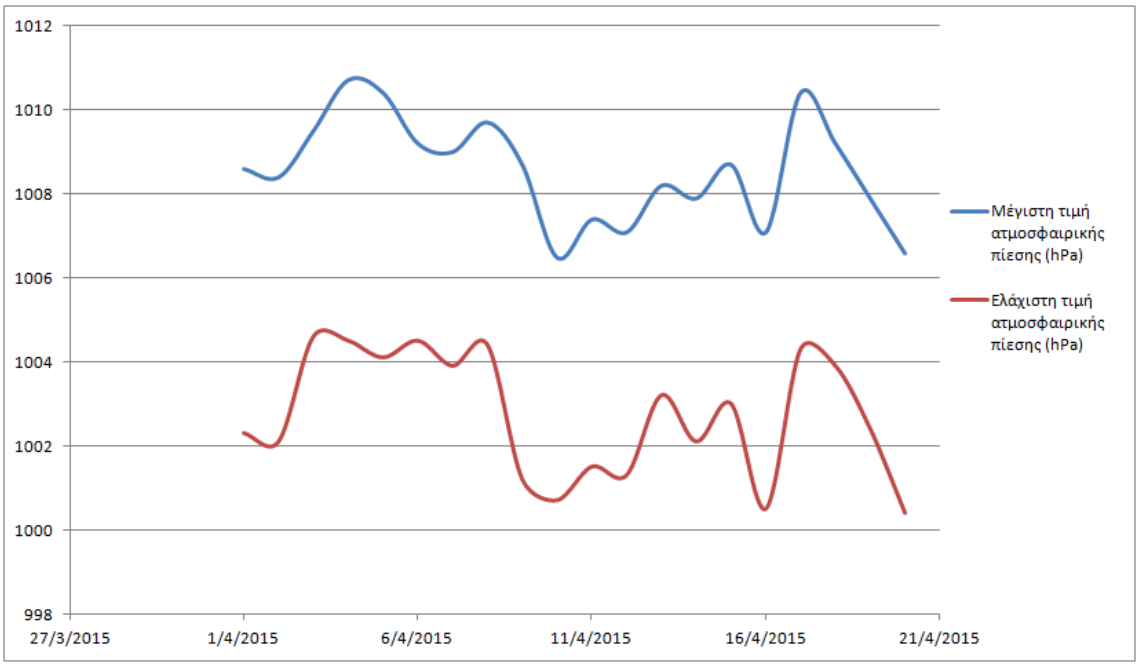
Αρχικά, στο γράφημα 5α γίνεται η παρουσίαση των μετρήσεων της μέγιστης και ελάχιστης τιμής της θερμοκρασίας, για κάθε ημέρα λήψης των μετρήσεων. Ομοίως, στο γράφημα 5β γίνεται η παρουσίαση των μετρήσεων της υγρασίας και στο γράφημα 5γ παρουσιάζονται οι μετρήσεις για την ατμοσφαιρική πίεση.



Γράφημα 5α, Μέγιστη και ελάχιστη τιμή της θερμοκρασίας



Γράφημα 5β, Μέγιστη και ελάχιστη τιμή της υγρασίας



Γράφημα 5γ, Μέγιστη και ελάχιστη τιμή της ατμοσφαιρικής πίεσης

7.1.5 Υλοποίηση απλοποιημένης εκδοχής του κώδικα Zambretti

Στο εξής κεφάλαιο πραγματοποιήσαμε την υλοποίηση μια απλοποιημένης εκδοχής του κώδικα Ζαμπρέττι, για την πρόγνωση του καιρού, βάσει της μεταβολής των τιμών της ατμοσφαιρικής πίεσης. Πιο συγκεκριμένα, κατασκευάσαμε έναν κώδικα, με τον οποίο το Arduino Mega λαμβάνει μετρήσεις από το αισθητήριο MPL 115A2 και τις αποθηκεύει σε έναν πίνακα. Στη συνέχεια χρησιμοποιεί την τρέχουσα τιμή της ατμοσφαιρικής πίεσης, καθώς και την τιμή αυτής 3 ώρες πρίν, για τον προσδιορισμό της κατάλληλης εξίσωσης z (Zr, για αυξανόμενη βαρομετρική τάση, Zs, για σταθερή βαρομετρική τάση και Zf, για μειούμενη βαρομετρική τάση). Τέλος, χρησιμοποιώντας την τρέχουσα αμοσφαιρική πίεση, πραγματοποιεί τον υπολογισμό της εξίσωσης z, και βάσει αυτού του αποτελέσματος εμφανίζει ένα μήνυμα για την πρόγνωση του καιρού. Ο κώδικας επεξηγείτε στο παράρτημα 1. Στην εικόνα 51 φαίνεται το αποτέλεσμα αυτού του κώδικα, με τις μετρήσεις του αισθητηρίου και το μήνυμα που έχει εμφανιστεί για την πρόγνωση του καιρού.

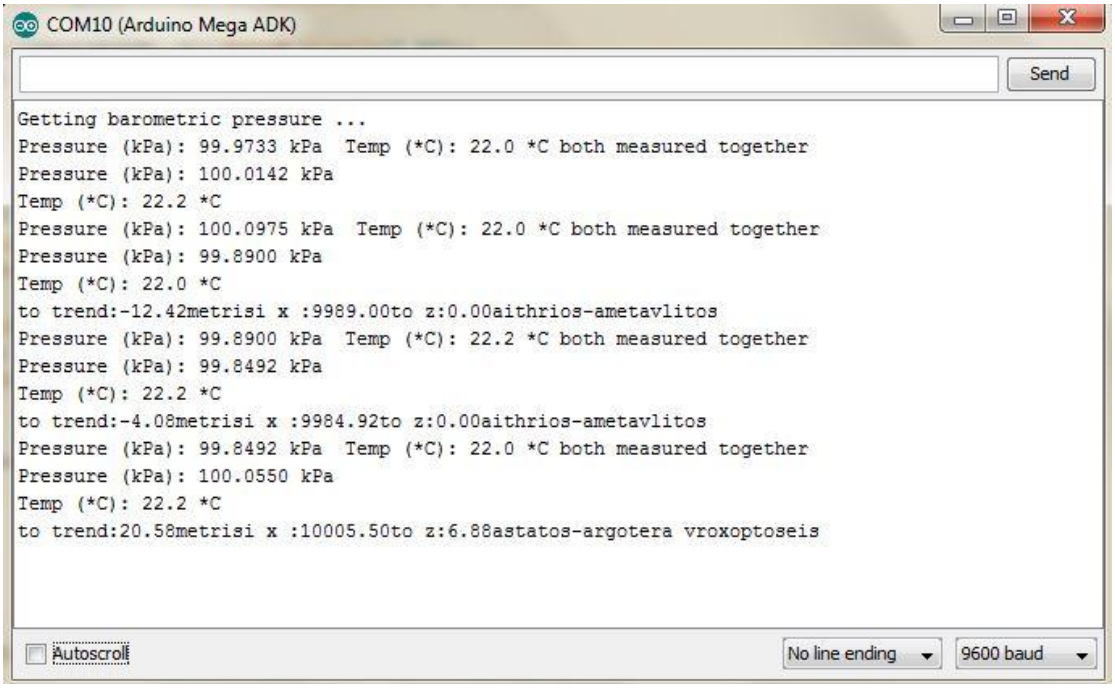
Κώδικας Zambretti:

```
#include <Wire.h>
#include <Adafruit_MPL115A2.h>

Adafruit_MPL115A2 mpl115a2;

void setup(void)
{
```

```
 Serial.begin(9600);
 float z=0;
 float trend=0;
 int x=0;
 float metriseis[100], diafores[100];
 Serial.println("Getting barometric pressure ...");
 mpl115a2.begin();
}

void loop(void)
{
 float pressureKPA = 0, temperatureC = 0;
  float z=0;
  float trend=0;
  int x;
  float metriseis[100], diafores[100],trexousa;
 mpl115a2.getPT(&pressureKPA,&temperatureC);
 Serial.print("Pressure (kPa): "); Serial.print(pressureKPA, 4); Serial.print(" kPa  ");
 Serial.print("Temp (*C): "); Serial.print(temperatureC, 1); Serial.println(" *C both measured
together");
 pressureKPA = mpl115a2.getPressure();
 Serial.print("Pressure (kPa): "); Serial.print(pressureKPA, 4); Serial.println(" kPa");
 temperatureC = mpl115a2.getTemperature();
 Serial.print("Temp (*C): "); Serial.print(temperatureC, 1); Serial.println(" *C");
 metriseis[1]=pressureKPA*100; //metatropi se hpa

for(x=2; x<=8; x=x+1)
  {
    delay (180000);
 mpl115a2.getPT(&pressureKPA,&temperatureC);
 Serial.print("Pressure (kPa): "); Serial.print(pressureKPA, 4); Serial.print(" kPa  ");
 Serial.print("Temp (*C): "); Serial.print(temperatureC, 1); Serial.println(" *C both measured
together");
 pressureKPA = mpl115a2.getPressure();
 Serial.print("Pressure (kPa): "); Serial.print(pressureKPA, 4); Serial.println(" kPa");
 temperatureC = mpl115a2.getTemperature();
 Serial.print("Temp (*C): "); Serial.print(temperatureC, 1); Serial.println(" *C");

  metriseis[x]=pressureKPA*100;
  diafores[x-1]=(metriseis[x]-metriseis[x-1]);
  trend=diafores[x-1];
  Serial.print("to trend:");
  Serial.print(trend);
  Serial.print("metrisi x :");
  Serial.print(metriseis[x]);
```

```
  mpl115a2.getPT(&pressureKPA,&temperatureC);
  pressureKPA = mpl115a2.getPressure();
  trexousa=pressureKPA*100;
    if(trend >0)
       z=(130-((trexousa)/81.0));
    else if ( trend = 0 )
       z=(147-((5*(trexousa))/376.0));
    else if (trend <0)
        z=(179-((2*(trexousa))/129.0));
    Serial.print("to z:");
    Serial.print(z);
if (z>=0.0 && z<2.0)
    Serial.print("aithrios-ametavlitos\n");
else if (z>=2.0 && z<3.0)
    Serial.print("aithrios\n");
else if (z>=3.0 && z<4.0)
    Serial.print("aithrios-metavallomenos\n");
else if (z>=4.0 && z<5.0)
    Serial.print("sxedon aithrios-argotera pithanon elafres vroxoptoseis\n");
else if (z>=5.0 && z<6.0)
    Serial.print("elafres vroxoptoseis-en sunexeia astatos\n");
else if (z>=6.0 && z<7.0)
    Serial.print("astatos-argotera vroxoptoseis\n");
else if (z>=7.0 && z<8.0)
    Serial.print("vroxoptoseis kata diastimata- argotera epidinosi\n");
else if (z>=8.0 && z<9.0)
    Serial.print("vroxoptoseis ana diastimata- en sunexeia edona astatos\n");
else if (z>=9.0 && z<10.0)
    Serial.print("edona astatos- vroxoptoseis\n");
else if (z>=10.0 && z<11.0)
    Serial.print("aithrios-ametavlitos\n");
else if (z>=11.0 && z<12.0)
    Serial.print("aithrios\n");
else if (z>=12.0 && z<13.0)
    Serial.print("aithrios- pithanes elafres vroxoptoseis\n");
else if (z>=13.0 && z<14.0)
    Serial.print("sxedon aithrios- polu pithanon elafres vroxoptoseis\n");
else if (z>=14.0 && z<15.0)
    Serial.print("elafres vroxoptoseis me diastimata iliofaneias\n");
else if (z>=15.0 && z<16.0)
    Serial.print("metavlitos me merikes vroxoptoseis\n");
else if (z>=16.0 && z<17.0)
    Serial.print("astatos-vroxoptoseis ana diastimata\n");
else if (z>=17.0 && z<18.0)
```

```
   Serial.print("suxnes vroxoptoseis\n");
else if (z>=18.0 && z<19.0)
   Serial.print("endona astatos- vroxoptoseis\n");
else if (z>=19.0 && z<20.0)
   Serial.print("thielodeis edones vroxoptoseis\n");
else if (z>=20.0 && z<21.0)
   Serial.print("aithrios- ametavlitos\n");
else if (z>=21.0 && z<22.0)
   Serial.print("aithrios\n");
else if (z>=22.0 && z<23.0)
   Serial.print("metavoli se aithrio\n");
else if (z>=23.0 && z<24.0)
   Serial.print("sxedon aithrios- se veltiosi\n");
else if (z>=24.0 && z<25.0)
   Serial.print("sxedon aithrios- grigora pithanes elafres vroxoptoseis\n");
else if (z>=25.0 && z<26.0)
   Serial.print("arxika elafres vroxoptoseis- veltiosi\n");
else if (z>=26.0 && z<27.0)
   Serial.print("metavlitos- se veltiosi\n");
else if (z>=27.0 && z<28.0)
   Serial.print("sxedon astatos- veltiosi stin imera");
else if (z>=28.0 && z<29.0)
   Serial.print("astatos- pithani veltiosi\n");
else if (z>=29.0 && z<30.0)
   Serial.print("astatos- aithrios ana diastimata\n");
else if (z>=30.0 && z<31.0)
   Serial.print("endona astatos- aithrios ana diastimata\n");
else if (z>=31.0 && z<32.0)
   Serial.print("thielodis- pithani veltiosi\n");
else if (z>=32.0)
   Serial.print("thielodeis- endones vroxoptoseis\n");
  }
}
```

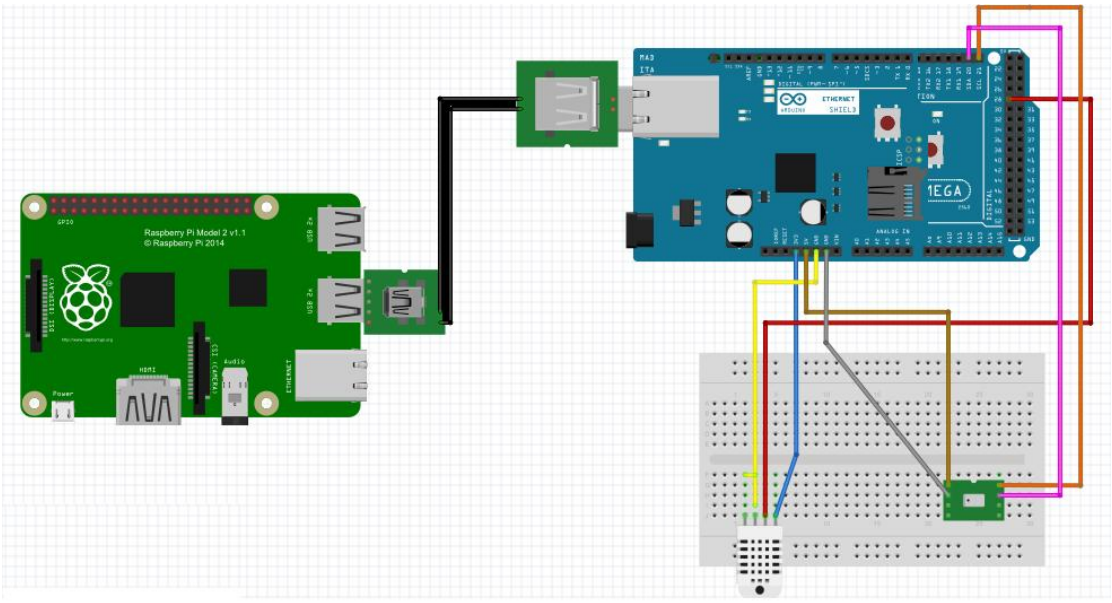


Εικόνα 51, Σειριακή οθόνη προγράμματος IDE,
όπου εμφανίζεται το αποτέλεσμα του κώδικα Zambretti

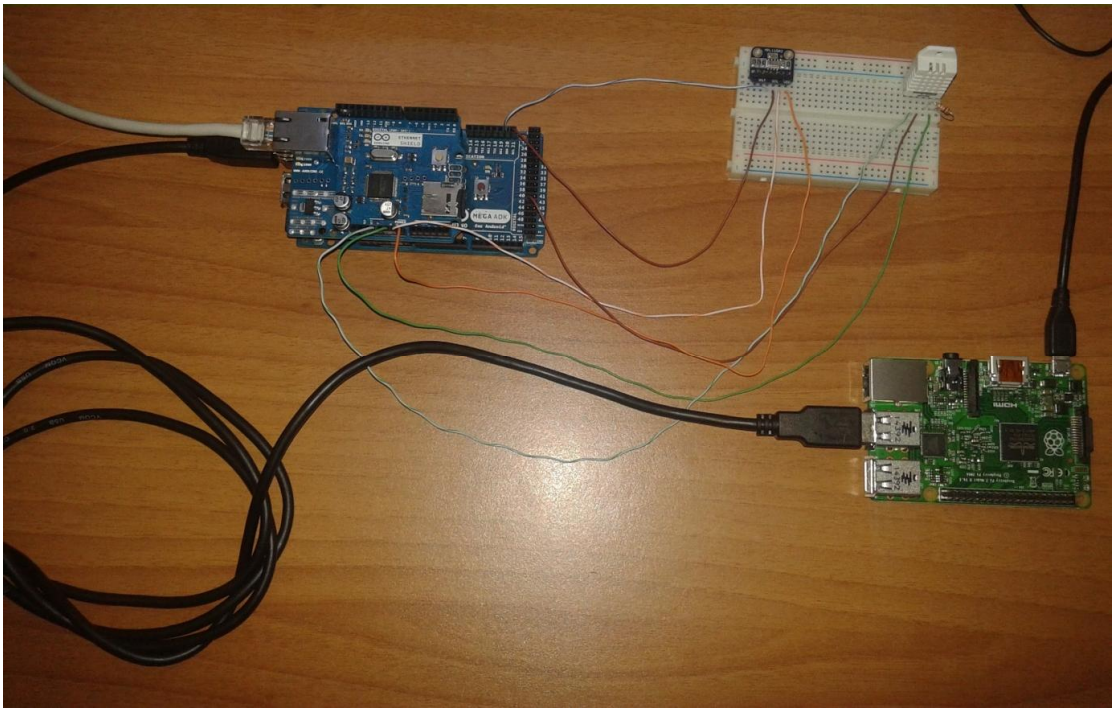
7.2 Ανάλυση και υλοποίηση του συστήματος DAQ - ΑΜΣ με τη χρήση Arduino Mega ADK Rev3 και Raspberry Pi Model B

Στο κεφάλαιο αυτό θα χρησιμοποιήσουμε το Raspberry Pi σε συνδυασμό με το Arduino Mega, ώστε να υλοποιήσουμε το σύστημα DAQ – ΑΜΣ. Σκοπός μας είναι να μεταφέρουμε τις μετρήσεις από τα αισθητήρια που λαμβάνει το Arduino στο Raspberry Pi. Με τον τρόπο αυτό μπορούμε να αναπτύξουμε ένα σύστημα DAQ, το οποίο θα είναι σε θέση να λαμβάνει και να επεξεργάζεται τα δεδομένα των αισθητηρίων αυτόνομα (χωρίς απαραίτητα να απαιτείται η επεξεργασία τους από το χρήστη όταν αυτά συγκεντρωθούν). Στις παρακάτω εικόνες (εικόνες 52α και 52β) φαίνεται η συνδεσμολογία του συστήματος.

Στο σημείο αυτό θα πρέπει να τονιστεί ότι η επικοινωνία του Raspberry Pi και του Arduino γίνεται σειριακά. Για το λόγο αυτό θα πρέπει να ενεργοποιήσουμε το serial port του Raspberry Pi. Η ενεργοποίηση πραγματοποιείται ανοίγοντας το root terminal του Raspberry Pi και πληκτρολογώντας “$ sudoedit /etc/inittab” .



Εικόνα 52α, Προσομοίωση της συνδεσμολογίας του συστήματος, για την επικοινωνία Raspberry Pi, Arduino και αισθητηρίων



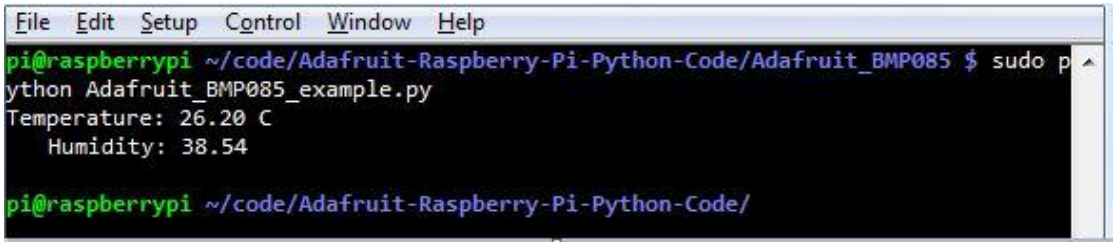
Εικόνα 52β, Συνδεσμολογία του συστήματος, για την επικοινωνία Raspberry Pi, Arduino και αισθητηρίων

7.2.1 Λήψη μετρήσεων στο Raspberry Pi, μέσω του Arduino, από το αισθητήριο RHT 03

Ο προγραμματισμός του Raspberry Pi πραγματοποιείται στη γλώσσα Python και για να συντάξουμε το πρόγραμμά μας ανοίγουμε το Gnome system menu, πηγαίνουμε στην καρτέλα Programming και ανοίγουμε το Geany. Στη συνέχεια ανοίγει το παράθυρο του περιβάλλοντος προγραμματισμού, στο οποίο επιλέγουμε την καρτέλα File, έπειτα το New (with Template) και επιλέγουμε το main.py, οπότε και είμαστε έτοιμοι να συντάξουμε τον κώδικά μας. Επίσης εδώ πρέπει να αναφερθεί ότι το Arduino θα πρέπει να είναι προγραμματισμένο για να λαμβάνει μετρήσεις από το αισθητήριο RHT 03.

Παρακάτω παρουσιάζεται ο κώδικας, με τον οποίο έχουμε προγραμματίσει το Raspberry Pi ώστε να λαμβάνει σειριακά τις μετρήσεις από το Arduino (Εικόνα 53).

```
import time
import serial #
def main():
port = serial.Serial("/dev/ttyACM0", baudrate=115200, timeout=None) #
while True:
line = port.readline() #
arr = line.split() #
if len(arr) < 3: #
continue #
dataType = arr[2]
data = float(arr[1]) #
if dataType == '%':
print("Humidity: %.1f %%" % data)
else:
print("Temperature: %.1f C" % data)
time.sleep(0.01)
if name == "__main__":
main()
```



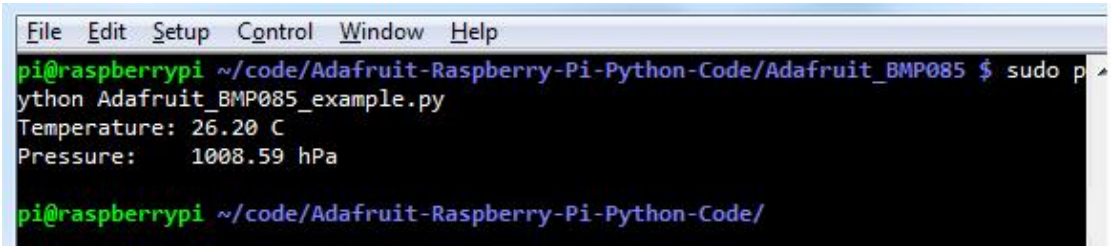
Εικόνα 53, Μέτρηση θερμοκρασίας και υγρασίας με το Raspberry Pi

7.2.2 Λήψη μετρήσεων στο Raspberry Pi, μέσω του Arduino, από το αισθητήριο MPL 115A2

Όπως και προηγουμένως, η λήψη μετρήσεων με το Raspberry Pi, προϋποθέτει να είναι προγραμματισμένο το Arduino, έτσι ώστε να λαμβάνει μετρήσεις με τον αισθητήρα MPL 115A2.

Παρακάτω παρουσιάζεται ο κώδικας, με τον οποίο έχουμε προγραμματίσει το Raspberry Pi ώστε να λαμβάνει σειριακά τις μετρήσεις από το Arduino (Εικόνα 54).

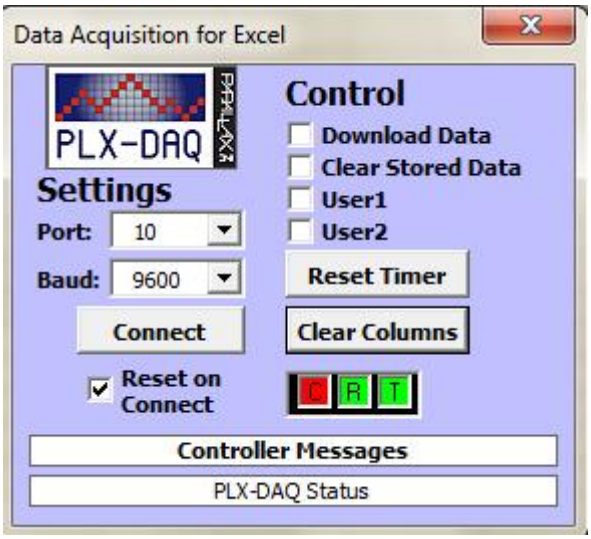
```
#include <Wire.h>
#include <gy_65.h>
void setup() {
Serial.begin(115200);
readCalibrationData();
}
void loop() {
float temp = readTemperature();
float pressure = readPressure();
Serial.print("Pressure: ");
Serial.print(pressure,2);
Serial.println(" Pa");
Serial.print("Temperature: ");
Serial.print(temp,2);
Serial.println("C");
delay(1000);
}
```



Εικόνα 54, Μέτρηση θερμοκρασίας και ατμοσφαιρικής πίεσης με το Raspberry Pi

7.3 Arduino and real time (data acquisition) charts in Excel

Στις περιπτώσεις που απαιτείται συλλογή μεγάλου όγκου δεδομένων σε πραγματικό χρόνο καθώς και η επεξεργασία αυτών με σύνθετα μαθηματικά μοντέλα (μεγάλη υπολογιστική ισχύ συγκριτικά με τους μικροελεγκτες και τους μικροϋπολογιστές) μπορούμε να χρησιμοποιήσουμε έναν υπολογιστή συνδεδεμένο με τον αεροελεγκτή ο οποίος λαμβάνει τις μετρήσεις. Η αποστολή τον δεδομένων από τον μικροελεγκτή προς τον υπολογιστή μπορεί να επιτευχθεί με ποικίλους τρόπους. Για παράδειγμα το Arduino μέσω των shields όπως το Ethernet Shield, Bluetooth Shield κ.α μπορεί να αποστείλει δεδομένα απευθείας σε έναν υπολογιστή. Παρακάτω επιλέξαμε να παρουσιάσουμε την διαδικασία συλλογής τον δεδομένων (θερμοκρασία και ατμοσφαιρική πίεση) και την άμεση καταγραφή τους σε υπολογιστικό φύλλο Excel μέσω της σειριακής επικοινωνίας.
Η υλοποίηση περιλαμβάνει την χρήση ενός δωρεάν προγράμματος που ονομάζεται PLX-DAQ (https://www.parallax.com/downloads/plx-daq, Εικόνα 55) το οποίο εγκαθιστούμε στον υπολογιστή μας καθώς και το Excel. Επιπρόσθετα απαιτείτε ο προγραμματισμός του Arduino.



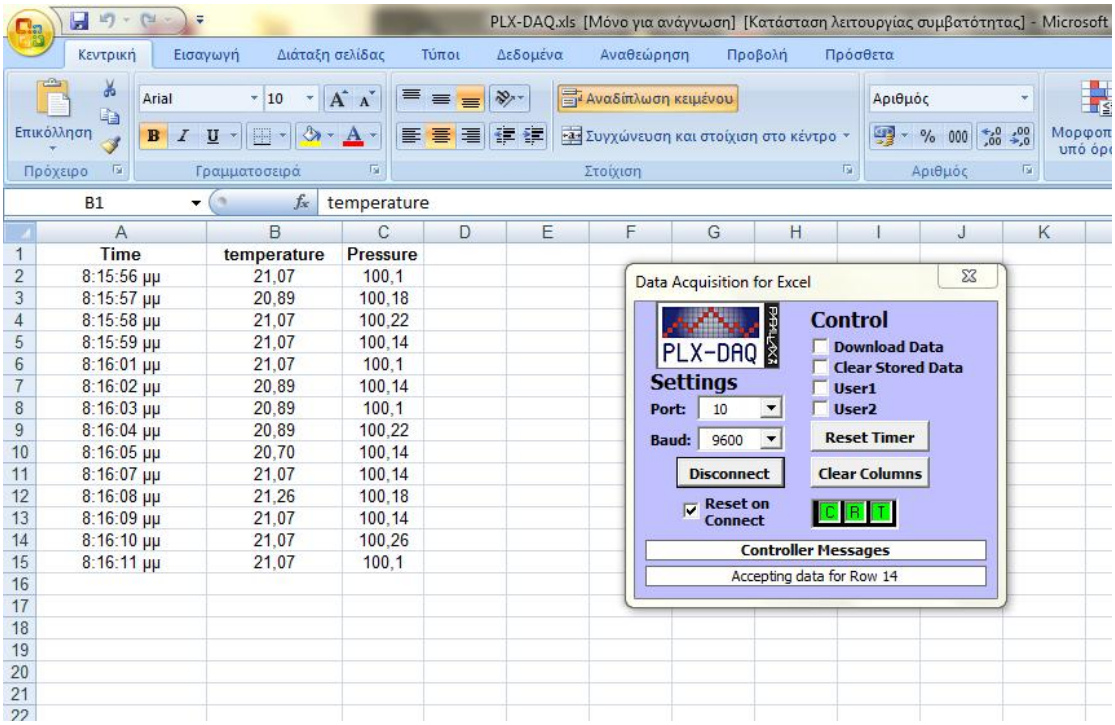
Εικόνα 55, Λογισμικό PLX-DAQ

Ο κώδικας Arduino (Επεξηγείτε στο παράρτημα):

```
#include <Wire.h>
#include <Adafruit_MPL115A2.h>
Adafruit_MPL115A2 mpl115a2;
int row = 0;
void setup(void)
{
  Serial.begin(9600);
  Serial.println("Getting barometric pressure ...");
  mpl115a2.begin();
```

```
  Serial.println("CLEARDATA");
  Serial.println("LABEL,Time,temperature,Pressure");
}
   void loop(void)
{
  float pressureKPA = 0, temperatureC = 0;

  mpl115a2.getPT(&pressureKPA,&temperatureC);
  Serial.print("Pressure (kPa): "); Serial.print(pressureKPA, 4); Serial.print(" kPa  ");
  Serial.print("Temp (*C): "); Serial.print(temperatureC, 1); Serial.println(" *C both
  measured together");

  pressureKPA = mpl115a2.getPressure();
  Serial.print("Pressure (kPa): "); Serial.print(pressureKPA, 4); Serial.println(" kPa");

  temperatureC = mpl115a2.getTemperature();
  Serial.print("Temp (*C): "); Serial.print(temperatureC, 1); Serial.println(" *C");
  Serial.print("DATA,TIME,"); Serial.print(temperatureC); Serial.print(",");
Serial.println(pressureKPA);
  row++;
  if (row > 360)
   {
    row=0;
    Serial.println("ROW,SET,2");
   }
  delay(100);
}
```

Αφού φορτώσουμε τον κώδικα στο Arduino και συνδέσουμε το αισθητήριο (MPL11A2) στην συνέχεια ανοίγουμε το Excel και το πρόγραμμα PLX-DAQ επιλέγουμε το Serial Port όπου συνδέσαμε το Arduino (και το ίδιο delay) και πατάμε Connect. Όλο αυτό έχει ως αποτέλεσμα την καταγραφή των μετρήσεων στο Excel (Εικόνα 56). Πλέον έχουμε στην διάθεσή μας μια ευρεία γκάμα επιλογών επεξεργασίας των μετρήσεων μας καθώς και την προβολή τους σε γραφήματα.



	A	B	C
1	Time	temperature	Pressure
2	8:15:56 μμ	21,07	100,1
3	8:15:57 μμ	20,89	100,18
4	8:15:58 μμ	21,07	100,22
5	8:15:59 μμ	21,07	100,14
6	8:16:01 μμ	21,07	100,1
7	8:16:02 μμ	20,89	100,14
8	8:16:03 μμ	20,89	100,1
9	8:16:04 μμ	20,89	100,22
10	8:16:05 μμ	20,70	100,14
11	8:16:07 μμ	21,07	100,14
12	8:16:08 μμ	21,26	100,18
13	8:16:09 μμ	21,07	100,14
14	8:16:10 μμ	21,07	100,26
15	8:16:11 μμ	21,07	100,1

Εικόνα 56 , Καταγραφή των μετρήσεων από το Arduino στο Excel

Τέλος θα αποτελούσε παράλειψη αν δεν αναφέραμε πως υπάρχουν πολλά προγράμματα για τα συστήματα DAQ επαγγελματικού τύπου όπως το γνωστό LabView. Το LabView είναι η πιο κομψή και δυνατή γλώσσα προγραμματισμού για τη συλλογή δεδομένων, την ανάλυση δεδομένων, την προσομοίωση και τον έλεγχο οργάνων και μετρήσεων μέσω υπολογιστή. Στηρίζεται στον γραφικό προγραμματισμό μέσω αντικειμένων και αποτελεί ένα καλό παράδειγμα του «αντικειμενοστραφή προγραμματισμού» (object oriented programming). Ο λόγος που αποφύγαμε να το χρησιμοποιήσουμε είναι πως δεν αποτελεί λογισμικό ανοιχτού κώδικα. Ωστόσο σε βιομηχανικό επίπεδο αποτελεί απαραίτητο εργαλείο για τα συστήματα DAQ.

8. Μελλοντική επέκταση

Η παρούσα διπλωματική εργασία είχε ως στόχο τη συλλογή δεδομένων από το περιβάλλον, μέσω αισθητηρίων , την πρόσβαση σε αυτά τοπικά ή απομακρυσμένα καθώς και την επεξεργασία - αξιοποίησή τους. Ωστόσο στα πλαίσια μιας διπλωματικής εργασίας είναι αδύνατο να εμβαθύνει κάποιος στα συστήματα DAQ - και να μελετήσει όλες τις εφαρμογές ή περιπτώσεις χρήσης τους. Εμείς επιλέξαμε να υλοποιήσουμε ένα σύστημα Α.Μ.Σ. με το οποίο ουσιαστικά συλλέγουμε τη θερμοκρασία, την υγρασία και την ατμοσφαιρική πίεση, αποθηκεύουμε τις τιμές τους και τις επεξεργαζόμαστε - αξιοποιούμε για την πρόγνωση του καιρού, μέσω του αλγορίθμου Zambretti. Φυσικά, τα δεδομένα των μετρήσεων αυτών παρουσιάζουν πληθώρα αξιοποιήσιμων εφαρμογών. Ενδεικτικά, θα μπορούσαμε να αξιοποιήσουμε τις μετρήσεις της θερμοκρασίας για την εξοικονόμηση ενέργειας σε ένα χώρο (κυρίως βιομηχανικό), ελέγχοντας τον κλιματισμό, σε εργαστήρια επισκευής ευαίσθητων ηλεκτρονικών (όπως κινητών), που απαιτείται ο έλεγχος και η σταθεροποίηση της υγρασίας και της ατμοσφαιρικής πίεσης ή σε θερμοκήπια μεγάλης παραγωγής, όπου απαιτείται ο έλεγχος της θερμοκρασίας και της υγρασίας.

Παρακάτω θα γίνει μια συνοπτική παρουσίαση της εξοικονόμησης ενέργειας σε μεγάλα βιομηχανικά κτήρια. Ειδικότερα, οικονομία στην ενέργεια σημαίνει αποδοτική χρήση ενέργειας, με προσεκτική η στοχευμένη επιλογή συσκευών που την χρησιμοποιούν. Επίσης σημαίνει τη βελτιστοποιημένη λειτουργία των συσκευών για την αποφυγή της άσκοπης σπατάλης της. Η σπατάλη ενέργειας έχει σοβαρές συνέπειες τόσο στο περιβάλλον, όσο και στην οικονομία. Κάποιες απώλειες ενέργειας με τα σημερινά δεδομένα δε μπορούν να αποφευχθούν, όμως είναι δυνατόν να μειωθούν σε μεγάλο βαθμό. Η εξοικονόμηση ενέργειας στα βιομηχανικά και μη κτήρια μπορεί να επιτευχθεί με τον έλεγχο της κατανάλωσής της από συσκευές ή τον περιορισμό των απωλειών από το ίδιο το κτήριο. Η οικονομία ενέργειας στη βιομηχανία μοιάζει με αυτή που εφαρμόζεται και στις κατοικίες. Κοινό σημείο τους είναι τα μέτρα που λαμβάνονται, ώστε να μη χάνεται η ενέργεια.

Η εξοικονόμηση ενέργειας στον κλιματισμό είναι ίσως από τους πιο δύσκολους τομείς επένδυσης για επιχειρήσεις και ιδιώτες κυρίως λόγω τεχνολογικών περιορισμών, συμβολαίων υποστήριξης ή χρόνου παρέμβασης αν αυτό είναι εφικτό.

Ο έλεγχος του κλιματισμού με αισθητήρες παρουσίας είναι ίσως από τους μόνους τρόπους εξοικονόμησης ενέργειας αν και φαίνεται δύσκολα υλοποιήσιμο λόγω διαφορετικών τεχνολογιών.

Γενικά για τον έλεγχο κλιματισμού θα μπορούσαν να χρησιμοποιηθούν οι αισθητήρες παρουσίας χωρίς αντιστάθμιση φωτός.

Χρήση και εγκατάσταση εξαρτώνται μερικώς από το σύστημα υπό διερεύνηση:

1. Υδρόψυκτα συστήματα

Σε αυτά μια κεντρική ψυκτική μονάδα, ψύχει νερό περίπου στους 4ο C, και κατόπιν αυτό το νερό δρομολογείται στην κτιριακή εγκατάσταση. Σε κάθε αίθουσα υπάρχουν fan-coils ο ανεμιστήρας των οποίων αναλαμβάνει να διοχετεύσει το ψυκτικό φορτίο στο χώρο. Η παροχή κρύου νερού ελέγχεται από μια ηλεκτροβάνα η οποία με τη σειρά της ελέγχεται από ένα θερμοστάτη, ο οποίος θα μπορούσε να συνδεθεί σε έναν Arduino ή οποιοδήποτε άλλο μικροελεγκτή.

Μερικοί θερμοστάτες έχουν είσοδο ελέγχου η οποία μπορεί να ελεγχθεί ώστε να κλείνει ο κλιματισμός στο χώρο αν δεν υπάρχει παρουσία ή να ανεβαίνει η θερμοκρασία σε ένα επιθυμητό επίπεδο (πχ από 21ο C σε 26ο C). Συχνά αποτελεί προτιμότερη και πιο ορθολογική μέθοδο σε σχέση με την πλήρη διακοπή κλιματισμού.

2. Μικρά κεντρικά συστήματα

Μια μικρού φορτίου κεντρική μονάδα ψύχει τον αέρα και μια σειρά αγωγών με τη βοήθεια ανεμιστήρων διοχετεύει το ψυκτικό φορτίο στο χώρο. Οι ανεμιστήρες συνήθως ελέγχονται από τους χωρικούς θερμοστάτες και ανοίγουν ή κλείνουν αναλόγως.

Ένας αισθητήρας θερμοκρασίας μπορεί εύκολα να κλείσει έναν ανεμιστήρα και κατ' επέκταση τη διοχέτευση του ψυκτικού φορτίου σε ένα χώρο.

3. Πολύ-Διαιρούμενα συστήματα

Μια κεντρική μονάδα αναλαμβάνει τη διοχέτευση ψυκτικού υγρού σε επιμέρους εξατμιστικές μονάδες (evaporator units). Σε αυτήν την περίπτωση δεν μεταφέρεται ψυχρός αέρας, αλλά ουσιαστικά παράγεται στον κάθε χώρο ξεχωριστά.

Έτσι ένας αισθητήρας θερμοκρασίας (με τη χρήση μικροελεγκτή όπως το Arduino) θα μπορεί να ελέγχει την κάθε επιμέρους μονάδα ξεχωριστά και να αποφασίζει σε ποια θα μετεφέρεται ο ψυχρός αέρας.

4. Απλά διαιρούμενα air-conditions

Η πρόκληση με αυτά τα συστήματα είναι ότι δε παρουσιάζουν όλα τα μοντέλα την ίδια συμπεριφορά αν κοπεί και επανέλθει η τροφοδοσία τους. Όταν κοπεί η τροφοδοσία και επανέλθει τότε έχουμε τις εξής περιπτώσεις:

Στις περισσότερες περιπτώσεις των βιομηχανικών κτηρίων (παλαιότερων κατασκευών) ο κλιματισμός είναι αυτόνομος, αφού ο κάθε χώρος είναι ανεξάρτητος από τους υπόλοιπους και εξαρτάται από τον εκάστοτε χρήστη του χώρου. Για να αποφευχθεί η σπατάλη ενέργειας, χωρίς όμως την επανεγκατάσταση μιας κεντρικής μονάδας κλιματισμού (αποφεύγοντας παράλληλα και το αντίστοιχο κόστος),

μπορούμε με τη χρήση μικροελεγκτή (Arduino) να ελέγχουμε τις επιμέρους κλιματιστικές μονάδες, ανάλογα με τη θερμοκρασία του χώρου στον οποίο βρίσκονται. Με άλλα λόγια, ο μικροελεγκτής θα λαμβάνει τις τιμές της θερμοκρασία κάθε χώρου (μέσω αισθητηρίων θερμοκρασίας) και θα αποφασίζει για τη θερμοκρασία αυτών. Οι κεντρικές μονάδες κλιματισμού δεν αποτελούν τη βέλτιστη λύση, αφού δε λαμβάνουν υπόψιν τους παράγοντες που επηρεάζουν τη θερμοκρασία του χώρου, όπως για παράδειγμα τον προσανατολισμό του χώρου, ο οποίος μπορεί να είναι βόρειος, νότιος κλπ.

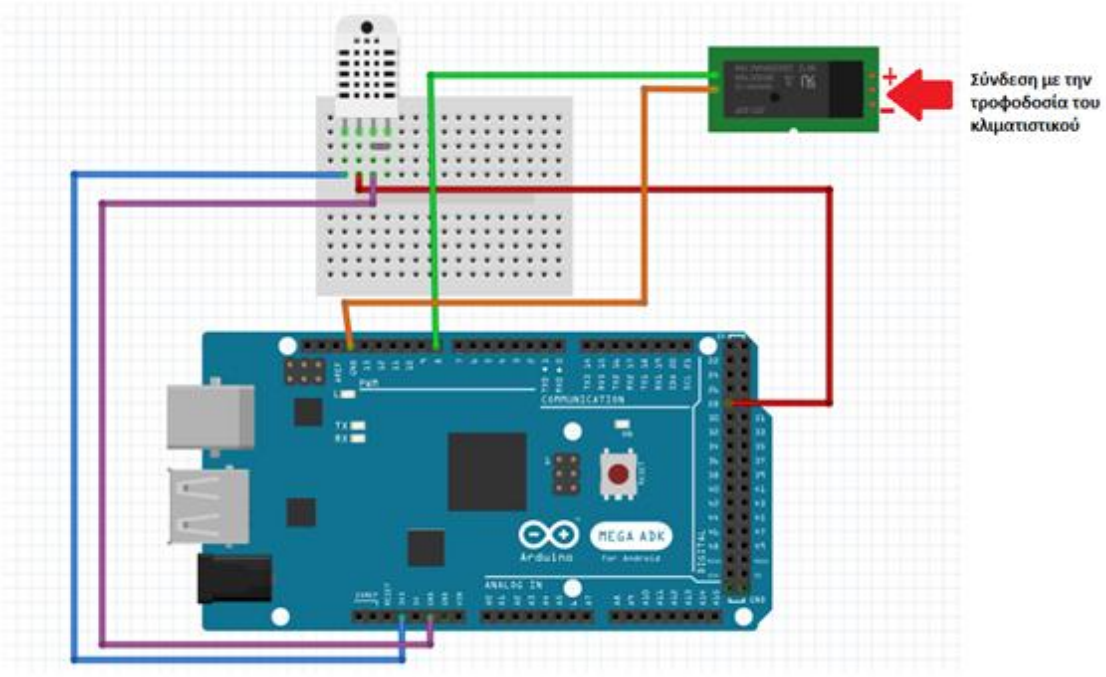
Σε γενικές γραμμές τα συστήματα κλιματισμού είναι από τις πιο ενεργοβόρες εφαρμογές στον κτιριακό τομέα και μια της οποίας το κόστος δύσκολα αντιμετωπίζεται.

Παρακάτω παρουσιάζεται ένας απλός έλεγχος μιας κλιματιστικής μονάδας με την χρήση του μικροελεγκτή Arduino. Ειδικότερα θα πραγματοποιείτε μέτρηση της θερμοκρασίας του χώρου μέσω του αισθητήρα RHT 03 και όταν αυτή είναι μεγαλύτερη από 23 oC τότε ο μικροελεγκτής Arduino Mega χρησιμοποιώντας ένα RC Switch θα απενεργοποιεί την κλιματιστική μονάδα. Αντίθετα όταν η θερμοκρασία του χώρου πέσει κάτω από 23 oC τότε θα ενεργοποιείται η κλιματιστική μονάδα. Η θερμοκρασία που έχουμε θέση σε αυτή είναι 23 oc. Με αυτόν τον πολύ απλό τρόπο μπορούμε να ελέγξουμε την θερμοκρασία ενός χώρου και να εξοικονομήσουμε ενέργεια. Με άλλα λόγια η παραπάνω εφαρμογή μπορεί να φανεί χρήσιμη στις περιπτώσεις που ο χρήστης έχει ξεχάσει να ρυθμίσει την θερμοκρασία στην ιδανικότερη τιμή της σε σχέση με την θερμοκρασία του περιβάλλοντος (για παράδειγμα η εξωτερική θερμοκρασία είναι 24oc και η κλιματιστική μονάδα είναι ρυθμισμένη στους 30 oC)και επομένως καταναλώνεται άσκοπα ενέργεια.

Για τον έλεγχο της κλιματιστικής χρειαζόμαστε:

1)Arduino
2)RC Switch
3)Breadbond
4)Καλώδια
4)Αισθητήρας θερμοκρασίας

Και πραγματοποιούμε την συνδεσμολογία της παρακάτω εικόνας (εικόνα 57):



Εικόνα 57, Έλεγχος κλιματιστικής μονάδας με το Arduino

Στην συνέχεια φορτώνουμε τον κώδικα (εικόνα 58) στον Arduino Mega μέσω του προγράμματος IDE.

klimatistiki_monada

```
#include <dht.h>
#include <RCSwitch.h> //Βιβλιοθήκη του RC Switch
dht DHT;
const int dhtPin = 10;
RCSwitch mySwitch = RCSwitch();

int chk;
float temp;
void setup()
{
     Serial.begin(9600);
     // If you want to set the aref to something other than 5v
    analogReference(EXTERNAL);
    //enable RC switch transmissions on pin 10.
     mySwitch.enableTransmit(8);
}

void loop()
{
    // READ DATA
    chk = DHT.read22(dhtPin);
    temp= DHT.temperature;

    Serial.print(" %, Temp: ");
    Serial.print(temp);
    Serial.println(" Celsius");

    if(temp < 23.00)
```

Εικόνα 58, Ο κώδικας ελέγχου της κλιματιστικής μονάδας

Ο κώδικας παρουσιάζεται παρακάτω:

```
#include <dht.h>
#include <RCSwitch.h> //Βιβλιοθήκη του RC Switch
dht DHT;
const int dhtPin = 10;
RCSwitch mySwitch = RCSwitch();
int chk;
float temp;
void setup()
{
   Serial.begin(9600);
   // If you want to set the aref to something other than 5v
   analogReference(EXTERNAL);
   //enable RC switch transmissions on pin 10.
   mySwitch.enableTransmit(8);
}
void loop()
{
   // READ DATA
   chk = DHT.read22(dhtPin);
   temp= DHT.temperature;
   Serial.print(" %, Temp: ");
   Serial.print(temp);
   Serial.println(" Celsius");

   if(temp < 23.00)
   {
   mySwitch.switchOn(1,1);
   Serial.println("Heater ON");
   }
 else {
   Serial.println("Heater OFF");
   mySwitch.switchOff(1,1);
      }
  delay(2000); //Delay 2 sec.

}
```

Συμπεράσματα

Ο στόχος της παρούσης διπλωματικής εργασίας είναι αφενός μεν να αποτελέσει μια εισαγωγή στα αυτοματοποιημένα συστήματα συλλογής και επεξεργασίας δεδομένων μέσω αισθητηρίων με την χρήση μικροελεγκτών (arduino) και μικροϋπολογιστών (raspberry pi) και αφετέρου να παρουσιάσει αναλυτικά την υλοποίηση ενός συγκεκριμένου συστήματος λήψης και αποθήκευσης δεδομένων καθώς και την απομακρυσμένη πρόσβαση σε αυτά μέσω του διαδικτύου. Στο πλαίσιο αυτό αποφασίσαμε να πραγματοποιήσουμε την υλοποίηση ενός αυτόνομου μετεωρολογικού σταθμού για την παρακολούθηση και καταγραφή και επεξεργασία των μετεωρολογικών φαινομένων και στην συνέχεια με βάση αυτά τα στοιχεία να γίνεται πρόγνωση του καιρού. Η δημιουργία αυτού του συστήματος ΑΜΣ-DAQ είναι μόνο η κορυφή του παγόβουνου καθώς οι εφαρμογές τέτοιων συστημάτων εξελίσσονται με τέτοιο τρόπο, που πλέον το μόνο που μπορεί να τα περιορίσει είναι η ανθρώπινη φαντασία. Μια παραλλαγή του παραπάνω συστήματος προτείναμε στην ενότητα της μελλοντικής επέκτασης όπου μπορεί εύκολα κανείς να οδηγηθεί στο συμπέρασμα της ευρείας χρήσεις τέτοιων συστημάτων. Καταλήγοντας, πρέπει να σημειώσουμε ότι η εφαρμογή που αναπτύχθηκε έχει αρκετούς περιορισμούς και ελλείψεις σε σχέση με ένα εμπορικό σύστημα, αποτελεί όμως έναν ισχυρό πυρήνα πάνω στον οποίο μπορεί να αναπτυχθεί μια αξιόπιστη και οικονομική λύση συστήματος DAQ- εποπτείας και καταγραφής δεδομένων για χρήση τόσο σε ερευνητικό επίπεδο όσο και σε βιομηχανικό επίπεδο.

Βιβλιογραφία

[1] Tanenbaum, Andrew S., «Δίκτυα Υπολογιστών», Κλειδάριθμος, 2012

[2] Jeremy Bloom, «Exploring Arduino, Tools and Techniques for Engineering . Wizardry**»,** Wiley 2013

[3] Tero Karvinen, Kimmo Karvinen, « Make Sensor (Arduino & Raspberry) **»** MakerMedia, 2014

[4] Rick Anderson, Dan Cervo, «Pro Arduino» Frlendsof, 2013

[5] Harold Timmis,«Practical Arduino Engineering **»,**Apress ,2011

[6] J.F. Kurose, K.W. Ross, «Δικτύωση Υπολογιστών», Α. Γκιούρδα, 2008

[7] Charles Bell - Beginning Sensor Networks with Arduino, APRESS, 2013

[8] Andrew K. Dennis - Raspberry Pi Home Automation with Arduino, Packt . Publishing 2013

[9] Rick Golden - Raspberry Pi NetworkingCookbook, Packt Publishing 2013

[10] Βίγκλας Π. - Εισαγωγή στη Μετεωρολογία: Μια εκπαιδευτική προσέγγιση - 2011 - edulll.ekt.gr

[11] Μπαλάρας Κ. – Στρατηγικές εξοικονόμησης ενέργειας - Τεκδοτική

[12] http://www.arduino.cc/

[13] https://www.raspberrypi.org/

[14] http://www.codecademy.com/learn

Παράρτημα 1

Α) Λήψη μετρήσεων θερμοκρασίας-Υγρασίας με τον αισθητήρα

```
#include <dht.h>
dht DHT;          // Εισαγωγή της βιβλιοθήκης του αισθητήρα RHTO3

const int dhtPin = 10;    // Δήλωση του ψηφιακού pin (ψηφιακή είσοδος
                             arduino) στο  οποίο θα γίνεται η ανάγνωση των
                             δεδομένων από τον  αισθητήρα
int chk;
float hum;              // Αρχικοποίηση σταθερών και μεταβλητών. Η
float temp;                θερμοκρασία και η υγρασία δηλώνονται ως δεκαδικοί
                           αριθμοί. Η ψηφιακή θύρα  από την οποία θα γίνεται
                           η ανάγνωση των δεδομένων από το αισθητήριο
                           (Η παραμετροποίηση και ο τρόπος επικοινωνίας
                           περιλαμβάνονται στη βιβλιοθήκη RHT03 της AdaFruit)
                           δηλώνεται ως ακέραιος αριθμός.
void setup()
 {
   Serial.begin(9600);     // Αρχικοποίηση κατάστασης/δηλώσεων η οποία
                              εκτελείται μόνο μια φορά.
       }

void loop()     //Εσωκλείεται (μέσα σε {}) το κυρίως πρόγραμμα το
                  οποίο εκτελείται συνέχεια
{
   chk = DHT.read22(dhtPin);   //  Γίνεται ανάγνωση των δεδομένων από τον
αισθητήρα και ανατίθενται στις μεταβλητές.
   hum = DHT.humidity;
    temp= DHT.temperature;  // Οι DHT.humidity και DHT.temperation είναι
εντολές της βιβλιοθήκης (ορίζουν τις λειτουργίες εισόδου/εξόδου) οι οποίες
αναλαμβάνουν την επικοινωνία με το αισθητήριο.

Serial.print("Humidity: ");   // Πραγματοποιείται εμφάνιση των τιμών   της
θερμοκρασίας και της υγρασίας μέσω  της σειριακής οθόνης που το πρόγραμμα IDE
του arduino περιλαμβάνει
Serial.print(hum);
Serial.print(" %, Temp: ");
Serial.print(temp);
Serial.println(" Celsius");
delay(2000);  // Καθορισμός συχνότητας λήψης των μετρήσεων
}
```

Β) Τροποποίηση του κώδικα και αποθήκευση των δεδομένων στην κάρτα SD για τον αισθητήρα RHT 03

```
#include <SPI.h> (1)
#include <SD.h> (2)
#include <pin.h> (3)
#include <dht.h>
dht DHT;

const int chipSelect = 4; // Δήλωση του pin 4 (υπεύθυνο για την σειριακή επικοινωνία) ως σταθερά
const int dhtPin = 42;     // Δήλωση του pin στο οποίο θα γίνεται η ανάγνωση των δεδομένων  από το αισθητήριο ως σταθερά
int chk;
float hum;
float temp;

void setup()
{
 Serial.begin(9600);     //Ενεργοποιεί την σειριακή επικοινωνία και περιμένει για την έναρξή της
 Serial.print("Initializing SD card..."); // Εμφανίζει το μήνυμα “εγκατάσταση SD card ”
 pinMode(53, OUTPUT);  // Το pin 53 πρέπει να δηλώνεται ως έξοδος στο arduino mega (ως έξοδος τηςβιβλιοθήκης της κάρτας SD) ακόμα και αν δεν χρησιμοποιηθεί στην σειριακή επικοινωνία
 if (!SD.begin(chipSelect)) {       //ελέγχεται αν η κάρτα είναι τοποθετημένη και μπορεί να αρχικοποιηθεί
    Serial.println("Card failed, or not present");       //Αν δεν είναι τοποθετημένη η κάρτα SD ή  δεν είναι δυνατή η επικοινωνία με αυτή τότε εμφάνισε το μήνυμα “Η επικοινωνία με την κάρτα απέτυχε ή αυτή δεν είναι τοποθετημένη”
return;
 }
 Serial.println("card initialized.");   //Αν η κάρτα είναι τοποθετημένη εμφάνισε το μήνυμα “Αρχικοποίηση επικοινωνίας με την κάρτα”
}

void loop()
{

        chk = DHT.read22(dhtPin);
        hum = DHT.humidity;
        temp= DHT.temperature;

        Serial.print("Humidity: ");
        Serial.print(hum);
        Serial.print(" %, Temp: ");
        Serial.print(temp);
        Serial.println(" Celsius");
```

```
        delay(60000); //Καθυστέρηση 2 sec.

 String dataString = "";      // Ορίζουμε τις μεταβλητές dataString και dataString_2 ως συμβολοσειρές (string)
 String dataString_2 = "";

 for (int digitalPin = 42; digitalPin < 43; digitalPin++) {
        int sensor = digitalRead(digitalPin);
        dataString += String(temp);
        dataString_2 += String(hum);
        if (digitalPin < 44) {
        dataString += ",";
        }                       // Ανάγνωση από το αισθητήριο και προσάρτηση των δεδομένων  στην συμβολοσειρά
}


 File dataFile = SD.open("datalog.txt", FILE_WRITE); //Δημιουργία και άνοιγμα του αρχείου datalog.txt από την κάρτα SD

  // Αν το αρχείο είναι διαθέσιμο γίνεται εγγραφή σε αυτό των δεδομένων υπό μορφή συμβολοσειράς
 if (dataFile) {
        dataFile.print("temp:"),dataFile.print(dataString),
        dataFile.print("hum:"), dataFile.println(dataString_2);
        dataFile.close();

        Serial.println("temp:"),Serial.println(dataString);
        Serial.println("hum:"), Serial.println(dataString_2);
 }
 else {
        Serial.println("error opening datalog.txt"); //Αν είναι αδύνατο το άνοιγμα του αρχείου εμφάνισε το μήνυμα “σφάλμα εντοπισμού του αρχείου datalog.txt ”
 }
}
```

(1)Η βιβλιοθήκη SPI επιτρέπει την επικοινωνία του Arduino ως master συσκευή με άλλες SPI συσκευές. Το Serial Peripherial Interface (SPI) αποτελεί ένα σύγχρονο πρωτόκολλο σειριακών δεδομένων για την επικοινωνία μεταξύ μικροελεγκτών και άλλων περιφερειακών συσκευών. Σε μια σύνδεση SPI πάντα υπάρχει μια master συσκευή (συνήθως ένας μικροελεγκτής) που ελέγχει τις περιφερειακές συσκευές.
Τυπικά υπάρχουν 3 γραμμές κοινές σε όλες τις συσκευές της SPI σύνδεσης:

- **MISO** (Master In Slave Out) - Η γραμμή Slave για αποστολή δεδομένων στον master,
- **MOSI** (Master Out Slave In) – Η γραμμή Master για αποστολή δεδομένων στις περιφερειακές συσκευές,
- **SCK** (Serial Clock) – Ο παλμός ρολογιού για το συγχρονισμό της μετάδοσης δεδομένων από την master συσκευή,
- **SS** (Slave Select) – Το pin σε κάθε συσκευή το οποίο χρησιμοποιεί ο master ώστε να ενεργοποιεί ή να απενεργοποιεί τις περιφεριακές συσκευές. Όταν σε μια συσκευή το Slave Select (SS) pin είναι low τότε αυτη η συσκευή

επικοινωνεί με την master , ενώ όταν είναι high την αγνοεί. Αυτό μας παρέχει την δυνατότητα χρήσης πολλαπλών συσκευών SPI, οι οποίες μοιράζονται τις ίδιες MISO, MOSI και SLK γραμμές.

(2) Η βιβλιοθήκη SD είναι υπεύθυνη για την ανάγνωση καρτών microSD. Με την χρήση της βιβλιοθήκης το SS είναι ενεργοποιημένο στο Pin 4. Η κάρτα sd τοποθετείται στην υποδοχή sd reader του ethernet shield και η επικοινωνία επιτυγχάνεται με την βοήθεια της βιβλιοθήκης SPI.
(3) Αποτελεί βιβλιοθήκη κατασκευασμένη και επεξεργασμένη με το πρόγραμμα blood shed dev c++ (ο κώδικας βρίσκεται στο παράρτημα) για την μετατροπή των pin ωστε να επιτευχθεί η επικοινωνία του ethernet shield και της κάρτας SD με το arduino mega. Η βιβλιοθήκη αυτή είναι απαραίτητη διότι χρησιμοποιούμε το arduino mega, και όχι το uno για το οποίο έχει αναπτυχθεί η βιβλιοθήκη SD.

Γ) Λήψη μετρήσεων ατμοσφαιρικής πίεσης και θερμοκρασίας

```
#include <Wire.h>  //Η βιβλιοθήκη Wire επιτρέπει την επικοινωνία με I2C / TWI συσκευές. Στα Arduino boards που έχουν το rev3 layout το pin 20 SDA (data line) και το pin 21 SCL (clock line) βρίσκονται δίπλα στο AREF pin.

#include <Adafruit_MPL115A2.h> // Εισαγωγή της βιβλιοθήκης του αισθητηρίου MPL115A2 της Adafruit

Adafruit_MPL115A2 mpl115a2;

void setup(void)
{
 Serial.begin(9600);
 Serial.println("Getting barometric pressure ..."); //εμφάνισε το μήνυμα ”Λήψη βαρομετρικής πίεσης ”
 mpl115a2.begin();  //Ενεργοποίηση του αισθητηρίου για λήψη δεδομένων (μετρήσεων) - γίνεται σύνδεση με την βιβλιοθήκη MPL115A και τις παραμέτρους που περιέχει
}

void loop(void)
{
 float pressureKPA = 0, temperatureC = 0;  //Δήλωση των μεταβλητών pressureKPA και temperatureC ως δεκαδικούς αριθμούς

 mpl115a2.getPT(&pressureKPA,&temperatureC); // Λήψη των μετρήσεων από το αισθητήριο (χρήση της βιβλιοθήκης MPL115A) μέσω της συνάρτησης getPT()
 Serial.print("Pressure (kPa): "); Serial.print(pressureKPA, 4); Serial.print(" kPa  ");
Serial.print("Temp (*C): "); Serial.print(temperatureC, 1); Serial.println(" *C both measured together"); // Εμφάνιση στην σειριακή οθόνη μηνύματος σχετικά με τις μετρήσεις που θα πραγματοποιηθούν
 pressureKPA = mpl115a2.getPressure();  //Λήψη της ατμοσφαιρικής πίεσης και ορισμός της τιμής της στην μεταβλητή pressureKPA
Serial.print("Pressure (kPa): "); Serial.print(pressureKPA, 4); Serial.println(" kPa");
//Εμφανίζει το μήνυμα της τιμής της ατμοσφαιρικής πίεσης σε kPa
```

```
 temperatureC = mpl115a2.getTemperature();   //Λήψη της θερμοκρασίας και ορισμός της  τιμής της στην μεταβλητή temperatureC
 Serial.print("Temp (*C): "); Serial.print(temperatureC, 1); Serial.println(" *C");
//Εμφανίζει το μήνυμα της τιμής της θερμοκρασίας σε βαθμούς C
 delay(1000); //ορισμός της συχνότητας λήψης και εμφάνισης μετρήσεων
}
```

Δ) Τροποποίηση του κώδικα και αποθήκευση των δεδομένων στην κάρτα SD για τον αισθητήρα MPL115A2

```
#include <Wire.h>
#include <SPI.h>
#include <SD.h>
#include <Adafruit_MPL115A2.h>
#include <pin.h>

Adafruit_MPL115A2 mpl115a2;
const int chipSelect = 4; //Δήλωση του pin 4 (υπεύθυνο για την σειριακή επικοινωνία) ως σταθερά

void setup()
{
 Serial.begin(9600);   //Εναρξη της σειριακής επικοινωνίας
 Serial.println("Getting barometric pressure ...") //Εμφάνισε το μήνυμα ”Λήψη βαρομετρικής πίεσης ”
 mpl115a2.begin(); //Ενεργοποίηση του αισθητήρα MPL115A2
Serial.print("Initializing SD card..."); //Εμφανίζει το μήνυμα στην σειριακή οθόνη “προετοιμασία της κάρτα SD”

 pinMode(53, OUTPUT); //Το pin 53 πρέπει να δηλωθεί ως έξοδος διότι στο Mega είναι το hardware SS pin, ώστε να λειτουργεί σωστά η βιβλιοθήκη SD.

 if (!SD.begin(chipSelect)) {  //Ελέγχεται αν η κάρτα είναι τοποθετημένη και μπορεί να αρχικοποιηθεί
 Serial.println("Card failed, or not present");         //Αν δεν είναι τοποθετημένη η κάρτα SD ή δεν είναι δυνατή η επικοινωνία με αυτή τότε εμφάνισε το μήνυμα “Η επικοινωνία με την κάρτα απέτυχε ή αυτή δεν είναι τοποθετημένη”
return;
 }
 Serial.println("card initialized.");   //Αν η κάρτα είναι τοποθετημένη εμφάνισε το μήνυμα  “Αρχικοποίηση επικοινωνίας με την κάρτα”
}

void loop()
{
 float pressureKPA = 0, temperatureC = 0;   //Δηλώνουμε τις μεταβλητές pressureKPA και temperatureC ως δεκαδικούς αριθμούς
```

```
 mpl115a2.getPT(&pressureKPA,&temperatureC); // Λήψη των μετρήσεων από το αισθητήριο (χρήση της βιβλιοθήκης MPL115A)  μέσω της συνάρτησης getPT()

 Serial.print("Pressure (kPa): "); Serial.print(pressureKPA, 4); Serial.print(" kPa  "); Serial.print("Temp (°C): "); Serial.print(temperatureC, 1); Serial.println(" °C both measured together");
 delay(4000);  //ορισμός της συχνότητας λήψης και εμφάνισης μετρήσεων

 File dataFile = SD.open("datalog.txt", FILE_WRITE);  //Εντολή δημιουργίας και ανοίγματος του αρχείου  datalog.txt από την κάρτα SD. Η εντολή αυτή εισχωρείται  στην μεταβλητή dataFile τύπου File (αρχείο δεδομένων)

if (dataFile) {
        dataFile.print("temp(°C):"),dataFile.print(temperatureC),
        dataFile.print("  press(Kpa):"), dataFile.println(pressureKPA);
        dataFile.close();
 }
 else {
        Serial.println("error opening datalog.txt"); //Αν είναι αδύνατο το άνοιγμα του αρχείου εμφάνισε το μήνυμα “σφάλμα εντοπισμού του αρχείου datalog.txt ”
 }
}
```

Ε) Απομακρυσμένη πρόσβαση στις μετρήσεις ατμοσφαιρικής πίεσης και θερμοκρασίας

```
#include <Wire.h>
#include <SPI.h>
#include <SD.h>
#include <Ethernet.h>  (1)

#include <Adafruit_MPL115A2.h>
#include <pin.h>

Adafruit_MPL115A2 mpl115a2;

byte mac[] = {0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED}; //Ορισμός του mac address που θα έχει  το Arduino και το οποίο θα πρέπει να είναι  μοναδικό μέσα στο lan.
IPAddress ip(192, 168, 1, 177);   //Ορισμός του ip address που θα έχει το Arduino (θα πρέπει να είναι μοναδικό μέσα στο lan.)

EthernetServer server(80); //Δημιουργία instance του EthernetServer object μαζί με το port number της συσκευής. Ο server στο Arduino θα ακούει για  εισερχόμενες συνδέσεις στο port 80.

const int chipSelect = 4; //Δήλωση του pin 4 (υπεύθυνο για την σειριακή επικοινωνία) ως σταθερά

void setup()
{
```

```
 Serial.begin(9600);
 Ethernet.begin(mac, ip);  //Εκκίνηση της επικοινωνίας Ethernet ρυθμίζοντας την IP και το MAC address της συσκευής.

 server.begin();  //Εκκίνηση του server για εισερχόμενες συνδέσεις χρησιμοποιώντας την εντολή begin().

 Serial.print("server is at ");
 Serial.println(Ethernet.localIP());
 Serial.println("Getting barometric pressure ..."); //Εμφάνισε το μήνυμα ”Λήψη βαρομετρικής πίεσης ”
 mpl115a2.begin();  //Ενεργοποίηση του αισθητήρα MPL115A2
Serial.print("Initializing SD card..."); //Εμφανίζει το μήνυμα στην σειριακή οθόνη “προετοιμασία της κάρτα SD”

 pinMode(53, OUTPUT);  //Το pin 53 πρέπει να δηλωθεί ως έξοδος διότι στο Mega είναι το  hardware SS pin, ώστε να λειτουργεί σωστά η βιβλιοθήκη SD.

  if (!SD.begin(chipSelect)) {  //Ελέγχεται αν η κάρτα είναι τοποθετημένη και μπορεί να αρχικοποιηθεί
Serial.println("Card failed, or not present"); //Αν δεν είναι τοποθετημένη η κάρτα SD ή  δεν είναι δυνατή η επικοινωνία με αυτή τότε εμφάνισε το μήνυμα “Η επικοινωνία με την κάρτα απέτυχε ή αυτή δεν είναι τοποθετημένη”
return;
 }
 Serial.println("card initialized.");  //Αν η κάρτα είναι τοποθετημένη, εμφάνισε το μήνυμα  “Αρχικοποίηση επικοινωνίας με την κάρτα”
}

void loop()
{
 float pressureKPA = 0, temperatureC = 0;  //Δηλώνουμε τις μεταβλητές pressureKPA και temperatureC ως δεκαδικούς αριθμούς

 mpl115a2.getPT(&pressureKPA,&temperatureC); // Λήψη των μετρήσεων από το αισθητήριο (χρήση της βιβλιοθήκης MPL115A) μέσω της συνάρτησης getPT()

 Serial.print("Pressure (kPa): "); Serial.print(pressureKPA, 4); Serial.print(" kPa  ");
 Serial.print("Temp (°C): "); Serial.print(temperatureC, 1); Serial.println(" °C both measured together");
 delay(4000);  //ορισμός της συχνότητας λήψης και εμφάνισης μετρήσεων

 File dataFile = SD.open("datalog.txt", FILE_WRITE); //Εντολή δημιουργίας και ανοίγματος  του αρχείου  datalog.txt από την  κάρτα SD. Η εντολή αυτή εισχωρείται  στην μεταβλητή dataFile τύπου File (αρχείο δεδομένων)

 if (dataFile) {
        dataFile.print("temp(°C):"),dataFile.print(temperatureC),
        dataFile.print("  press(Kpa):"), dataFile.println(pressureKPA);
        dataFile.close();
```

```
}

else {
        Serial.println("error opening datalog.txt"); //Αν είναι αδύνατο το άνοιγμα του αρχείου εμφάνισε το μήνυμα “σφάλμα εντοπισμού του αρχείου datalog.txt ”
}
EthernetClient client = server.available(); // Για τις εισερχόμενες αιτήσεις (clients) δηλαδή για απομακρυσμένη πρόσβαση στις σελίδες που είναι φορτωμένες στο arduino. απαιτείται η  δημιουργία ενός instance τύπου EthernetClient, που θα το χρησιμοποιήσουμε ώστε να ελέγχουμε αν υπάρχουν διαθέσιμα δεδομένα για διάβασμα.
if (client) {
       Serial.println("new client");

        boolean currentLineIsBlank = true;
        while (client.connected()) {
        if (client.available()) {
        char c = client.read();
        Serial.write(c);

//Αν υπάρχουν clients και δεδομένα να διαβαστούν, διάβασε 1 byte (έναν δηλαδή χαρακτήρα του client) και αποθήκευσέ τον στον c.


        if (c == '\n' && currentLineIsBlank) {

        client.println("HTTP/1.1 200 OK");
        client.println("Content-Type: text/html");
        client.println("Connection:  close");  // Η σύνδεση θα κλείσει αφού ολοκληρωθεί η  απάντηση
        client.println("Refresh: 5");  // Ανανέωνε τη σελίδα κάθε πέντε λεπτά
        client.println();
        client.println("<!DOCTYPE HTML>");
        client.println("<html>");

        //Εμφάνιση των μετρήσεων στην ιστοσελίδα
        client.print("Temperature");
        client.print(temperatureC);
        client.print("pressure");
        client.print(pressureKPA);
        client.println("<br />");

        client.println("</html>");
        break;
        } //Προβολή των μετρήσεων σε κώδικα html

        if (c == '\n') { // Έλεγχος συνθήκης. Αν αυτό που παίρνουμε ισοδυναμεί με “n”, τότε ξεκινάμε να γράφουμε μια νέα σειρά
        currentLineIsBlank = true;
        }
```

```
        else if (c != '\r') { //Συνέχεια συνθήκης. Αν αυτό που παίρνουμε δεν είναι το "r", τότε θα γράψουμε ένα χαρακτήρα, στην ίδια γραμμή
        currentLineIsBlank = false;
        }
        }
        }
        delay(1); // Δίνουμε λίγο χρόνο στον περιηγητή για να λάβει τα δεδομένα
       client.stop();  //Κλείνουμε τη σύνδεση
        Serial.println("client disconnected") //Εμφανίζει το μήνυμα "Η σύνδεση τερματίστηκε"
 }
}
```

(1) Αυτή η βιβλιοθήκη επιτρέπει στο Arduino board μέσω του Ethernet Shield να αποκτήσει δικτυακές δυνατότητες. Μπορεί να λειτουργήσει σαν server δεχόμενος εισερχόμενες συνδέσεις ή ως client δημιουργώντας εξερχόμενες. Η βιβλιοθήκη υποστηρίζει μέχρι 4 ταυτόχρονες συνδέσεις (εισερχόμενες ή εξερχόμενες ή συνδυασμός αυτών). Το Arduino επικοινωνεί με το Ethernet Shield μέσω του SPI bus. Αυτό στο Arduino Mega γίνεται μέσω των pins 50, 51, 52 και του pin 10 ως Slave Select (SS). Επιπλέον το hardware SS pin 53 πρέπει στο κώδικα μας να είναι δηλωμένο πάντα ως OUTPUT διαφορετικά δεν θα μπορεί να δουλέψει σωστά το SPI interface.

ΣΤ) Απομακρυσμένη πρόσβαση στις μετρήσεις υγρασίας και θερμοκρασίας

```
#include <SPI.h>
#include <SD.h>
#include <Ethernet.h>

#include <pin.h>
#include <dht.h>
dht DHT;

byte mac[] = {0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED}; //Ορισμός του mac address που θα έχει  το Arduino και το οποίο θα πρέπει να είναι  μοναδικό μέσα στο lan.
IPAddress ip(192, 168, 2, 4); //Ορισμός του ip address που θα έχει το Arduino (θα πρέπει να είναι μοναδικό μέσα στο lan.)
EthernetServer server(80); //Δημιουργία instance του EthernetServer object μαζί με το port number της συσκευής. Ο server στο Arduino θα ακούει για  εισερχόμενες συνδέσεις στο port 80.
const int chipSelect = 4; //Δήλωση του pin 4 (υπεύθυνο για την σειριακή επικοινωνία) ως σταθερά
const int dhtPin = 42;
int chk;
float hum;
float temp;

void setup()
{
```

```
  Serial.begin(9600);

}
  Ethernet.begin(mac, ip); //Εκκίνηση της επικοινωνίας Ethernet ρυθμίζοντας την IP και το MAC address της συσκευής.
  server.begin();//Εκκίνηση του server για εισερχόμενες συνδέσεις χρησιμοποιώντας την εντολή begin().
  Serial.print("server is at ");
   Serial.println(Ethernet.localIP());
  Serial.println("waiting for sync");
  Serial.print("Initializing SD card..."); //Εμφανίζει το μήνυμα στην σειριακή οθόνη “προετοιμασία της κάρτα SD”

  pinMode(53, OUTPUT); //Το pin 53 πρέπει να δηλωθεί ως έξοδος διότι στο Mega είναι το  hardware SS pin, ώστε να λειτουργεί σωστά η βιβλιοθήκη SD.

    if (!SD.begin(chipSelect)) {//Ελέγχεται αν η κάρτα είναι τοποθετημένη και μπορεί να αρχικοποιηθεί

    Serial.println("Card failed, or not present");//Αν δεν είναι τοποθετημένη η κάρτα SD ή  δεν είναι δυνατή η επικοινωνία με αυτή τότε εμφάνισε το μήνυμα “Η επικοινωνία με την κάρτα απέτυχε ή αυτή δεν είναι τοποθετημένη”
    return;
  }
  Serial.println("card initialized."); //Αν η κάρτα είναι τοποθετημένη, εμφάνισε το μήνυμα  “Αρχικοποίηση επικοινωνίας με την κάρτα”
}

void loop()
{
//Εντολές για τη μέτρηση της θερμοκρασίας και της υγρασίας
      chk = DHT.read22(dhtPin);
    hum = DHT.humidity;
    temp= DHT.temperature;

    //Εντολές εμφάνισης των μετρούμενων μεγεθών
    Serial.print("Humidity: ");
    Serial.print(hum);
    Serial.print(" %, Temp: ");
    Serial.print(temp);
    Serial.println(" Celsius");
    delay(6000);.

  String dataString = "";
  String dataString_2 = "";

  for (int digitalPin = 42; digitalPin < 43; digitalPin++) {
    int sensor = digitalRead(digitalPin);
    dataString += String(temp);
    dataString_2 += String(hum);
```

```
    if (digitalPin < 44) {
      dataString += ",";
    }

    EthernetClient client = server.available(); // Για τις εισερχόμενες αιτήσεις (clients) δηλαδή για απομακρυσμένη πρόσβαση στις σελίδες που είναι φορτωμένες στο arduino. απαιτείται η  δημιουργία ενός instance τύπου EthernetClient, που θα το χρησιμοποιήσουμε ώστε να ελέγχουμε αν υπάρχουν διαθέσιμα δεδομένα για διάβασμα.
     if (client) {
      Serial.println("new client");

      boolean currentLineIsBlank = true;
      while (client.connected()) {
        if (client.available()) {
          char c = client.read();
          Serial.write(c); //Αν υπάρχουν clients και δεδομένα να διαβαστούν, διάβασε 1 byte (έναν δηλαδή χαρακτήρα του client) και αποθήκευσέ τον στον c.


          if (c == '\n' && currentLineIsBlank) {

            client.println("HTTP/1.1 200 OK");
            client.println("Content-Type: text/html");
            client.println("Connection: close"); // Η σύνδεση θα κλείσει αφού ολοκληρωθεί η  απάντηση
             client.println("Refresh: 5");  // Ανανέωνε τη σελίδα κάθε πέντε λεπτά
            client.println();
            client.println("<!DOCTYPE HTML>");
            client.println("<html>"); //Προβολή των μετρήσεων σε κώδικα html

            //Εμφάνιση των μετρήσεων στην ιστοσελίδα
            for (int digitalPin = 42; digitalPin < 43; digitalPin++) {
              int sensor = digitalRead(digitalPin);
              client.print("temperature: ");
              client.print(temp);
              client.println(" Celsius");
              client.print(" ");
              client.print("humidity: ");
              client.print(hum);
              client.print(" % ");
              client.print(" ");

              client.println("<br />");
            }
            client.println("</html>");
            break;
          }
          if (c == '\n') { // Έλεγχος συνθήκης. Αν αυτό που παίρνουμε ισοδυναμεί με “n”, τότε ξεκινάμε να γράφουμε μια νέα σειρά
```

```
        currentLineIsBlank = true;
       }
       else if (c != '\r') { //Συνέχεια συνθήκης. Αν αυτό που παίρνουμε δεν είναι το “r”, τότε θα γράψουμε ένα χαρακτήρα, στην ίδια γραμμή
        currentLineIsBlank = false;
       }
      }
    }
      delay(100); // Δίνουμε λίγο χρόνο στον περιηγητή για να λάβει τα δεδομένα

    client.stop(); //Κλείνουμε τη σύνδεση
    Serial.println("client disconnected");//Εμφανίζει το μήνυμα “Η σύνδεση τερματίστηκε”
  }

  }

  File dataFile = SD.open("datalog.txt", FILE_WRITE); //Εντολή δημιουργίας και ανοίγματος του αρχείου datalog.txt από την κάρτα SD. Η εντολή αυτή εισχωρείται στην μεταβλητή dataFile τύπου File (αρχείο δεδομένων)

  if (dataFile) {
    dataFile.print("temp:"),dataFile.print(dataString),
    dataFile.print("hum:"), dataFile.print(dataString_2);
     dataFile.print(" ");

  dataFile.println();
    dataFile.close();
    // print to the serial port too:
    Serial.println("temp:"),Serial.println(dataString);
    Serial.println("hum:"), Serial.println(dataString_2);
  }
    else {
    Serial.println("error opening datalog.txt");//Αν είναι αδύνατο το άνοιγμα του αρχείου εμφάνισε το μήνυμα “σφάλμα εντοπισμού του αρχείου datalog.txt ”
  }
}

}
```

Ζ) Ολοκληρωμένος κώδικας για τον αισθητήρα RHT 03

```
#include <SPI.h>
#include <SD.h>
#include <Ethernet.h>
#include <Time.h> //Βιβλιοθήκη για τη λήψη ώρας και ημερομηνίας από το διαδίκτυο
                  ,μεσω του Ethernet Shield
#include <EthernetUdp.h>
#include <pin.h>
```

```
#include <dht.h>

dht DHT;

byte mac[] = {0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED};
IPAddress ip(192, 168, 2, 4);
EthernetServer server(80);

IPAddress timeServer(132, 163, 4, 101); // time-a.timefreq.bldrdoc.gov, σύνδεση με
                                        server για τη λήψη της τρέχουσας ώρας
                                        και ημερομηνίας
const int chipSelect = 4;
const int dhtPin = 42;
int chk;

float hum;
float temp;
const int timeZone = 3; //Η ζώνη ώρας της Ελλάδας
EthernetUDP Udp;
unsigned int localPort = 8888;

void setup()

{

  Serial.begin(9600);

 if (Ethernet.begin(mac) == 0) {

      while (1) {

    Serial.println("Failed to configure Ethernet using DHCP");

    delay(10000);

   }

}

 Ethernet.begin(mac, ip);

 server.begin();

 Serial.print("server is at ");

 Serial.println(Ethernet.localIP());

 Serial.print("IP number assigned by DHCP is ");

 Serial.println(Ethernet.localIP());

 Udp.begin(localPort); //Έναρξη επικοινωνίας με τον time server


 Serial.println("waiting for sync");
```

```
  setSyncProvider(getNtpTime);
  Serial.print("Initializing SD card...");
   pinMode(53, OUTPUT);
  if (!SD.begin(chipSelect)) {
    Serial.println("Card failed, or not present");
    return;
  }
  Serial.println("card initialized.");
}
time_t prevDisplay = 0;
void loop()
{
if (timeStatus() != timeNotSet) {
    if (now() != prevDisplay) {
      prevDisplay = now();
      digitalClockDisplay();
    }
  }

    chk = DHT.read22(dhtPin);
    hum = DHT.humidity;
    temp= DHT.temperature;
    Serial.print("Humidity: ");
    Serial.print(hum);
    Serial.print(" %, Temp: ");
    Serial.print(temp);
    Serial.println(" Celsius");
    delay(6000);
  String dataString = "";
  String dataString_2 = "";
  for (int digitalPin = 42; digitalPin < 43; digitalPin++) {
    int sensor = digitalRead(digitalPin);
```

```
  dataString += String(temp);
  dataString_2 += String(hum);
  if (digitalPin < 44) {
    dataString += ",";
  }

  EthernetClient client = server.available();
if (client) {
  Serial.println("new client");
  boolean currentLineIsBlank = true;
  while (client.connected()) {
    if (client.available()) {
      char c = client.read();
      Serial.write(c);
      if (c == '\n' && currentLineIsBlank) {
        client.println("HTTP/1.1 200 OK");
        client.println("Content-Type: text/html");
        client.println("Connection: close");
        client.println("Refresh: 5");
        client.println();
        client.println("<!DOCTYPE HTML>");
        client.println("<html>");

        for (int digitalPin = 42; digitalPin < 43; digitalPin++) {
          int sensor = digitalRead(digitalPin);
          client.print("temperature: ");
          client.print(temp);
          client.println(" Celsius");
          client.print(" ");
          client.print("humidity: ");
          client.print(hum);
          client.print(" % ");
```

```
      client.print("  ");
      client.print(" ");
      client.print(" ");
      client.print("time:"); //Εμφάνιση της ώρας και της ημερομηνίας κατά την
                                απομακρυσμένη πρόσβαση
        client.print(hour());
        client.print(":");
        client.print(minute());
        client.print(":");
        client.print(second());
        client.print(" ");
        client.print(" ");
        client.print("date:");
        client.print(" ");
        client.print(day());
        client.print("/");
        client.print(" ");
        client.print(month());
        client.print("/");
        client.print(" ");
        client.print(year());
        client.println();
       client.println("<br />");
    }
    client.println("</html>");
    break;
  }
  if (c == '\n') {
currentLineIsBlank = true;
  }
  else if (c != '\r') {
  currentLineIsBlank = false;
```

```
      }
     }
    }
      delay(100);
       client.stop();
   Serial.println("client disconnected");
  }

  }
   File dataFile = SD.open("datalog.txt", FILE_WRITE);

    if (dataFile) {
 dataFile.print("temp:"),dataFile.print(dataString),
 dataFile.print("hum:"), dataFile.print(dataString_2);
 dataFile.print(" ");
dataFile.print("time: "); //Εγγραφή της ώρας και της ημερομηνίας κατά τη λήψη των
                          μετρήσεων στην  κάρτα SD
  dataFile.print(hour());
   dataFile.print(":");
  dataFile.print(minute());
   dataFile.print(":");
  dataFile.print(second());
  dataFile.print(" ");
   dataFile.print("date:");
  dataFile.print(day());
  dataFile.print("/");
  dataFile.print(month());
  dataFile.print("/");
  dataFile.print(year());
  dataFile.println();
    dataFile.close();
    Serial.println("temp:"),Serial.println(dataString);
```

```
    Serial.println("hum:"), Serial.println(dataString_2);
  }
    else {
    Serial.println("error opening datalog.txt");
  }
}
void digitalClockDisplay(){ //Συνάρτηση της βιβλιοθήκης Time
  Serial.print(hour());
  printDigits(minute());
  printDigits(second());
  Serial.print(" ");
  Serial.print(day());
  Serial.print(" ");
  Serial.print(month());
  Serial.print(" ");
  Serial.print(year());
  Serial.println();
}

void printDigits(int digits){ //Συνάρτηση της βιβλιοθήκης Time
  Serial.print(":");
  if(digits < 10)
    Serial.print('0');
  Serial.print(digits);
}

const int NTP_PACKET_SIZE = 48;
byte packetBuffer[NTP_PACKET_SIZE];

time_t  getNtpTime()//Συνάρτηση της βιβλιοθήκης Time για το συγχρονισμό με τον
                    Time server
{
```

```
  while (Udp.parsePacket() > 0) ;
  Serial.println("Transmit NTP Request");
  sendNTPpacket(timeServer);
  uint32_t beginWait = millis();
  while (millis() - beginWait < 1500) {
    int size = Udp.parsePacket();
    if (size >= NTP_PACKET_SIZE) {
      Serial.println("Receive NTP Response");
      Udp.read(packetBuffer, NTP_PACKET_SIZE);  // read packet into the buffer
      unsigned long secsSince1900;
      secsSince1900 =  (unsigned long)packetBuffer[40] << 24;
      secsSince1900 |= (unsigned long)packetBuffer[41] << 16;
      secsSince1900 |= (unsigned long)packetBuffer[42] << 8;
      secsSince1900 |= (unsigned long)packetBuffer[43];
      return secsSince1900 - 2208988800UL + timeZone * SECS_PER_HOUR;
    }
  }
  Serial.println("No NTP Response ");
  return 0;
}
void sendNTPpacket(IPAddress &address) //Συνάρτηση της βιβλιοθήκης Time

{
  memset(packetBuffer, 0, NTP_PACKET_SIZE);

  packetBuffer[0] = 0b11100011;
  packetBuffer[1] = 0;
  packetBuffer[2] = 6;
  packetBuffer[3] = 0xEC;
    packetBuffer[12]  = 49;
  packetBuffer[13]  = 0x4E;
  packetBuffer[14]  = 49;
```

```
  packetBuffer[15]  = 52;
  Udp.beginPacket(address, 123);
  Udp.write(packetBuffer, NTP_PACKET_SIZE);
  Udp.endPacket();
}
```

Η) Ολοκληρωμένος κώδικας για τον αισθητήρα MPL 115A2

```
#include <SPI.h>
#include <SD.h>
#include <Ethernet.h>
#include <Wire.h>
#include <Time.h>  //Βιβλιοθήκη για τη λήψη ώρας και ημερομηνίας από το διαδίκτυο
                   ,μεσω του Ethernet Shield
#include <EthernetUdp.h>
#include <pin.h>
#include <Adafruit_MPL115A2.h>

Adafruit_MPL115A2 mpl115a2;

byte mac[] = {0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED};
IPAddress ip(192, 168, 2, 4);
EthernetServer server(80);
IPAddress timeServer(132, 163, 4, 101); // time-a.timefreq.bldrdoc.gov, σύνδεση με
                                           server για τη λήψη της τρέχουσας ώρας
                                           και ημερομηνίας
const int chipSelect = 4;
int chk;

const int timeZone = 3; //Η ζώνη ώρας της Ελλάδας
EthernetUDP Udp;
unsigned int localPort = 8888;
```

```
void setup()
{
  Serial.begin(9600);
  Serial.println("Getting barometric pressure ...");
  mpl115a2.begin();
if (Ethernet.begin(mac) == 0) {
    while (1) {
      Serial.println("Failed to configure Ethernet using DHCP");
      delay(10000);
    }
}
  Ethernet.begin(mac, ip);
  server.begin();
  Serial.print("server is at ");
  Serial.println(Ethernet.localIP());
  Serial.print("IP number assigned by DHCP is ");
  Serial.println(Ethernet.localIP());
  Udp.begin(localPort); //Έναρξη επικοινωνίας με τον time server
  Serial.println("waiting for sync");
  setSyncProvider(getNtpTime);
  Serial.print("Initializing SD card...");
  pinMode(53, OUTPUT);

  if (!SD.begin(chipSelect)) {
    Serial.println("Card failed, or not present");
    return;
  }
  Serial.println("card initialized.");
}
time_t prevDisplay = 0;
void loop()
{
```

```
  float pressureKPA = 0, temperatureC = 0;
if (timeStatus() != timeNotSet) {
   if (now() != prevDisplay) {
     prevDisplay = now();
     digitalClockDisplay();
   }
  }

  mpl115a2.getPT(&pressureKPA,&temperatureC);
  Serial.print("Pressure (kPa): "); Serial.print(pressureKPA, 4); Serial.print(" kPa  ");
  Serial.print("Temp (*C): "); Serial.print(temperatureC, 1); Serial.println(" *C both
measured together");

  pressureKPA = mpl115a2.getPressure();
  Serial.print("Pressure (kPa): "); Serial.print(pressureKPA, 4); Serial.println(" kPa");

  temperatureC = mpl115a2.getTemperature();
  Serial.print("Temp (*C): "); Serial.print(temperatureC, 1); Serial.println(" *C");

   Serial.print("Pressure: ");
   Serial.print(pressureKPA);
   Serial.print(" kPa , Temp: ");
   Serial.print(temperatureC);
   Serial.println(" Celsius");
   delay(6000);.

   EthernetClient client = server.available();
  if (client) {
   Serial.println("new client");
      boolean currentLineIsBlank = true;
   while (client.connected()) {
     if (client.available()) {
```

```
 char c = client.read();
 Serial.write(c);
 if (c == '\n' && currentLineIsBlank) {
client.println("HTTP/1.1 200 OK");
   client.println("Content-Type: text/html");
   client.println("Connection: close");
   client.println("Refresh: 5");
   client.println();
   client.println("<!DOCTYPE HTML>");
   client.println("<html>");

     client.print("temperature: ");
     client.print(temperatureC);
     client.println(" Celsius");
     client.print(" ");
     client.print("pressure: ");
     client.print(pressureKPA);
     client.print(" kPa ");
     client.print("  ");
     client.print(" ");
     client.print(" ");
     client.print("time:"); //Εμφάνιση της ώρας και της ημερομηνίας κατά την
                                απομακρυσμένη πρόσβαση
       client.print(hour());
       client.print(":");
       client.print(minute());
       client.print(":");
       client.print(second());
       client.print(" ");
       client.print(" ");
       client.print("date:");
       client.print(" ");
```

```
            client.print(day());
            client.print("/");
            client.print(" ");
            client.print(month());
            client.print("/");
            client.print(" ");
            client.print(year());
            client.println();

          client.println("<br />");


        client.println("</html>");
        break;
      }
      if (c == '\n') {
       currentLineIsBlank = true;
      }
      else if (c != '\r') {
       currentLineIsBlank = false;
      }
    }
  }
   delay(100);
  client.stop();
  Serial.println("client disconnected");
}

File dataFile = SD.open("datalog.txt", FILE_WRITE);

  if (dataFile) {
 dataFile.print("temp:"),dataFile.print(temperatureC),
```

```
  dataFile.print("press:"), dataFile.print(pressureKPA);
   dataFile.print(" ");
   dataFile.print("time: "); //Εγγραφή της ώρας και της ημερομηνίας κατά τη λήψη των
                               μετρήσεων στην  κάρτα SD
  dataFile.print(hour());
   dataFile.print(":");
  dataFile.print(minute());
   dataFile.print(":");
  dataFile.print(second());
  dataFile.print(" ");
   dataFile.print("date:");
  dataFile.print(day());
  dataFile.print("/");
  dataFile.print(month());
  dataFile.print("/");
  dataFile.print(year());
  dataFile.println();
    dataFile.close();
   Serial.println("temp:"),Serial.println(temperatureC);
    Serial.println("press:"), Serial.println(pressureKPA);
  }
    else {
    Serial.println("error opening datalog.txt");
  }
}
void digitalClockDisplay(){//Συνάρτηση της βιβλιοθήκης Time
Serial.print(hour());
  printDigits(minute());
  printDigits(second());
  Serial.print(" ");
  Serial.print(day());
  Serial.print(" ");
```

```
  Serial.print(month());
  Serial.print(" ");
  Serial.print(year());
  Serial.println();
}

void printDigits(int digits){ //Συνάρτηση της βιβλιοθήκης Time
  Serial.print(":");
  if(digits < 10)
    Serial.print('0');
  Serial.print(digits);
}
const int NTP_PACKET_SIZE = 48;
byte packetBuffer[NTP_PACKET_SIZE];

time_t  getNtpTime()//Συνάρτηση της βιβλιοθήκης Time για το συγχρονισμό με τον
                    Time server
{
  while (Udp.parsePacket() > 0) ;
  Serial.println("Transmit NTP Request");
  sendNTPpacket(timeServer);
  uint32_t beginWait = millis();
  while (millis() - beginWait < 1500) {
    int size = Udp.parsePacket();
    if (size >= NTP_PACKET_SIZE) {
      Serial.println("Receive NTP Response");
      Udp.read(packetBuffer, NTP_PACKET_SIZE);  // read packet into the buffer
      unsigned long secsSince1900;
      secsSince1900 =  (unsigned long)packetBuffer[40] << 24;
      secsSince1900 |= (unsigned long)packetBuffer[41] << 16;
      secsSince1900 |= (unsigned long)packetBuffer[42] << 8;
      secsSince1900 |= (unsigned long)packetBuffer[43];
```

```
      return secsSince1900 - 2208988800UL + timeZone * SECS_PER_HOUR;
    }
  }
  Serial.println("No NTP Response");
  return 0;
}
void sendNTPpacket(IPAddress &address) //Συνάρτηση της βιβλιοθήκης Time
{
    memset(packetBuffer, 0, NTP_PACKET_SIZE);
  packetBuffer[0] = 0b11100011;
  packetBuffer[1] = 0;
  packetBuffer[2] = 6;
  packetBuffer[3] = 0xEC;
    packetBuffer[12]  = 49;
  packetBuffer[13]  = 0x4E;
  packetBuffer[14]  = 49;
  packetBuffer[15]  = 52;
   Udp.beginPacket(address, 123);
  Udp.write(packetBuffer, NTP_PACKET_SIZE);
  Udp.endPacket();
```

Θ) Αλγόριθμος πρόβλεψης του καιρού (Zambretti)

```
#include <Wire.h>
#include <Adafruit_MPL115A2.h>
Adafruit_MPL115A2 mpl115a2;

void setup(void)
{
  Serial.begin(9600);
  float z=0; //Δήλωση της εξίσωσης z ως πραγματικό αριθμό
  float trend=0;  //Δήλωση της μεταβλητής trend, για τις διαφορές
```

```
  int x=0; //Δήλωση του x ως ακέραιο
  float metriseis[100], diafores[100]; //Δήλωση πινάκων
  Serial.println("Getting barometric pressure ...");
  mpl115a2.begin();
}

void loop(void)
{
  float pressureKPA = 0, temperatureC = 0;
   float z=0;
   float trend=0;
   int x;
   float metriseis[100], diafores[100],trexousa;
  mpl115a2.getPT(&pressureKPA,&temperatureC);
  Serial.print("Pressure (kPa): "); Serial.print(pressureKPA, 4); Serial.print(" kPa  ");
  Serial.print("Temp (*C): "); Serial.print(temperatureC, 1); Serial.println(" *C both
measured together");
  pressureKPA = mpl115a2.getPressure();
  Serial.print("Pressure (kPa): "); Serial.print(pressureKPA, 4); Serial.println(" kPa");
  temperatureC = mpl115a2.getTemperature();
  Serial.print("Temp (*C): "); Serial.print(temperatureC, 1); Serial.println(" *C");
  metriseis[1]=pressureKPA*100; //Μετατροπή από kpa σε  hpa, μέτρηση τρέχουσας
                                  – πρώτης τιμής

for(x=2; x<=8; x=x+1) //Εκκίνηση επανάληψης για τη λήψη μετρήσεων ανά 3 ώρες –
                        7 μετρήσεις τη μέρα
   {
     delay (180000); //Καθυστέρηση 3 ώρες
  mpl115a2.getPT(&pressureKPA,&temperatureC);
  Serial.print("Pressure (kPa): "); Serial.print(pressureKPA, 4); Serial.print(" kPa  ");
  Serial.print("Temp (*C): "); Serial.print(temperatureC, 1); Serial.println(" *C both
measured together");
  pressureKPA = mpl115a2.getPressure();
```

```
  Serial.print("Pressure (kPa): "); Serial.print(pressureKPA, 4); Serial.println(" kPa");
  temperatureC = mpl115a2.getTemperature();
  Serial.print("Temp (*C): "); Serial.print(temperatureC, 1); Serial.println(" *C");


    metriseis[x]=pressureKPA*100;
    diafores[x-1]=(metriseis[x]-metriseis[x-1]); //Υπολογισμός διαφοράς τρέχουσας
                                                  τιμής της ατμοσφαιρικής πίεσης
                                                  από την τιμή αυτής 3 ώρες πρίν
    trend=diafores[x-1]; //Η τιμή της διαφοράς
    Serial.print("to trend:");
    Serial.print(trend);
    Serial.print("metrisi x :");
    Serial.print(metriseis[x]);


  mpl115a2.getPT(&pressureKPA,&temperatureC);
  pressureKPA = mpl115a2.getPressure();



    trexousa=pressureKPA*100;
    if(trend >0) //Συνθήκες για την κατάλληλη επιλογή της εξίσωσης z - Αν η τιμή της
                 διαφοράς είναι θετική
      z=(130-((trexousa)/81.0));
    else if ( trend = 0 ) //Αν η τιμή της διαφοράς είναι μηδενική
      z=(147-((5*(trexousa))/376.0));
    else if (trend <0) //Αν η τιμή της διαφοράς είναι αρνητική
       z=(179-((2*(trexousa))/129.0));
    Serial.print("to z:");
    Serial.print(z);
if (z>=0.0 && z<2.0) // Συνθήκες για την κατάλληλη πρόβλεψη του καιρού, ανάλογα
                        με την τιμή της συνάρτησης z
    Serial.print("aithrios-ametavlitos\n");
else if (z>=2.0 && z<3.0)
    Serial.print("aithrios\n");
```

```
else if (z>=3.0 && z<4.0)
   Serial.print("aithrios-metavallomenos\n");
else if (z>=4.0 && z<5.0)
   Serial.print("sxedon aithrios-argotera pithanon elafres vroxoptoseis\n");
else if (z>=5.0 && z<6.0)
   Serial.print("elafres vroxoptoseis-en sunexeia astatos\n");
else if (z>=6.0 && z<7.0)
   Serial.print("astatos-argotera vroxoptoseis\n");
else if (z>=7.0 && z<8.0)
   Serial.print("vroxoptoseis kata diastimata- argotera epidinosi\n");
else if (z>=8.0 && z<9.0)
   Serial.print("vroxoptoseis ana diastimata- en sunexeia edona astatos\n");
else if (z>=9.0 && z<10.0)
   Serial.print("edona astatos- vroxoptoseis\n");
else if (z>=10.0 && z<11.0)
   Serial.print("aithrios-ametavlitos\n");
else if (z>=11.0 && z<12.0)
   Serial.print("aithrios\n");
else if (z>=12.0 && z<13.0)
   Serial.print("aithrios- pithanes elafres vroxoptoseis\n");
else if (z>=13.0 && z<14.0)
   Serial.print("sxedon aithrios- polu pithanon elafres vroxoptoseis\n");
else if (z>=14.0 && z<15.0)
   Serial.print("elafres vroxoptoseis me diastimata iliofaneias\n");
else if (z>=15.0 && z<16.0)
   Serial.print("metavlitos me merikes vroxoptoseis\n");
else if (z>=16.0 && z<17.0)
   Serial.print("astatos-vroxoptoseis ana diastimata\n");
else if (z>=17.0 && z<18.0)
   Serial.print("suxnes vroxoptoseis\n");
else if (z>=18.0 && z<19.0)
   Serial.print("endona astatos- vroxoptoseis\n");
```

```
else if (z>=19.0 && z<20.0)
   Serial.print("thielodeis edones vroxoptoseis\n");
else if (z>=20.0 && z<21.0)
   Serial.print("aithrios- ametavlitos\n");
else if (z>=21.0 && z<22.0)
   Serial.print("aithrios\n");
else if (z>=22.0 && z<23.0)
   Serial.print("metavoli se aithrio\n");
else if (z>=23.0 && z<24.0)
   Serial.print("sxedon aithrios- se veltiosi\n");
else if (z>=24.0 && z<25.0)
   Serial.print("sxedon aithrios- grigora pithanes elafres vroxoptoseis\n");
else if (z>=25.0 && z<26.0)
   Serial.print("arxika elafres vroxoptoseis- veltiosi\n");
else if (z>=26.0 && z<27.0)
   Serial.print("metavlitos- se veltiosi\n");
else if (z>=27.0 && z<28.0)
   Serial.print("sxedon astatos- veltiosi stin imera");
else if (z>=28.0 && z<29.0)
   Serial.print("astatos- pithani veltiosi\n");
else if (z>=29.0 && z<30.0)
   Serial.print("astatos- aithrios ana diastimata\n");
else if (z>=30.0 && z<31.0)
   Serial.print("endona astatos- aithrios ana diastimata\n");
else if (z>=31.0 && z<32.0)
   Serial.print("thielodis- pithani veltiosi\n");
else if (z>=32.0)
   Serial.print("thielodeis- endones vroxoptoseis\n");
  }
}
```

Η) Κώδικας PLX-DAQ

```
#include <Wire.h>
#include <Adafruit_MPL115A2.h>
Adafruit_MPL115A2 mpl115a2;
int row = 0;
void setup(void)
{
  Serial.begin(9600);
  Serial.println("Getting barometric pressure ...");
  mpl115a2.begin();
  Serial.println("CLEARDATA");
  Serial.println("LABEL,Time,temperature,Pressure");
}
   void loop(void)
{
  float pressureKPA = 0, temperatureC = 0;

  mpl115a2.getPT(&pressureKPA,&temperatureC);
  Serial.print("Pressure (kPa): "); Serial.print(pressureKPA, 4); Serial.print(" kPa  ");
  Serial.print("Temp (*C): "); Serial.print(temperatureC, 1); Serial.println(" *C both
  measured together");

  pressureKPA = mpl115a2.getPressure();
  Serial.print("Pressure (kPa): "); Serial.print(pressureKPA, 4); Serial.println(" kPa");

  temperatureC = mpl115a2.getTemperature();
  Serial.print("Temp (*C): "); Serial.print(temperatureC, 1); Serial.println(" *C");
  Serial.print("DATA,TIME,"); Serial.print(temperatureC); Serial.print(",");
Serial.println(pressureKPA);
  row++;
  if (row > 360)
    {
     row=0;
     Serial.println("ROW,SET,2");
    }
  delay(100);
}
```

Παράρτημα 2

Επειδή αντιμετωπίσαμε προβλήματα συμβατότητας και επικοινωνίας μεταξύ των shields και των αισθητηρίων, χρειάστηκε να παραμετροποιήσουμε κάποιες εισόδους και εξόδους του Arduino, ώστε να επιτευχθεί η επικοινωνία. Αυτό συνέβη επειδή τα περισσότερα αισθητήρια και shields είναι σχεδιασμένα για το Arduino Uno, το οποίο διαθέτει λιγότερες εισόδους και εξόδους από ότι το Mega. Επομένως χρειάστηκε η κατασκευή μιας βιβλιοθήκης, η οποία θα είχε ως στόχο την παραμετροποίηση των εισόδων και εξόδων του Arduino Mega, ώστε να επιτύχει τη συμβατότητα με τα αισθητήρια και τα shields. Η βιβλιοθήκη, την οποία ονομάσαμε pin, παρουσιάζεται παρακάτω:

```
#ifndef Pins_Arduino_h
#define Pins_Arduino_h
#include <avr/pgmspace.h>

#define NOT_A_PIN 0
#define NOT_A_PORT 0

#define NOT_ON_TIMER 0
#define TIMER0A 1
#define TIMER0B 2
#define TIMER1A 3
#define TIMER1B 4
#define TIMER2  5
#define TIMER2A 6
#define TIMER2B 7

#define TIMER3A 8
#define TIMER3B 9
#define TIMER3C 10
#define TIMER4A 11
#define TIMER4B 12
#define TIMER4C 13
#define TIMER5A 14
```

```
#define TIMER5B 15
#define TIMER5C 16

#if defined(__AVR_ATmega1280) || defined(__AVR_ATmega2560)
const static uint8_t SS   = 53;
const static uint8_t MOSI = 51;
const static uint8_t MISO = 50;
const static uint8_t SCK  = 52;
#else
const static uint8_t SS   = 10;
const static uint8_t MOSI = 11;
const static uint8_t MISO = 12;
const static uint8_t SCK  = 13;
#endif


in uint8_t's.
extern const uint16_t PROGMEM port_to_mode_PGM[];
extern const uint16_t PROGMEM port_to_input_PGM[];
extern const uint16_t PROGMEM port_to_output_PGM[];

extern const uint8_t PROGMEM digital_pin_to_port_PGM[];
 digital_pin_to_bit_PGM[];
extern const uint8_t PROGMEM digital_pin_to_bit_mask_PGM[];
extern const uint8_t PROGMEM digital_pin_to_timer_PGM[];

#define digitalPinToPort(P) ( pgm_read_byte( digital_pin_to_port_PGM + (P) ) )
#define digitalPinToBitMask(P) ( pgm_read_byte( digital_pin_to_bit_mask_PGM + (P) ) )
#define digitalPinToTimer(P) ( pgm_read_byte( digital_pin_to_timer_PGM + (P) ) )
#define analogInPinToBit(P) (P)
#define portOutputRegister(P) ( (volatile uint8_t *)( pgm_read_word( port_to_output_PGM + (P))) )
```

```
#define portInputRegister(P) ( (volatile uint8_t *)( pgm_read_word( port_to_input_PGM + (P))) )
#define portModeRegister(P) ( (volatile uint8_t *)( pgm_read_word( port_to_mode_PGM + (P))) )
#endif
```