ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ
ΣΧΟΛΗ ΕΠΙΣΤΗΜΩΝ ΑΓΩΓΗΣ
ΠΑΙΔΑΓΩΓΙΚΟ ΤΜΗΜΑ ΔΗΜΟΤΙΚΗΣ ΕΚΠΑΙΔΕΥΣΗΣ
ΜΕΤΑΠΤΥΧΙΑΚΟ ΠΡΟΓΡΑΜΜΑ ΣΠΟΥΔΩΝ
ΚΑΤΕΥΘΥΝΣΗ: ΘΕΤΙΚΕΣ ΕΠΙΣΤΗΜΕΣ ΣΤΗΝ ΕΚΠΑΙΔΕΥΣΗ

Δ ι π λ ω μ α τ ι κ ή Ε ρ γ α σ ί α

Δημιουργία περιβάλλοντος διεπαφής σε ένα σύστημα συγχρονικής λήψης και απεικόνισης δεδομένων γενικού σκοπού

Νικολού Αγγελική

ΙΩΑΝΝΙΝΑ 2017

ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ
ΣΧΟΛΗ ΕΠΙΣΤΗΜΩΝ ΑΓΩΓΗΣ
ΠΑΙΔΑΓΩΓΙΚΟ ΤΜΗΜΑ ΔΗΜΟΤΙΚΗΣ ΕΚΠΑΙΔΕΥΣΗΣ
ΜΕΤΑΠΤΥΧΙΑΚΟ ΠΡΟΓΡΑΜΜΑ ΣΠΟΥΔΩΝ
ΚΑΤΕΥΘΥΝΣΗ: ΘΕΤΙΚΕΣ ΕΠΙΣΤΗΜΕΣ ΣΤΗΝ ΕΚΠΑΙΔΕΥΣΗ

Δ ι π λ ω μ α τ ι κ ή Ε ρ γ α σ ί α

Δημιουργία περιβάλλοντος διεπαφής σε ένα σύστημα συγχρονικής λήψης και απεικόνισης δεδομένων γενικού σκοπού

Νικολού Αγγελική, Α.Μ. 324

Τριμελής εξεταστική επιτροπή:

Επιβλέπων: Αναστάσιος Μικρόπουλος, Καθηγητής Παιδαγωγικού Τμήματος Δημοτικής Εκπαίδευσης Πανεπιστημίου Ιωαννίνων.

Μέλος: Κωνσταντίνος Κώτσης, Καθηγητής Παιδαγωγικού Τμήματος Δημοτικής Εκπαίδευσης Πανεπιστημίου Ιωαννίνων.

Μέλος: Κωνσταντίνος Γαβριλάκης, Επίκουρος καθηγητής Παιδαγωγικού Τμήματος Δημοτικής Εκπαίδευσης Πανεπιστημίου Ιωαννίνων.

ΙΩΑΝΝΙΝΑ 2017

Περιεχόμενα

Ευρετήριο Σχημάτων

Ευρετήριο Πινάκων

Ευχαριστίες

Νοιώθω την υποχρέωση να ευχαριστήσω καταρχήν τον επιβλέποντα καθηγητή κ. Αναστάσιο Μικρόπουλο για την αμέριστη ηθική συμπαράσταση και την επιστημονική καθοδήγηση που μου παρείχε αυτό το χρονικό διάστημα. Θα ήθελα, επίσης, να εκφράσω τις ευχαριστίες μου και στα άλλα δύο μέλη της τριμελούς εξεταστικής επιτροπής, Καθ. Κωνσταντίνο Κώτση και Επ. Καθ. Κώστα Γαβριλάκη, που με βοήθησαν με τις χρήσιμες παρατηρήσεις τους. Ακόμη, επιθυμώ να ευχαριστήσω τον Σχολικό Σύμβουλο κ. Κωνσταντίνο Γεωργόπουλο, καθώς και τους συναδέλφους εκπαιδευτικούς κ. Δημήτριο Σιάνο και κ. Άλκη Γεωργόπουλο για τις χρήσιμες συμβουλές και τις προτάσεις τους. Τελειώνοντας θα ήθελα να εκφράσω την ευγνωμοσύνη μου στην οικογένειά μου για την υπομονή και τη συμπαράστασή τους.

Περίληψη

Στο σύγχρονο σχολικό εργαστηριακό περιβάλλον, η ανάπτυξη της ψηφιακής τεχνολογίας και οι Τεχνολογίες Πληροφορίας και Επικοινωνιών (ΤΠΕ) συμβάλλουν στην αναβάθμιση της διδακτικής διαδικασίας και βελτίωση της ποιότητας της μάθησης, καθώς παρέχουν νέα εργαλεία και περιβάλλοντα για τη συλλογή, επεξεργασία, οπτικοποίηση των δεδομένων και επικοινωνία των αποτελεσμάτων. Τα εργαστήρια βασισμένα σε υπολογιστή (Microcomputer-based laboratories, MBL), που στην Ελλάδα ονομάζονται Συστήματα Συγχρονικής Λήψης και Απεικόνισης, συνδέουν αισθητήρες, συσκευές πρόσκτησης δεδομένων, υπολογιστή και λογισμικό διαχείρισης, για την άμεση λήψη δεδομένων του πραγματικού κόσμου, επεξεργασία, καθώς και απεικόνισή τους σε πραγματικό χρόνο. Οι εμπορικά διαθέσιμες επιλογές για τα σχολικά εργαστήρια Φυσικών Επιστημών (ΦΕ) είναι δαπανηρές, δεν συνεργάζονται με εξοπλισμό διαφορετικού κατασκευαστή, ενώ είναι κλειστού χαρακτήρα, με αποτέλεσμα να μην είναι εφικτή οποιαδήποτε τροποποίηση, βελτίωση ή προσαρμογή από το χρήστη. Τα τελευταία χρόνια η πρόοδος στην τεχνολογία των ηλεκτρονικών και των ολοκληρωμένων κυκλωμάτων έχει οδηγήσει στην ανάπτυξη χαμηλού κόστους, ανοιχτής πρόσβασης και αρχιτεκτονικής τεχνολογιών, που βασίζονται σε μικροελεγκτές. Η πλατφόρμα Arduino αποτελεί ένα από τα πιο δημοφιλή project ανοιχτού υλικού/λογισμικού, ενώ παράλληλα υποστηρίζεται από μία εκτεταμένη κοινότητα χρηστών. Στην παρούσα εργασία δημιουργήθηκε περιβάλλον γραφικής διεπαφής χρήστη γενικού σκοπού, σε ένα ανοιχτό, χαμηλού κόστους σύστημα συγχρονικής λήψης και απεικόνισης δεδομένων. Η πλακέτα Arduino Uno προγραμματίστηκε να λειτουργεί ως συσκευή πρόσκτησης δεδομένων, η οποία λαμβάνει και επεξεργάζεται μετρήσεις από αναλογικούς ή ψηφιακούς αισθητήρες και στη συνέχεια αποστέλλει τα αποτελέσματα στον υπολογιστή μέσω σύνδεσης USB. Ταυτόχρονα, στην πλευρά του υπολογιστή, η εφαρμογή «libreMBL-GUI», η οποία αναπτύχθηκε με τη γλώσσα προγραμματισμού Python, επιτρέπει τη ρύθμιση της σειριακής επικοινωνίας για την αμφίδρομη επικοινωνία χρήστη-συστήματος, τον έλεγχο έναρξης και λήξης της γραφικής αναπαράστασης των μετρήσεων σε πραγματικό χρόνο, τη διαδραστική πλοήγηση του μαθητή στις γραφικές παραστάσεις και την αποθήκευσή δεδομένων/γραφικών παραστάσεων για μεταγενέστερη επεξεργασία. Επιπλέον είναι ανοιχτού κώδικα, ανεξάρτητη πλατφόρμας και διαφανής ως προς το υλικό, καθώς μπορεί να υποστηρίξει διαφορετικά είδη Arduino, ενώ δύναται να απεικονίσει τις μετρήσεις οποιουδήποτε συμβατού αισθητήρα που συνδέεται με την πλακέτα Arduino, με την προϋπόθεση να έχει δημιουργηθεί και μεταφορτωθεί στον μικροελεγκτή το κατάλληλο πρόγραμμα (sketch). Το σύστημα

δοκιμάστηκε με αισθητήρες θερμοκρασίας, πίεσης και απόστασης, που καλύπτουν αρκετά μεγάλο εύρος πειραμάτων στη Δευτεροβάθμια αλλά και στην Πρωτοβάθμια Εκπαίδευση. Τα προγράμματα (sketches) που δημιουργήθηκαν, ακολουθούν ένα κοινό μοτίβο κώδικα, ώστε να είναι εύκολη η επέκταση και προσαρμογή τους σε διαφορετικούς αισθητήρες.

Λέξεις κλειδιά: Arduino, Σύστημα συγχρονικής λήψης και απεικόνισης, Γραφική διεπαφή χρήστη.

Abstract

In the modern school lab environment, the development of digital technology and Information and Communication Technologies (ICT) contribute to upgrading the teaching process and improving the quality of learning as they provide new tools and environments for collecting, processing, visualizing data and communicating the results. Microcomputer-based laboratories (MBL), connect sensors, data acquisition devices, computer and software to instantly receive real-world data, process and graphical representation in real time. Commercially available choices for School Science Labs are costly, do not work with equipment from a different manufacturer, but are closed in nature, so that any modification, improvement, or customization by the user is not possible. In recent years, advances in electronics and integrated circuit technology have led to the development of low cost, open access technologies based on microcontrollers. The Arduino platform is one of the most popular open hardware/software projects and is supported by an extensive community of users. In this work, a general-purpose graphical user interface was created in an open, low-cost MBL system. The Arduino Uno board is programmed to act as a data acquisition device that receives and processes measurements from analogue or digital sensors and then sends the results to the computer via a USB connection. At the same time, on the computer, the libreMBL-GUI application, developed with the Python programming language, enables serial communication to be set up for interactive user-system communication, the start and end control of graphical representation of the measurements in real time, interactive student navigation on graphs and data storage for later processing. Additionally, it is open source, cross-platform, and transparent to the material, as it can support different Arduino types, while it can display the measurements of any compatible sensor connected to the Arduino board, provided that the appropriate sketch has been created and downloaded to the microcontroller. The system was tested with temperature, pressure and distance sensors,

covering a wide range of experiments in both Secondary and Primary Education. The created sketches follow a common pattern to make it easy to expand and adapt to different sensors.

Keywords: Arduino, Microcomputer-based laboratory, Graphical user interface.

Εισαγωγή

Η παρούσα εργασία δομείται σε τρία μέρη: α) τη θεωρητική τεκμηρίωση της μεθοδολογίας που αναπτύχθηκε, β) την περιγραφή της μεθοδολογίας και γ) την παρουσίαση των αποτελεσμάτων.

Στη θεωρητική τεκμηρίωση, αρχικά γίνεται μια σύντομη αναφορά στην εργαστηριακή εκπαίδευση στις Φυσικές Επιστήμες (ΦΕ). Στη συνέχεια συνοψίζονται τα παιδαγωγικά πλεονεκτήματα που αποδίδονται στα συστήματα Συγχρονικής Λήψης και Απεικόνισης, τα τεχνικά χαρακτηριστικά τους, καθώς και τα μειονεκτήματα των εμπορικών συστημάτων που χρησιμοποιούνται στα σχολικά εργαστήρια ΦΕ. Τέλος παρουσιάζεται μία περιεκτική βιβλιογραφική επισκόπηση των τελευταίων ετών, που αφορά την ανάπτυξη χαμηλού κόστους αξιόπιστων συστημάτων πρόσκτησης δεδομένων γενικού ή ειδικού τύπου, τα οποία βασίζονται σε μικροελεγκτές. Επιπλέον, συζητείται το αντικείμενο μελέτης της παρούσας εργασίας.

Στο δεύτερο μέρος παρουσιάζονται οι στόχοι και η πειραματική διάταξη που προτείνεται, ενώ γίνεται αναφορά στην πλατφόρμα Arduino, στους αισθητήρες που χρησιμοποιήθηκαν και στη γλώσσα προγραμματισμού Python, η οποία αξιοποιήθηκε για την ανάπτυξη της γραφικής διεπαφής. Για όλες τις παραπάνω επιλογές αναλύονται οι λόγοι που οδήγησαν σε αυτές, στα πλαίσια και της θεωρητικής τεκμηρίωσης που παρουσιάζεται στο πρώτο μέρος της παρούσας εργασίας.

Στο τρίτο μέρος της παρούσας εργασίας περιγράφεται αρχικά η διαδικασία προγραμματισμού της πλακέτας Arduino, ώστε να συνδεθεί με αναλογικούς ή ψηφιακούς αισθητήρες θερμοκρασίας, πίεσης και απόστασης, καθώς και ο τρόπος σύνδεσης στην πειραματική διάταξη. Στη συνέχεια περιγράφεται το περιβάλλον γραφικής διεπαφής και γίνεται σύντομη αναφορά στις βιβλιοθήκες και αρθρώματα της γλώσσας προγραμματισμού Python που χρησιμοποιήθηκαν για την ανάπτυξη της εν λόγω εφαρμογής. Επιπλέον, περιγράφεται αναλυτικά η διαδικασία προγραμματισμού της εφαρμογής. Τέλος, γίνεται αναφορά στους περιορισμούς της διάταξης που προτείνεται (λογισμικό), καθώς και στις προτάσεις για περεταίρω έρευνα και μελλοντικές επεκτάσεις.

1 ΘΕΩΡΗΤΙΚΗ ΤΕΚΜΗΡΙΩΣΗ

1.1 Η εργαστηριακή εκπαίδευση στις Φυσικές Επιστήμες

Η εργαστηριακή μάθηση αποτελεί αναπόσπαστο μέρος της εκπαίδευσης των Φυσικών Επιστημών (ΦΕ). Οι Lunetta, Hofstein, & Clough ορίζουν τις δραστηριότητες στα σχολικά εργαστήρια ΦΕ, ως «*μαθησιακές εμπειρίες στις οποίες οι μαθητές αλληλεπιδρούν με υλικά ή δευτερογενείς πηγές δεδομένων για να παρατηρήσουν και να κατανοήσουν το φυσικό κόσμο*» (2005:2). Παρόμοια το Εθνικό Συμβούλιο Έρευνας (National Research Council, NRC) συμπληρώνει τον ορισμό και αναφέρει ότι «*παρέχουν ευκαιρίες στους μαθητές να αλληλεπιδρούν άμεσα με τον υλικό κόσμο (ή με δεδομένα που προέρχονται από τον υλικό κόσμο), χρησιμοποιώντας εργαλεία, τεχνικές συλλογής δεδομένων, μοντέλα και θεωρίες της επιστήμης*» (2005:31). Οι δραστηριότητες περιλαμβάνουν φυσικό χειρισμό πραγματικών υλικών, αλληλεπίδραση με προσομοιώσεις ή με δεδομένα του πραγματικού κόσμου, που λαμβάνονται και απεικονίζονται σε μια ποικιλία μορφών, πρόσβαση σε μεγάλες βάσεις δεδομένων και απομακρυσμένη πρόσβαση σε επιστημονικά όργανα (NRC, 2005).

Το εργαστηριακό περιβάλλον, στο οποίο οι μαθητές συνεργάζονται - κυρίως σε μικρές ομάδες - και συμμετέχουν σε εργαστηριακές δραστηριότητες (Sesen & Tarhan, 2011), έχει αναγνωριστεί ως ένα μοναδικό εκπαιδευτικό περιβάλλον, το οποίο παρέχει κίνητρα στους μαθητές, ενισχύει την ενεργητική μάθηση, ενώ τους μεταφέρει την εμπειρία και την πρακτική της επιστημονικής έρευνας (Linn, 1997). Όπως αναφέρουν οι Hofstein & Mamlok-Naaman (2007), η έρευνα γενικά, αλλά και ειδικά στο πλαίσιο των εργαστηριακών δραστηριοτήτων, κατέχει κεντρική θέση για την επίτευξη του επιστημονικού εγγραμματισμού, που αποτελεί κεντρικό στόχο της εκπαίδευσης των ΦΕ (Chinn & Malhotra, 2002).

Στις παραδοσιακές εργαστηριακές δραστηριότητες τύπου «συνταγών μαγειρικής» ή κλειστού τύπου (closed-ended), οι μαθητές ακολουθούν συγκεκριμένες οδηγίες, ώστε να καταλήξουν σε προκαθορισμένα συμπεράσματα, ενώ έχουν ελάχιστες ευκαιρίες να προωθήσουν γνωστικές δεξιότητες υψηλότερου επιπέδου. Οι εκπαιδευτικοί επιλέγουν και ορίζουν το πρόβλημα, τον εξοπλισμό που θα χρησιμοποιηθεί, τις διαδικασίες και τα αναμενόμενα αποτελέσματα (Domin, 1999; Szott, 2014). Το ενδιαφέρον των μαθητών επικεντρώνεται κυρίως στη διεξαγωγή των πειραματικών εργασιών, στο χειρισμό του εξοπλισμού, στη συλλογή δεδομένων, χωρίς πολλές φορές να κατανοούν το σκοπό, τις διαδικασίες και τη σχέση του πειραματικού σχεδιασμού με την επιστημονική θεωρία, ή χωρίς

να έχουν ευκαιρίες για συζήτηση και προβληματισμό σχετικά με τις παρατηρήσεις τους (Hofstein & Lunetta, 2004; Κώτσης, 2005).

Όταν όμως οι εργαστηριακές εμπειρίες ολοκληρώνονται με άλλες μεταγνωστικές εμπειρίες μάθησης, όπως δραστηριότητες «πρόβλεψη-παρατήρηση-επεξήγηση», ενώ ενσωματώνουν όχι μόνο χειρισμό υλικών και διαδικασιών αλλά επικοινωνία ιδεών και συνεργασία, είναι εφικτό να προωθήσουν τη μάθηση των ΦΕ (Hofstein & Lunetta, 2004). Η μάθηση με διερεύνηση (Inquiry-based learning) είναι μια προσέγγιση που τοποθετεί τις ερωτήσεις, τις ιδέες και τις παρατηρήσεις των μαθητών στο κέντρο της μαθησιακής εμπειρίας (Literacy and Numeracy Secretariat, 2013). Οι μαθητές που συμμετέχουν σε ερευνητικές δραστηριότητες (inquiry-based activities) εμπλέκονται ενεργά στη διαδικασία της διατύπωσης ερωτήσεων, προβλέψεων και υποθέσεων, του σχεδιασμού πειραμάτων, της χρήσης εργαλείων για τη συλλογή/ανάλυση δεδομένων και της εξαγωγής συμπερασμάτων σχετικά με επιστημονικά προβλήματα ή φαινόμενα (Hofstein & Walberg, 1995; NRC, 2005). Παράλληλα ασκούνται στον επιστημονικό τρόπο σκέψης και κατανοούν τη μεθοδολογία που χρησιμοποιούν οι επιστήμονες για την παραγωγή, επαλήθευση και προώθηση της γνώσης.

Οι ερευνητικές δραστηριότητες στα εργαστήρια ΦΕ είναι ένας αποτελεσματικός τρόπος μάθησης, που ενθαρρύνει τη διερεύνηση του βαθμού κατανόησης των επιστημονικών εννοιών, νόμων και φαινομένων από τους μαθητές, την ανάπτυξη δεξιοτήτων επιστημονικής διαδικασίας, τη βελτίωση της στάσης τους απέναντι στη σχολική επιστήμη, την παροχή κινήτρων για να κατανοήσουν τη φύση της επιστήμης, την ανάπτυξη δεξιοτήτων επικοινωνίας και συνεργασίας (Sesen & Tarhan, 2011, Κώτσης, 2005). Επίσης ενθαρρύνουν την αυτενέργεια, την πρωτοβουλία, την πρωτοτυπία και τη δημιουργικότητα των μαθητών (Κώτσης, 2005). Οι εκπαιδευτικοί διαδραματίζουν ενεργό ρόλο σε όλη τη διαδικασία, θεσπίζοντας μία κουλτούρα όπου οι ιδέες με σεβασμό αμφισβητούνται, δοκιμάζονται, επαναπροσδιορίζονται, βελτιώνονται (Literacy and Numeracy Secretariat, 2013). Οι μαθητές με την υποστήριξη των εκπαιδευτικών αναλαμβάνουν την ευθύνη του πειραματικού σχεδιασμού και οδηγούνται από τον προβληματισμό στην κατανόηση και στη διατύπωση νέων ερωτήσεων.

Στην καθοδηγούμενη διερεύνηση (guided inquiry) ο εκπαιδευτικός παρουσιάζει τα ερευνητικά ερωτήματα και τις διαδικασίες, ενώ οι μαθητές σε ομάδες αποφασίζουν την κατεύθυνση και τις μεθόδους της έρευνας που θα ακολουθήσουν, συμμετέχοντας σε πρακτικές (hands-on), ανοιχτές (open-ended), μαθητοκεντρικές πειραματικές δραστηριότητες (Irinoye et al., 2014; Zion & Mendelovici, 2012). Οι μαθητές, εφόσον οι εκπαιδευτικοί

καθορίσουν τους στόχους, διατυπώσουν τις ερωτήσεις και παρέχουν μία λίστα από διαθέσιμα υλικά, οδηγούν τελικά την ερευνητική διαδικασία συμμετέχοντας στη λήψη αποφάσεων από το στάδιο της συλλογής δεδομένων. Τα αποτελέσματα της έρευνας δεν είναι εκ των προτέρων γνωστά ούτε στους μαθητές αλλά ούτε και στους εκπαιδευτικούς, οι οποίοι ως διαμεσολαβητές συντονίζουν την ανάλυση των δεδομένων, την ερμηνεία των αποτελεσμάτων και την εξαγωγή συμπερασμάτων (Igelsrud & Leonard, 1988).

Στην ανοιχτή διερεύνηση (open inquiry), το πιο πολύπλοκο επίπεδο της μάθησης με διερεύνηση, οι εκπαιδευτικοί καθορίζουν το γνωστικό ερευνητικό πλαίσιο, αλλά οι μαθητές διατυπώνουν τα ερευνητικά ερωτήματα και επιλέγουν την προσέγγιση που θα ακολουθήσουν, συμμετέχοντας στη λήψη αποφάσεων από το αρχικό στάδιο της εύρεσης του φαινομένου που θα εξερευνηθεί (Zion & Mendelovici, 2012). Η ανοιχτή διερεύνηση βασίζεται στην ανεξάρτητη δράση των μαθητών και εξαρτάται από την ικανότητα των εκπαιδευτικών να υποστηρίζουν/διευκολύνουν τους μαθητές σε κάθε στάδιο της έρευνας και να τους παρέχουν τα κατάλληλα εργαλεία και υλικά.

Η ολοένα και πιο διαδεδομένη αξιοποίηση της ψηφιακής τεχνολογίας και των ΤΠΕ στη διδασκαλία των ΦΕ, προσφέρει νέα εργαλεία και περιβάλλοντα για τη συλλογή, επεξεργασία και οπτικοποίηση των δεδομένων, την επικοινωνία ή διαμοιρασμό των αποτελεσμάτων και την υποστήριξη της διερευνητικής μάθησης. Όπως αναφέρουν οι Mikropoulos & Bellou (2006), Μικρόπουλος (2006) τα τεχνολογικά χαρακτηριστικά των ΤΠΕ αξιοποιούν τις ιδιότητες του χώρου και του χρόνου καθώς καταγράφουν, επεξεργάζονται, διαχειρίζονται και μεταφέρουν μεγάλο όγκο δεδομένων σε εξαιρετικά μικρό χρόνο. Επίσης παρουσιάζουν τα αποτελέσματα της επεξεργασίας μέσω δυναμικών, αλληλεπιδραστικών, πολλαπλών αναπαραστάσεων, επικοινωνούν την πληροφορία σύγχρονα ή ασύγχρονα και παρέχουν κίνητρα στους μαθητές.

1.2 Συστήματα Συγχρονικής Λήψης και Απεικόνισης

Τα εργαστήρια βασισμένα σε υπολογιστή (Microcomputer-based laboratories, MBL), που στην Ελλάδα ονομάζονται Συστήματα Συγχρονικής Λήψης και Απεικόνισης (ΣΣΛΑ), συνδέουν αισθητήρες, συσκευές καταγραφής ή πρόσκτησης δεδομένων, υπολογιστές και λογισμικό διαχείρισης, για την άμεση λήψη, απεικόνιση και ανάλυση δεδομένων, καταγράφοντας μετρήσεις φυσικών μεγεθών, όπως για παράδειγμα θερμοκρασία, πίεση, θέση, ταχύτητα, επιτάχυνση, δύναμη, ήχο, ένταση φωτός, ρεύμα, διαφορά δυναμικού. Δίνουν τη δυνατότητα στους μαθητές/φοιτητές να χειριστούν εξοπλισμό, όπως στα παραδοσιακά

εργαστήρια, να εξερευνήσουν, να ποσοτικοποιήσουν το φυσικό κόσμο και να κάνουν μετρήσεις σε μη συνηθισμένες χρονικές κλίμακες. Σύμφωνα με τον Hale (2000) προσφέρουν ένα ισχυρό μαθησιακό περιβάλλον, επιτρέποντας στους μαθητές να χρησιμοποιήσουν προηγμένες τεχνολογίες για να λύσουν προβλήματα και να κάνουν πραγματική επιστήμη. Τα συστήματα MBL παρέχουν ευκαιρίες για προβλέψεις, διατύπωση ερωτήσεων, σχεδιασμό πειραμάτων, συλλογή και ανάλυση δεδομένων, συζήτηση και επικοινωνία ιδεών και ευρημάτων, εξαγωγή συμπερασμάτων, και διατύπωση νέων ερωτήσεων (Krajcik & Layman, 1992).

Όπως αναφέρουν οι Thornton & Socoloff (1990), τα συστήματα MBL είναι ένα αποτελεσματικό εργαλείο για την ανάπτυξη μιας σταθερής εννοιολογικής βάσης για την κατανόηση του φυσικού κόσμου. Οι μαθητές προκειμένου να οικοδομήσουν τη γνώση για τα φαινόμενα και τις σχετικές επιστημονικές έννοιες, είναι απαραίτητο να έχουν τη δυνατότητα αλληλεπίδρασης με όργανα και υλικά, ενώ συνεργάζονται σε ομάδες και επιλύουν προβλήματα που τους ενδιαφέρουν (Tobin, 1990). Στα εργαστήρια MBL, οι μαθητές/φοιτητές εμπλέκονται στη διαδικασία εκτέλεσης πραγματικών πειραμάτων και όχι προσομοιώσεων, και ο υπολογιστής χρησιμοποιείται ως εργαλείο μέτρησης σε συνδυασμό με ένα εύχρηστο περιβάλλον διεπαφής, καθώς και τους κατάλληλους αισθητήρες. Ο Russell (2002) υπογραμμίζει ότι ο υπολογιστής δεν αντικαθιστά τις πειραματικές δραστηριότητες (όπως με τις προσομοιώσεις), αλλά τροποποιεί και επεκτείνει τον τρόπο διεξαγωγής των πειραμάτων. Ως ένα ισχυρό γνωστικό εργαλείο μέσω των τεχνολογικών χαρακτηριστικών του, ο υπολογιστής αποδεσμεύει τους μαθητές/φοιτητές από μηχανιστικές διεργασίες, εκτελεί υπολογισμούς, αναπαριστά μεγέθη, φαινόμενα και καταστάσεις (Μικρόπουλος, 2006).

Τα εργαλεία MBL επιτρέπουν στους μαθητές/φοιτητές να αναλάβουν ενεργό ρόλο στη μάθηση, να αξιοποιήσουν τις εμπειρίες της καθημερινής τους αλληλεπίδρασης με το φυσικό κόσμο και μέσω της άμεσης καταγραφής και γραφικής αναπαράστασης των πειραματικών δεδομένων σε πραγματικό χρόνο, να απομακρυνθούν από παρανοήσεις και να έχουν μια βαθύτερη επιστημονική κατανόηση αυτών των εμπειριών. Επιπλέον, η άμεση και γρήγορη καταγραφή και ανατροφοδότηση των δεδομένων σε κατανοητή μορφή υποστηρίζει τη συνεργατική μάθηση. Η αναπαράσταση των δεδομένων στον υπολογιστή είναι χωρίς νόημα χωρίς τη διαμεσολάβηση των μαθητών/φοιτητών, οι οποίοι έχοντας απαλλαγεί από κουραστικές και επαναλαμβανόμενες διεργασίες, έχουν χρόνο να εκφράσουν τις προβλέψεις τους, να δοκιμάσουν τις αρχικές τους αντιλήψεις, να συζητήσουν με τους συνομηλίκους τους

για την εγκυρότητα και τη σημασία των δεδομένων, να προβληματιστούν για απροσδόκητα αποτελέσματα και να εξάγουν συμπεράσματα (Kelly & Crawford,1996; Thornton,1987; Thornton & Sokoloff, 1990). Συμμετέχοντας σε δραστηριότητες «πρόβλεψη-παρατήρηση-επεξήγηση», με άμεση ανατροφοδότηση από τον υπολογιστή, έχουν τη δυνατότητα να προβληματιστούν και να συζητήσουν τους λόγους για τυχόν ανεπιτυχείς προβλέψεις. Ο χρόνος αξιοποιείται δημιουργικά - με την κατάλληλη καθοδήγηση από το διδάσκοντα - για αναστοχασμό, ανάλυση και ερμηνεία των αποτελεσμάτων (Gipps, 2002). Σύμφωνα με τον Newton (1998), στις δραστηριότητες καταγραφής δεδομένων πρέπει να ενθαρρύνεται η αλληλεπίδραση μεταξύ των μαθητών και να γίνεται ανάθεση ρόλων. Οι μαθητές πρέπει να έχουν ήδη κατακτήσει δεξιότητες προσδιορισμού μεταβλητών, να είναι σε θέση να αποφασίζουν τις μετρήσεις που θα πραγματοποιήσουν και να ενθαρρύνονται να κάνουν προβλέψεις. Εφόσον εξοικειωθούν με τον εργαστηριακό εξοπλισμό, μέσα από κατάλληλα σχεδιασμένες δραστηριότητες με σαφείς και κατανοητούς στόχους, θα βελτιώσουν τις ικανότητές τους στο χειρισμό, την ερμηνεία και την αξιολόγηση των δεδομένων.

Ο σχηματισμός των γραφικών παραστάσεων φυσικών μεγεθών ταυτόχρονα με την εξέλιξη του φυσικού φαινομένου βελτιώνει σημαντικά τις δεξιότητες των μαθητών στη δημιουργία και ερμηνεία των γραφικών παραστάσεων και τους βοηθά να εξοικειωθούν με τη διαδικασία διερεύνησης των επιστημονικών σχέσεων. Μπορούν να δουν το πείραμα ως κάτι που αναπτύσσεται με την πάροδο του χρόνου και ότι ο ίδιος ο χρόνος είναι μια μεταβλητή (Gipps, 2002). Όπως αναφέρουν οι Lapp & Cyrus (2000), οι μαθητές/φοιτητές αντιμετωπίζουν δυσκολίες να συνδέσουν τις γραφικές παραστάσεις με τις φυσικές έννοιες και τον πραγματικό κόσμο, και να πραγματοποιήσουν αμφίδρομες μεταβάσεις μεταξύ του φυσικού φαινομένου και των αντίστοιχων γραφικών αναπαραστάσεων ή μεταξύ γραφικών παραστάσεων διαφορετικών φυσικών μεγεθών που αναπαριστούν το ίδιο φαινόμενο. Η δυναμική του συστήματος MBL να συσχετίζει τις γραφικές παραστάσεις των φυσικών μεγεθών με κινηματικά φαινόμενα που εξελίσσονται, αξιοποιήθηκε σε μελέτη της συνολικής κατανόησης του πλαισίου των αναπαραστάσεων, τόσο σε διαφορετικές δραστηριότητες όσο και σε φοιτητές διαφορετικού μαθηματικού επιπέδου (Γεωργόπουλος κ.α., 2010). Τα αποτελέσματα της μελέτης έδειξαν σημαντική βελτίωση, ενώ φοιτητές με χαμηλότερο μαθηματικό επίπεδο βελτιώθηκαν σε μεγαλύτερο βαθμό εκμεταλλευόμενοι κυρίως τις απλούστερες δραστηριότητες. Η Brasell (1987) υποστηρίζει ότι η άμεση γραφική αναπαράσταση των πειραματικών δεδομένων σε πραγματικό χρόνο είναι ίσως το πιο σημαντικό χαρακτηριστικό των δραστηριοτήτων με συστήματα MBL για τα κίνητρα και τη

γνωστική λειτουργία των μαθητών. Σύμφωνα με την έρευνά της, ακόμη και μια μικρή καθυστέρηση είκοσι δευτερολέπτων στην εμφάνιση του γραφήματος μετά τη λήξη του φυσικού φαινομένου, επηρεάζει την ικανότητα των μαθητών να συνδέσουν τη γραφική παράσταση με τη φυσική έννοια. Τα εργαλεία MBL, επιτρέπουν στους μαθητές να συλλέξουν τα δικά τους δεδομένα, να χειριστούν μεταβλητές και να παρακολουθήσουν σε πραγματικό χρόνο τη δυναμική γραφική αναπαράσταση της σχέσης μεταξύ αυτών των μεταβλητών. Η άμεση συσχέτιση των γραφημάτων με τα αντίστοιχα φυσικά φαινόμενα τους βοηθά να συνδέσουν τις αφηρημένες έννοιες με φαινόμενα του πραγματικού κόσμου (Kelly & Crawford,1996; McKenzie & Padilla,1986; Russell et al., 2004; Thornton,1987). Τα αποτελέσματα της έρευνας των McFarlane et al. (1995) υποδηλώνουν ότι μαθητές ηλικίας επτά και οκτώ χρόνων που ήρθαν σε επαφή με συστήματα καταγραφής δεδομένων, παρουσίασαν μια αυξημένη ικανότητα στην ανάγνωση, ερμηνεία και δημιουργία γραφημάτων σε σύγκριση με μαθητές της ίδιας ηλικίας που χρησιμοποίησαν πιο παραδοσιακές πειραματικές διατάξεις.

Η έρευνα υποδεικνύει ότι με τα συστήματα MBLs γίνεται εξοικονόμηση του χρόνου που δαπανάται για χρονοβόρες διεργασίες που συνήθως συνδέονται με τη συλλογή και την απεικόνιση των δεδομένων στο εργαστήριο φυσικής (Kelly & Crawford, 1996; Thornton & Sokoloff, 1990). Ο χρόνος των μαθητών/φοιτητών αξιοποιείται με δημιουργικές διεργασίες, όπως παρατήρηση των φυσικών φαινομένων, ανάλυση και ερμηνεία των δεδομένων και των γραφικών αναπαραστάσεων των φαινομένων αυτών. Οι μαθητές/φοιτητές έχουν περισσότερο χρόνο για να επαναλάβουν, αξιολογήσουν, βελτιώσουν ένα πείραμα, να χειριστούν διαφορετικές μεταβλητές, να επικεντρωθούν περισσότερο στην ανακάλυψη και την κατανόηση επιστημονικών εννοιών και να αναπτύξουν δεξιότητες κριτικής σκέψης και ερμηνείας των επιστημονικών δεδομένων. Η ευκολία και η ταχύτητα για επαναλαμβανόμενες μετρήσεις κάτω από διαφορετικές πειραματικές συνθήκες ενθαρρύνουν τη διατύπωση και τον έλεγχο υποθέσεων (Kennedy & Finn, 2000; Thornton & Sokoloff, 1990; Russell et al., 2004; Krajcik & Layman, 1992). Οι Barton & Rogers (1991) υποστηρίζουν ότι μετατοπίζοντας την έμφαση από τον τρόπο συλλογής των δεδομένων στην αξιοποίηση δεξιοτήτων ερμηνείας και ανάλυσης δεδομένων, ενισχύεται η επιστημονική σκέψη, η δημιουργικότητα και η ικανότητα επίλυσης προβλήματος.

Ο Russell (2002) συνοψίζει τα πλεονεκτήματα που αποδίδονται στα συστήματα MBL και ξεκινώντας από τον εξοπλισμό (αισθητήρες-διεπαφή-υπολογιστής-λογισμικό) αναφέρει την ταχύτητα λήψης και τη δυνατότητα αποθήκευσης μεγάλου όγκου δεδομένων, την άμεση

επεξεργασία και απεικόνισή τους, τη δυνατότητα για μετρήσεις σε μη συνηθισμένες χρονικές κλίμακες (π.χ. κλάσματα του δευτερολέπτου ή ημέρες), το μεγάλο εύρος πιθανών πειραμάτων, τη δυνατότητα χρήσης πολλαπλών αισθητήρων ταυτόχρονα για λήψη δεδομένων. Ο υψηλής τεχνολογίας εξοπλισμός δίνει κίνητρα και η ευκολία στη χρήση μειώνει το εργαστηριακό άγχος, ιδίως σε μαθητές με χαμηλότερη αυτοπεποίθηση. Μειώνεται επίσης ο χρόνος αναμονής καθώς και οι χρονοβόρες διεργασίες συλλογής και επεξεργασίας δεδομένων. Τέλος, όσον αφορά στα παιδαγωγικά οφέλη, δίνεται η δυνατότητα στους μαθητές να διερευνήσουν, να επαναλάβουν/βελτιώσουν πειράματα, να επικεντρωθούν στην ερμηνεία/αξιολόγηση των δεδομένων, να συνδέσουν τις γραφικές αναπαραστάσεις με τις φυσικές έννοιες, να αναπτύξουν δεξιότητες δημιουργίας και ερμηνείας γραφικών παραστάσεων και να συνεργαστούν σε ομάδες.

Στην εργασία των Chen et al. (2012) διερευνάται η εμπλοκή της τεχνολογίας στη δημιουργία καινοτόμων εργαστηρίων φυσικής, καθώς και η επίδραση της τεχνολογίας στη διερεύνηση, στην επικοινωνία/συνεργασία, στα μαθησιακά αποτελέσματα. Επίσης, παρουσιάζεται ένα πλαίσιο (framework) ώστε να χρησιμοποιηθεί από εκπαιδευτικούς, επιστήμονες και προγραμματιστές για τον σχεδιασμό και την αξιολόγηση τεχνολογικά ολοκληρωμένων εργαστηρίων (technology-integrated laboratories). Στον Πίνακα 1.1 συνοψίζονται τα μοναδικά χαρακτηριστικά τεσσάρων τύπων εργαστηρίων (MBL, προσομοιώσεις, απομακρυσμένα εργαστήρια και εργαστήρια επαυξημένης πραγματικότητας (AR)), ως προς την υποστήριξη στις μαθησιακές δραστηριότητες και την ανοιχτή διερεύνηση, τους τρόπους αξιοποίησης και τους εκπαιδευτικούς στόχους (Chen et al., 2012).

Όσον αφορά στους εκπαιδευτικούς στόχους, τα συστήματα MBL θα μπορούσαν να καλύψουν όλα τα πεδία, ιδιαίτερα την ανάπτυξη ψυχοκινητικών δεξιοτήτων από τους μαθητές/φοιτητές, οι οποίοι χειρίζονται πραγματικό εξοπλισμό (όπως στα παραδοσιακά εργαστήρια) και βιώνουν τον ενθουσιασμό της επιστημονικής διαδικασίας και της αυθεντικής έρευνας. Θεωρητικά, μεταξύ των τεσσάρων τύπων εργαστηρίων, τα συστήματα MBL παρουσιάζουν την υψηλότερη ευελιξία και υποστηρίζουν την ανοιχτή διερεύνηση, καθώς δεν υπαγορεύουν τα φαινόμενα που πρέπει να διερευνηθούν, την ερευνητική διαδικασία, το πλαίσιο ή την πολυπλοκότητα. Μια ποικιλία αισθητήρων μπορεί να συνδεθεί στην ίδια διεπαφή, να υποστηριχθεί από το ίδιο λογισμικό και να αξιοποιηθεί σε πολλά πειράματα (Thorton & Sokoloff, 1990).

Πίνακας 1.1 Χαρακτηριστικά διαφορετικών τεχνολογιών στα εργαστήρια ΦΕ.
++: μεγάλες δυνατότητες, +: δυνατότητες με περιορισμούς, -: λιγότερο φιλικές τεχνολογίες

	MBL	**Προσομοιώσεις**	**Απομακρυσμένα εργαστήρια**	**AR**
Αντικείμενα	Πραγματικά	Εικονικά	Πραγματικά	Μεικτά
Δεδομένα	Αυθεντικά	Τεχνητά	Αυθεντικά	Μεικτά
Κόστος ανά χρήση	Υψηλό	Χαμηλό	Χαμηλό	Μεικτό
Υποστήριξη στις μαθησιακές δραστηριότητες				
Εμπλοκή (εξερεύνηση, εξοικείωση με φαινόμενα ή αντικείμενα)	++	++	+	++
Εύρεση πληροφοριών	+	++	+	+
Διατύπωση και επεξεργασία ερωτήσεων, διατύπωση προβλέψεων ή υποθέσεων	++	++	+	++
Σχεδιασμός έρευνας (σχεδιασμός πειραματικής διαδικασίας, προσδιορισμός, έλεγχος, χειρισμός μεταβλητών)	++	+	+	+
Εκτέλεση διαδικασίας (χειρισμός εξοπλισμού, παρατήρηση, συλλογή δεδομένων)	++	++	++	++
Επεξεργασία δεδομένων (ανάλυση, μετασχηματισμός και οπτικοποίηση δεδομένων, μοντελοποίηση, εξαγωγή συμπερασμάτων)	++	++	++	++
Διαμοιρασμός ευρημάτων (επικοινωνία ευρημάτων, συνεργασία, παρουσίαση ιδεών, ανατροφοδότηση)	++	++	++	++
Υποστήριξη στην ανοιχτή διερεύνηση				
Διατύπωση ερωτήσεων	++	+	-	++
Επιλογή εξοπλισμού	++	+	-	+
Επιλογή διαδικασίας	++	+	-	+
Επιλογή μεταβλητών	++	++	-	+
Ανάλυση δεδομένων	++	++	++	++
Ερμηνεία αποτελεσμάτων	++	++	++	++
Αναφορά αποτελεσμάτων	++	++	+	++
Τρόποι αξιοποίησης				
Επίδειξη	++	++	++	++
Ατομική εργασία	+	++	++	++
Ομαδική εργασία	++	-	-	+
Εκπαιδευτικοί στόχοι				
Γνωστικοί	++	++	++	++
Συναισθηματικοί	++	++	+	++
Ψυχοκινητικοί	++	-	-	+

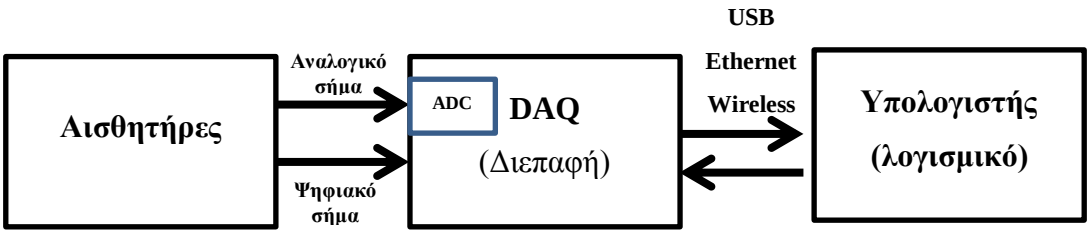
Οι μαθητές/φοιτητές αποκτούν πρωτοφανή ευελιξία να θέσουν ερευνητικά ερωτήματα, να επιλέξουν εξοπλισμό και διαδικασία, να εξερευνήσουν το φυσικό κόσμο, να κάνουν μετρήσεις, να αναλύσουν δεδομένα, να εξάγουν συμπεράσματα και να γενικεύσουν την πρόσφατα αποκτηθείσα επιστημονική γνώση τους για φαινόμενα που αποτελούν μέρος του καθημερινού τους κόσμου. Ένα ευρύ φάσμα μαθητών από το Δημοτικό μέχρι το Πανεπιστήμιο, αρχάριοι ή προχωρημένοι, χρησιμοποιώντας το ίδιο σύνολο εργαλείων (υλικό, λογισμικό), έχουν τη δυνατότητα να επικεντρωθούν στην έρευνα πολλών διαφορετικών φυσικών φαινομένων.

Όμως τα εργαλεία από μόνα τους δεν είναι αρκετά. Η αποτελεσματικότητα των συστημάτων MBL εξαρτάται σε μεγάλο βαθμό από την υποστήριξη, ενθάρρυνση και καθοδήγηση που προσφέρει ο διδάσκων (Chen et al., 2012; Krajcik & Layman, 1992), με κατάλληλα σχεδιασμένες δραστηριότητες (Newton, 1998; Nicolaou et. al., 2007), καθώς και από προγράμματα σπουδών που εστιάζουν στη διερεύνηση κατάλληλων φαινόμενων (Thorton & Sokoloff, 1990). Στην έρευνα των Nicolaou, Nicolaidou, Zacharia, και Constantinou (2007), μελετήθηκε κατά πόσο η χρήση συστημάτων MBL σε συνδυασμό με μια ακολουθία ερευνητικών δραστηριοτήτων (inquiry-based activity sequence) που αφορούσαν την τήξη και πήξη, συμβάλλει στην ανάπτυξη της εννοιολογικής κατανόησης μαθητών Δημοτικού, καθώς και στην ανάπτυξη της ικανότητάς τους στη δημιουργία και ερμηνεία γραφικών παραστάσεων. Ο συνδυασμός MBL-ερευνητικές δραστηριότητες, φάνηκε να είναι πιο αποτελεσματικός από την παραδοσιακή προσέγγιση ή ακόμη και από τον συνδυασμό των ερευνητικών δραστηριοτήτων με συμβατικές εργαστηριακές διαδικασίες και συσκευές.

Το βασικό μειονέκτημα των MBL, που τα καθιστά λιγότερο δημοφιλή σε σύγκριση με τα παραδοσιακά εργαστήρια ή τις προσομοιώσεις, είναι το υψηλό κόστος (Chen et al., 2012; 2014). Ο μη επαρκής (λόγω κόστους) εξοπλισμός αξιοποιείται κυρίως σε πειράματα επίδειξης, περιορίζοντας στο ελάχιστο τις ομαδικές ή ατομικές δραστηριότητες.

1.2.1 Τεχνικά χαρακτηριστικά συστημάτων Συγχρονικής Λήψης και Απεικόνισης

Ένα σύστημα MBL συνήθως αποτελείται από αισθητήρες (sensors), μια συσκευή καταγραφής (Data Logger) ή πρόσκτησης δεδομένων (Data Acquisition, DAQ) και έναν υπολογιστή με εγκατεστημένο το κατάλληλο λογισμικό διαχείρισης για αυτόματη λήψη, ανάλυση και γραφική απεικόνιση των δεδομένων (Σχήμα 1.1).



Σχήμα 1.1: Σύστημα MBL

Οι αισθητήρες (sensors) ανιχνεύουν τις μεταβολές φυσικών μεγεθών (δεδομένα του πραγματικού κόσμου) και τις μετατρέπουν -συνήθως- σε ηλεκτρικά σήματα (τάση ή ρεύμα), παράγοντας μια μετρήσιμη έξοδο. Η συσκευή καταγραφής ή πρόσκτησης δεδομένων, λειτουργεί ως ενδιάμεση διεπαφή μεταξύ του υπολογιστή και των σημάτων από τους αισθητήρες. Μία από τις βασικές της λειτουργίες είναι η μετατροπή των εισερχόμενων αναλογικών σημάτων σε ψηφιακή μορφή, μέσω αναλογοψηφιακού μετατροπέα (Analog-to-Digital Converter, ADC), ώστε στη συνέχεια να επεξεργαστούν από τον υπολογιστή.

Με τον όρο πρόσκτηση δεδομένων (Data Acquisition) περιγράφουμε τη διαδικασία με την οποία φυσικά φαινόμενα από τον πραγματικό κόσμο μετασχηματίζονται σε ηλεκτρικά σήματα, που μετρώνται και μετατρέπονται σε ψηφιακή μορφή, για την επεξεργασία, ανάλυση, αποθήκευση και απεικόνισή τους σε έναν υπολογιστή.

Οι παραδοσιακοί καταγραφείς δεδομένων (Data Loggers) είναι αυτόνομες (stand-alone) συσκευές, οι οποίες εκτελούν τις επιθυμητές μετρήσεις μέσω αισθητήρων, μετατρέπουν τα σήματα σε ψηφιακή μορφή και αποθηκεύουν τα δεδομένα σε εσωτερική μνήμη. Συνήθως περιλαμβάνουν ενσωματωμένη οθόνη και την ικανότητα να μεταφέρουν τα ήδη αποθηκευμένα δεδομένα σε έναν υπολογιστή για «offline» ανάλυση, επεξεργασία και μόνιμη αποθήκευση.

Οι καταγραφείς δεδομένων που βασίζονται σε υπολογιστή (Computer-based data loggers) είναι ένας συνδυασμός μιας συσκευής πρόσκτησης δεδομένων (DAQ) και ενός υπολογιστή. Σε σύγκριση με τα παραδοσιακά αυτόνομα καταγραφικά, προσφέρουν οπτικοποίηση σε πραγματικό χρόνο, «inline» ανάλυση, λειτουργικότητα που καθορίζεται

από τον χρήστη, μεγάλο αποθηκευτικό χώρο και δυνατότητα σύνδεσης με τον υπολογιστή μέσω σειριακής (USB) ή δικτυακής σύνδεσης (Ethernet, Wireless). Οι ειδικού τύπου εμπορικοί καταγραφείς δεδομένων είναι συνήθως συσκευές μονού καναλιού (single-channel) με σχετικά χαμηλό κόστος, ενώ οι συσκευές πολλαπλών καναλιών (multi-channel) έχουν τη δυνατότητα να πραγματοποιήσουν ταυτόχρονα μια ποικιλία μετρήσεων.

1.2.2 Συστήματα Συγχρονικής Λήψης και Απεικόνισης στα εκπαιδευτικά εργαστήρια Φυσικών Επιστημών

Είναι πολύ σημαντική η επιλογή του κατάλληλου ΣΣΛΑ για ένα εκπαιδευτικό εργαστήριο ΦΕ ώστε να είναι αξιόπιστο, λειτουργικό, να συνεργάζεται με αρκετούς συμβατούς αισθητήρες και να έχει προσιτή τιμή. Το αντίστοιχο λογισμικό ενδείκνυται να είναι εύχρηστο, ανεξάρτητο πλατφόρμας και να μπορεί να υποστηρίξει τους διδακτικούς και εκπαιδευτικούς στόχους του προγράμματος σπουδών. Επίσης υλικό και λογισμικό να είναι προσαρμοσμένα σε ένα σύνολο προτύπων και συμβάσεων, ώστε το σύστημα να μπορεί να αξιοποιηθεί σε ένα εύρος πειραματικών εφαρμογών.

Τα τελευταία είκοσι χρόνια διάφορες εταιρίες, όπως για παράδειγμα οι Pasco (www.pasco.com), Vernier (www.vernier.com), Fourier Systems (www.fouriersystems.com), Philip Harris Scientific (www.philipharris.co.uk), παρέχουν συσκευές καταγραφής ή πρόσκτησης δεδομένων, συμβατούς αισθητήρες και λογισμικά, ώστε να αξιοποιηθούν στις εκπαιδευτικές εργαστηριακές δραστηριότητες. Όμως οι εμπορικά διαθέσιμες επιλογές είναι δαπανηρές, δεν συνεργάζονται με εξοπλισμό διαφορετικού κατασκευαστή και είναι κλειστού χαρακτήρα με αποτέλεσμα να μην είναι εφικτή οποιαδήποτε τροποποίηση, βελτίωση ή προσαρμογή από το χρήστη.

Μερικά από τα μειονεκτήματα των εμπορικών συστημάτων όπως τα αναφέρουν οι Real, Raviola, Jauré, Vitali (2015) και Pearce (2014), είναι:

- Η ανάπτυξη, υποστήριξη και παραμονή των προϊόντων στην αγορά εξαρτάται από τις εμπορικές στρατηγικές και αποφάσεις της εταιρίας και συνήθως δεν υποστηρίζεται η διαλειτουργικότητα με αποτέλεσμα το κλείδωμα στον προμηθευτή (vendor lock-in).
- Είναι «μαύρα κουτιά», δηλαδή η εσωτερική λειτουργία τους δεν είναι διαφανής στο χρήστη.
- Η βελτίωση/αναβάθμιση υλικού και λογισμικού για συνεργασία με νέες αρχιτεκτονικές και λειτουργικά συστήματα δεν είναι εφικτή από το χρήστη,

εξαρτάται από τις εμπορικές στρατηγικές και αποφάσεις της εταιρίας, συνήθως συνοδεύεται από επιπρόσθετο κόστος και επηρεάζει τον κύκλο ζωής της συσκευής.

- Έχουν υψηλό κόστος, το οποίο πολλές φορές είναι απαγορευτικό για φτωχές ή αναπτυσσόμενες χώρες και για εκπαιδευτικά ιδρύματα με χαμηλό προϋπολογισμό, όπως τα σχολεία.

Στην Ελλάδα, από το Σεπτέμβριο του 2001, τα νέα εργαστήρια ΦΕ των Λυκείων προμηθεύτηκαν το υψηλού κόστους, εμπορικό σύστημα συγχρονικής λήψης και απεικόνισης MultiLog, το οποίο αποτελείται από τον καταγραφέα δεδομένων (Data Logger) DB-526 της εταιρίας Fourier και μια μεγάλη γκάμα αισθητήρων (Δύναμης, Θερμοκρασίας, Απόλυτης πίεσης αερίων, Μαγνητικού πεδίου, Φωτεινής έντασης, Μέτρησης pH, Κίνησης, Διαφοράς δυναμικού, Έντασης ρεύματος, μικρόφωνο, Χρόνου (Φωτοπύλη), Ραδιενέργειας Geiger-Müller). Το σύστημα μπορεί να συγκεντρώσει μετρήσεις από έξι (6) αισθητήρες ταυτόχρονα. Επίσης συνοδεύεται από το απαραίτητο λογισμικό (DB-Lab ή MultiLab), το οποίο αναλαμβάνει τη μεταφορά των δεδομένων στον Η/Υ, τη μαθηματική επεξεργασία και την γραφική αναπαράστασή τους στην οθόνη.

Στην εργασία των Βαμβακούση & Μακρυωνίτη (2003), επιχειρήθηκε μια καταγραφή-αποτίμηση της ενός χρόνου συμβολής του συστήματος MultiLog στην διεξαγωγή πειραματικών ασκήσεων. Στα υπέρ του συστήματος καταλογίζονται με υψηλά ποσοστά, η καινοτομία (55,5%) και η ελκυστικότητα προς τους μαθητές (58,3%), ενώ αντίθετα εμφανίζεται μειωμένη η ευκολία χρήσης του συστήματος (σπάνια-καθόλου 44,4%). Τη διετία 2014-2016 οι Ντούλας & Νίκας (2017) πραγματοποίησαν έρευνα σε 36 Λύκεια συγκεκριμένης περιφέρειας της Αττικής, όπου μελέτησαν τα βαθύτερα αίτια για την εγκατάλειψη του Multilog από τους εκπαιδευτικούς, οι οποίοι πολλές φορές αγνοούν την ύπαρξή του. Πολλά από τα προβλήματα προκύπτουν από τη μη ανανέωση/αναβάθμιση του υλικού/λογισμικού ώστε να συνεργαστεί με νέες αρχιτεκτονικές και λειτουργικά συστήματα. Κρίνεται επίσης απαραίτητη η επιμόρφωση των εκπαιδευτικών σε θέματα λογισμικού, εγκατάστασης και εκτέλεσης εργαστηριακών ασκήσεων, καθώς και η συνεργασία τους με το τοπικό Εργαστηριακό Κέντρο Φυσικών Επιστημών (ΕΚΦΕ).

Σε αποτύπωση της γνώμης των υπευθύνων ΕΚΦΕ για τις ελάχιστες υποχρεωτικές εργαστηριακές δραστηριότητες Γενικού Λυκείου, όπου το 1/5 απαιτεί τη χρήση αισθητήρων, ένα μεγάλο ποσοστό αναφέρεται στις δυσκολίες χρήσης της συσκευής MultiLog. Οι υπέυθυνοι ΕΚΦΕ πιστεύουν ότι τα πειράματα πρέπει να γίνονται με όσο το δυνατόν

απλούστερες συσκευές και υλικά, ώστε να μπορούν να πραγματοποιηθούν με άνεση από τους μαθητές. Επίσης εκφράζουν την επιθυμία επιμόρφωσης στη χρήση συσκευών/λογισμικών MBL και στην απεικόνιση/επεξεργασία δεδομένων με τη χρήση αισθητήρων. Για τις υποχρεωτικές δραστηριότητες του Γυμνασίου, φαίνεται ότι ένα ποσοστό περίπου 50% διαφωνούν ή διαφωνούν απόλυτα για πειραματικές δραστηριότητες με τη χρήση αισθητήρων και βρίσκουν ότι οι προτεινόμενες δραστηριότητες δεν μπορούν να υλοποιηθούν με τον εργαστηριακό εξοπλισμό που κατά μέσο όρο υπάρχει στα Γυμνάσια (Ραγιαδάκος κ.α., 2008).

Εκτός από το υψηλό κόστος και τη δυσκολία χρήσης, οι Πάλλας & Ορφανάκης (2016) αναφέρουν τη μη συμβατότητα της συσκευής MultiLog με αισθητήρες διαφορετικής τεχνολογίας και παρουσιάζουν όπως και οι Νούσης & Νούση (2013) τις δυνατότητες αξιοποίησης της πλατφόρμας Arduino σε ένα εναλλακτικό, χαμηλού κόστους, σύγχρονο σύστημα λήψης και απεικόνισης δεδομένων.

Η πρόοδος στην τεχνολογία των ηλεκτρονικών και των ολοκληρωμένων κυκλωμάτων (Integrated Circuits, IC) έχει οδηγήσει στην ανάπτυξη χαμηλού κόστους και ανοιχτής πρόσβασης εργαστηριακών συστημάτων που βασίζονται κυρίως σε ανοιχτό υλικό και λογισμικό. Τα τελευταία χρόνια αποτελούν μία αξιόπιστη λύση για την αντιμετώπιση των προβλημάτων που αναφέρθηκαν. Σε αντίθεση με το λογισμικό ανοιχτού κώδικα (Open Source Software, OSS) το οποίο χρησιμοποιείται ευρέως, το ανοιχτό υλικό (Open Source Harware, OSH) είναι αρκετά νέο (Faugel & Bobkov, 2013). Η πλατφόρμα Arduino, αποτελεί ένα από τα πιο δημοφιλή project ανοιχτού υλικού και η αξιοποίηση τεχνολογιών που βασίζονται σε μικροελεγκτές προσφέρει ευέλικτους τρόπους για τον έλεγχο των πειραμάτων, την πρόσκτηση δεδομένων, και τη διασύνδεση εργαστηριακών οργάνων σε προσωπικούς υπολογιστές. Ο μικροελεγκτής, ως ένα προγραμματιζόμενο ολοκληρωμένο κύκλωμα το οποίο διαθέτει κεντρική μονάδα επεξεργασίας, μνήμη και κυκλώματα ελέγχου περιφερειακών συσκευών, δύναται να αυτοματοποιήσει διαδικασίες, να διαδραματίσει κεντρικό ρόλο στην ανάπτυξη μικρού μεγέθους εργαστηριακών οργάνων για τη μελέτη πεδίου, να διευκολύνει την τηλεανίχνευση και την κατασκευή χαμηλού κόστους συστημάτων πρόσκτησης και ελέγχου δεδομένων (Mabbott, 2014).

Ένα από τα σημαντικότερα πλεονεκτήματα των ανοιχτών συστημάτων, εκτός από τη μείωση του κόστους και την υποστήριξη της διαλειτουργικότητας, είναι η ανάπτυξη κοινοτήτων χρηστών που συνεργάζονται, σχεδιάζουν και αναπτύσσουν υλικό και λογισμικό το οποίο μπορεί να τροποποιηθεί, βελτιωθεί επεκταθεί, ώστε να καλύψει συγκεκριμένες

προδιαγραφές, καθώς και τις ερευνητικές, εκπαιδευτικές ή παιδαγωγικές ανάγκες των εκπαιδευτικών ιδρυμάτων.

Στη συνέχεια παρουσιάζεται περιεκτική βιβλιογραφική επισκόπηση των τελευταίων ετών που αφορά την ανάπτυξη χαμηλού κόστους αξιόπιστων συστημάτων καταγραφής ή πρόσκτησης δεδομένων γενικού ή ειδικού τύπου, που βασίζονται σε μικροελεγκτές.

1.3 Συστήματα πρόσκτησης δεδομένων χαμηλού κόστους: βιβλιογραφική επισκόπηση

Ερευνητικά ή εκπαιδευτικά εργαστήρια σε όλο τον κόσμο εκτελούν πειράματα χρησιμοποιώντας διάφορους τύπους αισθητήρων, που συνδέονται μέσω κατάλληλης διεπαφής με υπολογιστή για την πρόσκτηση και γραφική αναπαράσταση των δεδομένων. Διατυπώνουν υποθέσεις, σχεδιάζουν και εκτελούν πειράματα, καταγράφουν, αναλύουν και ερμηνεύουν τα πειραματικά δεδομένα, εξάγουν επιστημονικά συμπεράσματα.

Τα τελευταία χρόνια τα εκπαιδευτικά ιδρύματα, ακολουθώντας μια νέα προσέγγιση που αφορά την εκπαιδευτική μεθοδολογία και τα εργαστηριακά όργανα μέτρησης και καταγραφής φυσικών μεγεθών, αναπτύσσουν, σύμφωνα με τις ειδικές εκπαιδευτικές ανάγκες τους, ευέλικτα, εκπαιδευτικά αξιόπιστα, χαμηλού κόστους και ανοιχτής πρόσβασης επιστημονικά εργαλεία για εργαστηριακά πειράματα (Grinias et al., 2016; OpenLabTools Project, 2017; Zachariadou et. al., 2012). Ακολουθώντας τη φιλοσοφία του ανοιχτού υλικού και λογισμικού έχουν υλοποιηθεί αρκετά projects ανάπτυξης ολοκληρωμένων γενικού τύπου συστημάτων DAQ (Data Acquisition System) με στόχο να κάνουν τις μετρήσεις στα εκπαιδευτικά εργαστήρια απλές, ακριβείς και διαφανείς, όπως το Plasduino (Baldini et al., 2014), το Edaq530 (Kopasz et al., 2011), το openDAQ (Martín et al., 2014) ή το LibreLab (Real et al., 2015). Επίσης το Instrumentino (Koenka et al., 2014; 2015) είναι ένα ανοιχτού κώδικα πλαίσιο (framework) ανάπτυξης προσαρμοσμένων προγραμμάτων GUI, γραμμένο σε γλώσσα προγραμματισμού Python, για τον έλεγχο και παρακολούθηση πειραματικών συστημάτων που βασίζονται στην πλατφόρμα Arduino. Αρκετές ερευνητικές εργασίες, κυρίως από φτωχές ή αναπτυσσόμενες χώρες, αναφέρονται στο σχεδιασμό και την ανάπτυξη χαμηλού κόστους αξιόπιστων συστημάτων DAQ, που βασίζονται σε μικροελεγκτές, για τη μέτρηση και γραφική απεικόνιση είτε συγκεκριμένων παραμέτρων είτε συστημάτων με πολλαπλές λειτουργίες, ώστε να αξιοποιηθούν σε περιβαλλοντικές μετρήσεις (Al-Mamum et al., 2013.; Davitadze et al., 2016; Saraswati et al., 2015; Simić, 2014), βιομηχανικές εφαρμογές (Nath et al., 2014), αλλά και στην εκπαίδευση ως συστήματα MBL (Atkin, 2016;

Naveenkumar & Prasad, 2013; Rosenberg et al., 2012; Tho & Hussain, 2011). Ειδικά για τα σχολικά εργαστήρια, το χαμηλό κόστος καθιστά εφικτό για τους εκπαιδευτικούς με πολύ περιορισμένους οικονομικούς πόρους να παρέχουν χρήσιμο εργαστηριακό εξοπλισμό σε ομάδες μαθητών, ώστε να εμπλακούν σε πραγματικά "hands-on" πειράματα (Huang, 2015; Zieris, 2014; Νούσης & Νούση, 2013; Πάλλας & Ορφανάκης, 2016).

Η ανοιχτού υλικού και ανοιχτού κώδικα πλατφόρμα Arduino υιοθετείται σε αρκετά συστήματα (Baldini et al., 2014; Davitadze, 2016; Grinias et al., 2016; Naveenkumar & Prasad, 2013; Saraswati et al., 2015; Zachariadou et. al., 2012; Νούσης & Νούση, 2013), όπως και οι μικροελεγκτές της σειράς AVR ATmega (Al-Mamum et al., 2013; Martín et al., 2014; Real et al.; Simić, 2014). Για την ανάπτυξη του λογισμικού διαχείρισης και γραφικής διεπαφής χρήστη αξιοποιούνται εμπορικά πακέτα (Zachariadou et. al., 2012; Νούσης & Νούση, 2013; Al-Mamum et al., 2013; Nath et al., 2014; Naveenkumar & Prasad, 2013), ελεύθερο λογισμικό (Simić, 2014) ή γλώσσες προγραμματισμού όπως η Python (Baldini et al., 2014; Grinias et al., 2016; Martín et al., 2014; Real et al.; Tho & Hussain, 2011).

Το χαμηλού κόστους, πλήρως ελεγχόμενο από υπολογιστή σύστημα SolarInsight που ανέπτυξαν οι Zachariadou et al. (2012), έχει ως κύριο στόχο να εφοδιάσει τους φοιτητές με τα απαραίτητα όργανα μέτρησης και καταγραφής φυσικών μεγεθών, εργαλεία λογισμικού και μεθοδολογία, προκειμένου να μάθουν βασικές έννοιες της Φυσικής ημιαγωγών. Το σύστημα βασίζεται στην πλατφόρμα Arduino UNO, με ενσωματωμένο τον 8-bit μικροελεγκτή AVR ATmega328P. Το εύχρηστο γραφικό περιβάλλον διεπαφής, που αποτελείται από πέντε ανεξάρτητες καρτέλες - κάθε μία για ένα διαφορετικό πείραμα - αναπτύχθηκε με τη γλώσσα προγραμματισμού C#, χρησιμοποιώντας το περιβάλλον Visual Studio 2010 Professional, που διατίθεται ελεύθερα στους φοιτητές. Επίσης, αξιοποιήθηκε το ανοιχτού κώδικα πακέτο ανάλυσης δεδομένων μεγάλης κλίμακας ROOT (C++) -που δημιουργήθηκε στο CERN-, για την επεξεργασία των πειραματικών μετρήσεων (γραφική αναπαράσταση, προσαρμογή θεωρητικών μοντέλων στις πειραματικές μετρήσεις).

Επίσης οι Νούσης και Νούση (2013) ανέπτυξαν ένα πρωτότυπο σύστημα συγχρονικής λήψης και απεικόνισης βασισμένο στην πλατφόρμα Arduino Uno για χρήση στα εργαστήρια ΦΕ των Γυμνασίων και Λυκείων. Το πρωτότυπο διαθέτει τρεις (3) εισόδους για αναλογικούς ή ψηφιακούς αισθητήρες, μία για ειδικού τύπου αισθητήρες και δύο στις οποίες μπορούν να συνδεθούν αισθητήρες που χρησιμοποιούν το πρωτόκολλο OneWire. Όμως αποτελεί προϋπόθεση η ανάπτυξη των κατάλληλων αισθητήρων ή άλλων συσκευών επέκτασης που θα είναι συμβατοί με το πρωτότυπο. Στις δυνατότητες της εφαρμογής datArdu που

προγραμμάτισαν σε περιβάλλον Delphi περιλαμβάνονται: ρύθμιση της σειριακής επικοινωνίας, έλεγχος έναρξης και λήξης της δειγματοληψίας, καθορισμός του ρυθμού δειγματοληψίας, ενημέρωση για το πλήθος και το είδος των συνδεδεμένων αισθητήρων, αποθήκευση των δεδομένων σε αρχείο κειμένου ή αυτόματη αποστολή σε φύλλο του Excel, ευέλικτος τρόπος γραφικής απεικόνισης των δεδομένων.

Στην πλατφόρμα Arduino βασίζεται επίσης το Plasduino, ένα ανοικτού υλικού/λογισμικού, ευέλικτο και εύχρηστο σύστημα πρόσκτησης δεδομένων γενικής χρήσης (general-purpose data acquisition system), ειδικά σχεδιασμένο για εκπαιδευτικά πειράματα φυσικής. Η πολυνηματική (multi-threaded) εφαρμογή για τον έλεγχο του συστήματος αναπτύχθηκε εξ 'ολοκλήρου με τη γλώσσα προγραμματισμού Python, ενώ για τη γραφική διεπαφή χρήστη χρησιμοποιήθηκε η βιβλιοθήκη PyQt. Μερικά από τα βασικά χαρακτηριστικά της εφαρμογής είναι η αυτόματη ανίχνευση της πλατφόρμας Arduino, η αυτόματη μεταφόρτωση του προγράμματος στον μικροελεγκτή και η διαχείριση της συλλογής, επεξεργασίας και αποθήκευσης των δεδομένων (Baldini et. al., 2014).

Παρόμοια οι Grinias et al. (2016) ανέπτυξαν μια ανοιχτού κώδικα συσκευή πρόσκτησης δεδομένων, χρησιμοποιώντας τη διεπαφή Arduino Uno και έναν 16-bit αναλογοψηφιακό μετατροπέα, ώστε να αξιοποιηθεί σε πειράματα αναλυτικής χημείας και να συνεργαστεί με εμπορικά ή χαμηλού κόστους DIY εργαστηριακά όργανα (π.χ. χρωματογράφος αερίων). Το λογισμικό ανοικτού κώδικα για τον έλεγχο της συσκευής αναπτύχθηκε σε Python, ενώ για τη γραφική διεπαφή χρήστη χρησιμοποιήθηκαν οι βιβλιοθήκες PyQt και PyQTGraph. Ο χρήστης έχει τη δυνατότητα να επιλέξει τη σειριακή θύρα για τη μεταφορά των δεδομένων στον συνδεδεμένο υπολογιστή, το αναλογικό ή το ψηφιακό κανάλι εισόδου του Arduino, το ρυθμό πρόσκτησης των δεδομένων (Hz) και την αντίστοιχη χρονική διάρκεια ανανέωσης αρχείου/γραφικής παράστασης.

Η γλώσσα προγραμματισμού Python και η βιβλιοθήκη Tkinter χρησιμοποιήθηκε και από τους Tho & Hussain (2011) για τη σχεδίαση του γραφικού περιβάλλοντος διεπαφής του συστήματος MBL, που ανέπτυξαν για τη μελέτη του νόμου πίεσης του αερίου (Gay-Lussac), ώστε να αξιοποιηθεί στην εκπαιδευτική διαδικασία. Επιλέχθηκε η διεπαφή PHOENIX (Physics with Home-made Equipment and Innovative Experiments, Inter-University Accelerator Centre), καθώς και αισθητήρες πίεσης και θερμοκρασίας. Η διεπαφή βασίζεται στον 8-bit μικροελεγκτή AVR ATmega16. Μετά τη ρύθμιση της συσκευής οι μαθητές/φοιτητές είναι σε θέση να παρατηρήσουν τις τιμές της θερμοκρασίας και της πίεσης, που καταγράφονται και απεικονίζονται γραφικά στην οθόνη του υπολογιστή κάθε 5 sec.

Επιπλέον παρέχεται η δυνατότητα εκτύπωσης και αποθήκευσης των αποτελεσμάτων (αριθμητικές τιμές, γραφική παράσταση).

Σε μικροελεγκτή της σειράς ATmega (AVR ATmega128) βασίζεται και το σύστημα πρόσκτησης δεδομένων (data acquisition system) για περιβαλλοντικούς αισθητήρες που ανέπτυξε ο Simić (2014). Για την επαλήθευση του συστήματος χρησιμοποιήθηκε ο ψηφιακός αισθητήρας υγρασίας και θερμοκρασίας SHT11. Το σύστημα παρέχει σε πραγματικό χρόνο πρόσβαση στις μετρήσεις των αισθητήρων και γραφική αναπαράσταση. Μπορεί να χρησιμοποιηθεί σαν αυτόνομη (standalone) διάταξη, αλλά επίσης μπορεί να συνδεθεί με τον υπολογιστή μέσω σειριακής επικοινωνίας (USB), για πιο λεπτομερή ανάλυση των αποτελεσμάτων μέσω εφαρμογής που αναπτύχθηκε με το ελεύθερο λογισμικό Microsoft Visual Studio Express. Ο χρήστης έχει τη δυνατότητα να ρυθμίσει τη θύρα επικοινωνίας, το ρυθμό δειγματοληψίας, τις τιμές συναγερμού για τη θερμοκρασία και την σχετική υγρασία, καθώς και την έναρξη/τερματισμό σχεδίασης της γραφικής παράστασης των μετρήσεων. Παρέχεται επίσης δυνατότητα καταγραφής των μετρήσεων και της χρονοσήμανσης (ημερομηνία και ώρα) σε αρχείο με μορφότυπο συμβατό με MS Excel.

Παρόμοια, οι Al-Mamum et al. (2013) σχεδίασαν και ανέπτυξαν ένα χαμηλού κόστους και πλήρως αυτοματοποιημένο σύστημα καταγραφής δεδομένων (data logging system), βασισμένο στον 8-bit μικροελεγκτή AVR ATmega8, για τη μέτρηση της θερμοκρασίας μέσω του αναλογικού αισθητήρα θερμοκρασίας LM35. Επίσης, ανέπτυξαν ένα απλό και φιλικό προς το χρήστη περιβάλλον διεπαφής (GUI), χρησιμοποιώντας το εμπορικό πακέτο λογισμικού LabVIEW της εταιρίας National Instruments, για περαιτέρω επεξεργασία και αποθήκευση των δεδομένων. Το πρόγραμμα λαμβάνει με αίτημα τη θερμοκρασία σε περιοδική βάση (30 sec) και μπορεί να αποθηκεύσει τις διαφορετικές τιμές ως συνάρτηση του χρόνου στον υπολογιστή. Στο περιβάλλον εμφανίζεται η θερμοκρασία σε βαθμούς Κελσίου (αριθμητική τιμή, γραφική παράσταση, εικονικό θερμόμετρο) με την τρέχουσα ημερομηνία και ώρα. Τα αποτελέσματα αποθηκεύονται σε μορφή απλού κειμένου.

Οι Davitadze, Partenadze και Djincharadze (2016), ανέπτυξαν ένα πειραματικό πρωτότυπο για τη μέτρηση της ποιότητας του αέρα χρησιμοποιώντας την πλατφόρμα Arduino Uno και τέσσερεις αισθητήρες (υγρασίας DHT-11, θερμοκρασίας και ατμοσφαιρικής πίεσης BMP085, συγκέντρωσης μονοξειδίου του άνθρακα (CO) MQ-7, συγκέντρωσης διοξειδίου του αζώτου (NO2) WSP1110), οι οποίοι προγραμματίστηκαν με τα κατάλληλα sketches (προγράμματα στην ορολογία του Arduino). Χρησιμοποιώντας Java, Javascript, Jquery, Html, Css, PhP ανέπτυξαν μία εφαρμογή για την γραφική απεικόνιση των

δεδομένων σε πρόγραμμα πλοήγησης στο Διαδίκτυο. Τα δεδομένα αποθηκεύονται σε ένα αρχείο κειμένου στον διακομιστή (server) ανά δευτερόλεπτο, και στη συνέχεια δημιουργείται η γραφική αναπαράστασή τους. Στο γραφικό περιβάλλον διεπαφής, ο χρήστης μπορεί να ορίσει για κάθε ένα από τα πέντε γραφήματα που εμφανίζονται, το όνομα, το χρόνο ανανέωσης, τα όρια των τιμών και το χρώμα.

Στο σύστημα παρακολούθησης και πρόσκτησης δεδομένων θερμοκρασίας (Monitoring Data Acquisition System) που ανέπτυξαν οι Nath et al. (2014), ώστε να αξιοποιηθεί σε βιομηχανικές εφαρμογές, όπου απαιτείται υψηλή ευαισθησία ανίχνευσης αλλαγών θερμοκρασίας, ο ψηφιακός I2C αισθητήρας θερμοκρασίας TMP275 με διακριτική ικανότητα 0.0625°C συνδέθηκε στον 8-bit Freescale Semiconductor μικροελεγκτή MC9S08JM60. Για την γραφική αναπαράσταση (σε πραγματικό χρόνο) και αποθήκευση των δεδομένων στον υπολογιστή, δημιουργήθηκε γραφικό περιβάλλον διεπαφής με το λογισμικό LABVIEW, στο οποίο ο χρήστης μπορεί να ρυθμίσει την οριακή τιμή της θερμοκρασίας και το ρυθμό δειγματοληψίας.

Η εργασία του Atkin (2016) πραγματεύεται την κατασκευή μιας πολύ απλής συσκευής, για τη μέτρηση της πυκνότητας μαγνητικού πεδίου (teslameter), χρησιμοποιώντας συνήθη εργαστηριακό εξοπλισμό και χαμηλού κόστους εξαρτήματα, ώστε να αξιοποιηθεί στην εκπαίδευση για να διευκολυνθεί η μελέτη των μαγνητικών πεδίων που παράγονται από ένα ρευματοφόρο πηνίο. Για να παραχθεί η γραφική παράσταση της πυκνότητας μαγνητικής ροής ως συνάρτηση της απόστασης, η συσκευή συνδέθηκε με τον μικροελεγκτή Arduino και επιλέχθηκε το χαμηλού κόστους λογισμικό MakerPlot (Windows) για τη γραφική σχεδίαση των δεδομένων που λαμβάνονται μέσω σειριακής σύνδεσης.

Τα τελευταία είκοσι χρόνια πολλά πανεπιστήμια και ερευνητικά κέντρα αναπτύσσουν εύχρηστα και χαμηλού κόστους πρωτότυπα συστήματα (υλικού και λογισμικού). Όμως σε πολλές περιπτώσεις αυτά τα εναλλακτικά ερευνητικά συστήματα δεν είναι γενικής χρήσης ή ανοιχτού υλικού/λογισμικού, ενώ για την αξιοποίησή τους εκτός του πανεπιστημιακού χώρου απαιτείται συνεργασία και υποστήριξη από την αντίστοιχη ερευνητική ομάδα υλοποίησης. Η χαμηλού κόστους, ανοιχτού υλικού και ανοιχτού κώδικα πλατφόρμα ανάπτυξης ηλεκτρονικών «πρωτοτύπων» Arduino, έφερε στο προσκήνιο την έννοια του ανοιχτού υλικού (Free and Open-Source Hardware, FOSH) (Kushner, 2011).

"*Το ανοικτό υλικό είναι υλικό του οποίου ο σχεδιασμός είναι δημοσίως διαθέσιμος έτσι ώστε ο καθένας να μπορεί να μελετήσει, μετατρέψει, διανείμει, κατασκευάσει και πουλήσει τον*

σχεδιασμό ή το υλικό με βάση αυτόν τον σχεδιασμό. Χρησιμοποιεί προσιτά εξαρτήματα και υλικά, τυποποιημένες διαδικασίες, ανοικτές υποδομές, χωρίς περιορισμούς περιεχόμενο και ανοικτού κώδικα εργαλεία σχεδιασμού για να μεγιστοποιήσει την ικανότητα του καθενός για να δημιουργεί και να κάνει χρήση του υλικού" (http://www.oshwa.org/definition/greek/).

Όπως αναφέρουν οι Mellis et al. (2007), το Arduino έχει μείνει ανοιχτό με πολλούς διαφορετικούς τρόπους. Τα αρχεία σχεδιασμού υπόκεινται στην άδεια Creative Commons Αναφορά Δημιουργού-Παρόμοια Διανομή, ο πηγαίος κώδικας του Java περιβάλλοντος ανάπτυξης υπόκεινται στη Γενική Άδεια Δημόσιας Χρήσης (GNU GPL) και οι C/C++ βιβλιοθήκες στην Ελάσσονα Γενική Άδεια Δημόσιας Χρήσης (GNU LGPL). Το σύστημα (υλικό και λογισμικό) μπορεί να μελετηθεί, να κατανοηθεί η λειτουργία του, να μετατραπεί, ενώ επιτρέπεται η επαναχρησιμοποίηση του σχεδιασμού σε άλλα πλαίσια. Τέλος είναι ανοιχτή η δημιουργία εκπαιδευτικού υλικού και η οργάνωση εργαστηρίων (workshops).

Με την ανάπτυξη του ανοιχτού υλικού και ανοιχτού λογισμικού, μειώνεται δραστικά το κόστος της πειραματικής έρευνας στις επιστήμες. Επίσης, δίνεται η ευκαιρία στα εκπαιδευτικά ιδρύματα να αγοράσουν ή να αναπτύξουν χαμηλού κόστους εργαστηριακό εξοπλισμό. Όπως αναφέρουν οι Bouquet & Bobroff (2017), η αξιοποίηση χαμηλού κόστους εξοπλισμού σε πειράματα Φυσικής, ανοίγει ενδιαφέρουσες δυνατότητες για νέα προγράμματα σπουδών στις αναπτυσσόμενες χώρες, στα εκπαιδευτικά ιδρύματα που προωθούν πειράματα «do-it-yourself», ή στα διαδικτυακά μαθήματα όπως το MOOC (Μαζικά Ελεύθερα Διαδικτυακά Μαθήματα). Επίσης η ενσωμάτωση του Arduino σε εκπαιδευτικά εργαστηριακά project δίνει τη δυνατότητα σε φοιτητές/μαθητές να αποκτήσουν δεξιότητες προγραμματισμού και κατασκευής απλών ηλεκτρονικών κυκλωμάτων, να εμπλακούν σε πρακτικές (hands-on) πειραματικές δραστηριότητες που προωθούν τη δημιουργικότητα και την ανταλλαγή ιδεών, να αναπτύξουν και να προγραμματίσουν εργαστηριακά όργανα ανίχνευσης και συλλογής δεδομένων (Bouquet and Bobroff, 2017; Haugen & Moore, 2014; Kuan et al., 2016; Mabbott, 2014; Petry et al.,2016; Urban, 2016).

Τα διάφορα μοντέλα της πλατφόρμας Arduino που διατίθενται προσυναρμολογημένα στην αγορά με χαμηλό κόστος, αυθεντικά ή συμβατά (Arduino-compatible) αλλά όχι «μαύρα κουτιά», ανοίγουν ενδιαφέρουσες προοπτικές στα σχολικά εργαστήρια ΦΕ. Είναι σε θέση να διαβάζουν ένα ευρύ φάσμα αισθητήρων, να ελέγχουν ένα ευρύ φάσμα συσκευών εξόδου, να επικοινωνούν με λογισμικό που τρέχει σε έναν υπολογιστή, ή να ανταλλάσσουν πληροφορίες μέσω δικτύου (Banzi, 2011; Mellis et al, 2007).

1.4 Το αντικείμενο ανάλυσης και ο σκοπός της παρούσας μελέτης

Όπως προκύπτει από την επισκόπηση της σχετικής με το θέμα βιβλιογραφίας, τα τελευταία χρόνια αναδεικνύεται μεγάλο ενδιαφέρον στην ανάπτυξη χαμηλού κόστους συστημάτων που βασίζονται σε μικροελεγκτή, για την πρόσκτηση δεδομένων και τον έλεγχο πειραμάτων, ειδικού ή γενικού τύπου.

Τα ειδικού τύπου συστήματα αφορούν πρωτότυπες πειραματικές διατάξεις και εργαλεία λογισμικού για

- εμπλουτισμό των εργαστηρίων Πανεπιστημίων, Τεχνολογικών Ιδρυμάτων ή Σχολείων με χαμηλού κόστους ειδικό εργαστηριακό, εκπαιδευτικό και ερευνητικό εξοπλισμό,
- αξιοποίηση σε εξειδικευμένο εύρος εργαστηριακών πειραμάτων και συνεργασία με εμπορικά ή αυτοσχέδια όργανα του εργαστηρίου,
- περιβαλλοντικές μετρήσεις και μελέτες

Τα ολοκληρωμένα γενικού τύπου ανοιχτά συστήματα, απευθύνονται κυρίως στην τριτοβάθμια εκπαίδευση, ώστε να αναπτυχθούν ή προσαρμοστούν κατάλληλα και αξιοποιηθούν σε ερευνητικά projects ή εργαστηριακά πειράματα, ενώ είναι εφικτή και η εμπορική αξιοποίησή τους.

Όσον αφορά τα σχολικά εργαστήρια ΦΕ, οι Kopasz et al. (2011) αναφέρουν ότι ο πρωταρχικός στόχος στην ανάπτυξη του ερευνητικού συστήματος Edaq530 (www.noise.inf.u-szeged.hu/edudev/EDAQ530/), ήταν να εξυπηρετήσει τις ανάγκες των εκπαιδευτικών ΦΕ δευτεροβάθμιας εκπαίδευσης για πειράματα επίδειξης ή μετρήσεις στη Βιολογία και Χημεία. Επίσης οι Baldini et al. (2014) ξεκίνησαν με προοπτική, μία συνεργασία με σχολεία δευτεροβάθμιας εκπαίδευσης στην Πίζα της Ιταλίας, στα οποία παρείχαν πλήρως συναρμολογημένα συστήματα Plasduino (pythonhosted.org/plasduino/). Στην Ελλάδα οι Νούσης & Νούση (2013) ανέπτυξαν ένα πρωτότυπο σύστημα συγχρονικής λήψης και απεικόνισης βασισμένο στην πλατφόρμα Arduino Uno για χρήση στα εργαστήρια ΦΕ, με την προϋπόθεση της ανάπτυξης των κατάλληλων αισθητήρων ή άλλων συσκευών επέκτασης, ενώ για την ανάπτυξη της εφαρμογής datArdu χρησιμοποίησαν το εμπορικό περιβάλλον προγραμματισμού Delphi.

Όμως οι εκπαιδευτικοί ΦΕ δευτεροβάθμιας εκπαίδευσης δεν έχουν τη γνώση ή τον εξοπλισμό για να συναρμολογήσουν εξαρχής εργαστηριακά συστήματα ακολουθώντας

οδηγίες από ανοιχτά αρχεία σχεδιασμού ή να προσαρμόσουν ένα πολύπλοκο λογισμικό διαχείρισης ώστε να καλύπτει τις εργαστηριακές και εκπαιδευτικές ανάγκες τους. Επομένως για την αξιοποίηση των προαναφερθέντων γενικού τύπου ανοιχτών συστημάτων στα σχολικά εργαστήρια ΦΕ κρίνεται απαραίτητη η συνεχής συνεργασία με τις αντίστοιχες ερευνητικές ομάδες ανάπτυξης, καθώς και η παροχή υποστήριξης στην προμήθεια, συντήρηση και αναβάθμιση υλικού και λογισμικού.

Συμπερασματικά, τα υπάρχοντα συστήματα εμφανίζουν τα ακόλουθα μειονεκτήματα, όσον αφορά στη χρήση τους σε σχολικά εργαστήρια ΦΕ:

- Κάποια από τα υπάρχοντα είναι ειδικού τύπου και ως εκ τούτου δεν μπορούν να καλύψουν ικανοποιητικά όλες τις ανάγκες των παραπάνω εργαστηρίων, παρά μόνο μετά από σημαντική επέμβαση (στο λογισμικό και στο υλικό).
- Τα ολοκληρωμένα γενικού τύπου συστήματα υπερκαλύπτουν τις ανάγκες των παραπάνω εργαστηρίων. Όμως η πολυπλοκότητα που εμφανίζουν απαιτεί σημαντική επένδυση χρόνου από τους εκπαιδευτικούς ΦΕ για την εκμάθηση της χρήσης τους (κάτι που δεν μπορεί να είναι εφικτό σε όλες τις περιπτώσεις).
- Επιπλέον, η ορθή χρήση των παραπάνω συστημάτων γενικού τύπου προϋποθέτει τη συνεχή υποστήριξή τους από την Ομάδα Ανάπτυξης, το οποίο δεν μπορεί να συμβεί για όλα τα διαθέσιμα συστήματα (και μάλιστα στο εύρος που απαιτείται εάν ένα σύστημα υιοθετηθεί από μεγάλο αριθμό σχολικών εργαστηρίων).
- Κάποια από τα ολοκληρωμένα γενικού τύπου συστήματα δεν υποστηρίζονται πλέον από την ομάδα που τα ανέπτυξε, γεγονός που δείχνει ότι η πιθανή σημαντική επένδυση σε γνώση που μπορεί να έχει γίνει από κάποιον εκπαιδευτικό σε μία πλατφόρμα αυτού του τύπου μπορεί να αποβεί τελικά ατελέσφορη.
- Σε κάποια από τα συστήματα ειδικού ή γενικού τύπου η προσφερόμενη διαδικτυακή υποστήριξη είναι ελλιπής ή ανεπαρκής, το οποίο τα καθιστά μη χρηστικά έξω από την ερευνητική ομάδα που τα ανέπτυξε.

Τα παραπάνω μειονεκτήματα των υπαρχόντων συστημάτων οδήγησαν στην ανάπτυξη του προτεινόμενου συστήματος Συγχρονικής Λήψης και Απεικόνισης, το οποίο εμφανίζει τα ακόλουθα πλεονεκτήματα σε σχέση με τα διαθέσιμα συστήματα:

- Είναι γενικού τύπου, ανοικτού υλικού και ανοικτού λογισμικού, επομένως μπορεί να προσαρμοστεί και επεκταθεί για την κάλυψη οποιασδήποτε σχετικής ανάγκης Σχολικού Εργαστηρίου.

- Βασίζεται στην πλατφόρμα Arduino, η οποία διατίθεται προσυναρμολογημένη στην αγορά και ως εκ τούτου μπορεί να χρησιμοποιηθεί χωρίς ανάγκες προσαρμογών από τον χρήστη και με πολύ μικρό κόστος κτήσης. Επίσης υποστηρίζεται από μία εκτεταμένη κοινότητα χρηστών, οι οποίοι επικοινωνούν τη γνώση τους μέσω Διαδικτύου.
- Το περιβάλλον προγραμματισμού του Arduino, η γλώσσα προγραμματισμού Python, καθώς και οι βιβλιοθήκες που χρησιμοποιήθηκαν, διατίθενται ελεύθερα και ως εκ τούτου οι πιθανοί χρήστες δεν χρειάζεται να πληρώνουν το αντίστοιχο κόστος για την προσαρμογή του κώδικα στις ανάγκες τους.
- Ο κώδικας που αναπτύχθηκε, είτε για να φορτωθεί στην πλακέτα Arduino είτε για τη γραφική διεπαφή, είναι απλός και συμπαγής, οπότε ο πιθανός χρήστης μπορεί να τον χρησιμοποιήσει ως έχει ή με μικρές επεμβάσεις/επεκτάσεις, χωρίς να απαιτείται η συνεχής υποστήριξη του προγραμματιστή (developer).
- Ο κώδικας που αναπτύχθηκε για τη γραφική διεπαφή είναι διαφανής (transparent) ως προς το υλικό και μπορεί να υποστηρίξει διαφορετικά είδη Arduino και συμβατών αισθητήρων. Επίσης μπορεί να τρέξει σε διαφορετικά λειτουργικά συστήματα (όπως Linux και Windows που συναντώνται στα σχολικά εργαστήρια).
- Η ανάπτυξη του συστήματος έγινε από την αρχή με στόχο τα σχολικά εργαστήρια ΦΕ και τη χρήση του από εκπαιδευτικούς ΦΕ, οι οποίοι έχουν δυνατότητα συνεργασίας με εκπαιδευτικούς Πληροφορικής της σχολικής μονάδας για προσαρμογή/επέκταση του κώδικα.
- Στο GUI που αναπτύχθηκε έγινε προσπάθεια διατήρησης της απλότητας της διεπαφής με τον χρήστη, για να είναι προσαρμοσμένη στο επίπεδο των μαθητών.

2 ΜΕΘΟΔΟΛΟΓΙΑ

2.1 Στόχοι

Στην παρούσα εργασία προτείνεται η συνεργασία της πλατφόρμας Arduino (ανοιχτό υλικό/λογισμικό) με την γλώσσα προγραμματισμού Python, για την ανάπτυξη Συστήματος Συγχρονικής Λήψης και Απεικόνισης για τα σχολικά εργαστήρια ΦΕ, με τις παρακάτω προδιαγραφές:

- να είναι χαμηλού κόστους, ώστε να είναι δυνατή η προμήθεια εξοπλισμού για ομάδες μαθητών και όχι περιορισμός της χρήσης του μόνο σε πειράματα επίδειξης,
- να μην είναι «μαύρο κουτί», με διαφανή την εσωτερική του λειτουργία, το οποίο αυξάνει και την παιδαγωγική αξία του συστήματος,
- να είναι εύκολη η δημιουργία, ρύθμιση, χρήση, τροποποίηση, επέκτασή του,
- να λειτουργεί άμεσα, εφόσον έχουν εγκατασταθεί οι drivers για την πλακέτα Arduino,
- να είναι ανοιχτού λογισμικού, ώστε ο κώδικας να μπορεί ελεύθερα να χρησιμοποιηθεί, τροποποιηθεί, επεκταθεί από τους χρήστες του,
- να είναι ανεξάρτητο πλατφόρμας
- να ελέγχεται πλήρως από τον υπολογιστή για οπτικοποίηση των δεδομένων σε πραγματικό χρόνο, «inline» ανάλυση και μεγάλο αποθηκευτικό χώρο,
- να προσφέρει σε εκπαιδευτικούς και μαθητές ένα απλό, εύχρηστο, γενικού σκοπού γραφικό περιβάλλον διεπαφής για την πρόσκτηση δεδομένων που λαμβάνονται από ένα ευρύ φάσμα αισθητήρων, την αποθήκευση και γραφική αναπαράστασή τους σε πραγματικό χρόνο,
- να εμπνεύσει εκπαιδευτικούς και μαθητές για συμμετοχή σε κοινότητες χρηστών ανοιχτών τεχνολογιών και λογισμικού ανοιχτού κώδικα.

2.2 Πειραματική Διάταξη

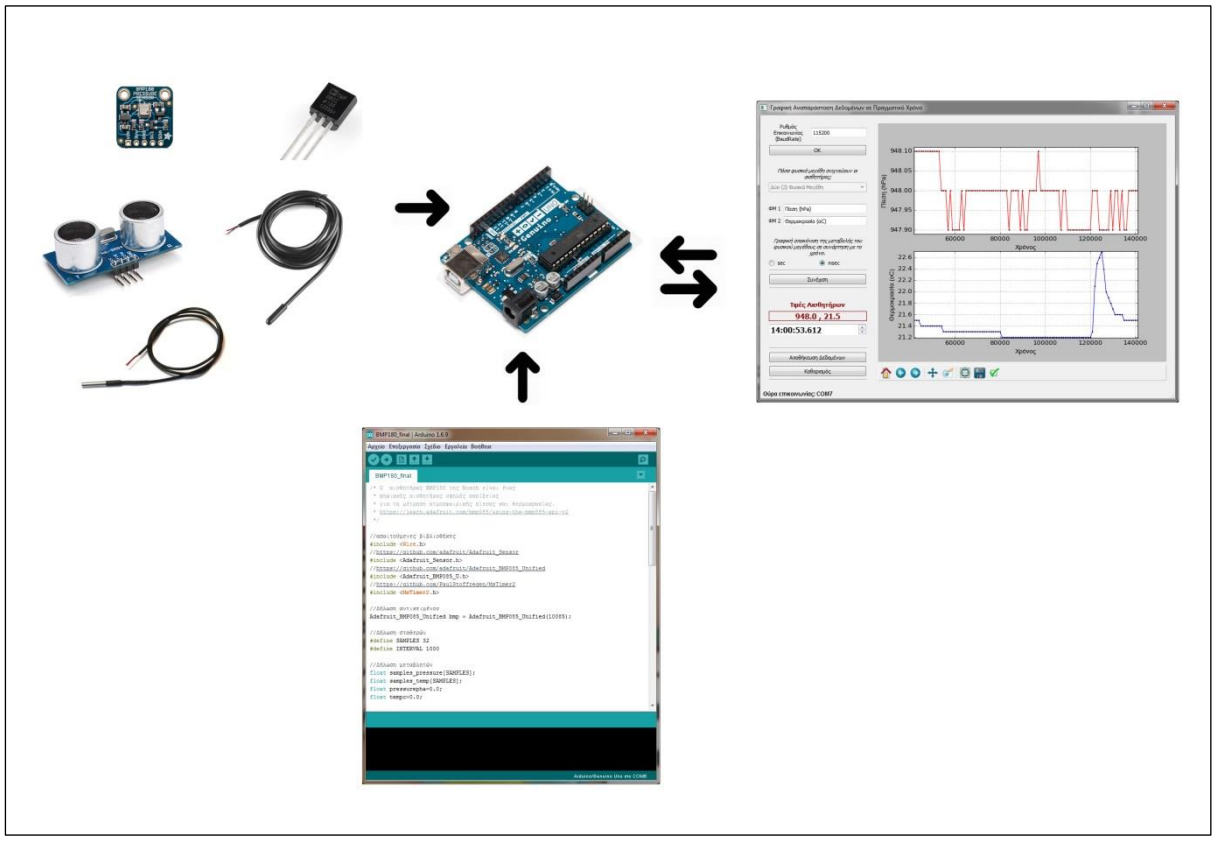
Το σύστημα αποτελείται από μία συσκευή πρόσκτησης δεδομένων (πλακέτα Arduino/Genuino Uno), η οποία διαβάζει ένα ευρύ φάσμα αισθητήρων και συνδέεται στον υπολογιστή με ένα καλώδιο USB. Οι τέσσερεις βασικές συνιστώσες από τις οποίες αποτελείται το σύστημα (Σχήμα 2.1) είναι:

(i) η πλακέτα Arduino/Genuino Uno,

(ii) αναλογικοί και ψηφιακοί αισθητήρες που συνδέονται κατάλληλα με την πλακέτα Arduino,

(iii) τα προγράμματα (sketches στην ορολογία του Arduino) για κάθε αισθητήρα ή συνδυασμό αισθητήρων, τα οποία τρέχουν στον μικροελεγκτή της πλακέτας Arduino,

(iv) η γραφική διεπαφή χρήστη (Graphical User Interface, GUI) για την αμφίδρομη επικοινωνία χρήστη-συστήματος.



Σχήμα 2.1: Πειραματική Διάταξη Συστήματος

Η πλατφόρμα Arduino προγραμματίστηκε να λειτουργεί ως συσκευή πρόσκτησης δεδομένων, η οποία λαμβάνει και επεξεργάζεται τις μετρήσεις από αναλογικούς ή ψηφιακούς αισθητήρες και αποστέλλει τα αποτελέσματα στον υπολογιστή μέσω σειριακής σύνδεσης. Από τα μοντέλα της πλατφόρμας Arduino, επιλέχθηκε η πλακέτα Arduino/Genuino Uno, εφόσον αποτελεί την πιο οικονομική λύση (~ 25€ αυθεντική έκδοση και ~12€ συμβατές εκδόσεις) και καλύπτει τις ανάγκες του ΣΣΛΑ που προτείνεται. Επίσης δοκιμάστηκε η πλακέτα Arduino/Genuino Mega (~ 45€ αυθεντική έκδοση και ~20€ συμβατές εκδόσεις), η οποία υπερκαλύπτει τις ανάγκες του συστήματος.

Για την επεξεργασία των δεδομένων που λαμβάνονται από τους αισθητήρες και αποστολή τους –με κατάλληλη μορφή- στον υπολογιστή (μέσω σύνδεσης USB), δημιουργήθηκαν ενδεικτικά προγράμματα (sketches στην ορολογία του Arduino) για αναλογικούς και ψηφιακούς αισθητήρες πίεσης, θερμοκρασίας και απόστασης. Τα sketches ακολουθούν ένα κοινό μοτίβο κώδικα ώστε να είναι εύκολη η επέκταση και προσαρμογή τους σε διαφορετικούς αισθητήρες.

Στην πλευρά του υπολογιστή, η ανοιχτού κώδικα εφαρμογή «libreMBL-GUI», η οποία αναπτύχθηκε με τη γλώσσα προγραμματισμού Python, προσφέρει ένα απλό και εύχρηστο περιβάλλον γραφικής διεπαφής για τον έλεγχο έναρξης και λήξης της γραφικής αναπαράστασης των μετρήσεων σε πραγματικό χρόνο, τη διαδραστική πλοήγηση του μαθητή στις γραφικές παραστάσεις και την αποθήκευσή δεδομένων/γραφικών παραστάσεων για μεταγενέστερη επεξεργασία.

2.3 Πλατφόρμα Arduino

Η πλατφόρμα Arduino δεν θεωρείται μόνο ένας μικροελεγκτής, αλλά ένα «Κίνημα Ανοιχτού Υλικού (Open Source Hardware Movement)» (Nayyar & Puri, 2016) που περιλαμβάνει το υλικό/λογισμικό, την ομάδα ανάπτυξης, τη φιλοσοφία σχεδιασμού και τη συναδελφική αλληλεγγύη της κοινότητας χρηστών που το υποστηρίζει (Wheat, 2011). Όπως αναφέρεται στον επίσημο δικτυακό τόπο (https://www.arduino.cc), είναι μια χαμηλού κόστους, ανοιχτού υλικού και ανοιχτού κώδικα πλατφόρμα ανάπτυξης ηλεκτρονικών «πρωτοτύπων», που βασίζεται σε εύκολο στη χρήση - αλλά και αρκετά ευέλικτο για προχωρημένους χρήστες - υλικό (hardware) και λογισμικό (software).

Ο κύριος ενσωματωμένος μικροελεγκτής (σειρά Atmega), που βασίζεται στην 8, 16 ή 32 bit αρχιτεκτονική AVR της εταιρίας Atmel (Advanced Technology for MEmory and Logic), είναι ένα προγραμματιζόμενο ολοκληρωμένο κύκλωμα, το οποίο διαθέτει επεξεργαστή, μνήμη, διάφορα περιφερειακά κυκλώματα, καθώς επίσης και θύρες εισόδου/εξόδου για επικοινωνία με εξωτερικές συσκευές. Η πλακέτα προγραμματίζεται μέσω διεπαφής USB και το πρόγραμμα αποθηκεύεται στην εσωτερική μνήμη του μικροελεγκτή.

Το περιβάλλον προγραμματισμού αποτελείται από το ανοιχτού κώδικα και ανεξάρτητο πλατφόρμας ολοκληρωμένο περιβάλλον ανάπτυξης IDE (Integrated Development Environment) που βασίζεται στο λογισμικό Processing, καθώς και στη γλώσσα προγραμματισμού Arduino, που βασίζεται στη C/C++. Το Processing είναι ένα προγραμματιστικό περιβάλλον και παράλληλα μια γλώσσα προγραμματισμού ανοικτού

κώδικα (βασίζεται στη Java), για την εύκολη εκμάθηση προγραμματισμού στο χώρο των εικαστικών τεχνών, μέσω της δημιουργίας διαδραστικών γραφικών (Processing, 2017; Reas & Make, 2015).

Το Arduino γεννήθηκε στο Ivrea Interaction Design Institute στην βόρεια Ιταλία, που προσφέρει ένα μεταπτυχιακό πρόγραμμα στη σχεδίαση διαδραστικών ψηφιακών προϊόντων, περιβαλλόντων, συστημάτων και υπηρεσιών (interaction design), υιοθετεί τη φιλοσοφία της μάθησης μέσα από την πράξη (learning by doing) και υποστηρίζει τη δημιουργία εργαλείων για τη γρήγορη δημιουργία πρωτοτύπων από μαθητές/φοιτητές που δεν έχουν υπόβαθρο στα ηλεκτρονικά και στον προγραμματισμό (Mellis et al, 2007).

Τα τελευταία χρόνια η πλακέτα Arduino αλλάζει και προσαρμόζεται στις νέες ανάγκες και προκλήσεις για εφαρμογές IoT, 3D εκτυπώσεις, ενσωματωμένα συστήματα, ενώ χάρη στη φιλοσοφία του ανοιχτού υλικού/ λογισμικού, υποστηρίζεται από μια μεγάλη κοινότητα χρηστών, που μοιράζονται ιδέες, οδηγίες και λύσεις για πρωτότυπα έργα που αλληλεπιδρούν με άλλα αντικείμενα, ανθρώπους και δίκτυα. Το Arduino είναι μια ευρέως αποδεκτή πλατφόρμα για physical computing (χρήση λογισμικού και κατάλληλου υλικού για την υλοποίηση κατασκευών που μπορούν να αισθανθούν και να ανταποκριθούν σε ερεθίσματα από τον πραγματικό κόσμο). Χρησιμοποιώντας αισθητήρες (sensors) και ενεργοποιητές (actuators) που ελέγχονται από κατάλληλο λογισμικό που τρέχει μέσα στον μικροελεγκτή, εκπαιδευτικοί και μαθητές διερευνούν φυσικά και χημικά φαινόμενα, κατασκευάζουν χαμηλού κόστους επιστημονικά όργανα, ή έρχονται σε πρώτη επαφή με τον προγραμματισμό και τη ρομποτική με έναν προσιτό και παιγνιώδη τρόπο. Σχεδιαστές και αρχιτέκτονες δημιουργούν διαδραστικά πρωτότυπα και έξυπνα συστήματα, μουσικοί και καλλιτέχνες το χρησιμοποιούν για εγκαταστάσεις και πειραματισμό με νέα μουσικά όργανα (Arduino, 2017).

2.3.1 Η *πλακέτα* Arduino/Genuino Uno

Το τελευταίο μοντέλο Arduino/Genuino Uno (Arduino Uno μόνο στις ΗΠΑ και Genuino Uno εκτός ΗΠΑ) βασίζεται στον 8-bit μικροελεγκτή ATmega328P της Atmel, με ταχύτητα χρονισμού 16MHz. Διαθέτει 32KB μνήμη τύπου Flash για την αποθήκευση του προγράμματος, εκ των οποίων τα 0,5 KB χρησιμοποιούνται από τον φορτωτή εκκίνησης (bootloader), 2KB SRAM για την εκτέλεση του προγράμματος και 1 KB EEPROM. Η πλακέτα επεκτείνει τους ακροδέκτες εισόδου/εξόδου (I/O pins) του μικροελεγκτή, ώστε να χρησιμοποιηθούν για τη διασύνδεση εξωτερικών συσκευών. Διαθέτει συνολικά 20

ακροδέκτες, εκ των οποίων οι 14 είναι ψηφιακοί ακροδέκτες εισόδου/εξόδου (digital I/O pins) και οι 6 είναι αναλογικοί ακροδέκτες εισόδου (analog input pins) (Σχήμα 2.2).

Από τους 14 ψηφιακούς ακροδέκτες, οι έξι (3, 5, 6, 9, 10, 11) είναι εφικτό να χρησιμοποιηθούν ως «ψευδοαναλογικές» έξοδοι PWM (Pulse Width Modulation) των 8-bit, υποστηρίζοντας την ελεγχόμενη παροχή ισχύος προς συνδεδεμένες συσκευές. Οι ψηφιακοί ακροδέκτες 2 και 3 μπορεί να ρυθμιστούν ώστε να προκαλέσουν εξωτερική διακοπή (external interrupt) του εκτελούμενου κώδικα στον μικροελεγκτή.

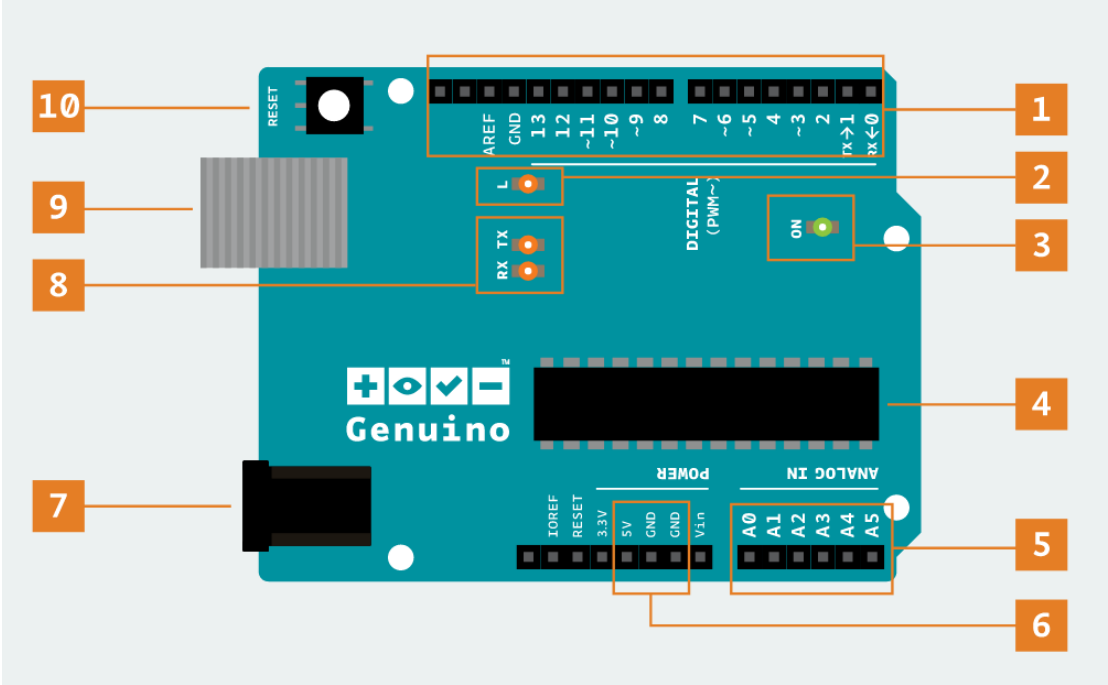
Κάθε αναλογική είσοδος (A0 έως A5) έχει ενσωματωμένο έναν αναλογοψηφιακό μετατροπέα (Analog-to-Digital Converter, ADC) με ανάλυση 10 bit (2^{10}=1024 διαφορετικές τιμές), δηλαδή μπορεί να ανιχνεύσει έως και 1024 διαφορετικά επίπεδα τάσης σε κάθε αναλογικό κανάλι. Οι είσοδοι A4 (SDA) και A5 (SCL), με χρήση της κατάλληλης βιβλιοθήκης (Wire Library), χρησιμοποιούνται για την υλοποίηση του δισύρματου πρωτοκόλλου επικοινωνίας I2C/TWI (Inter-Integrated Circuit/Two wire interface) για επικοινωνία με I2C συσκευές. Η γραμμή SCL (Serial Clock Line) είναι η γραμμή ρολογιού, ενώ η SDA (Serial Data Line) είναι η γραμμή δεδομένων.

Σχήμα 2.2: Πλακέτα Arduino/Genuino Uno (Πηγή: https://goo.gl/LUBeIG)

Η πλακέτα λειτουργεί στα 5V και οι ψηφιακοί ακροδέκτες έχουν τη δυνατότητα να παρέχουν ρεύμα μέχρι 40 mA, επαρκές για να οδηγήσουν τα περισσότερα ηλεκτρονικά

εξαρτήματα, με εξαίρεση τους κινητήρες με υψηλές απαιτήσεις ρεύματος. Διαθέτει δύο ακροδέκτες τροφοδοσίας στα 5V και 3,3 V, καθώς και ακροδέκτες γείωσης στις δύο πλευρές. Οι ψηφιακοί ακροδέκτες D0 και D1 υποστηρίζουν τη σειριακή επικοινωνία λειτουργώντας ως πομπός Tx (transmitter) και δέκτης Rx (receiver) αντίστοιχα. Η θύρα USB μπορεί να χρησιμοποιηθεί για τη σύνδεση με υπολογιστή. Η πλακέτα τροφοδοτείται μέσω της σύνδεσης USB από τον υπολογιστή ή με εξωτερική τροφοδοσία (συνιστώμενη τάση 7 έως 12V) (π.χ. μπαταρία 9-V, ή τροφοδοτικό συνεχούς ρεύματος) (Arduino, 2016; Desai, 2015).

Οι αναλογικές είσοδοι της πλακέτας έχουν τη δυνατότητα να επεξεργαστούν σήματα από αισθητήρες που μετατρέπουν το προς μέτρηση φυσικό μέγεθος σε τάση (π.χ. αισθητήρες θερμοκρασίας ή πίεσης). Οι ψηφιακές είσοδοι επεξεργάζονται δεδομένα από αισθητήρες που παρέχουν ψηφιακή έξοδο. Επίσης για σκοπούς επίδειξης μπορεί να χρησιμοποιηθούν ψηφιακές κάρτες επέκτασης (π.χ. οθόνη LED με μεγάλα ψηφία) που συνδέονται με τις ψηφιακές εξόδους. Για τα δεδομένα υπάρχει δυνατότητα να αποθηκευτούν σε μια κάρτα μνήμης SD ή να αποσταλούν στον υπολογιστή μέσω της θύρας USB.



Σχήμα 2.3: Στοιχεία της πλακέτας Arduino/Genuino Uno.
Πηγή: Arduino/Genuino Uno Board Anatomy (https://goo.gl/bttaVy)

Στο Σχήμα 2.3 παρουσιάζονται τα στοιχεία από τα οποία αποτελείται η πλακέτα Arduino/Genuino Uno:

1. **Ψηφιακοί ακροδέκτες εισόδου/εξόδου:** Χρησιμοποιούνται με τις συναρτήσεις της γλώσσας προγραμματισμού Arduino *digitalRead()*, *digitalWrite()*, και *analogWrite()*.

Η συνάρτηση *analogWrite()* λειτουργεί μόνο στους ακροδέκτες που έχουν το σύμβολο PWM (~).

2. **Pin13 LED:** Ο ακροδέκτης 13 είναι συνδεδεμένος με ένα LED, τον μόνο ενεργοποιητή (actuator) που είναι ενσωματωμένος στην πλακέτα, ο οποίος είναι πολύ χρήσιμος για τον εντοπισμό σφαλμάτων (debugging).
3. **Power LED:** Υποδεικνύει ότι η πλακέτα τροφοδοτείται με ρεύμα, ενώ είναι χρήσιμο για τον εντοπισμό σφαλμάτων.
4. **ATmega μικροελεγκτής**
5. **Αναλογικοί ακροδέκτες εισόδου:** Χρησιμοποιούνται με τη συνάρτηση της γλώσσας προγραμματισμού Arduino *analogRead()*.
6. **Ακροδέκτες γείωσης (GND) και 5V**
7. **Υποδοχή τροφοδοσίας:** Χρησιμοποιείται για την τροφοδοσία της πλακέτας όταν δεν είναι συνδεδεμένη σε μια θύρα USB, ενώ δέχεται τάση μεταξύ 7-12V.
8. **TX and RX LEDs:** Υποδεικνύουν την επικοινωνία μεταξύ της πλακέτας και του υπολογιστή αναβοσβήνοντας γρήγορα κατά τη μεταφόρτωση του προγράμματος (sketch) στον μικροελεγκτή, ή κατά τη διάρκεια της σειριακής επικοινωνίας. Είναι χρήσιμα για τον εντοπισμό σφαλμάτων.
9. **Θύρα USB:** Χρησιμοποιείται για την τροφοδοσία της πλακέτας, τη φόρτωση των προγραμμάτων (sketches) στον μικροελεγκτή, καθώς και τη μεταφορά των δεδομένων που λαμβάνει η πλακέτα Arduino από τους αισθητήρες προς τον υπολογιστή (*π.χ. μέσω Serial.println()*).
10. **Κουμπί επανεκκίνησης (reset button)**

2.3.2 Επικοινωνία Arduino - Υπολογιστή

Η πλακέτα Arduino/Genuino Uno υποστηρίζει τη σειριακή επικοινωνία. Περιλαμβάνει έναν δεύτερο 8-bit μικροελεγκτή (Atmega16U2), κατάλληλα προγραμματισμένο, ο οποίος λειτουργεί ως γέφυρα μεταξύ της θύρας USB του υπολογιστή και της σειριακής θύρας του κυρίως μικροελεγκτή (USB-to-Serial Converter).

Για να επιτευχθεί η σωστή επικοινωνία μεταξύ Arduino και υπολογιστή θα πρέπει αμφότεροι να έχουν τον ίδιο ρυθμό μετάδοσης δεδομένων (baudrate), που συνήθως εκφράζεται σε bits ανά δευτερόλεπτο (bps). Ένας από τους πιο κοινούς ρυθμούς μετάδοσης είναι 9600 bps. Σε προγραμματιστικό επίπεδο χρησιμοποιείται η εντολή *Serial.begin(speed),* όπου *speed* ο ρυθμός μετάδοσης σε bps. Για επικοινωνία με τη σειριακή κονσόλα του IDE,

μπορεί να χρησιμοποιηθούν ρυθμοί μετάδοσης 300, 1200, 2400, 4800, 9600, 19.200, 38400, 57600, 115200, 230400 ή 250000 bps.

Οι Νούσης & Νούση (2013) αναφέρουν ότι, εφόσον η σειριακή επικοινωνία Arduino-υπολογιστή υλοποιείται μέσω ενός μετατροπέα σειριακού σε USB, δύνανται να χρησιμοποιηθούν και μη προκαθορισμένοι ρυθμοί επικοινωνίας εκμεταλλευόμενοι την υψηλή ταχύτητα μεταφοράς του διαύλου USB, και προτείνουν ρυθμούς επικοινωνίας μεγαλύτερους ή ίσους από 500000 bps για σειριακή αποστολή μεγάλης ποσότητας δεδομένων σε μικρό χρόνο.

Στο σύστημα που προτείνεται στην παρούσα εργασία, επιλέχθηκε ρυθμός επικοινωνίας (baudrate) 115200 bps, ο οποίος είναι συμβατός με τα περισσότερα υπολογιστικά συστήματα και καλύπτει τις ανάγκες μετάδοσης δεδομένων για τα πειράματα που θα υλοποιηθούν σε ένα σχολικό εργαστήριο ΦΕ.

2.3.3 Χρονιστές (Timers) και Διακοπές (Interrupts)

Στο υλικό του μικροελεγκτή ATmega328P συμπεριλαμβάνονται τρεις χρονιστές (timers) που εξαρτώνται από το ρολόι του συστήματος. Οι timer0 και timer2 είναι χρονιστές διακριτικής ικανότητας 8 bit, ενώ ο timer1 είναι χρονιστής διακριτικής ικανότητας 16 bit. Αξιοποιούνται είτε για εισαγωγή καθυστέρησης κατά την εκτέλεση των εντολών του προγράμματος είτε για εκτέλεση συγκεκριμένων εντολών σε τακτά χρονικά διαστήματα. Ο χρονιστής timer0 χρησιμοποιείται στη συνάρτηση καθυστέρησης *delay()* και στις συναρτήσεις χρονομέτρησης σε πραγματικό χρόνο *millis()* και *micros()*, ενώ ο timer2 στη συνάρτηση *tone()* για τη δημιουργία ήχων.

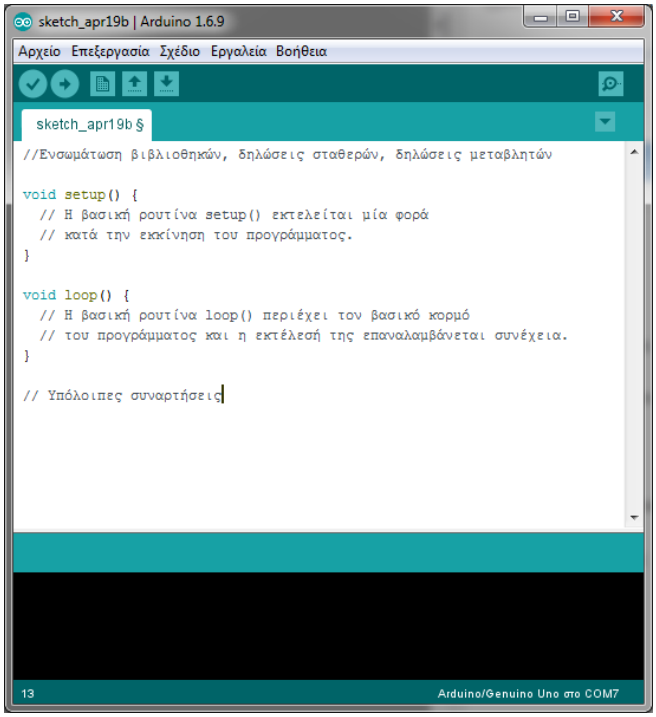
Μια διακοπή (interrupt) αποτελεί ένα σήμα που εκπέμπεται είτε από υλικό είτε από λογισμικό προς τον επεξεργαστή, ώστε να διακόψει την τρέχουσα δραστηριότητα του και να χειριστεί διεργασίες υψηλής προτεραιότητας. Μόλις ενεργοποιηθεί η διακοπή, ο επεξεργαστής ανταποκρίνεται στο αίτημα διακοπής, αναστέλλει την διεργασία που εκτελούσε, αποθηκεύει την κατάστασή του και καλεί το πρόγραμμα να εκτελέσει μια διαφορετική διεργασία που ονομάζεται χειριστής διακοπής (interrupt handler) ή ρουτίνα εξυπηρέτησης διακοπής (Interrupt Service Routine, ISR). Μόλις ολοκληρωθεί, το πρόγραμμα επιστρέφει σε αυτό που έκανε πριν ενεργοποιηθεί η διακοπή. Οι διακοπές χωρίζονται σε διακοπές υλικού (hardware interrupts) και διακοπές λογισμικού (software interrupts) όπως:

- Timer Interrupts: Διακοπές που ενεργοποιούνται από κάποιον χρονιστή του Arduino.

- External interrupts: Οι ψηφιακοί ακροδέκτες D2 και D3 του Arduino/Genuino Uno μπορεί να ρυθμιστούν ώστε να προκαλέσουν εξωτερική διακοπή.
- Pin Change Interrupts: Διακοπές που ενεργοποιούνται από την αλλαγή της κατάστασης οποιουδήποτε από μια ομάδα ακροδεκτών.

2.3.4 Περιβάλλον προγραμματισμού

Τα προγράμματα, τα οποία στην ορολογία του Arduino καλούνται "sketches", δημιουργούνται στο ανεξάρτητο πλατφόρμας ολοκληρωμένο περιβάλλον ανάπτυξης IDE (Integrated Development Environment), το οποίο είναι γραμμένο σε Java. Το IDE ενσωματώνει δυνατότητες συγγραφής και επεξεργασίας του κώδικα με συντακτική χρωματική σήμανση, μεταγλώττισης του κώδικα σε οδηγίες που αντιλαμβάνεται το υλικό και μεταφόρτωσης (upload) του μεταγλωττισμένου κώδικα στο Arduino μέσω σύνδεσης USB. Επιπλέον περιλαμβάνει τη σειριακή κονσόλα (Serial Monitor) μέσω της οποίας ο χρήστης μπορεί να αλληλεπιδρά με το σύστημα Arduino στέλνοντας κατάλληλες εντολές ή λαμβάνοντας δεδομένα. Η γλώσσα προγραμματισμού Arduino που βασίζεται στη C/C++ είναι αρκετά εύκολη για αρχάριους προγραμματιστές, ενώ υποστηρίζει κοινά χαρακτηριστικά σύγχρονων αντικειμενοστραφών γλωσσών προγραμματισμού.



Σχήμα 2.4: Ολοκληρωμένο περιβάλλον ανάπτυξης IDE

Σε ένα Arduino πρόγραμμα (sketch), περιλαμβάνονται πάντα οι δύο βασικές συναρτήσεις setup() και loop() (Σχήμα 2.4). Η βασική συνάρτηση setup(), η οποία είναι

υπεύθυνη για τη ρύθμιση του Arduino, εκτελείται μία φορά κατά την εκκίνηση του προγράμματος, ενώ η βασική συνάρτηση loop() είναι ένας ατέρμων βρόχος (infinite loop), στον οποίο ο κώδικας επαναλαμβάνεται συνέχεια. Επιπρόσθετα, υπάρχει δυνατότητα ορισμού και άλλων συναρτήσεων, οι οποίες καλούνται από τις βασικές συναρτήσεις setup() και loop().

Το περιβάλλον IDE περιέχει αρκετά έτοιμα παραδείγματα για την ευκολότερη κατανόηση της λειτουργίας του, ενώ επεκτείνεται μέσω της χρήσης βιβλιοθηκών, που παρέχουν στα sketches πρόσθετη λειτουργικότητα και προγράμματα οδήγησης συσκευών (drivers), για χειρισμό δεδομένων ή συνεργασία με εξαρτήματα που συνδέονται στην πλακέτα Arduino (π.χ. αισθητήρες, οθόνες Character LCD, σερβοκινητήρες). Εκτός από τις έτοιμες βιβλιοθήκες που συμπεριλαμβάνονται στο IDE, υπάρχουν διαθέσιμες εκατοντάδες επιπλέον βιβλιοθήκες στο Διαδίκτυο σε τοποθεσίες όπως Arduino Playground (http://playground.arduino.cc/), Github (https://github.com/) και Google Code Archive (https://code.google.com/archive/), για λήψη και εγκατάσταση μέσω του Διαχειριστή Βιβλιοθήκης (Library Manager) ή με εισαγωγή σε μορφή zip.

2.4 Αισθητήρες

Οι αισθητήρες (sensors) ή αισθητήρια είναι μετρητικές διατάξεις που ανιχνεύουν ένα φυσικό μέγεθος (δεδομένα του πραγματικού κόσμου) και το μετατρέπουν -συνήθως- σε ηλεκτρικό σήμα (τάση ή ρεύμα), παράγοντας μια μετρήσιμη έξοδο.

2.4.1 Γενικά Χαρακτηριστικά αισθητήρων

Τα γενικά χαρακτηριστικά των αισθητήρων, όπως αναφέρονται στο National Instruments (2013), Μπισδούνης (2009) και παρουσιάζονται στη συνέχεια, αποτυπώνουν την απόδοση και συμπεριφορά των αισθητήρων κατά τη διάρκεια των μετρήσεων, ενώ καθορίζουν την ποιότητα εξόδου.

- Το **εύρος λειτουργίας (operating range)** ενός αισθητήρα ορίζεται από τα όρια, εντός των οποίων μπορεί να λειτουργεί αξιόπιστα. Συνήθως, εκφράζεται με την ελάχιστη και τη μέγιστη τιμή που μπορεί να μετρήσει. Οι προδιαγραφές περιλαμβάνουν συνήθως και άλλες έννοιες εύρους (π.χ. θερμοκρασίας, πίεσης, υγρασίας).
- Η **ακρίβεια (accuracy)** είναι η μέγιστη διαφορά ανάμεσα στην έξοδο του αισθητήρα και την μετρούμενη (πραγματική) τιμή. Μπορεί να εκφράζεται είτε ως ποσοστό του εύρους τιμών του είτε ως απόλυτη τιμή.

- Η έννοια της **επαναληψιμότητας (precision),** αναφέρεται στο βαθμό κατά τον οποίο ο αισθητήρας δίνει την ίδια τιμή εξόδου, κάθε φορά που τροφοδοτείται με την ίδια είσοδο.
- Η **διακριτική ικανότητα (resolution)** ενός αισθητήρα, αναφέρεται στην ελάχιστη μεταβολή της εισόδου που θα οδηγήσει σε μια ανιχνεύσιμη μεταβολή στην έξοδό του.
- Η **ευαισθησία (sensitivity)** είναι ο λόγος της αλλαγής της εξόδου προς την αντίστοιχη αλλαγή της εισόδου. Όταν ο αισθητήρας έχει γραμμική συμπεριφορά, τότε η ευαισθησία είναι σταθερή σε όλη την περιοχή λειτουργίας του.
- Η **βαθμονόμηση (calibration)** είναι η διαδικασία στην οποία γνωστές τιμές (συνήθως πρότυπες) εισόδου εφαρμόζονται στον αισθητήρα με σκοπό την παρατήρηση της εξόδου του. Καθορίζει τις μονάδες στις οποίες βαθμολογείται η κλίμακα του οργάνου.
- Η **μετατόπιση (Offset)** ορίζεται ως η έξοδος για μηδενική τιμή εισόδου του μετρούμενου μεγέθους ή εναλλακτικά η διαφορά μεταξύ της πραγματικής τιμής εξόδου και της καθορισμένης τιμής εξόδου κάτω από συγκεκριμένες συνθήκες περιβάλλοντος.
- Η **γραμμικότητα (linearity)** εκφράζει το βαθμό στον οποίο η γραφική παράσταση της εξόδου ως προς την είσοδο του αισθητήρα προσεγγίζει μια ευθεία γραμμή.

2.4.2 Αισθητήρες συστήματος και κριτήρια επιλογής

Το σύστημα δοκιμάστηκε με αισθητήρες θερμοκρασίας, πίεσης και απόστασης, που καλύπτουν αρκετά μεγάλο εύρος πειραμάτων στη Δευτεροβάθμια αλλά και στην Πρωτοβάθμια Εκπαίδευση. Ενδεικτικά αναφέρονται πειράματα μέτρησης θερμοκρασίας και μετάδοσης θερμότητας, πειραματική προσέγγιση των νόμων των αερίων, μελέτη της απλής αρμονικής ταλάντωσης ή κίνησης σε κεκλιμένο επίπεδο, πειράματα στη βιολογία, περιβαλλοντικές μετρήσεις.

Οι αισθητήρες επιλέχθηκαν με τα παρακάτω κριτήρια:

- Γενικά χαρακτηριστικά
- Χαμηλό κόστος
- Διαθεσιμότητα στην ελληνική αγορά
- Επαρκής τεκμηρίωση από τις κατασκευάστριες εταιρίες και την κοινότητα χρηστών του Arduino

- Ευχρηστία σύνδεσης
- Δυνατότητα σύγκρισης αναλογικών και ψηφιακών αισθητήρων που ανιχνεύουν το ίδιο φυσικό μέγεθος (θερμοκρασία).
- Πειραματισμός με ψηφιακούς αισθητήρες που υποστηρίζουν τα πρωτόκολλα επικοινωνίας I2C και 1-wire.

Στον Πίνακα 2.1 παρουσιάζονται οι αισθητήρες που χρησιμοποιήθηκαν για τη δοκιμή του συστήματος.

Πίνακας 2.1: Αισθητήρες Συστήματος

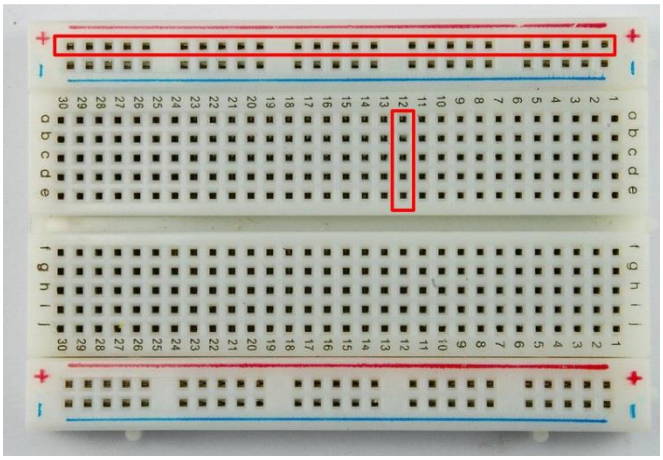
Τύπος	**Χαρακτηρισμός**	**Κόστος**
Θερμοκρασίας	Αναλογικός αισθητήρας θερμοκρασίας TMP36	~2,00 €
	Αναλογικός αδιάβροχος (waterproof) αισθητήρας θερμοκρασίας σειράς LM35	~6,00 €
	Ψηφιακός αδιάβροχος αισθητήρας θερμοκρασίας 1-wire DS18B20	~5,00 €
Πίεσης	Ψηφιακός I2C Αισθητήρας Βαρομετρικής πίεσης / θερμοκρασίας BMP180	~12,00 €
Απόστασης	Αισθητήρας Υπερήχων HC-SR04 για τον υπολογισμό απόστασης	~3,00 €

2.4.3 Διάταξη σύνδεσης αισθητήρων (breadboard)

Για την εύκολη σύνδεση και αποσύνδεση των αισθητήρων με την πλακέτα Arduino, ενδείκνυται η χρήση μιας πλακέτας πειραματισμού για ηλεκτρονικά κυκλώματα (breadboard), η οποία χρησιμοποιείται στη δημιουργία προσωρινών κυκλωμάτων και πρωτοτύπων, χωρίς να απαιτείται η συγκόλληση των εξαρτημάτων. Οι ακροδέκτες των αισθητήρων τοποθετούνται στις κατάλληλες οπές υποδοχής του breadboard και στη συνέχεια χρησιμοποιούνται πρόσθετα καλώδια δοκιμών (breadboard jumper wires) (Σχήμα 2.6) για τη διασύνδεση με την πλακέτα.

Η πλακέτα στο Σχήμα 2.5 αποτελείται από οπές υποδοχής σε δέκα (10) γραμμές (a,b,c,d,e,f,g,h,,I,j) και τριάντα (30) στήλες (1-30). Οι οπές υποδοχής, σε ομάδες των πέντε (5), συνδέονται ηλεκτρικά κάτω από την επιφάνεια της πλακέτας με μεταλλικό συνδετήρα

(π.χ. a12,b12,c12,d12,e12), με αποτέλεσμα να έχουν ηλεκτρική επαφή καλώδια ή ακροδέκτες αισθητήρων που τοποθετούνται σε αυτές.



Σχήμα 2.5: Breadboard

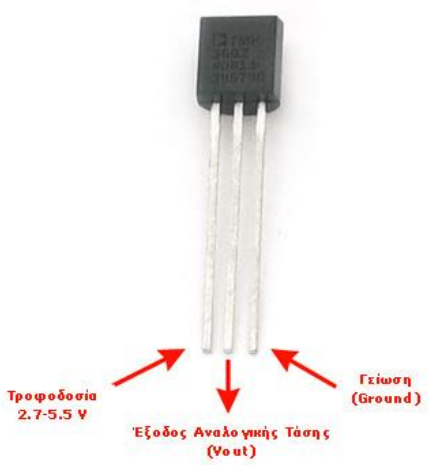
Στην πάνω ή κάτω πλευρά της πλακέτας υπάρχουν οι όμοιες κατασκευαστικά και μονωμένες μεταξύ τους γραμμές τροφοδοσίας (+) και γείωσης (-), για τη σύνδεση με την ηλεκτρική πηγή τροφοδοσίας (π.χ. ακροδέκτης τροφοδοσίας 5V της πλακέτας Arduino) και γείωσης (π.χ. ακροδέκτης γείωσης GND της πλακέτας Arduino) αντίστοιχα. Οι οπές της γραμμής (+) ή (-), συνδέονται ηλεκτρικά μεταξύ τους με μεταλλικό συνδετήρα.

Σχήμα 2.6: Καλώδια διασύνδεσης (jumper wires)

2.4.4 Αναλογικός Αισθητήρας Θερμοκρασίας TMP36

Ο TMP36 είναι ένας αισθητήρας θερμοκρασίας χαμηλής τάσης (2.7 V έως 5.5 V) και με χαμηλό κόστος. Έχει τρεις ακροδέκτες: έναν που συνδέεται με την τροφοδοσία, έναν που

συνδέεται με τη γείωση (GND), και έναν τρίτο που εξάγει μια μεταβλητή τάση, ανάλογα με τη θερμοκρασία που αισθάνεται (Σχήμα 2.7).

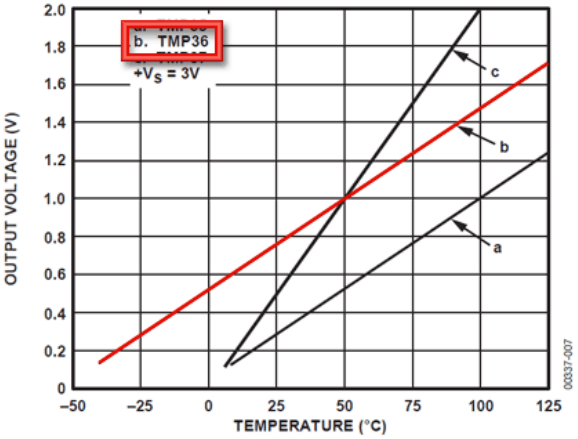


Σχήμα 2.7: Ο αισθητήρας θερμοκρασίας TMP36 και οι ακροδέκτες του (Πηγή: https://goo.gl/IRWRvf)

Η τάση εξόδου είναι γραμμικά ανάλογη με τη θερμοκρασία σε βαθμούς Κελσίου (°C), με συντελεστή μετατροπής 10 mV / °C (ή 0.01 V/°C), και είναι μετατοπισμένη κατά 0.5 V (Offset Voltage) ώστε να μετρά και αρνητικές τιμές θερμοκρασίας (Σχήμα 2.8). Επομένως για τη μετατροπή της τάσης εξόδου (Vout), σε βαθμούς Κελσίου ^{o}C, χρησιμοποιείται η σχέση:

$$\text{Temp (°C)} = [(\text{Vout σε V}) - 0.5]*100 \text{ ή}$$

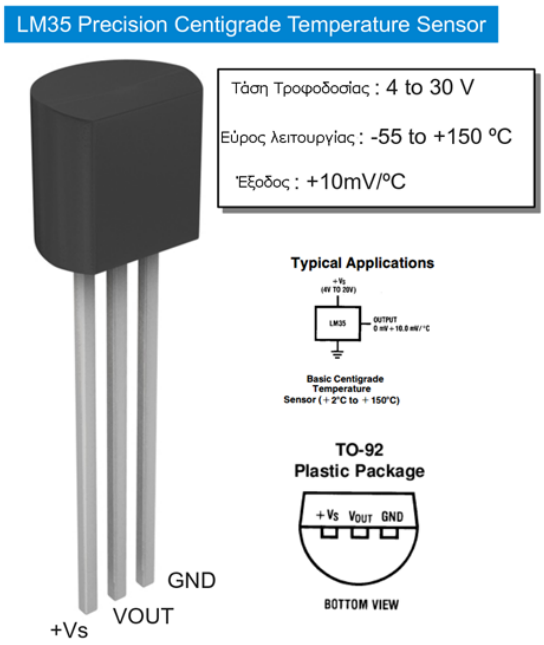
$$\text{Temp (°C)} = [(\text{Vout σε mV}) - 500] / 10$$



Σχήμα 2.8: Τάση εξόδου ως προς τη θερμοκρασία (γραμμή b) για τον αισθητήρα θερμοκρασίας TMP36 (Πηγή: https://goo.gl/IRWRvf)

Ο αισθητήρας TMP36 δεν απαιτεί εξωτερική βαθμονόμηση και μπορεί να μετρήσει θερμοκρασίες από -40 έως +125 βαθμούς Κελσίου με ακρίβεια ±2°C, ενώ κοντά στους +25°C η ακρίβεια βελτιώνεται στο ±1°C. Το ρεύμα λειτουργίας του είναι κάτω από 50 μA, προκαλώντας μικρή αυτοθέρμανση (λιγότερο από 0.1 °C σε νηνεμία). Θεωρείται κατάλληλος για εφαρμογές στην αυτοκινητοβιομηχανία, σε συστήματα περιβαλλοντικού ελέγχου, αυτοματισμούς ελέγχου θερμοκρασίας, βιομηχανικές εφαρμογές, συναγερμούς πυρκαγιάς, κλπ. (TMP36 DataSheet, https://goo.gl/oDvHSY).

2.4.5 Αναλογικοί αισθητήρες Θερμοκρασίας Σειράς LM35



Σχήμα 2.9: Αισθητήρας Θερμοκρασίας LM35, χαρακτηριστικά, τυπική σύνδεση (Πηγή: https://goo.gl/zywe3l)

Ο αισθητήρας σειράς LM35 έχει τρεις ακροδέκτες: έναν που συνδέεται με την τροφοδοσία, έναν που συνδέεται με τη γείωση (GND), και έναν τρίτο που εξάγει μια μεταβλητή τάση, ανάλογα με τη θερμοκρασία που αισθάνεται (Σχήμα 2.9, 2.10).

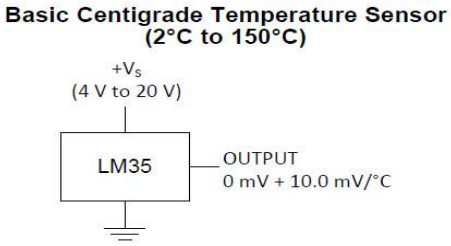
Το LM35 είναι ένα ολοκληρωμένο κύκλωμα, του οποίου η τάση εξόδου (Output Voltage) είναι γραμμικά ανάλογη με τη θερμοκρασία σε βαθμούς Κελσίου. Ο συντελεστής μετατροπής είναι 10.0 mV/°C (ή 0.01 V/°C). Για τη μετατροπή της τάσης εξόδου (Vout), σε βαθμούς Κελσίου oC, χρησιμοποιείται η σχέση:

$$\text{Temp (°C)} = [(\text{Vout σε V})]*100 \text{ ή}$$

$$\text{Temp (°C)} = [(\text{Vout σε mV})] / 10$$

Σχήμα 2.10: Αδιάβροχος (waterproof) αισθητήρας θερμοκρασίας σειράς LM35
(Πηγή: https://goo.gl/x1s34c)

Το κύκλωμα LM35 δεν απαιτεί εξωτερική βαθμονόμηση και μπορεί να μετρήσει θερμοκρασίες από -55 έως +150 βαθμούς Κελσίου. Εμφανίζει ακρίβεια ±0.25°C σε θερμοκρασία δωματίου και ±0.75°C στο θερμοκρασιακό εύρος από −55 έως +150 oC. Το ρεύμα λειτουργίας του είναι κάτω από 60 μA, προκαλώντας μικρή αυτοθέρμανση (0.08 °C σε νηνεμία). Η χαμηλή αντίσταση εξόδου (0.1 Ω για ρεύμα εξόδου 1 mA), η γραμμικότητα και η υψηλή ευαισθησία το καθιστούν κατάλληλο να συνεργάζεται με ηλεκτρονικά συστήματα συλλογής δεδομένων ή κυκλώματα ελέγχου. Αν ο αισθητήρας LM35 συνδεθεί όπως φαίνεται στο Σχήμα 2.11, η μικρότερη θερμοκρασία που μπορεί να μετρήσει είναι 2^oC. Στον Πίνακα 2.2 παρουσιάζονται οι συνιστώμενες συνθήκες λειτουργίας για διαφορετικούς αισθητήρες σειράς LM35 (LM35 Datasheet, https://goo.gl/rqh0sI).

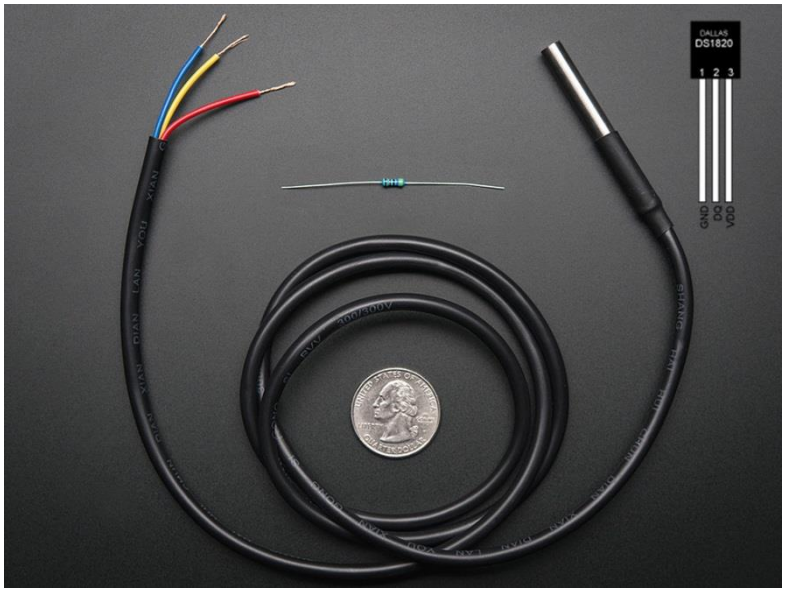


Σχήμα 2.11:Σύνδεση αισθητήρα LM35

Πίνακας 2.2: Συνιστώμενες συνθήκες λειτουργίας για αισθητήρες σειράς LM35

Συνιστώμενες συνθήκες λειτουργίας		**MIN**	**MAX**	
Θερμοκρασία λειτουργίας: T_{MIN} to T_{MAX}	LM35, LM35A	-55	150	°C
	LM35C, LM35CA	-40	110	
	LM35D	0	100	
Τάση Τροφοδοσίας ($+V_s$)		4	30	V

2.4.6 Ψηφιακός Αισθητήρας Θερμοκρασίας 1-wire DS18B20



Σχήμα 2.12: Αδιάβροχος αισθητήρας θερμοκρασίας 1-wire DS18B20
Πηγή: https://goo.gl/5sDPnH

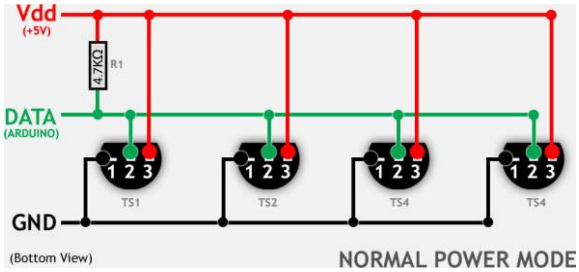
Ο ψηφιακός αισθητήρας θερμοκρασίας (Digital Thermometer) DS18B20 της εταιρίας Maxim Integrated Products, συνδυάζει ένα αισθητήρα θερμοκρασίας και έναν αναλογοψηφιακό μετατροπέα (Σχήμα 2.12). Η τάση λειτουργίας του κυμαίνεται από 3V έως 5,5V. Ο αισθητήρας μετράει θερμοκρασίες από -55 έως +125 βαθμούς Κελσίου με ακρίβεια ±2°C. Στην περιοχή από -10°C έως +85°C η ακρίβεια είναι ±0,5°C. Η προεπιλεγμένη ανάλυση του αναλογοψηφιακού μετατροπέα είναι 12 bit, αλλά μπορεί να διαμορφωθεί προγραμματιστικά από το χρήστη στα 9, 10, 11 ή 12 bit, που ισοδυναμεί με διακριτική ικανότητα (temperature resolution) 0,5°C, 0,25°C, 0,125°C ή 0,0625°C αντίστοιχα. Η ψηφιοποίηση της θερμοκρασίας διαρκεί 750 ms ή λιγότερο (Πίνακας 2.3).

Πίνακας 2.3: Μέγιστοι χρόνοι ψηφιοποίησης της θερμοκρασίας, ανάλογα την ανάλυση του ADC

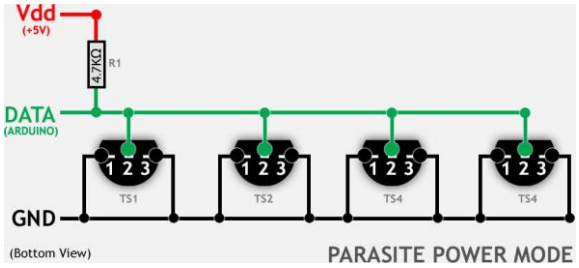
Ανάλυση ADC (resolution)	**Χρόνος ψηφιοποίησης θερμοκρασίας (MAX)**
9-bit	93,75 ms
10-bit	187,5 ms
11-bit	375 ms
12-bit	750 ms

Το ψηφιακό θερμόμετρο DS18B20 επικοινωνεί με τον κεντρικό μικροεπεξεργαστή (π.χ. Arduino) μέσω του διαύλου 1-Wire, ο οποίος (εξ' ορισμού) απαιτεί μόνο μια γραμμή δεδομένων (καθώς και τη γραμμή γείωσης). Η ενέργεια που απαιτείται για την ανάγνωση,

εγγραφή και εκτέλεση μετατροπών θερμοκρασίας μπορεί να προέρχεται από την ίδια τη γραμμή δεδομένων (Data Line) χωρίς την ανάγκη για εξωτερική τροφοδοσία ("παρασιτικός" τρόπος τροφοδοσίας). Ο δίαυλος 1-Wire, απαιτεί μια εξωτερική αντίσταση, περίπου 5kΩ (π.χ. 4,7 kΩ).



Σχήμα 2.13: Σύνδεση πολλαπλών DS18B20 στο δίαυλο 1-Wire (κανονικός τρόπος τροφοδοσίας) (Πηγή: https://goo.gl/Ykkqm5)



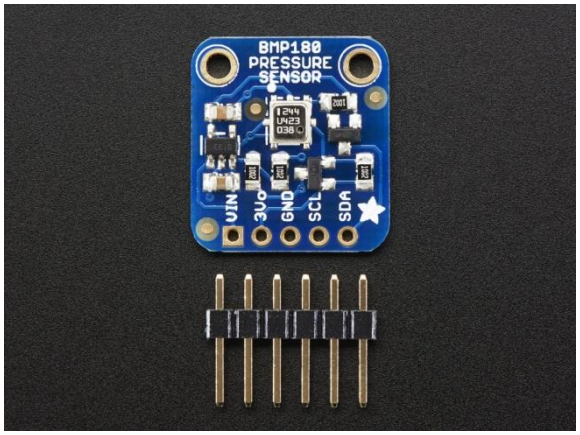
Σχήμα 2.14: Σύνδεση πολλαπλών DS18B20 στο δίαυλο 1-Wire ("παρασιτικός" τρόπος τροφοδοσίας) (Πηγή: https://goo.gl/Ykkqm5)

Επειδή κάθε DS18B20 περιέχει ένα μοναδικό κωδικό αριθμό 64 bits αποθηκευμένο σε ενσωματωμένη ROM, πολλαπλά DS18B20 (μέχρι και 127) μπορούν να συνδεθούν παράλληλα στον ίδιο 1-Wire δίαυλο είτε με κανονικό τρόπο τροφοδοσία (Σχήμα 2.13), είτε με «παρασιτικό» τρόπο τροφοδοσίας (Σχήμα 2.14). Αυτό επιτρέπει τον έλεγχο -μέσω ενός μικροεπεξεργαστή- πολλών DS18B20, τα οποία βρίσκονται κατανεμημένα σε μια μεγάλη περιοχή. Θεωρείται κατάλληλος για εφαρμογές σε συστήματα περιβαλλοντικού ελέγχου HVAC (θέρμανση (H), εξαερισμός (V), κλιματισμός (AC)) καθώς και σε ανιχνευτές θερμοκρασίας στο εσωτερικό κτιρίων (DS18B20 Datasheet, https://goo.gl/xngXUk).

2.4.7 Ψηφιακός I2C Αισθητήρας Βαρομετρικής πίεσης / θερμοκρασίας BMP180

Ο αισθητήρας BMP180 της Bosch είναι ένας ψηφιακός αισθητήρας υψηλής ακρίβειας και χαμηλού κόστους για τη μέτρηση ατμοσφαιρικής πίεσης και θερμοκρασίας (Σχήμα 2.15). Επειδή η πίεση αλλάζει με το υψόμετρο μπορεί επίσης να χρησιμοποιηθεί ως ένα αλτίμετρο για τη μέτρηση του υψομέτρου.

Αποτελείται από έναν αισθητήρα του οποίου η ηλεκτρική αντίσταση μεταβάλλεται σε σχέση με την ατμοσφαιρική πίεση που του εφαρμόζεται. Επίσης περιέχει έναν αισθητήρα θερμοκρασίας, έναν αναλογοψηφιακό μετατροπέα, έναν ελεγκτή με μνήμη EEPROM και μια σειριακή διασύνδεση I2C. Η διεπαφή I2C επιτρέπει την εύκολη διασύνδεση του συστήματος με ένα μικροελεγκτή. Οι τιμές θερμοκρασίας και πίεσης που μας παρέχει ο αισθητήρας, σταθμίζονται με χρήση των δεδομένων από την μνήμη EEPROM. Είναι κατάλληλος για χρήση σε κινητά τηλέφωνα, PDAs, συσκευές πλοήγησης, κ.α.



Σχήμα 2.15: Αισθητήρας BMP180 της adafruit
(Πηγή: https://goo.gl/NZ4RgZ)

- Τάση τροφοδοσίας (Vin): από 3 έως 5V DC (για τον αισθητήρα της adafruit)
- Εύρος λειτουργίας του αισθητήρα για πίεση: 300-1100 hPa (9000m έως -500m από την επιφάνεια της θάλλασας)
- Διακριτική ικανότητα έως 0.03hPa / 0.25m
- Εύρος λειτουργίας του αισθητήρα για θερμοκρασία:-40 έως +85°C, με ακρίβεια +-2°C
- Η διευθυνσιοδότηση του διαδρόμου I2C, γίνεται με αριθμούς των 7-bit (έως 128 αισθητήρες στον ίδιο διάδρομο)

Η προεπιλεγμένη ανάλυση του αναλογοψηφιακού μετατροπέα είναι 16 bit, αλλά ειδικά για την πίεση μπορεί να διαμορφωθεί προγραμματιστικά από το χρήστη μέχρι 19 bit, το οποίο αυξάνει τη διακριτική ικανότητα του αισθητήρα. Στον Πίνακα 2.4 αποτυπώνονται οι αντίστοιχοι χρόνοι ψηφιοποίησης (BMP180 datasheet, https://goo.gl/PHgbqR).

Πίνακας 2.4: Χρόνοι ψηφιοποίησης Πίεσης/Θερμοκρασίας

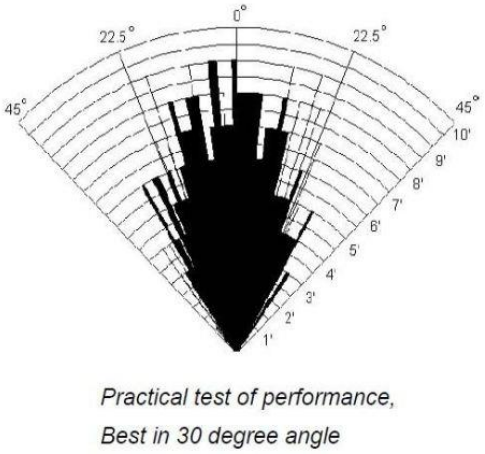
Χρόνοι ψηφιοποίησης	**Ανάλυση ADC**	**Τυπική τιμή**	**Μέγιστη τιμή**	**Μονάδα μέτρησης**
Χρόνος μετατροπής Πίεσης	ultra low power mode (16 bit)	3	4,5	ms
	standard mode (17 bit)	5	7,5	ms
	high resolution mode (18 bit)	9	13,5	ms
	ultra high res. Mode (19 bit)	17	25,5	ms
Χρόνος μετατροπής Θερμοκρασίας	standard mode (16 bit)	3	4,5	ms

2.4.8 Αισθητήρας Υπερήχων HC-SR04 για τον υπολογισμό απόστασης

Σχήμα 2.16: Αισθητήρας Υπερήχων HC-SR04 (Πηγή: https://goo.gl/h3gmIP)

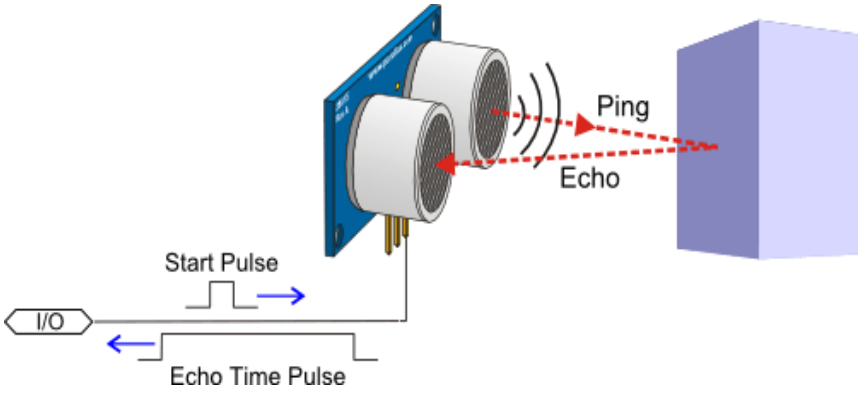
Ο αισθητήρας υπερήχων HC-SR04 χρησιμοποιείται για την ανίχνευση κοντινών αντικειμένων και την ακριβή μέτρηση της απόστασής τους από τον αισθητήρα, χρησιμοποιώντας έναν πομπό και έναν δέκτη υπερήχων (Σχήμα 2.16).

Το εύρος λειτουργίας του είναι από 2cm έως 400 cm, με διακριτική ικανότητα (resolution) 0,3 cm και η λειτουργία του δεν επηρεάζεται από το φως του ήλιου ή το χρώμα του υλικού. Λειτουργεί στα 5V DC και το ρεύμα λειτουργίας του είναι μικρότερο από 20mA. Η εφικτή γωνία μέτρησης για αξιόπιστα αποτελέσματα είναι μέχρι 15^{o}, με μέγιστη γωνία μέτρησης 30^{o} (Σχήμα 2.17).



Σχήμα 2.17: Γωνία μέτρησης αισθητήρα HC-SR04
(Πηγή: https://goo.gl/h3gmIP)

Ο υπολογισμός της απόστασης γίνεται με τη μέτρηση του χρόνου που μεσολαβεί από την εκπομπή ενός σύντομου παλμού υπερήχων (40 kHz) από τον πομπό, μέχρι την ανίχνευση του ήχου επιστροφής (echo) από τον δέκτη, όταν ανακλάται από ένα αντικείμενο στην ευθεία του αισθητήρα (Σχήμα 2.18), με δεδομένη και γνωστή την ταχύτητα του ήχου στον αέρα, που είναι περίπου 340 μέτρα το δευτερόλεπτο. Ο χρόνος μεταξύ δύο διαδοχικών μετρήσεων προτείνεται να είναι μεγαλύτερος από 60ms (HC-SR04 datasheet, https://goo.gl/ZTkIZs).



Σχήμα 2.18: Εκπομπή ενός σύντομου παλμού υπερήχων από τον πομπό και ανίχνευση του ήχου επιστροφής από τον δέκτη μετά την ανάκλαση, Πηγή: https://goo.gl/RuJ3eq

2.5 Γλώσσα Προγραμματισμού Python

Η γλώσσα προγραμματισμού Python (https://www.python.org/) δημιουργήθηκε από τον Guido van Rossum και κυκλοφόρησε για πρώτη φορά το 1991. Έχει εξελιχθεί σε μία ευρέως χρησιμοποιούμενη γλώσσα προγραμματισμού γενικού σκοπού. Είναι μία ισχυρά αντικειμενοστραφής γλώσσα υψηλού επιπέδου, που υποστηρίζει ικανοποιητικά και τις προγραμματιστικές δομές του συναρτησιακού και του διαδικαστικού προγραμματισμού.

Η Python είναι μια δυναμική γλώσσα προγραμματισμού. Χρησιμοποιεί διερμηνευτή εντολών (και όχι μεταγλωττιστή) για τη μετάφραση και εκτέλεση επί τόπου μίας εντολής κάθε φορά, παρέχοντας τη δυνατότητα πιο γρήγορου ελέγχου και άμεσης διόρθωσης των λαθών. Θεωρείται από τις ευκολότερες γλώσσες για αρχάριους προγραμματιστές καθώς η φιλοσοφία πίσω από την ανάπτυξη της είναι η δημιουργία ευέλικτου, ευανάγνωστου και σαφή κώδικα. Παρέχει δυνατότητες για γρήγορη προτυποποίηση και ανάπτυξη εφαρμογών. Διαθέτει πληθώρα έτοιμων βιβλιοθηκών που μπορούν να χρησιμοποιηθούν εύκολα και άμεσα. Συνοδεύεται από ένα εύχρηστο περιβάλλον ανάπτυξης που ονομάζεται IDLE (Interactive DeveLopment Environment), το οποίο ενσωματώνει δυνατότητες συγγραφής, επεξεργασίας, αποθήκευσης, εκτέλεσης και αποσφαλμάτωσης των προγραμμάτων.

Είναι ανοιχτού κώδικα (open source), διατίθεται δωρεάν από την επίσημη ιστοσελίδα και είναι ανεξάρτητη πλατφόρμας. Υποστηρίζεται από τον μη κερδοσκοπικό οργανισμό Python Software Foundation (https://www.python.org/psf/) που έχει ως αποστολή την ανάδειξη, προστασία και προώθηση της γλώσσας καθώς και την ανάπτυξη και υποστήριξη μιας διεθνούς κοινότητας προγραμματιστών Python. Η κοινότητα Python, εκτός από τη συνεχή βελτίωση της γλώσσας, είναι επίσης υπεύθυνη για την συνεχή ανάπτυξη ανοικτών πακέτων βιβλιοθηκών, το καλύτερο χαρακτηριστικό της Python, που την καθιστά από τις πιο επεκτάσιμες πλατφόρμες. Οι βιβλιοθήκες χρησιμοποιούνται για τη δημιουργία εφαρμογών που εκτείνονται από δυναμικές ιστοσελίδες σε πολύπλοκες εφαρμογές ανάλυσης δεδομένων, καθώς και την ανάπτυξη απλών GUI-based εφαρμογών για το σχεδιασμό γραφημάτων, όπως η εφαρμογή «libreMBL-GUI» που παρουσιάζεται στην παρούσα εργασία.

3 ΑΠΟΤΕΛΕΣΜΑΤΑ

Για τη λειτουργία της πειραματικής διάταξης που προτείνεται στην παρούσα εργασία, ακολουθήθηκαν τα εξής βήματα:

- Προγραμματισμός της πλακέτας Arduino για κάθε αισθητήρα ή συνδυασμό αισθητήρων (π.χ. δύο αισθητήρες θερμοκρασίας).
- Σύνδεση του αισθητήρα (ή αισθητήρων) στην πλακέτα.
- Ανάπτυξη του κώδικα της εφαρμογής «libreMBL-GUI».

3.1 Προγραμματισμός της πλακέτας Arduino

Για κάθε αισθητήρα ή συνδυασμό αισθητήρων που συνδέονται στην πλακέτα Arduino, είναι απαραίτητη η δημιουργία προγράμματος (sketch), το οποίο:

- διαβάζει περιοδικά τις μετρήσεις του αισθητήρα (ή συνδυασμό αισθητήρων) μέσω των αναλογικών ή ψηφιακών ακροδεκτών της πλακέτας στην οποία συνδέεται,
- επεξεργάζεται τις μετρήσεις,
- αποστέλλει τα αποτελέσματα της επεξεργασίας στον υπολογιστή, μέσω της σειριακής θύρας, για απεικόνιση σε κατανοητή μορφή από τον μαθητή.

Στο πλαίσιο της παρούσας εργασίας, δημιουργήθηκαν ενδεικτικά πέντε (5) προγράμματα (sketches) για τους αισθητήρες θερμοκρασίας, πίεσης και απόστασης που προτείνονται. Όλα τα προγράμματα ακολουθούν ένα κοινό μοτίβο, ώστε να είναι εύκολη η τροποποίηση/επέκταση του κώδικα για διαφορετικούς αισθητήρες ή συνδυασμό αισθητήρων. Για την ανάπτυξη των προγραμμάτων δόθηκε ιδιαίτερη έμφαση στα εξής:

- Οι αναλογικοί αισθητήρες έχουν συνήθως 3 ακροδέκτες: έναν που συνδέεται με την τροφοδοσία (5V ή 3,3V), έναν που συνδέεται με τη γείωση (GND), και έναν τρίτο που εξάγει μια μεταβλητή τάση και συνδέεται σε μία από τις έξι (6) αναλογικές εισόδους (A0, A1, A2, A3, A4,A5) της πλακέτας. Ο αναλογοψηφιακός μετατροπέας (ADC) διαβάζει την τάση που έρχεται από τον αισθητήρα και τη μετατρέπει σε ψηφιακή με ανάλυση 10 bit (1024 στάθμες). Η εντολή *AnalogRead(sensorPin)* της γλώσσας προγραμματισμού του Arduino, επιστρέφει μία τιμή στο πεδίο τιμών [0, 1023], όπου *sensorPin* είναι η αναλογική είσοδος στην οποίο συνδέεται ο ακροδέκτης εξόδου τάσης (Vout) του αισθητήρα.

```
int sensorVal = analogRead (sensorPin);
```

Στη συνέχεια γίνεται μετατροπή της τιμής εξόδου του ADC σε τάση, λαμβάνοντας υπόψη την τάση αναφοράς της πλακέτας.

```
float voltage = (sensorVal / 1023.0) * arefVoltage;
```

Η προεπιλεγμένη τάση αναφοράς έχει ρυθμιστεί στα 5V. Η εντολή *analogReference()*, ρυθμίζει την τάση αναφοράς εσωτερικά (INTERNAL) στα 1,1 V ή εξωτερικά (EXTERNAL) σε οποιαδήποτε άλλη τιμή (0-5V) εξωτερικής τάσης που θα συνδεθεί στον ακροδέκτη AREF, αυξάνοντας έτσι τη διακριτική ικανότητα του συστήματος αλλά μειώνοντας ταυτόχρονα το εύρος της μετρούμενης τάσης.

- Στους αναλογικούς αισθητήρες η διακριτική ικανότητα καθορίζεται από τον ADC της πλακέτας Arduino, ενώ στους ψηφιακούς από τον ενσωματωμένο ADC που διαθέτουν.
- Για τη συνεργασία του Arduino με ψηφιακούς αισθητήρες υπάρχει δυνατότητα αξιοποίησης κατάλληλων βιβλιοθηκών που συνήθως προτείνονται από τις κατασκευάστριες εταιρίες (π.χ. Adafruit, Sparkfun) και συνοδεύονται από ενδεικτικά παραδείγματα χρήσης της βιβλιοθήκης για την επικοινωνία του ψηφιακού αισθητήρα με την πλακέτα Arduino.
- Για την μείωση του τυχαίου θορύβου σε αργά μεταβαλλόμενο σήμα (π.χ. θερμοκρασία) μια απλή λύση αποτελεί η εξομάλυνση μέσου όρου, δηλαδή η λήψη ενός μεγάλου αριθμού μετρήσεων (συνήθως πολλαπλάσιο του 2) και υπολογισμός του μέσου όρου. (Νούσης & Νούση, 2013).
- Για την επίτευξη ενός σταθερού ρυθμού δειγματοληψίας έγινε χρήση διακοπών χρονιστή (timer Interrupts) μέσω της βιβλιοθήκης MsTimer2 (https://goo.gl/gqISc1), που επιτρέπει την περιοδική εκτέλεση μιας συνάρτησης ανά ρυθμιζόμενο αριθμό χιλιοστών του δευτερολέπτου (milliseconds).
- Για να επιλέγει ο χρήστης την έναρξη και λήξη της διαδικασίας των μετρήσεων, μέσω της εφαρμογής «libreMBL-GUI», αξιοποιήθηκε η εντολή *Serial.available(),* η οποία λαμβάνει τον αριθμό των bytes (χαρακτήρων) που διατίθενται για ανάγνωση από τη σειριακή θύρα. Με την εντολή

```
if (Serial.available()>0){
/* Έναρξη των μετρήσεων */
… }
```

οι ενσωματωμένες εντολές εκτελούνται εφόσον υπάρχουν δεδομένα που έχουν ληφθεί ή έχουν ήδη αποθηκευτεί στη μνήμη σειριακής λήψης (Serial Receive Buffer). Επίσης για το άδειασμα (flush) της Serial Receive Buffer, ώστε να μην υπάρχουν αποθηκευμένα δεδομένα, χρησιμοποιήθηκε η παρακάτω συνάρτηση:

```
void serialFlush(){
    while(Serial.available() > 0)
      Serial.read();
    }
```

- Η εντολή *Serial.print()* διοχετεύει τα δεδομένα για αποστολή μέσω της σειριακής θύρας ως ASCII κείμενο, όπου για τους ακέραιους αριθμούς (int) ή αριθμούς κινητής υποδιαστολής (float) χρησιμοποιείται ένας χαρακτήρας ASCII για κάθε ψηφίο. Η προεπιλεγμένη ακρίβεια των αριθμών κινητής υποδιαστολής είναι δύο δεκαδικά ψηφία. Επιπλέον, η εντολή *Serial.println()* διοχετεύει τα δεδομένα για αποστολή μέσω της σειριακής θύρας ως ASCII κείμενο, ακολουθούμενο από ένα χαρακτήρα carriage return (carriage return character, '\r') και έναν χαρακτήρα νέας γραμμής (newline character, '\n'). Για τη συνεργασία του προγράμματος (sketch) με την εφαρμογή «libreMBL-GUI» που τρέχει στον υπολογιστή, τα δεδομένα που διοχετεύονται για αποστολή μέσω της σειριακής θύρας, πρέπει να είναι χωρισμένα με κόμμα (,) ακολουθούμενα στο τέλος από αλλαγή γραμμής (\r\n). Αφορούν:
    - το ρυθμό δειγματοληψίας, δηλαδή το χρόνο (ms), που μέσω της βιβλιοθήκης MsTimer2, ρυθμίζει την περιοδική εκτέλεση της συνάρτησης υπολογισμού της μέτρησης (ή του ΜΟ των μετρήσεων) για κάθε φυσικό μέγεθος.
    - τη μέτρηση (ή τον μέσο όρο των μετρήσεων) για κάθε φυσικό μέγεθος.

```
Serial.print (χρόνος_ms);
Serial.print (", ");
Serial.print (μέτρηση_φυσικό_μέγεθος_1);
Serial.print (", ");
Serial.print (μέτρηση_φυσικό_μεγεθος_2);
Serial.print (", ");
Serial.println (μέτρηση_φυσικό_μεγεθος_3);
```

- Τα προγράμματα (sketches) για κάθε αισθητήρα (ή συνδυασμό αισθητήρων) που δημιουργήθηκαν ώστε να μεταφορτωθούν στην πλακέτα, ακολουθούν ένα κοινό μοτίβο κώδικα:

```
//Δήλωση βιβλιοθηκών
…
//Δήλωση σταθερών
#define SAMPLES αριθμός // αφορά τον αριθμό των μετρήσεων αργά μεταβαλλόμενων
//σημάτων όπως θερμοκρασία ή πίεση που θα πραγματοποιηθούν, και στη συνέχεια θα
// υπολογιστεί ο μέσος όρος τους
#define INTERVAL αριθμός // αφορά το χρόνο σε ms που ρυθμίζει την περιοδική
//εκτέλεση της συνάρτησης συνάρτησηΜετρήσεων() μέσω της βιβλιοθήκης MsTimer2
…
//Για ψηφιακούς αισθητήρες που χρησιμοποιούν συγκεκριμένες βιβλιοθήκες
//εδώ γίνεται η δήλωση των αντικειμένων
//Δήλωση αντικείμενων
…
//Δήλωση μεταβλητών
…

void συνάρτησηΜετρήσεων()//η συνάρτηση που καλείται περιοδικά μέσω της MsTimer2
{
  interrupts(); //ενεργοποίηση των διακοπών
   /* Έλεγχος αν υπάρχουν δεδομένα που έχουν ληφθεί ή έχουν ήδη αποθηκευτεί
    * στη Serial Receive Buffer της σειριακής θύρας. Με αυτόν τον τρόπο
    * η έναρξη των μετρήσεων ξεκινά όταν ο χρήστης επιλέξει την έναρξη
    * της διαδικασίας μέσα από τη γραφική διεπαφή που τρέχει στον υπολογιστή
    */
  if (Serial.available()>0){
   //άδειασμα της Serial Receive Buffer
   serialFlush();
   //ή
   // Λήψη αριθμού μετρήσεων ίσο με SAMPLES και υπολογισμός του Μ.Ο.
   for (int i=0; i < SAMPLES; i++)
   {
         /* Νέο συμβάν αισθητήρα */
         …
   }
   // υπολογισμός του Μέσου όρου
   …
   //ή
   /* Νέο συμβάν αισθητήρα */
         …

  /* Οι τιμές που στέλνονται στη σειριακή θύρα χωρισμένες με (,)αφορούν:
   * (α)το ρυθμό δειγματοληψίας, δηλαδή το χρόνο (ms) που ρυθμίζει την περιοδική
   * εκτέλεση της συνάρτησης υπολογισμού της μέτρησης (ή του ΜΟ των μετρήσεων)
   * για κάθε φυσικό μέγεθος.
   * (β) τη μέτρηση (ή τον μέσο όρο των μετρήσεων) για κάθε φυσικό μέγεθος.
   */
   Serial.print(INTERVAL);
   Serial.print(", ");
   Serial.print(μέτρηση_φυσικό_μέγεθος_1);
   Serial.print(", ");
   Serial.print(μέτρηση_φυσικό_μέγεθος_2);
   Serial.print(", ");
```

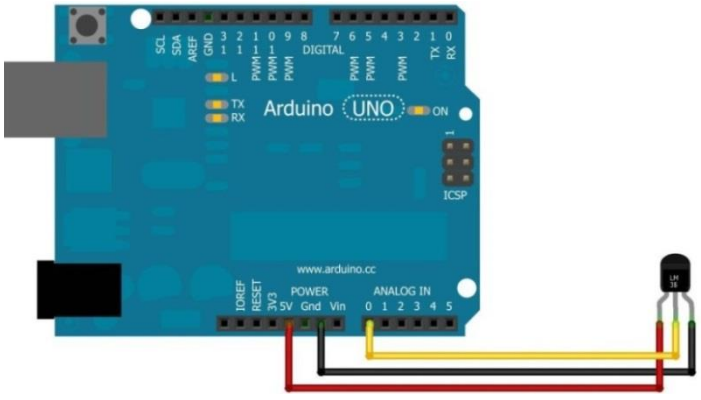
```
    Serial.println(μέτρηση_φυσικό_μέγεθος_3);
     } }

void setup() {
  Serial.begin(115200); // εκκίνηση της σειριακής επικοινωνίας με τον υπολογιστή με
                        //ρυθμό επικοινωνίας (baudrate)115200 bps
  //Αρχικοποίηση αντικειμένων
  …
  // ρύθμιση της συνάρτησης set της βιβλιοθήκης MsTimer2
  // περιοδική εκτέλεση της συνάρτησης συνάρτησηΜετρήσεων() ανά ρυθμιζόμενο αριθμό
  // χιλιοστών του δευτερολέπτου INTERVAL
  MsTimer2::set(INTERVAL, συνάρτησηΜετρήσεων);
  MsTimer2::start(); //εκκίνηση της διακοπής
}
  void loop() {
   }
//συνάρτηση που αδειάζει τη Serial Receive Buffer
  void serialFlush(){
    while(Serial.available() > 0)
      Serial.read();
    }
```

3.2 Σύνδεση αισθητήρων στην πλακέτα Arduino/Genuino Uno

3.2.1 Σύνδεση του αισθητήρα TMP36 με Arduino/Genuino Uno

Μία τυπική σύνδεση (Σχήμα 3.1) αφορά τη:

- Σύνδεση του ακροδέκτη τροφοδοσίας του αισθητήρα στον ακροδέκτη 5V του Arduino/Genuino Uno.
- Σύνδεση του ακροδέκτη γείωσης του αισθητήρα στον ακροδέκτη γείωσης του Arduino/Genuino Uno.
- Σύνδεση του ακροδέκτη εξόδου τάσης σε αναλογικό ακροδέκτη (π.χ. A0) του Arduino/Genuino Uno.



Σχήμα 3.1: Τυπική σύνδεση του αισθητήρα TMP36 με Arduino/Genuino Uno
Πηγή: https://goo.gl/o9vFTc

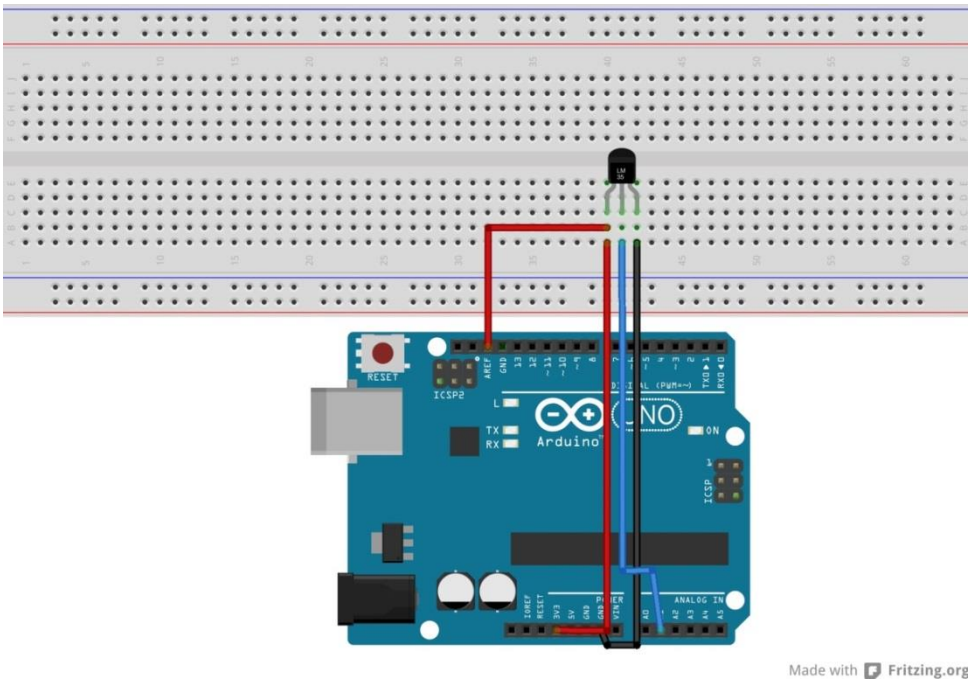
Για να αυξηθεί η διακριτική ικανότητα του συστήματος, επιλέχθηκε να γίνει αλλαγή της τάσης αναφοράς από 5V σε 3.3V, διότι μειώνοντας την τάση αναφοράς, μειώνεται το βήμα (step) του αναλογοψηφιακού μετατροπέα (ADC). Για παράδειγμα με τάση αναφοράς 5V (5000mV), η ελάχιστη διαφορά τάσης που μπορεί να ανιχνεύσει ο ADC, ο οποίος έχει ανάλυση 10 bit (1024 στάθμες), είναι 5000/1024 ≃ 4,9mV που αντιστοιχεί σε ελάχιστο βήμα θερμοκρασιακής διαφοράς περίπου 0,5 oC, εφόσον η τάση εξόδου του αισθητήρα είναι 10 mV / °C. Αντίστοιχα, με τάση αναφοράς 3.3 V (3300mV) είναι 3300/1024 ≃ 3,3 mV, που αντιστοιχεί σε ελάχιστο βήμα θερμοκρασιακής διαφοράς περίπου 0,33 oC.

Η τάση αναφοράς μπορεί να αλλάξει σε 3.3 V με χρήση της εντολής

analogReference(EXTERNAL);

και η τάση που εφαρμόζεται στον ακροδέκτη AREF (0 έως 5V μόνο) χρησιμοποιείται πλέον ως τάση αναφοράς. Ο ακροδέκτης 3.3 V του Arduino/Genuino Uno, μπορεί να παρέχει εξωτερική τάση 3.3 V στον ακροδέκτη AREF (Σχήμα 3.2).

- Σύνδεση του ακροδέκτη τροφοδοσίας του αισθητήρα στον ακροδέκτη 3.3V του Arduino/Genuino Uno.
- Σύνδεση του ακροδέκτη 3.3V του Arduino/Genuino Uno στον ακροδέκτη AREF του Arduino/Genuino Uno, ώστε να παρέχει εξωτερική τάση αναφοράς 3.3V.

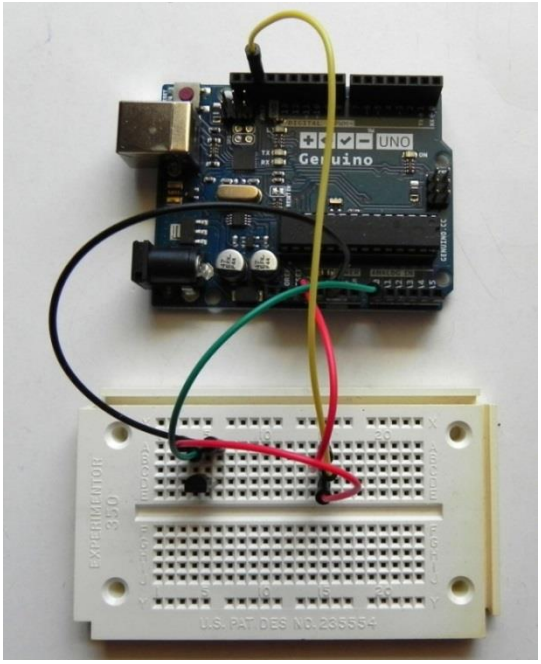


Σχήμα 3.2: Σύνδεση του αισθητήρα TMP36 στο Arduino/Genuino Uno με εξωτερική τάση αναφοράς 3.3V (Πηγή: https://goo.gl/EQVOMr)

- Σύνδεση του ακροδέκτη γείωσης του αισθητήρα στον ακροδέκτη γείωσης του Arduino/Genuino Uno.
- Σύνδεση του ακροδέκτη εξόδου τάσης σε αναλογικό ακροδέκτη (π.χ. A1) του Arduino/Genuino Uno.

Πρόγραμμα (sketch) για τον αισθητήρα θερμοκρασίας TMP36

Το πρόγραμμα (Παράρτημα Α), το οποίο μεταφορτώθηκε στον μικροελεγκτή της πλακέτας Genuino Uno (Σχήμα 3.3), υπολογίζει κάθε 1000 milliseconds, μέσω διακοπής χρονιστή (βιβλιοθήκη MsTimer2), το μέσο όρο 100 μετρήσεων θερμοκρασίας σε βαθμούς Κελσίου, που λαμβάνονται με καθυστέρηση 9500 microseconds. Οι τιμές που αποστέλλονται στη σειριακή θύρα χωρισμένες με κόμμα (,), αφορούν:



Σχήμα 3.3: Σύνδεση του αισθητήρα TMP36 σε Genuino Uno με εξωτερική τάση αναφοράς 3.3V

- το χρόνο (1000 ms) που ρυθμίζει την περιοδική εκτέλεση της συνάρτησης υπολογισμού των μετρήσεων και του ΜΟ. Η περίοδος (ms) και η συνάρτηση ορίζονται στη μέθοδο set της βιβλιοθήκης MsTimer2.
- το μέσο όρο των 100 μετρήσεων θερμοκρασίας σε βαθμούς Κελσίου.

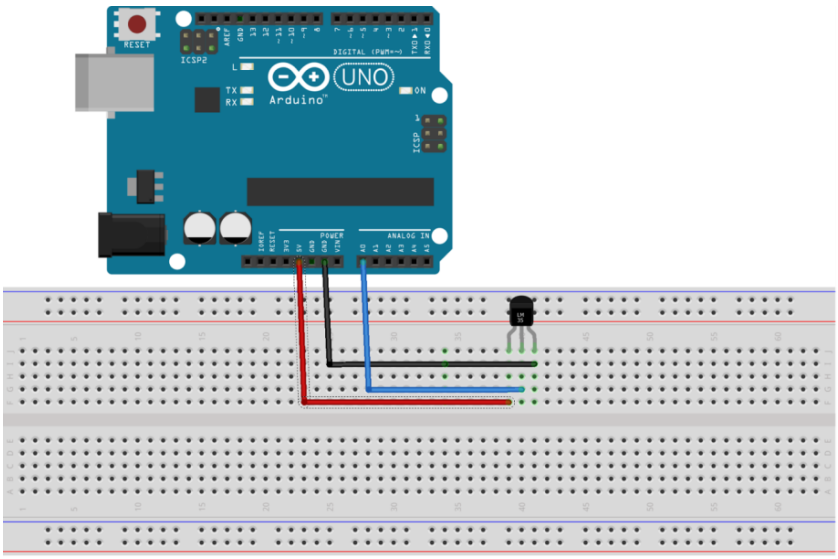
```
for (int i=0; i <= SAMPLES-1; i++) //συλλογή 100 μετρήσεων
{
  /* ο αναλογοψηφιακός μετατροπέας (ADC) διαβάζει
   * την τάση που έρχεται από τον αισθητήρα
   * και τη μετατρέπει σε ψηφιακή με ανάλυση 10 bit (1024 στάθμες)
   */
```

```
     int sensorVal = analogRead (sensorPin);
//μετατροπή της τιμής εξόδου του ADC σε τάση, που βασίζεται στην τάση
//αναφοράς
     float voltage = (sensorVal / 1023.0) * arefVoltage;

/* μετατροπή της τάσης σε θερμοκρασία (βαθμούς Κελσίου)
 * ο συντελεστής μετατροπής είναι 10 mV / βαθμό Κελσίου με 500 mV
 * μετατόπιση (offset) ώστε να μετρά και αρνητικές τιμές θερμοκρασίας
 */
     float temperature = (voltage - 0.5) * 100;
     samples_temp[i]=temperature;
     tempc += samples_temp[i];
     delayMicroseconds(9500); //καθυστέρηση 9500 μs
    }
  tempc /= SAMPLES; //υπολογισμός του ΜΟ για καλύτερη ακρίβεια
```

3.2.2 Σύνδεση του αισθητήρα σειράς LM35 με Arduino/Genuino Uno

Μία τυπική σύνδεση (Σχήμα 3.4) αφορά τη:

- Σύνδεση του ακροδέκτη τροφοδοσίας του αισθητήρα στον ακροδέκτη 5V του Arduino/Genuino Uno.
- Σύνδεση του ακροδέκτη γείωσης του αισθητήρα στον ακροδέκτη γείωσης του Arduino/Genuino Uno.
- Σύνδεση του ακροδέκτη εξόδου τάσης σε αναλογικό ακροδέκτη (π.χ. A0) του Arduino/Genuino Uno.

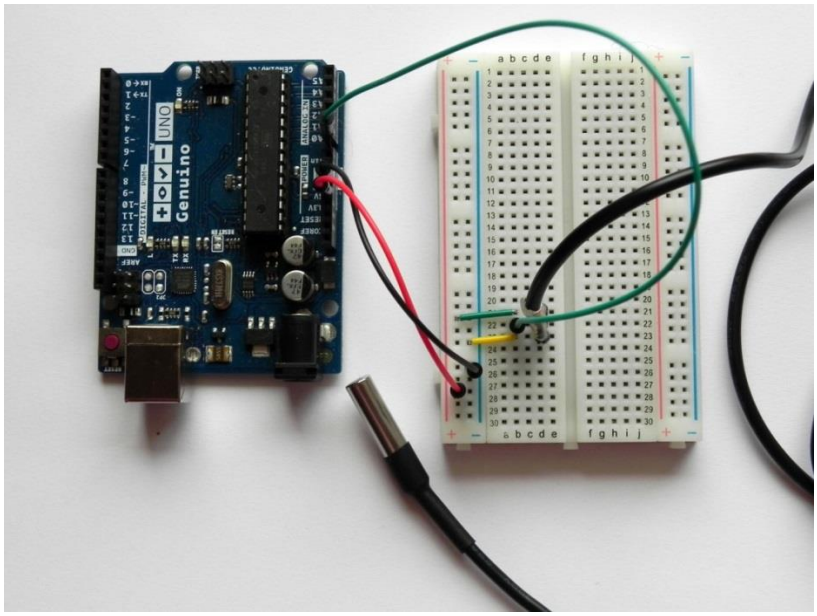


Σχήμα 3.4: Σύνδεση του αισθητήρα LM35 με Arduino Uno
Πηγή: https://goo.gl/cS321E

Η έξοδος του αναλογικού αισθητήρα θερμοκρασίας LM35 είναι 10 mV (0,01 V) ανά βαθμό Κελσίου και στη μέγιστη θερμοκρασία που μετράει, 150oC, θα δώσει μια έξοδο μόνο 1,5V. Επιλέχθηκε να χρησιμοποιηθεί η εντολή

analogReference(INTERNAL);

η οποία αλλάζει την ενσωματωμένη (build-in) τάση αναφοράς σε 1.1 V, που συμφωνεί καλύτερα με τη τάση εξόδου του αισθητήρα. Η διακριτική ικανότητα του αισθητήρα με τάση αναφορά ADC 5V είναι περίπου 0,5 βαθμούς Κελσίου ανά βήμα, ενώ με τάση αναφορά ADC 1.1 V είναι 0.107 βαθμούς Κελσίου ανά βήμα. Αυτό όμως έχει ως αποτέλεσμα τη μείωση του εύρους λειτουργίας του αισθητήρα που πλέον μπορεί να μετρήσει από +2°C έως +110°C.



Σχήμα 3.5: Σύνδεση του αδιάβροχου αισθητήρα LM35 σε Genuino Uno

Πρόγραμμα (sketch) για τον αισθητήρα θερμοκρασίας σειράς LM35

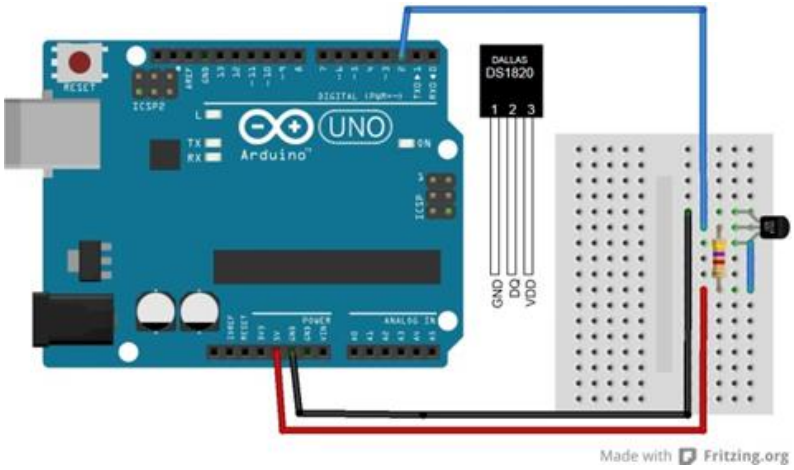
Το πρόγραμμα (Παράρτημα Α), το οποίο μεταφορτώθηκε στον μικροελεγκτή της πλακέτας Genuino Uno (Σχήμα 3.5), υπολογίζει κάθε 1000 milliseconds, μέσω διακοπής χρονιστή (βιβλιοθήκη MsTimer2), το μέσο όρο 100 μετρήσεων θερμοκρασίας σε βαθμούς Κελσίου, που λαμβάνονται με καθυστέρηση 9500 microseconds. Οι τιμές που αποστέλλονται στη σειριακή θύρα χωρισμένες με κόμμα (,), αφορούν:

- το χρόνο (1000 ms) που ρυθμίζει την περιοδική εκτέλεση της συνάρτησης υπολογισμού των μετρήσεων και του ΜΟ. Η περίοδος (ms) και η συνάρτηση ορίζονται στη μέθοδο set της βιβλιοθήκης MsTimer2.
- το μέσο όρο των 100 μετρήσεων.

3.2.3 Σύνδεση του ψηφιακού αισθητήρα 1-wire DS18B20 με Arduino/Genuino Uno

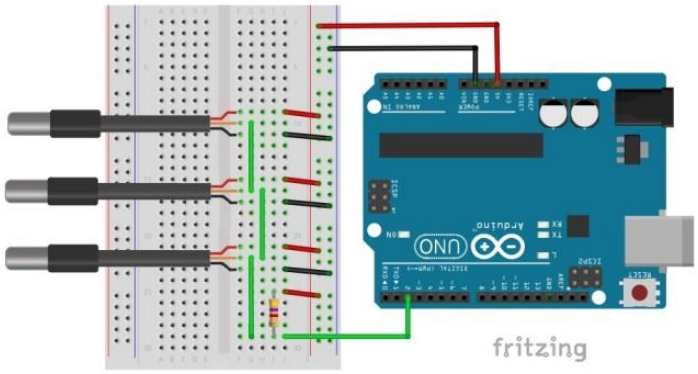
Μία τυπική σύνδεση (Σχήμα 3.6) αφορά τη:

- Σύνδεση του ακροδέκτη τροφοδοσίας του αισθητήρα στον ακροδέκτη 5V του Arduino/Genuino Uno.
- Σύνδεση του ακροδέκτη γείωσης του αισθητήρα στον ακροδέκτη γείωσης του Arduino/Genuino Uno.
- Σύνδεση της ψηφιακής εξόδου του αισθητήρα στην ψηφιακή είσοδο (π.χ. D2) του Arduino/Genuino Uno.
- Σύνδεση της ψηφιακής εξόδου του αισθητήρα μέσω αντίστασης 4,7KΩ στην τροφοδοσία (5V) του Arduino/Genuino Uno (pull-up resistor).



Σχήμα 3.6: Σύνδεση του ψηφιακού αισθητήρα 1-wire DS18B20 με Arduino/Genuino Uno
Πηγή: https://goo.gl/8tY4J6

Στο Σχήμα 3.7 απεικονίζεται ο τρόπος σύνδεσης πολλαπλών αισθητήρων DS18B20 παράλληλα στον ίδιο 1-Wire δίαυλο (με κανονικό τρόπο τροφοδοσίας).

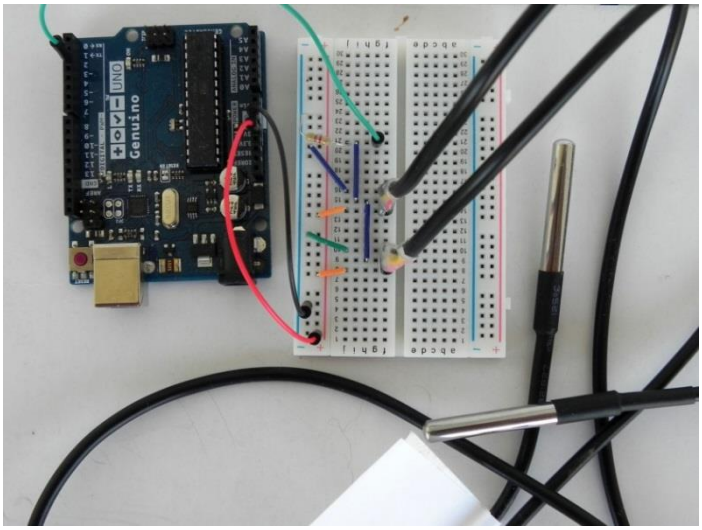


Σχήμα 3.7: Σύνδεση πολλαπλών 1-wire DS18B20 σε Arduino Uno
Πηγή: https://goo.gl/YPBa1z

Πρόγραμμα (sketch) για αισθητήρες θερμοκρασίας 1-wire DS18B20

Στο Genuino Uno, συνδέθηκαν δύο αδιάβροχοι αισθητήρες 1-wire DS18B20 (Σχήμα 3.8), ενώ για τη λειτουργία τους χρησιμοποιήθηκαν οι βιβλιοθήκες:

- **OneWire** (https://github.com/PaulStoffregen/OneWire), για την υλοποίηση του πρωτοκόλλου επικοινωνίας 1-Wire.
- **DallasTemperature** (https://github.com/milesburton/Arduino-Temperature-Control-Library)



Σχήμα 3.8: Σύνδεση δύο(2)αισθητήρων 1-wire DS18B20 σε Genuino Uno

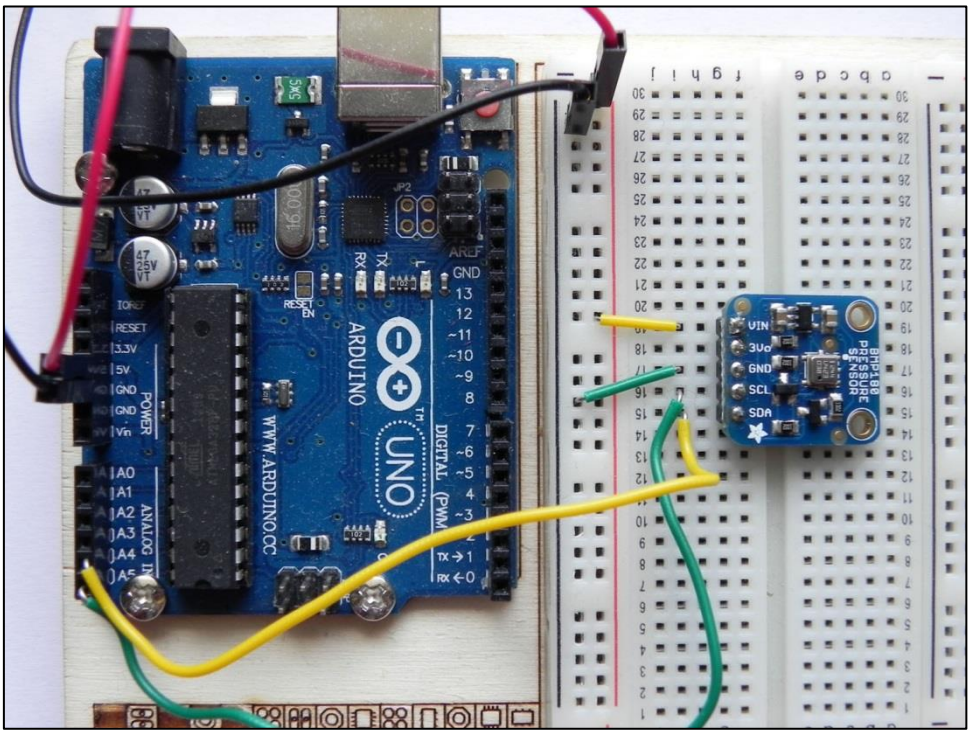
Εφόσον είναι γνωστός ο μοναδικός 64-bit κωδικός του κάθε αισθητήρα DS18B20 και γνωρίζοντας ότι η ψηφιοποίηση της θερμοκρασίας διαρκεί περίπου 750 ms (για ανάλυση ADC 12 bit), το πρόγραμμα (παράρτημα Α) λαμβάνει κάθε 1000 milliseconds, μέσω διακοπής χρονιστή (βιβλιοθήκη MsTimer2), μετρήσεις θερμοκρασίας σε βαθμούς Κελσίου από δύο αισθητήρες που είναι παράλληλα συνδεδεμένοι στον ίδιο δίαυλο 1-Wire. Οι τιμές που αποστέλλονται στη σειριακή θύρα χωρισμένες με κόμμα (,), αφορούν:

- το χρόνο (1000 ms) που ρυθμίζει την περιοδική εκτέλεση της συνάρτησης υπολογισμού της μέτρησης για κάθε αισθητήρα και έχει οριστεί στη μέθοδο set της βιβλιοθήκης MsTimer2.
- τη θερμοκρασία (oC) που ανίχνευσε ο πρώτος αισθητήρας (ακρίβεια δύο δεκαδικών ψηφίων).
- τη θερμοκρασία (oC) που ανίχνευσε ο δεύτερος αισθητήρας (ακρίβεια δύο δεκαδικών ψηφίων).

3.2.4 Σύνδεση του ψηφιακού I2C αισθητήρα BMP180 με Arduino/Genuino Uno

Για τη σύνδεση του ψηφιακού I2C αισθητήρα BMP180 με Arduino/Genuino Uno (Σχήμα 3.9):

- Ο ακροδέκτης τροφοδοσίας **Vin** του αισθητήρα συνδέεται στον ακροδέκτη των 5V ή 3.3V του Arduino/Genuino Uno.
- Ο ακροδέκτης γείωσης (**Gnd**) του αισθητήρα συνδέεται στον ακροδέκτη γείωσης του Arduino/Genuino Uno.
- Ο ακροδέκτης **I^2C SCL clock** του αισθητήρα συνδέεται στον αναλογικό ακροδέκτη A5 (SCL) του Arduino/Genuino Uno.
- Ο ακροδέκτης **I^2C SDA data** του αισθητήρα συνδέεται στον αναλογικό ακροδέκτη A4 (SDA) του Arduino/Genuino Uno.



Σχήμα 3.9: Σύνδεση του ψηφιακού I2C αισθητήρα BMP180 με Arduino/Genuino Uno

Πρόγραμμα (sketch) για τον ψηφιακό I2C αισθητήρα BMP180

Για τη λειτουργία του αισθητήρα BMP180 χρησιμοποιήθηκαν οι βιβλιοθήκες:

- **Wire Library**: Είναι ενσωματωμένη στο περιβάλλον ανάπτυξης (IDE) του Arduino και χρησιμοποιήθηκε για την υλοποίηση του πρωτοκόλλου I2C.
- **Adafruit_Sensor library** (https://github.com/adafruit/Adafruit_Sensor)

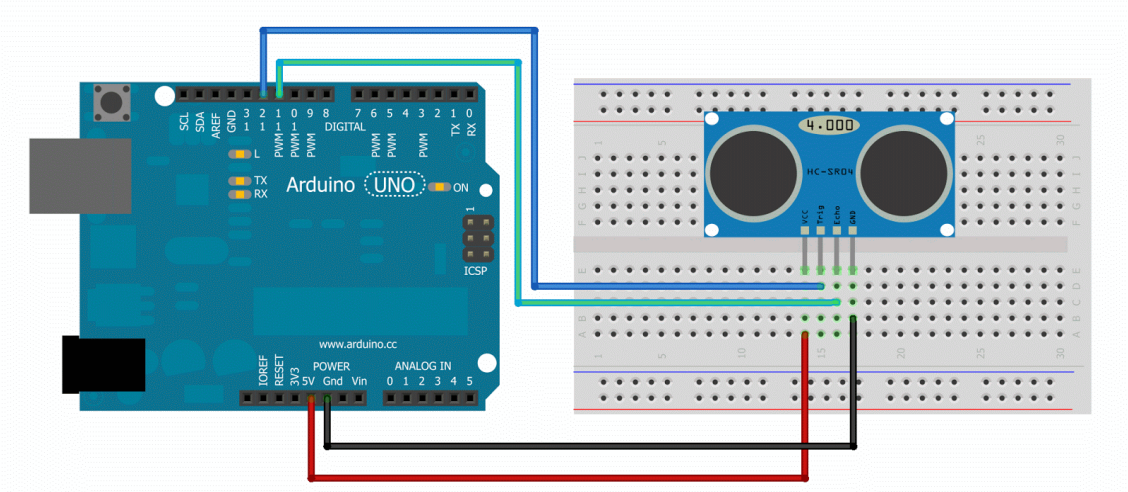
- **Adafruit Unified BMP085/BMP180 Driver**
  (https://github.com/adafruit/Adafruit_BMP085_Unified)

Το πρόγραμμα (παράρτημα Α) υπολογίζει κάθε 1000 milliseconds, μέσω διακοπής χρονιστή (βιβλιοθήκη MsTimer2), το μέσο όρο 32 μετρήσεων Πίεσης σε Εκτοπασκάλ (hPa), καθώς και θερμοκρασίας σε βαθμούς Κελσίου (oC). Για την ψηφιοποίηση της Πίεσης χρησιμοποιήθηκε το high resolution mode (Πίνακας 2.4).

Οι τιμές που αποστέλλονται στη σειριακή θύρα χωρισμένες με κόμμα (,), αφορούν:

- το χρόνο (1000 ms) που ρυθμίζει την περιοδική εκτέλεση της συνάρτησης υπολογισμού των μετρήσεων και του ΜΟ για κάθε φυσικό μέγεθος. Η περίοδος (ms) και η συνάρτηση ορίζονται στη μέθοδο set της βιβλιοθήκης MsTimer2.
- το μέσο όρο 32 μετρήσεων Πίεσης (hPa) (ακρίβεια ενός δεκαδικού ψηφίου).
- το μέσο όρο 32 μετρήσεων Θερμοκρασίας (oC) (ακρίβεια ενός δεκαδικού ψηφίου).

3.2.5 Σύνδεση του αισθητήρα Υπερήχων HC-SR04 με Arduino/Genuino Uno



Σχήμα 3.10: Σύνδεση του αισθητήρα Υπερήχων HC-SR04 με Arduino/Genuino Uno (Πηγή: https://goo.gl/liLMUU)

Για τη σύνδεση του αισθητήρα Υπερήχων HC-SR04 με Arduino/Genuino Uno (Σχήμα 3.10):

- Ο ακροδέκτης γείωσης **(Gnd)** του αισθητήρα συνδέεται στον ακροδέκτη γείωσης του Arduino.
- Ο ακροδέκτης **Trig** του αισθητήρα συνδέεται σε έναν ψηφιακό ακροδέκτη του Arduino (π.χ. 12).

- Ο ακροδέκτης **Echo** του αισθητήρα συνδέεται σε έναν ψηφιακό ή PWM ακροδέκτη του Arduino (π.χ. 11).
- Ο ακροδέκτης τροφοδοσίας **(Vcc)** του αισθητήρα συνδέεται στον ακροδέκτη 5V του Arduino.

Πρόγραμμα (sketch) για τον αισθητήρα Υπερήχων HC-SR04

Για τη λειτουργία του αισθητήρας Υπερήχων HC-SR04 με την πλακέτα Arduino χρησιμοποιήθηκε η βιβλιοθήκη **NewPing** (http://playground.arduino.cc/Code/NewPing).

Το πρόγραμμα (παράρτημα Α) λαμβάνει κάθε 200 milliseconds, μέσω διακοπής χρονιστή (βιβλιοθήκη MsTimer2), μετρήσεις της απόστασης (cm) από ένα αντικείμενο που κινείται και υπολογίζει τη στιγμιαία ταχύτητα σε m/sec. Οι τιμές που αποστέλλονται στη σειριακή θύρα χωρισμένες με κόμμα (,), αφορούν:

- το χρόνο (1000 ms) που ρυθμίζει την περιοδική εκτέλεση της συνάρτησης υπολογισμού των μετρήσεων για κάθε φυσικό μέγεθος. Η περίοδος (ms) και η συνάρτηση ορίζονται στη μέθοδο set της βιβλιοθήκης MsTimer2.
- την τιμή της απόστασης (cm)
- την τιμή της στιγμιαίας ταχύτητας (m/sec)

3.3 Περιβάλλον γραφικής διεπαφής

Η εφαρμογή «libreMBL-GUI» που αναπτύχθηκε για το σκοπό της παρούσας εργασίας, αποτελεί την τέταρτη και πιο σημαντική συνιστώσα του χαμηλού κόστους συστήματος Συγχρονικής Λήψης και Απεικόνισης και είναι:

- Ανεξάρτητη πλατφόρμας (εκτέλεση της εφαρμογής σε διαφορετικά λειτουργικά συστήματα όπως Linux και Windows).
- Ελεύθερο Λογισμικό / Λογισμικό Ανοικτού Κώδικα (ΕΛ/ΛΑΚ), ώστε να μπορεί ελεύθερα να χρησιμοποιηθεί, τροποποιηθεί και επεκταθεί από τους χρήστες της.
- Διαφανής (transparent) ως προς το υλικό, καθώς μπορεί να υποστηρίξει διαφορετικά είδη Arduino και

- δύναται να απεικονίσει τις μετρήσεις οποιουδήποτε συμβατού αισθητήρα που συνδέεται με την πλακέτα Arduino, με την προϋπόθεση να έχει δημιουργηθεί και μεταφορτωθεί στον μικροελεγκτή το κατάλληλο πρόγραμμα (sketch).

Το απλό και εύχρηστο περιβάλλον γραφικής διεπαφής (Graphical User Interface, GUI), επιτρέπει την αμφίδρομη επικοινωνία χρήστη-συστήματος για πρόσκτηση των μετρήσεων που λαμβάνονται από ένα ευρύ φάσμα αισθητήρων, την γραφική αναπαράστασή τους σε πραγματικό χρόνο και την αποθήκευση των δεδομένων για μεταγενέστερη επεξεργασία. Στις δυνατότητες της εφαρμογής «libreMBL-GUI» περιλαμβάνονται:

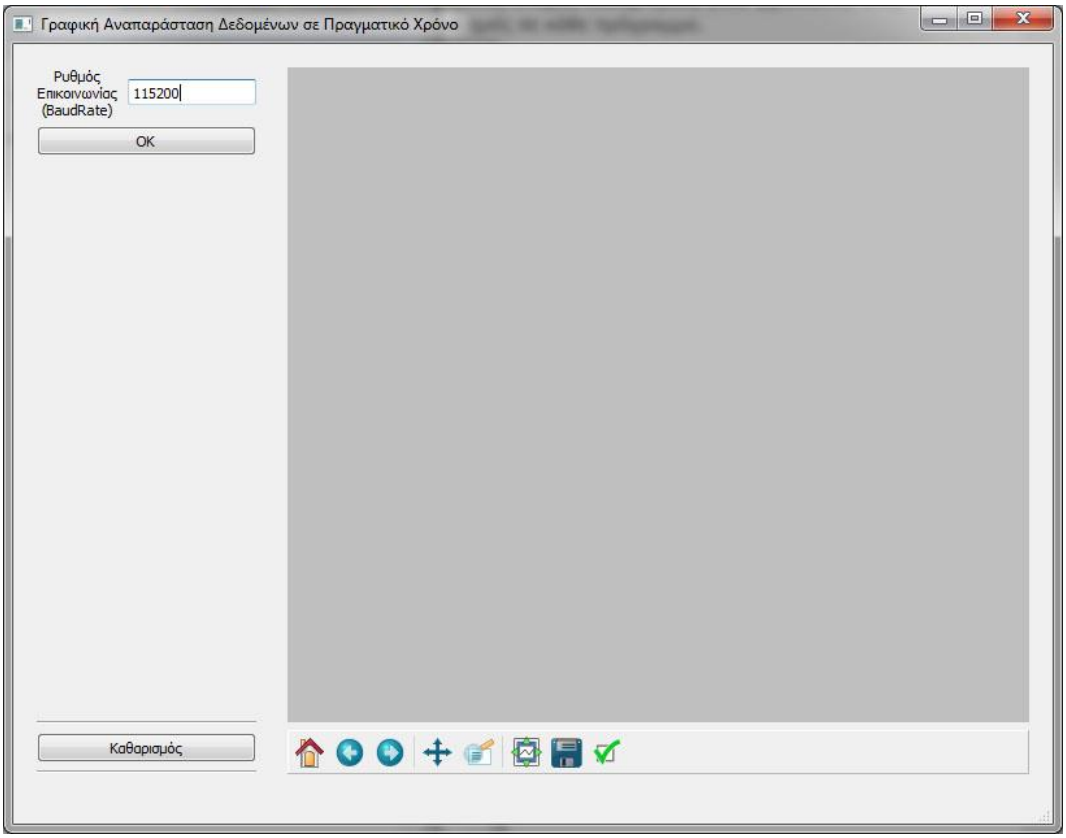
- ρύθμιση της σειριακής επικοινωνίας,
- έλεγχος έναρξης και λήξης της δειγματοληψίας,
- ευέλικτος τρόπος γραφικής απεικόνισης των δεδομένων,
- διαδραστική πλοήγηση του χρήστη στις γραφικές παραστάσεις,
- αποθήκευσή των δεδομένων σε αρχείο κειμένου,
- αποθήκευση στιγμιότυπου των γραφικών παραστάσεων σε αρχείο με επέκταση png, ps, eps, svg ή pdf.

Πριν τη διαδικασία έναρξης των μετρήσεων μέσω της εφαρμογής, πρέπει να έχουν προηγηθεί τα εξής:

- Σύνδεση του αισθητήρα (ή συνδυασμού αισθητήρων) στην πλακέτα Arduino Uno (όπως έχει αναλυθεί στην ενότητα 3.2).
- Δημιουργία του προγράμματος (sketch), για την περιοδική λήψη των μετρήσεων και αποστολή των δεδομένων στη σειριακή θύρα (όπως έχει αναλυθεί στην ενότητα 3.1). Τα δεδομένα είναι χωρισμένα με κόμμα (,), με πρώτο το ρυθμό δειγματοληψίας (ms) ακολουθούμενο από τη μέτρηση για κάθε φυσικό μέγεθος. Στο τέλος υπάρχει αλλαγή γραμμής (\r\n).
- Σύνδεση του Arduino στο υπολογιστή μέσω καλωδίου USB. Ο υπολογιστής αναγνωρίζει τη συσκευή ως σειριακή θύρα. Σε λειτουργικό σύστημα Windows είναι της μορφής COMX (π.χ. COM3, COM11), ενώ σε Linux εμφανίζεται ως /dev/ttyXXX.
- Μεταφόρτωση του προγράμματος (sketch) στην πλακέτα Arduino Uno.

Για να επιτευχθεί η σωστή επικοινωνία μεταξύ Arduino και υπολογιστή θα πρέπει αμφότεροι να έχουν τον ίδιο ρυθμό επικοινωνίας (BaudRate), τον οποίο καλείται ο χρήστης

(μαθητής ή εκπαιδευτικός) να γνωρίζει και πληκτρολογήσει στην εισαγωγική αλληλεπιδραστική οθόνη του περιβάλλοντος γραφικής διεπαφής. Ο προεπιλεγμένος ρυθμός επικοινωνίας είναι 115200 bps (Σχήμα 3.11).

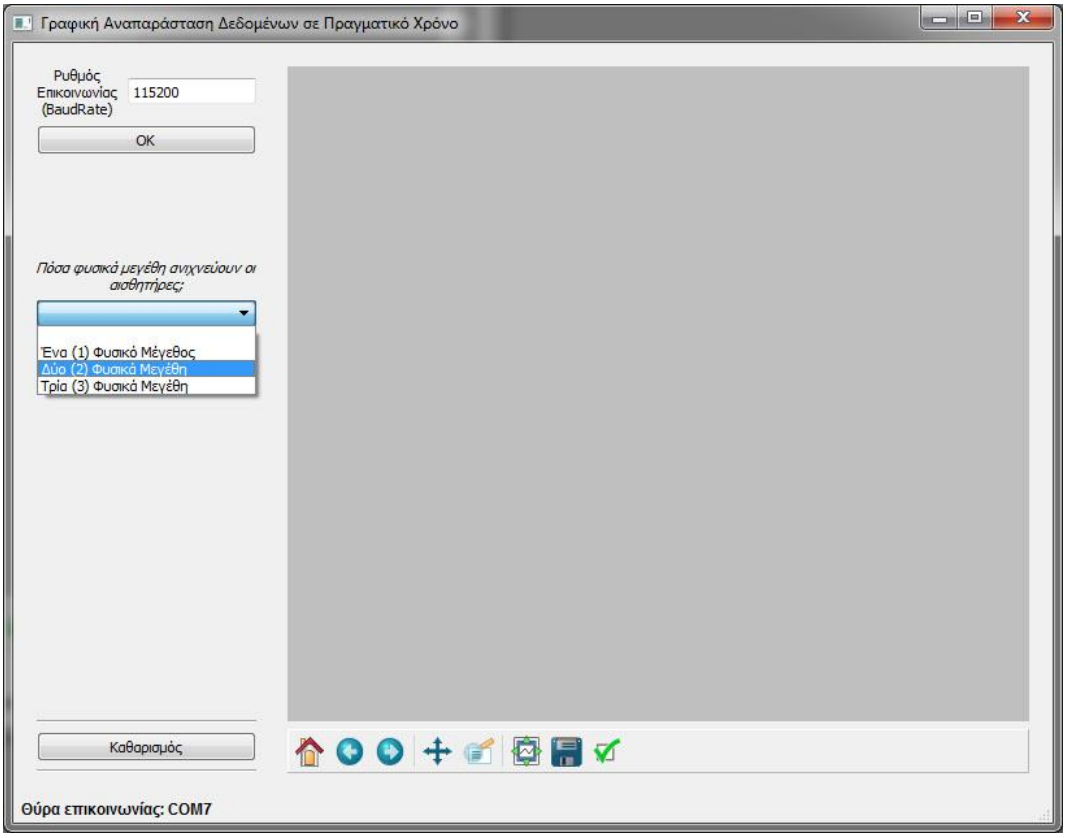


Σχήμα 3.11: Εισαγωγική αλληλεπιδραστική οθόνη της γραφικής διεπαφής

Με την επιβεβαίωση από το χρήστη (OK), το πρόγραμμα αναγνωρίζει τη σειριακή θύρα (COMX για Windows, /dev/ttyXXX για GNU/Linux), την εμφανίζει στη γραμμή κατάστασης (statusbar) και ρυθμίζει την επικοινωνία με το Arduino. Το πρόγραμμα δίνει τη δυνατότητα γραφικής απεικόνισης της μεταβολής ενός φυσικού μεγέθους σε συνάρτηση με το χρόνο (ξεκινώντας από τη χρονική στιγμή 0) και δύναται να εμφανίζει ταυτόχρονα τις γραφικές παραστάσεις μέχρι τριών (3) φυσικών μεγεθών (Σχήμα 3.12).

Σε αυτό το σημείο καλείται ο μαθητής να επιλέξει και πληκτρολογήσει (προαιρετικά), πόσα και ποια φυσικά μεγέθη ανιχνεύει ο αισθητήρας (ή οι αισθητήρες) που έχουν συνδεθεί στην πλακέτα Arduino καθώς και τις αντίστοιχες μονάδες μέτρησης. Ο εκπαιδευτικός, σύμφωνα με τις οδηγίες που διατίθενται για τους αισθητήρες της Πειραματικής διάταξης (Πίνακας 3.1), καθοδηγεί και υποστηρίζει τους μαθητές ώστε να εμπλακούν στη διαδικασία των μετρήσεων. Οι πληροφορίες που πληκτρολογεί ο μαθητής στα πλαίσια κειμένου, είτε σε

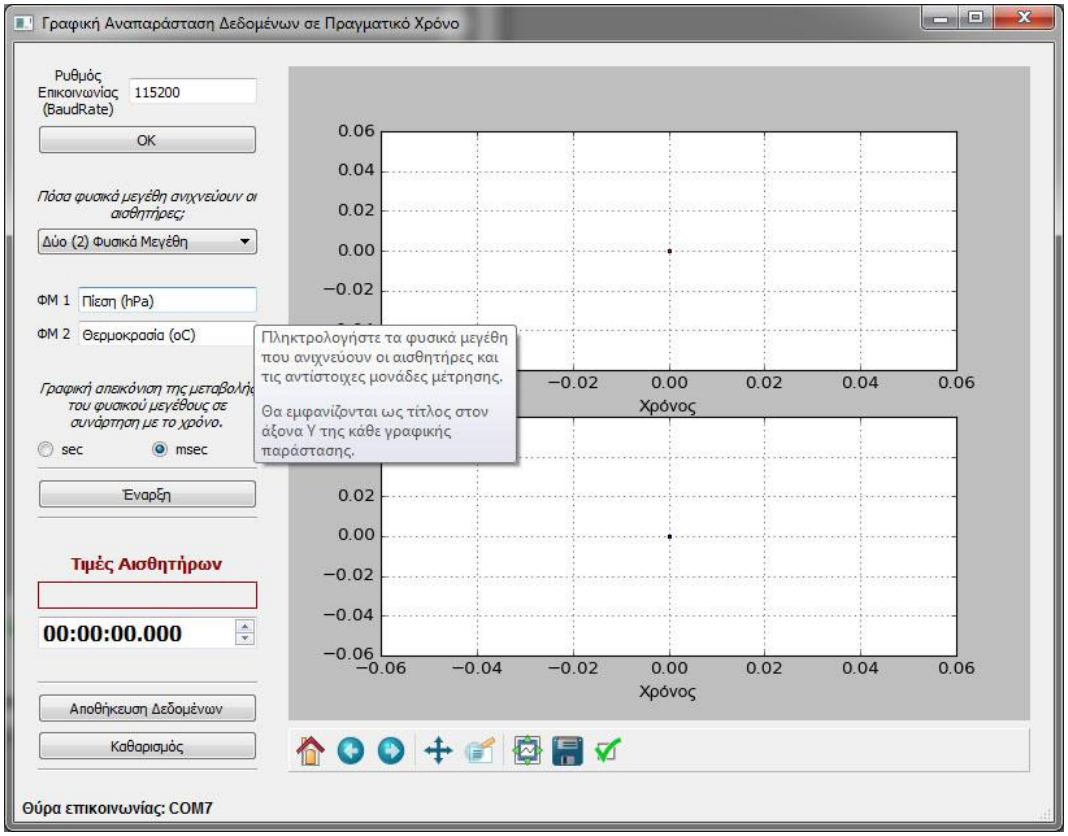
αυτό το σημείο ή κατά τη διάρκεια της γραφικής αναπαράστασης, θα εμφανίζονται ως τίτλος στον άξονα Υ της κάθε γραφικής παράστασης (Σχήμα 3.13).



Σχήμα 3.12 Το πρόγραμμα εμφανίζει ταυτόχρονα τις γραφικές παραστάσεις μέχρι τριών (3) φυσικών μεγεθών.

Πίνακας 3.1:Αισθητήρες της Πειραματικής διάταξης και τα Φυσικά Μεγέθη που ανιχνεύουν

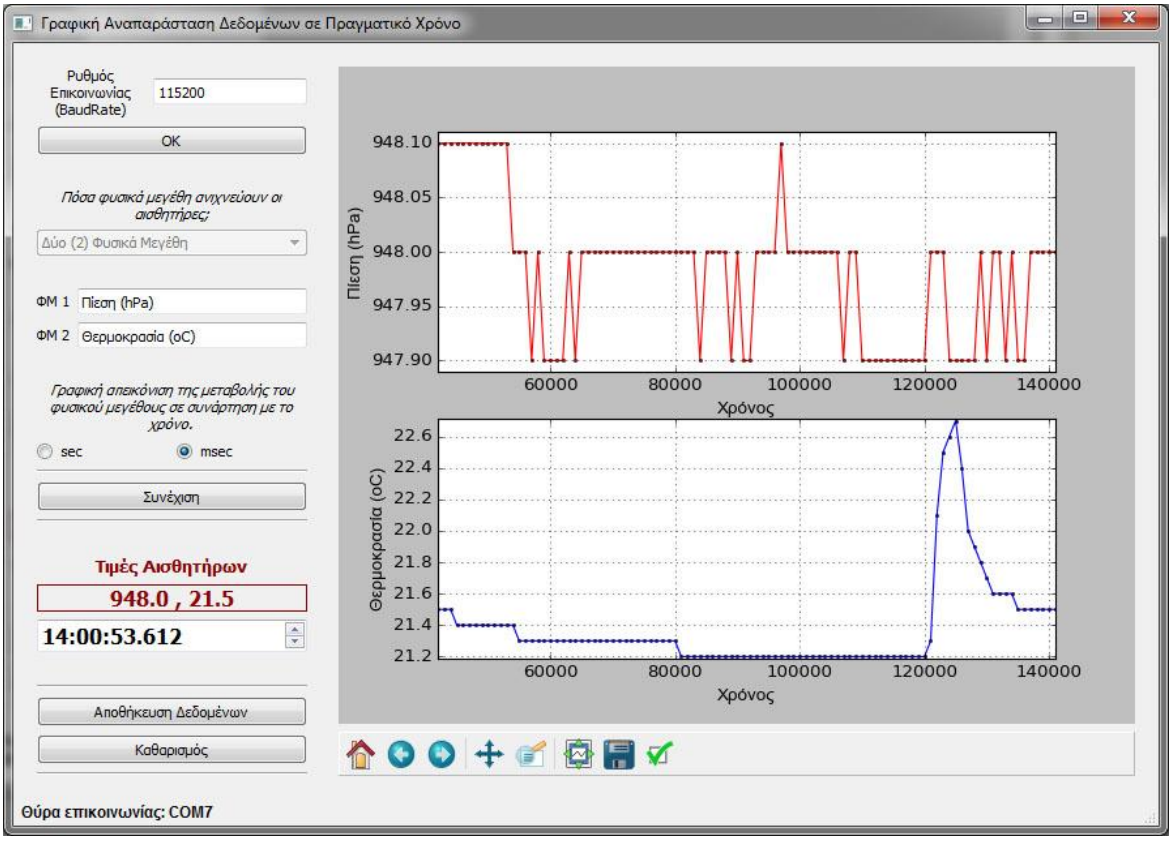
Πειραματική διάταξη - Αισθητήρες	**Φυσικά Μεγέθη**
Αναλογικός αισθητήρας θερμοκρασίας TMP36	ΦΜ 1: Θερμοκρασία (oC)
Αναλογικός αδιάβροχος (waterproof) αισθητήρας θερμοκρασίας σειράς LM35	ΦΜ 1: Θερμοκρασία (oC)
Δύο ψηφιακοί αδιάβροχοι αισθητήρες θερμοκρασίας 1-wire DS18B20	ΦΜ 1: Θερμοκρασία (oC) ($1^{ος}$ αισθητήρας) ΦΜ 2: Θερμοκρασία (oC) ($2^{ος}$ αισθητήρας)
Ψηφιακός I2C Αισθητήρας Βαρομετρικής πίεσης / θερμοκρασίας BMP180	ΦΜ 1: Πίεση (hPa) ΦΜ 2: Θερμοκρασία (oC)
Αισθητήρας Υπερήχων HC-SR04 για τον υπολογισμό απόστασης	ΦΜ 1: Απόσταση (cm) ΦΜ 2: Στιγμιαία ταχύτητα (m/sec)



Σχήμα 3.13: Σύνδεση του ψηφιακού αισθητήρα Πίεσης/Θερμοκρασίας BMP180 στην πλακέτα Arduino και προετοιμασία του προγράμματος γραφικής διεπαφής για λήψη και γραφική απεικόνιση των μετρήσεων Πίεσης (hPa) και Θερμοκρασίας (oC).

Ο χρήστης με το πάτημα του κουμπιού «Έναρξη» που μετονομάζεται άμεσα σε «Παύση», επιβεβαιώνει την έναρξη της διαδικασίας. Αυτόματα ενημερώνεται το sketch που τρέχει στον μικροελεγκτή, ώστε να προβεί στη διαδικασία περιοδικής μέτρησης της μεταβολής του φυσικού μεγέθους (ή των φυσικών μεγεθών) μέσω του αισθητήρα (ή αισθητήρων) και αποστολή των δεδομένων στη σειριακή θύρα. Το πρόγραμμα λαμβάνοντας τα δεδομένα από τη σειριακή θύρα, εμφανίζει τις μετρήσεις και σχεδιάζει σε πραγματικό χρόνο τη γραφική αναπαράστασή τους σε συνάρτηση με το χρόνο (second ή millisecond), ξεκινώντας από τη χρονική στιγμή μηδέν (0).

Επίσης για κάθε τιμή (ή τιμές αισθητήρων) που λαμβάνει, εμφανίζει τον πραγματικό χρόνο λήψης (current time) από το ρολόι του συστήματος (system clock). Η ακρίβεια εξαρτάται από την ακρίβεια του υποκείμενου λειτουργικού συστήματος, ενώ δεν παρέχουν όλα τα συστήματα ακρίβεια ενός χιλιοστού του δευτερολέπτου (1-millisecond accuracy).

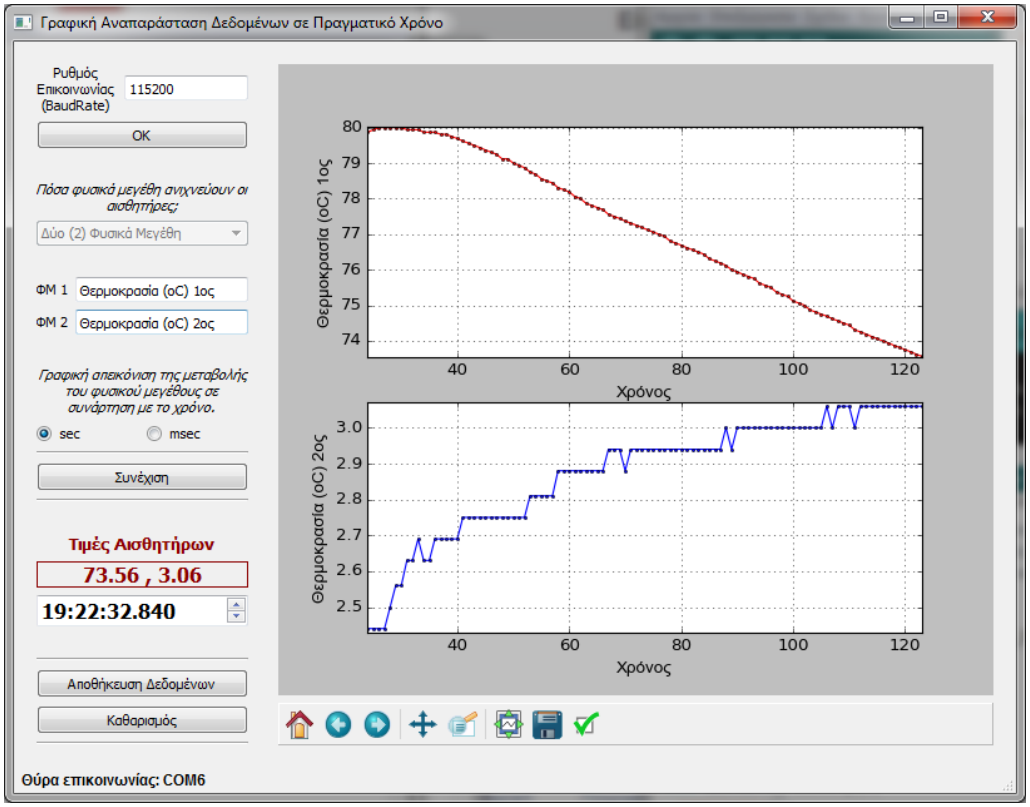


Σχήμα 3.14: Γραφική αναπαράσταση της Πίεσης και Θερμοκρασίας σε συνάρτηση με το χρόνο (msec). Το πρόγραμμα βρίσκεται σε κατάσταση Παύσης.

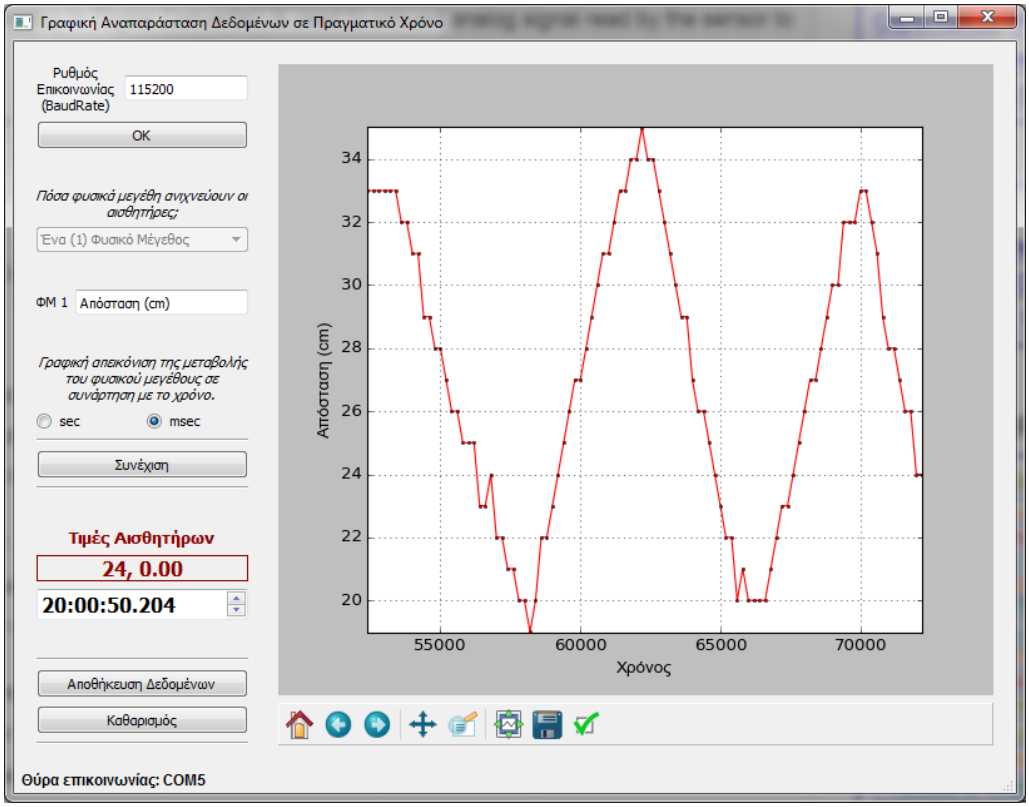
Η γραφική αναπαράσταση των δεδομένων ανανεώνεται συνεχώς, εμφανίζοντας τις τελευταίες εκατό (100) τιμές των αντίστοιχων μετρήσεων (Σχήμα 3.14, 3.15, 3.16). Με το πάτημα του κουμπιού «Παύση» που μετονομάζεται άμεσα σε «Συνέχιση», γίνεται προσωρινή παύση της διαδικασίας. Ο τερματισμός της εφαρμογής γίνεται με την επιλογή του κόκκινου κουμπιού κλεισίματος του παραθύρου.

Στην κατάσταση παύσης δίνονται στο χρήστη οι εξής δυνατότητες:

- Επιλογή συνέχισης της διαδικασίας και λήψη επιπλέον μετρήσεων (κουμπί «Συνέχιση»).
- Διαδραστική πλοήγηση στην περιοχή γραφήματος.
- Αποθήκευση των δεδομένων (κουμπί «Αποθήκευση δεδομένων»).
- Επανεκκίνηση της διαδικασίας (κουμπί «Καθαρισμός») για νέες μετρήσεις με τους ίδιους ή διαφορετικούς αισθητήρες.



Σχήμα 3.15:Μετρήσεις από δύο ψηφιακούς αδιάβροχους αισθητήρες θερμοκρασίας 1-wire DS18B20 που έχουν τοποθετηθεί σε βραστό και παγωμένο νερό αντίστοιχα.



Σχήμα 3.16: Γραφική αναπαράσταση της απόστασης, αντικειμένου που κινείται μπροστά από τον αισθητήρα Υπερήχων HC-SR04 σε συνάρτηση με το χρόνο (msec)

3.3.1 Διαδραστική πλοήγηση στη γραφική παράσταση

Η βιβλιοθήκη Matplotlib που χρησιμοποιήθηκε για την απεικόνιση των γραφικών παραστάσεων, παρέχει τη δυνατότητα ενσωμάτωσης εργαλείων (Interactive Navigation Toolbar) ή γεγονότων (events) για τη διαδραστική πλοήγηση του χρήστη –μετά την παύση της διαδικασίας- στα δεδομένα που απεικονίζονται γραφικά.

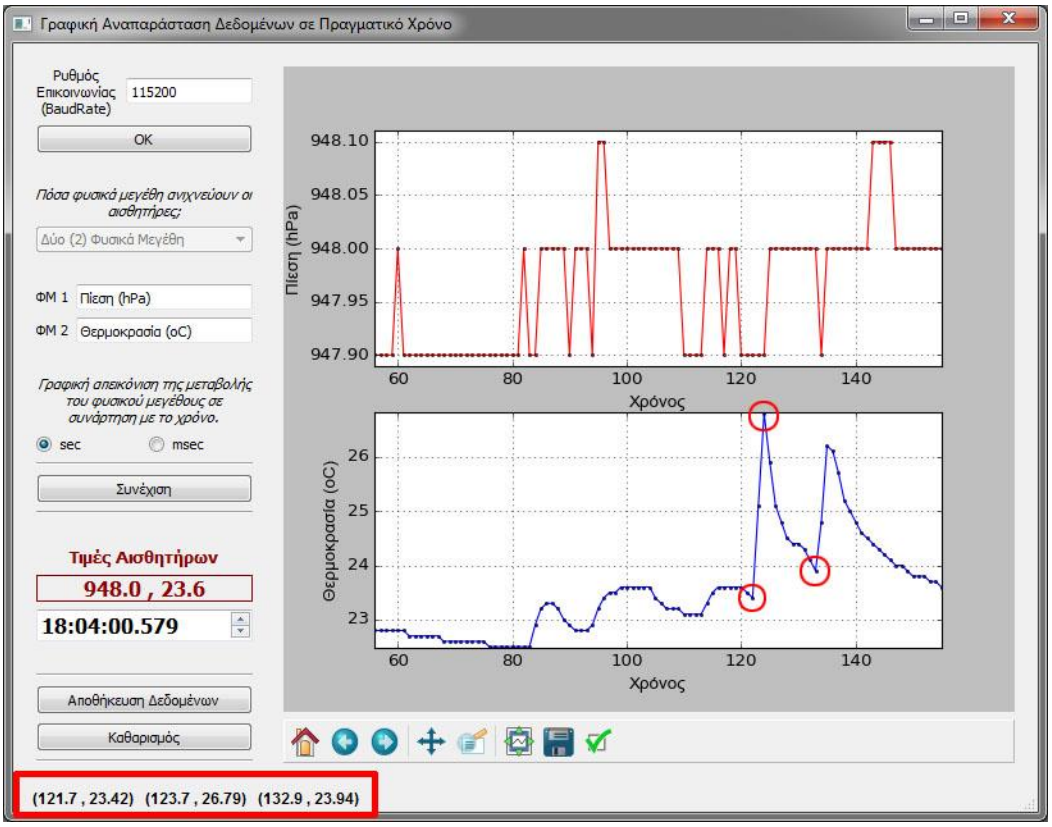
Στον Πίνακα 3.2 περιγράφονται τα κουμπιά της Εργαλειοθήκης Διαδραστικής Πλοήγησης (Interactive Navigation Toolbar).

Πίνακας 3.2: Περιγραφή της Εργαλειοθήκης Διαδραστικής Πλοήγησης

Κουμπί	**Περιγραφή**
	Το κουμπί **Pan/Zoom** έχει δύο λειτουργίες: Πανοραμική (pan) και Μεγέθυνσης (zoom). Ο χρήστης κρατώντας πατημένο τα δεξί πλήκτρο του ποντικιού, έχει τη δυνατότητα να κάνει μεγέθυνση (zoom in) ή σμίκρυνση (zoom out), γύρω από μια περιοχή σημείων που παρουσιάζουν ενδιαφέρον, ενώ με κρατημένο το αριστερό πλήκτρο του ποντικού να πλοηγείται πανοραμικά.
	Με το κουμπί **Zoom-to-rectangle**, ο χρήστης έχει τη δυνατότητα να μεγεθύνει/σμικρύνει μια περιοχή ενδιαφέροντος σχεδιάζοντας γύρω της ένα ορθογώνιο, κρατώντας πατημένο τα αριστερό ή το δεξί πλήκτρο του ποντικιού αντίστοιχα.
	Εφ’ όσον ο χρήστης χρησιμοποιώντας τα κουμπί Pan/Zoom και Zoom-to-rectangle έχει δημιουργήσει διάφορες προβολές, με τα κουμπιά **Forward** και **Back** έχει τη δυνατότητα να μετακινείτε εμπρός ή πίσω σε αυτές τις προβολές. Το κουμπί **Home** εμφανίζει την πρώτη προεπιλεγμένη προβολή των δεδομένων.
	Το κουμπί **Save** (Αποθήκευση), εμφανίζει πλαίσιο διαλόγου για αποθήκευση της τρέχουσας προβολής σε αρχείο με επεκτάσεις png, ps, eps, svg ή pdf.

Όταν τα κουμπιά Pan/Zoom και Zoom-to-rectangle της Εργαλειοθήκης Διαδραστικής Πλοήγησης δεν είναι επιλεγμένα, ο χρήστης δύναται να επιλέξει (με αριστερό κλικ) σημεία της γραφικής παράστασης στην περιοχή σχεδίασης. Οι συντεταγμένες των σημείων (χρόνος, φυσικό μέγεθος) εμφανίζονται στην γραμμή κατάστασης (status bar), ώστε ο μαθητής

συγκρίνοντας τις τιμές να εξάγει συμπεράσματα (Σχήμα 3.17). Για διαγραφή των δεδομένων από τη γραμμή κατάστασης, είναι αρκετό ένα κλικ οπουδήποτε στην περιοχή γραφήματος, εκτός της περιοχής σχεδίασης.

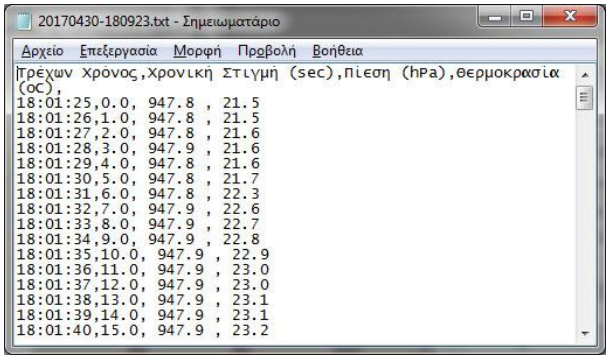


Σχήμα 3.17: Οι συντεταγμένες (χρόνος, θερμοκρασία) των σημείων που επέλεξε ο χρήστης, όπως εμφανίζονται στη γραμμή κατάστασης.

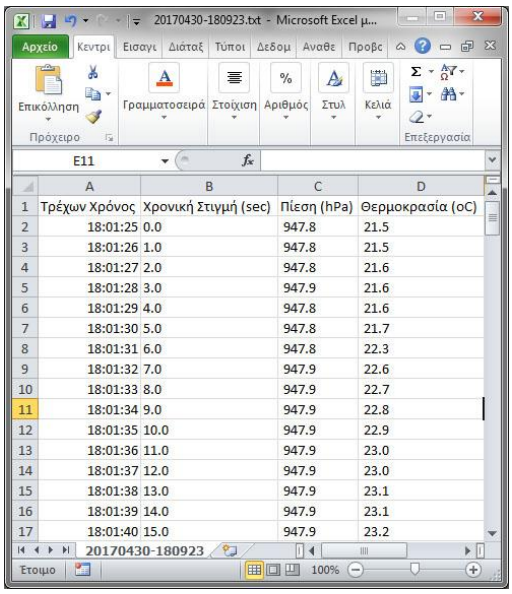
3.3.2 Αποθήκευση των δεδομένων

Η εφαρμογή δίνει τη δυνατότητα στο χρήστη να αποθηκεύσει τις τιμές εξέλιξης των φαινομένων σε συνάρτηση με το χρόνο, σε αρχείο κειμένου (.txt) στον υπολογιστή, για μεταγενέστερη επεξεργασία. Για κάθε χρονική στιγμή μέτρησης (ξεκινώντας από τη χρονική στιγμή 0) αποθηκεύονται σε νέα γραμμή (\n) χωρισμένα με κόμμα (,), ο τρέχων χρόνος από το ρολόι συστήματος (current time), η χρονική στιγμή (sec, msec) και οι μετρήσεις για κάθε φυσικό μέγεθος (Σχήμα 3.18, 3.19).

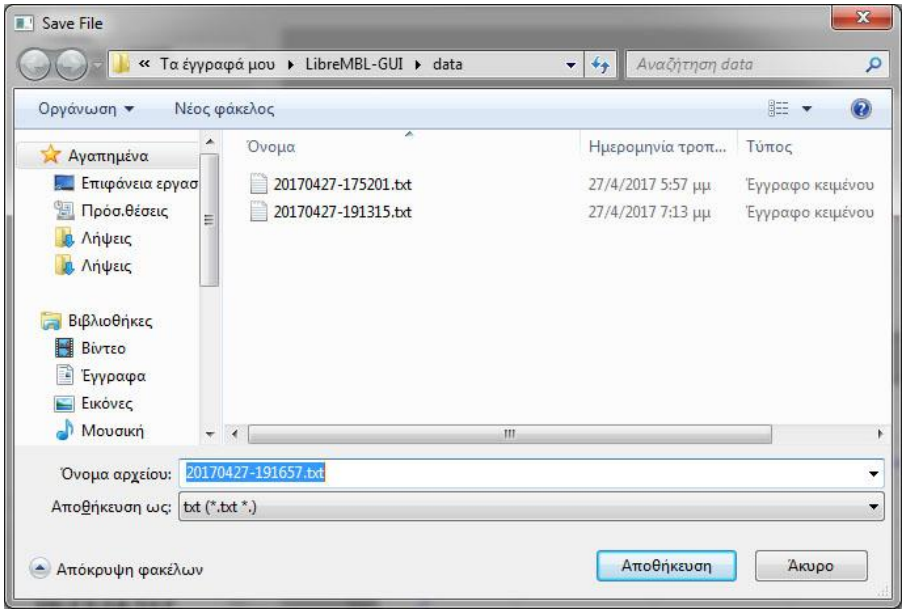
Στον φάκελο της εφαρμογής, δημιουργείται αυτόματα (αν δεν υπάρχει) φάκελος με το όνομα «data», ενώ εμφανίζεται παράθυρο διαλόγου για την αποθήκευση του αρχείου από το χρήστη. Το όνομα αρχείου που προτείνεται, προκύπτει από την τρέχουσα ημερομηνία και ώρα του συστήματος (timestamp) (Σχήμα 3.20).



Σχήμα 3.18: Τα δεδομένα των μετρήσεων όπως εμφανίζονται στο Σημειωματάριο των Windows



Σχήμα 3.19: Τα δεδομένα των μετρήσεων όπως εμφανίζονται στην εφαρμογή Microsoft Excel



Σχήμα 3.20: Το παράθυρο διαλόγου για αποθήκευση των δεδομένων.
Το προτεινόμενο όνομα αρχείου είναι η τρέχουσα ημερομηνία και ώρα (timestamp).

3.4 Βιβλιοθήκες (libraries) και αρθρώματα (modules)

Για την ανάπτυξη της εφαρμογής επιλέχθηκε η γλώσσα προγραμματισμού Python (έκδοση 2.7). Για την ανάπτυξη της γραφικής διεπαφής αξιοποιήθηκε η βιβλιοθήκη PyQt4 (έκδοση 12), για την απεικόνιση των γραφημάτων η βιβλιοθήκη Matplotlib (έκδοση 2.0.0), ενώ για την επικοινωνία του υπολογιστή με την πλατφόρμα Arduino το άρθρωμα (module) PySerial (έκδοση 3.3). Επίσης από την πρότυπη βιβλιοθήκη της Python ενσωματώθηκαν τα αρθρώματα sys, os, threading και time.

3.4.1 Το άρθρωμα sys

Το άρθρωμα sys (πρότυπη βιβλιοθήκη της Python) περιέχει συναρτήσεις και μεταβλητές για το χειρισμό διαφόρων τμημάτων του περιβάλλοντος λειτουργίας της Python. Παρέχει πρόσβαση σε ορισμένες μεταβλητές που χρησιμοποιούνται ή υποστηρίζονται από τον διερμηνευτή και σε συναρτήσεις που αλληλεπιδρούν με τον διερμηνευτή (https://docs.python.org/2/library/sys.html).

Η εισαγωγή του έγινε στην αρχή του προγράμματος με

import sys

Στην εφαρμογή χρησιμοποιήθηκαν:

- sys.argv: παρέχει πρόσβαση σε κάθε όρισμα που δίνεται από τη γραμμή εντολών.
- sys.getfilesystemencoding(): επιστρέφει το όνομα της κωδικοποίησης που χρησιμοποιείται για την μετατροπή των Unicode ονομάτων αρχείων. Το αποτέλεσμα εξαρτάται από το λειτουργικό σύστημα.

3.4.2 Τα αρθρώματα os και os.path

Το άρθρωμα os (πρότυπη βιβλιοθήκη της Python) (https://docs.python.org/2/library/os.html) παρέχει ένα φορητό (portable) τρόπο για αλληλεπίδραση με πολλές συναρτήσεις, που εξαρτώνται από το λειτουργικό σύστημα, για χειρισμό διαδικασιών, αρχείων, καταλόγων και άλλων χαρακτηριστικών "χαμηλού επιπέδου" του λειτουργικού συστήματος, ενώ το άρθρωμα os.path παρέχει χρήσιμες συναρτήσεις για χειρισμό διαδρομών (pathnames). Τα προγράμματα που εισάγουν και χρησιμοποιούν το άρθρωμα os έχουν μεγαλύτερες πιθανότητες να είναι μεταφέρσιμα σε διαφορετικές πλατφόρμες. Η εισαγωγή του αρθρώματος os έγινε στην αρχή του προγράμματος με

import os

Στην εφαρμογή χρησιμοποιήθηκαν οι Συναρτήσεις:

- os.makedirs(*path*): αναδρομική συνάρτηση δημιουργίας καταλόγου.
- os.sep: ο χαρακτήρας που χρησιμοποιείται από το λειτουργικό σύστημα για το διαχωρισμό των στοιχείων μιας διαδρομής (pathname separator : '/' ή ':' ή '\').
- os.path.dirname(*path*): Επιστρέφει το όνομα καταλόγου της διαδρομής *path.*
- os.path.exists(*path*): Επιστρέφει True αν το *path* αναφέρεται σε υπάρχουσα διαδρομή.
- os.path.join (*path, *paths*): Ενώνει ένα ή περισσότερα στοιχεία διαδρομής με έξυπνο τρόπο. Η τιμή επιστροφής είναι η συσχέτιση του *path* και όλων των μελών του **path* με ακριβώς έναν διαχωριστή καταλόγου (os.sep) μετά από κάθε μη κενό μέρος εκτός από το τελευταίο.
- os.path.realpath(*path*): Επιστρέφει την κανονική διαδρομή του καθορισμένου αρχείου.

3.4.3 Το άρθρωμα threading

Το άρθρωμα threading (https://docs.python.org/2/library/threading.html) παρέχει λειτουργικότητα πολυνημάτωσης υψηλότερου επιπέδου και είναι χτισμένο πάνω από το άρθρωμα χαμηλότερου επιπέδου thread. Η εισαγωγή του αρθρώματος threading έγινε στην αρχή του προγράμματος με:

import threading

Για τη δημιουργία ενός νήματος (thread), καλείται ο κατασκευαστής της έτοιμης κλάσης *Thread*.

comt=**threading.Thread**(*target* = συνάρτηση)

όπου *target* είναι συνήθως η συνάρτηση που θα κληθεί για να εκτελεστεί από το νήμα. Αν έχει την τιμή None δεν θα κληθεί τίποτα.

- start(): προκαλεί την εκκίνηση της δραστηριότητας του νήματος. Πρέπει να κληθεί το πολύ μια φορά για κάθε νήμα
- join*([timeout]*): περιμένει το νήμα να τελειώσει την εκτέλεσή του κώδικά του. Μπλοκάρει το νήμα που την καλεί μέχρι το σημείο που το νήμα για το οποίο καλείται τερματίσει ή έως ότου επέλθει το χρονικό όριο (timeout).

3.4.4 Το άρθρωμα PySerial

Το άρθρωμα PySerial (http://pythonhosted.org/pyserial/), επιτρέπει την επικοινωνία του υπολογιστή με την πλατφόρμα Arduino, καθώς υποστηρίζει τη σειριακή σύνδεση. Παρέχει πρόσβαση στις ρυθμίσεις της σειριακής θύρας μέσω των ιδιοτήτων της Python και επιτρέπει την άμεση ρύθμιση της σειριακής θύρας μέσω του διερμηνευτή εντολών. Η PySerial είναι η γέφυρα για οποιαδήποτε επικοινωνία μεταξύ της Python και του Arduino.

Η εισαγωγή του αρθρώματος PySerial ώστε να χρησιμοποιηθεί από την Python έγινε στην αρχή του προγράμματος με:

import serial

Για το άνοιγμα της σειριακής θύρας και την έναρξη της επικοινωνίας καλείται ο κατασκευαστής της έτοιμης κλάσης *Serial.*

ser=**serial.Serial** (*port, baudrate, timeout=x*) #ανοίγει τη σειριακή θύρα

port: το όνομα της συσκευής ανάλογα το λειτουργικό σύστημα (π.χ. /Dev/ttyUSB0 σε GNU/Linux ή COM3 στα Windows).

baudrate: ο ρυθμός επικοινωνίας

Σε όλες τις πλατφόρμες υποστηρίζονται οι τυπικές τιμές όπως 9600, 19200, 38400, 57600, 115200. Επίσης υποστηρίζονται από αρκετές πλατφόρμες και συσκευές, τυπικές τιμές ρυθμού επικοινωνίας πάνω από 115200, όπως 230400, 460800, 500000, 576000, 921600, 1000000, 1152000, 1500000, 2000000, 2500000, 3000000, 3500000, 4000000.

timeout=x, όπου x δευτερόλεπτα*:* ρυθμίζει τη συμπεριφορά της συνάρτησης read(). Όταν ο απαιτούμενος αριθμός bytes δεν είναι διαθέσιμος, περιμένει μέχρι να λήξει το χρονικό όριο και επιστρέφει όλα τα bytes που ελήφθησαν μέχρι τότε.

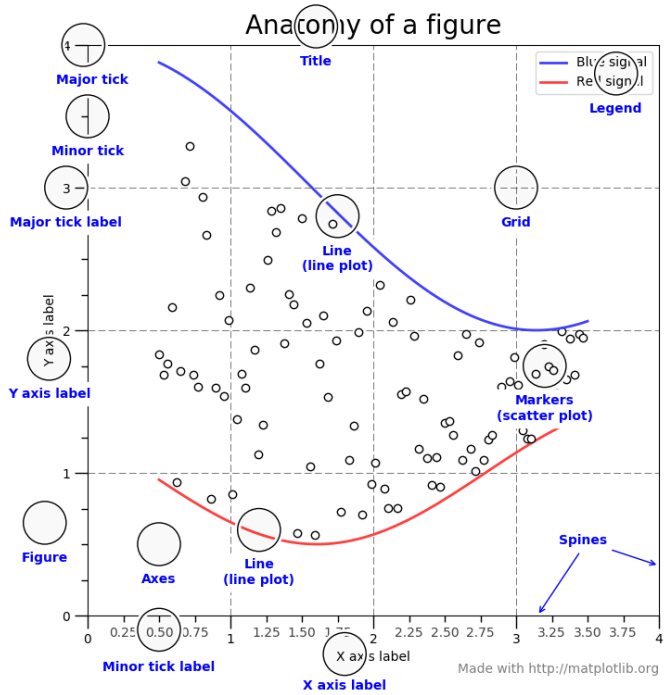
Στην εφαρμογή χρησιμοποιήθηκαν οι μέθοδοι της κλάσης serial.Serial:

- in_waiting: επιστρέφει τον αριθμό των bytes που βρίσκονται στη buffer λήψης
- read(*size=αριθμός*): διαβάζει *size* bytes από τη σειριακή θύρα. Εάν έχει οριστεί χρονικό όριο (timeout), ενδέχεται να επιστρέφει λιγότερους χαρακτήρες.
- write(*data*): γράφει τα δεδομένα (*data)* στη σειριακή θύρα.
- reset_input_buffer(): κάνει εκκαθάριση (flush) της buffer εισόδου, απορρίπτοντας όλο το περιεχόμενό της.

- reset_output_buffer(): κάνει εκκαθάριση (flush) της buffer εξόδου, αναστέλλοντας την τρέχουσα έξοδο και απορρίπτοντας όλο το περιεχόμενο της buffer.
- close(): κλείνει άμεσα τη σειριακή θύρα.

3.4.5 Η βιβλιοθήκη Matplotlib

Η βιβλιοθήκη matplotlib της Python (http://matplotlib.org/), είναι μια από τις πιο δημοφιλείς, ευέλικτες και εύχρηστες βιβλιοθήκες για απεικόνιση γραφημάτων αλλά και εικόνων. Αν και έχει εμπνευστεί από τον τομέα γραφικών του MATLAB, είναι ανεξάρτητη του MATLAB και έχει σχεδιαστεί με τη φιλοσοφία να δίνει τη δυνατότητα δημιουργίας απλών (κυρίως 2D) γραφικών παραστάσεων με λίγες γραμμές κώδικα και με εύκολες στη χρήση ενσωματωμένες συναρτήσεις και μεθόδους. Υποστηρίζει διαδραστικά και μη διαδραστικά γραφήματα σε μια μεγάλη ποικιλία από τύπους (γραφήματα γραμμών, στηλών, πίτας, ιστογράμματα, κ.α.), χαρακτηριστικά και επιλογές διαμόρφωσης. Είναι ανεξάρτητη πλατφόρμας και μεταφέρσιμη. Μπορεί να ενσωματωθεί σε μια γραφική διεπαφή χρήστη για την ανάπτυξη εφαρμογών και χρησιμοποιείται ευρέως σε Python εφαρμογές για οπτικοποίηση και ανάλυση δεδομένων.



Σχήμα 3.21: Ανατομία του figure, Πηγή: https://goo.gl/BI7etq

Τα βασικά Matplotlib αντικείμενα που συνθέτουν ένα *figure* (Σχήμα 3.21), δηλαδή ολόκληρη την περιοχή γραφήματος αποτυπώνονται στην Πίνακα 3.3.

Πίνακας 3.3: Τα βασικά Matplotlib αντικείμενα που συνθέτουν ένα figure

Αντικείμενο	**Περιγραφή**
FigureCanvas	Ο καμβάς, κλάση container για το στιγμιότυπο Figure
Figure	Η περιοχή γραφήματος, container για ένα ή περισσότερα στιγμιότυπα Axes
Axes	Η περιοχή σχεδίασης, δηλαδή η ορθογώνια περιοχή που συγκρατεί τα βασικά στοιχεία, όπως άξονες, γραμμές, κείμενο και ορίζει το σύστημα συντεταγμένων.

Υπάρχουν τρία στρώματα στο matplotlib API.

- Το *matplotlib.backend_bases.FigureCanvas* είναι η περιοχή πάνω στην οποία σχεδιάζεται το figure,
- το *matplotlib.backend_bases.Renderer* είναι το αντικείμενο που σχεδιάζει το FigureCanvas και
- το *matplotlib.artist.Artist* είναι το αντικείμενο που χρησιμοποιεί ένα renderer για να ζωγραφίσει πάνω στον καμβά.

Τα FigureCanvas και *Renderer* χειρίζονται όλες τις λεπτομέρειες για επικοινωνία με το GUI, ενώ το *Artist* διαχειρίζεται όλες τις κατασκευές υψηλού επιπέδου, όπως την απεικόνιση του figure, του κειμένου και των γραμμών. Υπάρχουν δύο τύποι *Artist*, οι πρωτογενείς (primitives) και οι containers. Οι πρωτογενείς αντιπροσωπεύουν τα τυπικά γραφικά αντικείμενα όπως *Line2D, Rectangle, Text, AxesImage* που τοποθετούνται στους containers (*Axis, Axes και Figure*) (Artist tutorial matplotlib, 2017). Στην εφαρμογή δημιουργείται ένα στιγμιότυπο *Figure*, το οποίο χρησιμοποιείται για τη δημιουργία ενός ή περισσοτέρων στιγμιότυπων *Axes* (*Subplot)*. Στη συνέχεια χρησιμοποιείται η ειδική βοηθητική μέθοδος *plot()* του στιγμιότυπου *Axes* για να δημιουργηθεί ο πρωτογενής τύπος *Line2D*.

Οι μέθοδοι της κλάσης Axes που χρησιμοποιήθηκαν στην εφαρμογή:

- plot(): σχεδιάζει γραμμές και/ή σημεία στον container *Axes*
- set_xlim(): ορίζει τα όρια για τα δεδομένα του άξονα x
- set_ylim(): ορίζει τα όρια για τα δεδομένα του άξονα y
- set_xlabel(): ορίζει τον τίτλο του άξονα x
- set_ylabel(): ορίζει τον τίτλο του άξονα y

- relim(): επαναπροσδιορίζει τα όρια των δεδομένων

- autoscale_view(): αυτόματη προσαρμογή της προβολής του γραφήματος, ανάλογα με τα όρια των δεδομένων.

- grid(): ενεργοποιεί η απενεργοποιεί το πλέγμα στους άξονες

3.4.6 Η βιβλιοθήκη PyQt

Η Qt είναι ένα ολοκληρωμένο -ανεξάρτητο πλατφόρμας- πλαίσιο ανάπτυξης εφαρμογών σε C++, που χρησιμοποιείται ευρέως για την ανάπτυξη γραφικών διεπαφών χρήστη (Graphical User Interfaces, GUIs) για cross-platform εφαρμογές. Μπορεί να χρησιμοποιηθεί σε διάφορες γλώσσες προγραμματισμού (Ruby, Java, Perl, Python) μέσω δεσμών (bindings). Η PyQt είναι ο δεσμός της Python με το Qt, δηλαδή, ένα σύνολο από Python βιβλιοθήκες, ώστε οι προγραμματιστές Python να έχουν πρόσβαση στις βιβλιοθήκες της Qt που είναι γραμμένες σε C++. Επομένως η PyQt συνδυάζει όλα τα πλεονεκτήματα του ανεξάρτητου πλατφόρμας πλαισίου ανάπτυξης εφαρμογών C++/Qt και της ανεξάρτητη πλατφόρμας, διερμηνευμένης γλώσσας προγραμματισμού Python.

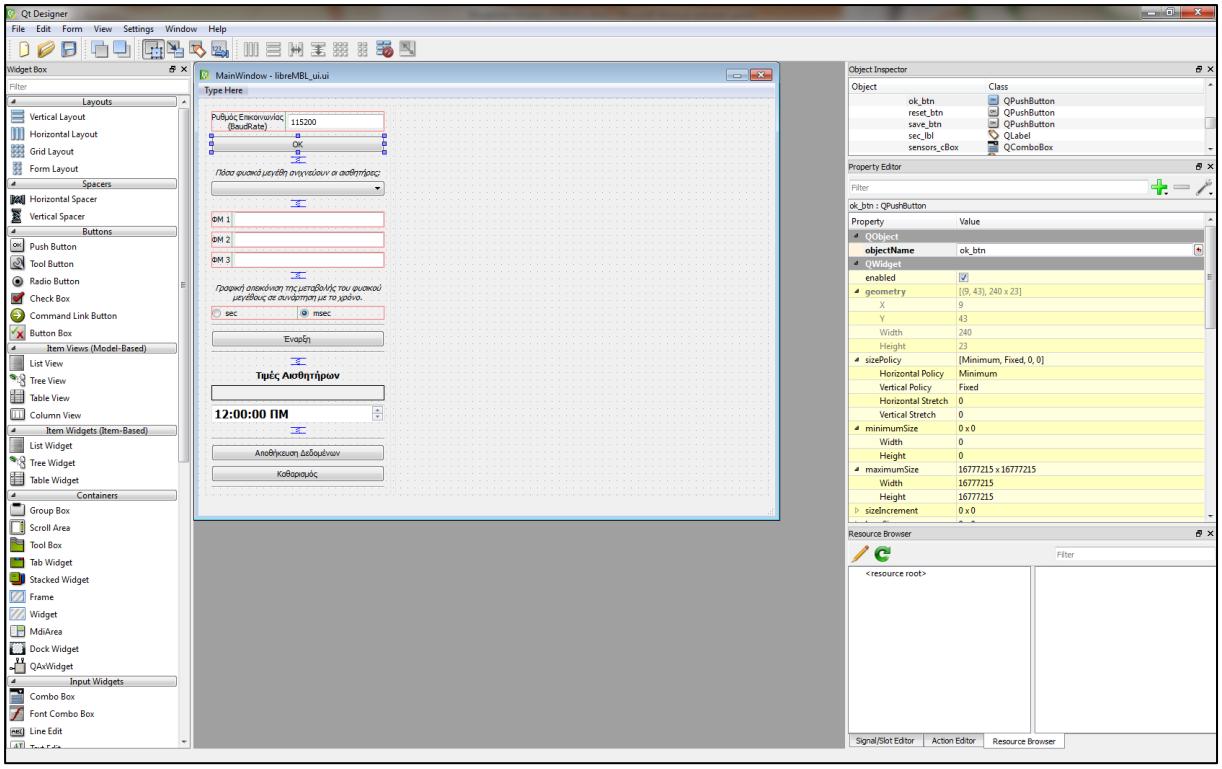
Τα συστατικά Qt αντιστοιχίζονται σε διάφορα υπο-αρθρώματα (submodules), όπου το PyQt είναι το κύριο άρθρωμα. Το άρθρωμα QtCore περιέχει τον πυρήνα των μη GUI κλάσεων, ενώ το άρθρωμα QtGui περιέχει την πλειοψηφία των GUI κλάσεων.

from PyQt4 import QtGui, QtCore

Η Qt περιλαμβάνει επίσης το Qt Designer, ένα εργαλείο σχεδιασμού και κατασκευής GUI από συστατικά Qt, που επιτρέπει τη γρήγορη και αποτελεσματική ανάπτυξη γραφικών διεπαφών για όλες τις υποστηριζόμενες πλατφόρμες (Windows, Linux, MacOS), και το οποίο χρησιμοποιήθηκε για την ανάπτυξη της γραφικής διεπαφής της εφαρμογής «libreMBL-GUI». (Σχήμα 3.22). Ενώ το εργαλείο Qt Designer δεν έχει τη δυνατότητα να κάνει αποσφαλμάτωση (debugging) και να «χτίσει» την εφαρμογή, οποιαδήποτε στοιχεία δημιουργηθούν με το Qt Designer μπορούν αργότερα να αλλάξουν δυναμικά μέσω του κώδικα.

Για να καλείται από την εφαρμογή «libreMBL-GUI» το αρχείο (libreMBL_ui.ui) που δημιουργήθηκε με το Qt Designer, χρησιμοποιήθηκε από το άρθρωμα *uic*, η συνάρτηση *loadUi(),* η οποία φορτώνει (load) το αρχείο και επιστρέφει ένα στιγμιότυπο της γραφικής διεπαφής.

from PyQt4.uic import loadUi



Σχήμα 3.22: Στιγμιότυπο του εργαλείου Qt Designer, με το αρχείο libreMBL_ui.ui της εφαρμογής

Ενσωμάτωση ενός Matplotlib figure σε ένα παράθυρο Qt

Για την ενσωμάτωση ενός Matplotlib figure σε ένα παράθυρο Qt, αρχικά εισάγεται το Matplotlib αντικείμενο *Figure*, που αποτελεί την ανεξάρτητη-από-backend περιοχή γραφήματος.

```
from matplotlib.figure import Figure
```

Στη συνέχεια από το άρθρωμα *matplotlib.backends.backend_qt4agg*, εισάγεται η κλάση *FigureCanvasQTAgg*, που είναι ο εξαρτώμενος-από-backend καμβάς του figure και δύναται να αποδώσει (render) στο Qt4 backend, το αντικείμενο *Figure*. Επίσης κληρονομεί από την κλαση QWidget, η οποία είναι η βασική κλάση όλων των αντικείμενων της γραφικής διεπαφής.

```
from matplotlib.backends.backend_qt4agg import (
FigureCanvasQTAgg as FigureCanvas,
NavigationToolbar2QT as NavigationToolbar)
```

Για την εμφάνιση της εργαλειοθήκης διαδραστικής πλοήγησης (Interactive Navigation Toolbar) εισάγεται η κλάση *NavigationToolbar2QT*, η οποία κληρονομεί επίσης από την κλαση QWidget.

3.5 Περιγραφή του κώδικα της εφαρμογής

Η εφαρμογή αποτελείται από τις κλάσεις *Main(QtGui.QMainWindow)* και *DynamicMplCanvas(FigureCanvas)* (Παράρτημα Β).

Η κλάση *Main(QtGui.QMainWindow)* συντονίζει την εκτέλεση του προγράμματος και έχει ως κύριο καθήκον τη δημιουργία της γραφικής διεπαφής (GUI). Εφ' όσον κληρονομεί από την κλάση *QtGui.QMainWindow* έχει όλες τις ιδιότητες για να συμπεριφέρεται ως το κύριο παράθυρο μιας GUI εφαρμογής. Περιλαμβάνει τις συναρτήσεις:

- initUI()
- portBaudIn()
- itemsHide()
- itemsShow()
- onComboBoxCurrentIndexChanged()
- receiving()
- dataToGui()
- onGraphPointsClick(event)
- startPause(state)
- onStartButtonPress()
- onSaveButtonPress()
- onResetButtonPress()
- closeEvent(event)

Η κλάση *DynamicMplCanvas(FigureCanvas)* είναι υπεύθυνη για τη δημιουργία, ανανέωση και ενσωμάτωση των γραφημάτων στο κεντρικό παράθυρο της GUI εφαρμογής. Κληρονομεί από την κλάση *FigureCanvas* που είναι ο εξαρτώμενος-από-backend καμβάς. Επίσης είναι ένα QWidget που μπορεί να ενσωματωθεί σε μια Qt εφαρμογή. Περιλαμβάνει τις συναρτήσεις:

- resetFigure ()
- initFigure (numplot)
- updateFigure (numplot)

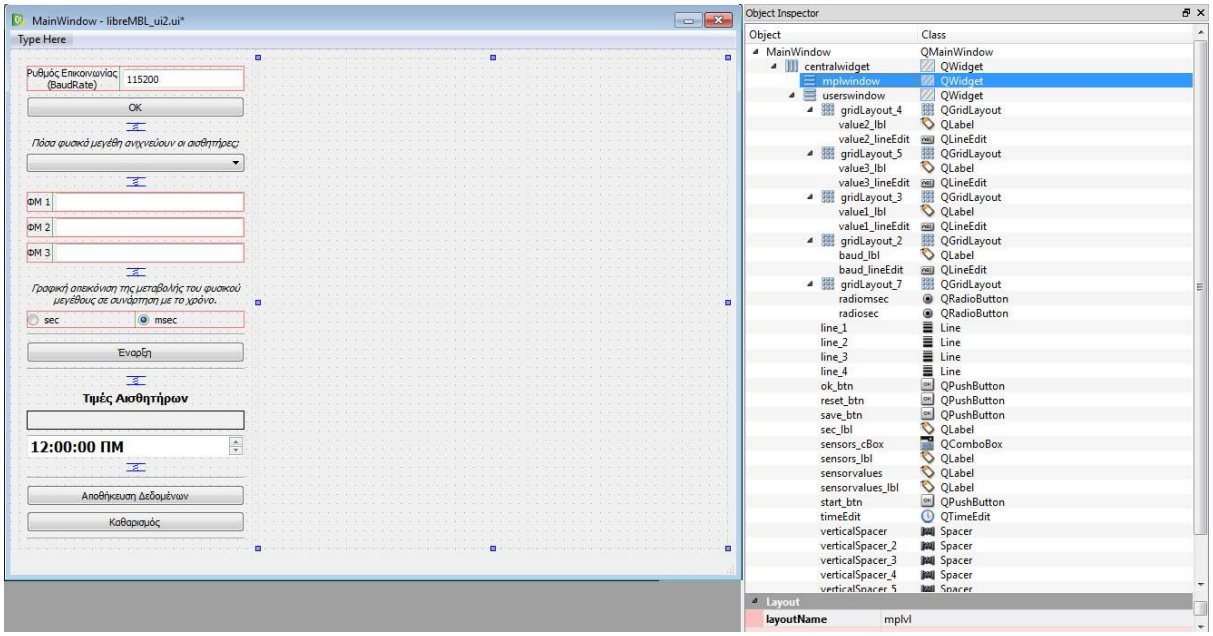
3.5.1 Ο κατασκευαστής της κλάσης Main

```
class Main(QtGui.QMainWindow):
  def __init__(self, ):
       super(Main, self).__init__()
       loadUi(os.path.join(os.path.dirname(__file__),'libreMBL_ui.ui'), self)
       self.setWindowTitle(u'Γραφική Αναπαράσταση Δεδομένων σε Πραγματικό Χρόνο')
       self.initUI()
```

Στην αρχή της κλάσης ορίζεται ο κατασκευαστής. Με *super*, επιστρέφεται το γονικό (parent) αντικείμενο της κλάσης, έτσι ώστε να ενεργεί σαν ένα αντικείμενο QT. Η συνάρτηση *loadUi()*, φορτώνει το αρχείο *'libreMBL_ui.ui'* και επιστρέφει ένα στιγμιότυπο της γραφικής διεπαφής που σχεδιάστηκε με το εργαλείο QtDesigner. Τέλος ορίζεται ο τίτλος του παραθύρου της εφαρμογής και καλείται η συνάρτηση *self.initUI()*.

3.5.2 Συνάρτηση initUI() της κλάσης Main

Στη συνάρτηση *initUI()*, δημιουργείται το στιγμιότυπο *self.dc* της κλάσης *DynamicMplCanvas()* και προστίθεται (addWidget) στο κατακόρυφο (vertical) layout *mplvl*, του container *mplwindow (Σχήμα 3.23)*. Ο καμβάς (*self.dc)* περιέχει και εμφανίζει ένα στιγμιότυπο *Figure*.



Σχήμα 3.23: Container mplwindow, με vertical layout mplvl

Επίσης στο layout *mplvl* προστίθεται η εργαλειοθήκη πλοήγησης (Navigation Toolbar), δηλαδή το widget που περιέχει τα κουμπιά για αλληλεπίδραση με τα γραφήματα. Το layout συγκρατεί τα widget σε ορισμένες θέσεις σε σχέση με τα άλλα στοιχεία και διασφαλίζει τη σωστή εμφάνισή τους (resize) όταν αλλάζει το μέγεθος του κυρίου παραθύρου της εφαρμογής.

```
self.dc = DynamicMplCanvas()
self.mplvl.addWidget(self.dc)
self.toolbar = NavigationToolbar(self.dc,
             self.mplwindow, coordinates=True)
self.mplvl.addWidget(self.toolbar)
```

Στη συνέχεια ορίζεται για κάθε κουμπί της εφαρμογής, η συνάρτηση που θα εκτελείται όταν ο χρήστης με αριστερό κλικ το επιλέξει. Παρόμοια για το comboBox widget *sensors_cBox* ορίζεται η συνάρτηση που θα καλείται όταν ο χρήστης επιλέξει διαφορετική τιμή από την πτυσσόμενη λίστα επιλογών.

```
self.ok_btn.clicked.connect(self.portBaudIn)
self.start_btn.clicked.connect(self.onStartButtonPress)
self.reset_btn.clicked.connect(self.onResetButtonPress)
self.save_btn.clicked.connect(self.onSaveButtonPress)
self.sensors_cBox.currentIndexChanged[int].connect(self.onComboBoxCurrentIndexChang
ed)
```

Τέλος καλείται η συνάρτηση *self.onResetButtonPress()* που καθορίζει την αρχική εμφάνιση της εφαρμογής.

3.5.3 Συνάρτηση portBaudIn() της κλάσης Main

Η συνάρτηση καλείται όταν ο χρήστης καθορίσει τον ρυθμό επικοινωνίας (baudrate) στο αντίστοιχο widget πλαίσιο κειμένου (*baud_lineEdit*), με προεπιλεγμένη τιμή 115200, και επιλέξει το κουμπί «OK» (*ok_btn*). Η συνάρτηση είναι υπεύθυνη για το άνοιγμα της σειριακής θύρας ώστε να είναι εφικτή η επικοινωνία με την πλατφόρμα Arduino σε κοινό ρυθμό επικοινωνίας.

Αν η σειριακή θύρα είναι ήδη ανοιχτή, γίνεται εκκαθάριση της buffer εισόδου και εξόδου και κλείνει άμεσα η σειριακή θύρα. Στη συνέχεια αναζητείται, σε λίστα που περιέχει τις πιο γνωστές σειριακές θύρες για Linux και Windows, η συσκευή που συνδέθηκε. Όταν βρεθεί, ανοίγει η σειριακή θύρα και εμφανίζεται το όνομά της στη γραμμή κατάστασης. Στην αντίθετη περίπτωση εμφανίζεται προειδοποιητικό μήνυμα με *QtGui.QMessageBox()*.

```
def portBaudIn(self):
        locations=['/dev/ttyACM0', '/dev/ttyACM1','/dev/ttyACM2', '/dev/ttyACM3',
           '/dev/ttyACM4', '/dev/ttyACM5','/dev/ttyUSB0','/dev/ttyUSB1',
           '/dev/ttyUSB2','/dev/ttyUSB3', '/dev/ttyUSB4', '/dev/ttyUSB5',
           '/dev/ttyUSB6', '/dev/ttyUSB7', '/dev/ttyUSB8', '/dev/ttyUSB9',
           '/dev/ttyUSB10','/dev/ttyS0', '/dev/ttyS1', '/dev/ttyS2', 'com2',
           'com3', 'com4', 'com5', 'com6', 'com7', 'com8', 'com9', 'com10',
           'com11', 'com12', 'com13', 'com14', 'com15', 'com16', 'com17',
           'com18', 'com19', 'com20', 'com21', 'com1', 'end']
        if hasattr(self, 'ser'):
            try:
                self.ser.reset_input_buffer()
                self.ser.reset_output_buffer()
                self.ser.close()
            except:
                pass
        for device in locations:
            try:
```

```
                self.ser    =    serial.Serial(device,int(self.baud_lineEdit.text()),
                                                                    timeout=2)
                self.statusbar.showMessage(u'Θύρα επικοινωνίας: '+ /
                                                    self.ser.name.upper())
                self.ser.reset_input_buffer()
                break
            except:
                if device == 'end':
                     warn = QtGui.QMessageBox()
                     warn.setWindowTitle(u'Προειδοποίηση')
                     warn.setText(u'Έχετε συνδέσει το καλώδιο;')
                     warn.exec_()
                     return
```

3.5.4 Συνάρτηση onComboBoxCurrentIndexChanged() της κλάσης Main

Η συνάρτηση καλείται όταν ο χρήστης επιλέξει διαφορετική τιμή από την πτυσσόμενη λίστα των τεσσάρων επιλογών του comboBox *sensors_cBox*. Ανάλογα την τιμή του ευρετηρίου (1,2 ή 3) εμφανίζονται οι κατάλληλες ετικέτες και πλαίσια κειμένου, ώστε ο χρήστης να πληκτρολογήσει τα (1, 2, ή 3) φυσικά μεγέθη που ανιχνεύουν οι αισθητήρες. Επίσης για τον καμβά (*self.dc*) καλείται η συνάρτηση *initFigure (numplot)* της κλάσης *DynamicMplCanvas(FigureCanvas)* με όρισμα την τιμή του ευρετηρίου, για την αρχικοποίηση των αντικειμένων *Axes και Line2D*.

```
i=int(self.sensors_cBox.currentIndex())
self.dc.initFigure(i)
```

3.5.5 Συνάρτηση onStartButtonPress () της κλάσης Main

Η συνάρτηση καλείται όταν ο χρήστης επιλέξει το κουμπί *start_btn* («Έναρξη», «Παύση» ή «Συνέχιση»). Άμεσα καλείται η συνάρτηση *self.startPause(state)*, με όρισμα την άρνηση (not) της μεταβλητής αντικειμένου *self. running* που έχει αρχική τιμή False.

```
def onStartButtonPress(self):
        self.startPause(not self.running)
```

3.5.6 Συνάρτηση startPause(state) της κλάσης Main

Αρχικά αντιστοιχίζεται στην τιμή της μεταβλητής αντικειμένου *self. running*, η τιμή του ορίσματος state.

Αν η τιμή της μεταβλητής αντικειμένου *self. running* είναι True (δηλαδή, έχει πατηθεί το κουμπί «Έναρξη» ή «Συνέχιση»):

- Δημιουργείται νήμα (thread), για να μπορεί το πρόγραμμα να επικοινωνεί με τη σειριακή θύρα *(self.ser)* για λήψη και αποστολή δεδομένων, ενώ ταυτόχρονα εκτελείται το GUI (κύριο νήμα ελέγχου).
- Το νήμα παίρνει ως όρισμα τη συνάρτηση *self.receiving()*, η οποία θα εκτελέσει το νήμα.
- Με τη μέθοδο *start()*, ξεκινά η εκτέλεση του νήματος.

Αν η τιμή της μεταβλητής αντικειμένου *self. running* είναι False (δηλαδή, έχει πατηθεί το κουμπί «Παύση»):

- Το κύριο νήμα ελέγχου περιμένει έως ότου επέλθει το χρονικό όριο των 2 seconds.
- Στον καμβά (*self.dc*) συνδέεται η συνάρτηση *self.onGraphPointsClick(event)* η οποία εκτελείται όταν ο χρήστης επιλέξει με κλικ *(button_press_event)* κάποιο σημείο της περιοχής του γραφήματος.
- Γίνεται εκκαθάριση της σειριακής buffer εισόδου και εξόδου.

```
self.running = state
if self.running:
       self.start_btn.setText(u'Παύση')
       self.comt=threading.Thread(target = self.receiving)
       self.comt.start()
else:
       if hasattr(self, 'comt'):
            self.comt.join(2)
       self.start_btn.setText(u'Συνέχιση')
       self.dc.mpl_connect('button_press_event', self.onGraphPointsClick)
```

3.5.7 Συνάρτηση receiving() της κλάσης Main

Η συνάρτηση *receiving()* είναι υπεύθυνη για την λήψη και αποστολή δεδομένων από και προς τη σειριακή θύρα αντίστοιχα. Τα δεδομένα που διοχετεύονται για αποστολή από την πλατφόρμα Arduino, μέσω της σειριακής θύρας ως ASCII κείμενο, είναι χωρισμένα με κόμμα (,), με πρώτο το ρυθμό δειγματοληψίας (ms) ακολουθούμενο από τη μέτρηση για κάθε φυσικό μέγεθος. Στο τέλος υπάρχει αλλαγή γραμμής (\r\n).

Ο χρήστης έχει πατήσει το κουμπί «Εναρξη» ή «Συνέχιση» και η τιμή της μεταβλητής αντικειμένου *self. running* είναι True. Για όσο είναι True (δηλαδή μέχρι ο χρήστης να πατήσει το κουμπί «Παύση») επαναλαμβάνονται από το πρόγραμμα τα παρακάτω:

- Γράφει στη σειριακή θύρα έναν χαρακτήρα (π.χ. ‘*’). Αυτό έχει ως αποτέλεσμα, να υπάρχουν δεδομένα στη μνήμη σειριακής λήψης (Serial Receive Buffer) της

πλατφόρμας Arduino ώστε να ξεκινήσει η διαδικασία των μετρήσεων και η αποστολή των δεδομένων στον υπολογιστή.

- Περιμένει μέχρι να υπάρχουν δεδομένα στη buffer λήψης, και όταν υπάρχουν τα διαβάζει και τα προσθέτει στο αλφαριθμητικό *mybuffer*.
- Αν υπάρχει ο χαρακτήρας νέας γραμμής (‘\n’) στο αλφαριθμητικό *mybuffer* τότε*:*
  - Το αλφαριθμητικό *mybuffer* χωρίζεται με τη χρήση της συνάρτησης *split ('\n')* με όρισμα το χαρακτήρα νέας γραμμής. Επιστρέφεται μία λίστα (*lines)* που περιέχει όλα τα κομμάτια.
  - Το τελευταίο στοιχείο της λίστας *lines* (*lines[-1]*) ανατίθεται ως τιμή στο mybuffer.
  - Το προτελευταίο στοιχείο της λίστας *lines* (*lines[-2])* που περιέχει τα δεδομένα, χωρίζεται (split) με βάση το κόμμα (‘,’) και τα στοιχεία της λίστας που δημιουργείται, ανατίθενται στη λίστα *lastReceived* που είναι καθολική μεταβλητή. Επομένως τα στοιχεία της λίστας *lastReceived* αφορούν τα δεδομένα των μετρήσεων για μια χρονική στιγμή.
  - Καλείται η συνάρτηση *self.dataToGui()* που θα διαχειριστεί αυτά τα δεδομένα ώστε είτε να εμφανιστούν στα widget του GUI, είτε στις γραφικές παραστάσεις μέσω συνάρτησης της κλάσης *DynamicMplCanvas(FigureCanvas).*

```
def receiving(self):
    global lastReceived
    mybuffer = ''
    while self.running:
        self.ser.write('*')
        while (self.ser.in_waiting==0):
            pass #do nothing
        mybuffer = mybuffer + self.ser.read(self.ser.in_waiting)
        if '\n' in mybuffer:
            lines = mybuffer.split('\n')
            mybuffer = lines[-1]
            lastReceived=lines[-2].split(',')
            self.dataToGui()
```

3.5.8 Συνάρτηση dataToGui() της κλάσης Main

- Όλα τα στοιχεία της λίστας *lastReceived* (καθολική μεταβλητή) ξεκινώντας από το δεύτερο (*lastReceived[1:]*), ενώνονται με τη συνάρτηση *join*(), με ενδιάμεσο διαχωριστικό το κόμμα και το αποτέλεσμα ανατίθεται στην τοπική μεταβλητή *values*. Η *values* επομένως περιέχει όλες τις μετρήσεις των φυσικών μεγεθών μιας χρονικής στιγμής, οι οποίες είναι χωρισμένες με κόμμα και εμφανίζονται στο label widget

sensorvalues του GUI. Επίσης εμφανίζεται στο widget *timeEdit* η τρέχουσα ώρα *QtCore.QTime.currentTime()* με μορφή hh:mm:ss.zzz.

```
values=','.join(lastReceived[1:])
self.sensorvalues.setText(values)
self.timeEdit.setTime(QtCore.QTime.currentTime())
```

- Εφ' όσον το πρώτο στοιχείο της λίστας (*lastReceived[0]*) περιέχει το χρόνο που μεσολαβεί μεταξύ των μετρήσεων, συμπληρώνεται κατάλληλα η λίστα stime (καθολική μεταβλητή), ώστε να περιέχει την τελευταία χρονική στιγμή των μετρήσεων (sec ή msec), ξεκινώντας από τη χρονική στιγμή 0.

```
if self.radiosec.isChecked():
      multiplier=0.001
else:
 multiplier=1
if (stime):
      stime.append(int(lastReceived[0])*multiplier+stime[-1])
      stime.pop(0)
else:
      stime.append(int(lastReceived[0])*multiplier)
      stime.append(int(lastReceived[0])*multiplier-stime[0])
      stime.pop(0)
```

- Με χρήση της συνάρτηση *join()*, με ενδιάμεσο διαχωριστικό το κόμμα, ενώνονται η τρέχουσα ώρα, η τελευταία χρονική στιγμή και οι μετρήσεις (μεταβλητή *values*). Το αποτέλεσμα προστίθεται (*append()*) στη λίστα *self.myList* (μεταβλητή αντικειμένου) που θα περιέχει όλα τα δεδομένα. Στη συνέχεια θα αξιοποιηθεί από τη συνάρτηση *self.onSaveButtonPress* για αποθήκευση των δεδομένων.

```
ctime=str(QtCore.QTime.currentTime().toString('hh:mm:ss'))
self.myList.append(','.join((ctime, str(stime[-1]), values)))
```

- Για τον καμβά (*self.dc*) καλείται η συνάρτηση *updateFigure (numplot)* της κλάσης *DynamicMplCanvas* με όρισμα την τιμή ευρετηρίου (1,2 ή 3) του comboBox widget *sensors_cBox*, ώστε να ανανεωθούν τα γραφήματα με τις νέες τιμές. Επίσης ανανεώνονται τα στοιχεία της λίστας *ylabel* (καθολική μεταβλητή) με τα περιεχόμενα των πλαισίων κειμένου που αφορούν τα φυσικά μεγέθη που έχει πληκτρολογήσει ο χρήστης, και τα οποία θα εμφανιστούν ως τίτλοι στον άξονα y των γραφημάτων.

```
i=int(self.sensors_cBox.currentIndex())
if i >= 1:
 ylabel[0]=unicode(self.value1_lineEdit.text())
if i >= 2:
 ylabel[1]=unicode(self.value2_lineEdit.text())
if i >= 3:
 ylabel[2]=unicode(self.value3_lineEdit.text())
```

```
self.dc.updateFigure(i)
```

3.5.9 Συνάρτηση onSaveButtonPress () της κλάσης Main

Η συνάρτηση καλείται όταν ο χρήστης επιλέξει το κουμπί «Αποθήκευση Δεδομένων» (*save_btn*), ώστε να αποθηκευτούν σε αρχείο κειμένου (.txt) τα δεδομένα όλων των μετρήσεων που βρίσκονται στη μεταβλητή αντικειμένου *self.myList*.

Στο φάκελο που υπάρχει το αρχείο του προγράμματος, δημιουργείται (αν δεν υπάρχει) φάκελος με όνομα *'data'*. Καλώντας τη συνάρτηση *getSaveFileName()* της κλάσης *QtGui.QFileDialog*, παρέχεται στο χρήστη ένα παράθυρο διαλόγου στο οποίο δύναται να επιλέξει το φάκελο που θα αποθηκευτεί το αρχείο καθώς και το όνομα αρχείου. Ως προεπιλογή προτείνετε ο φάκελος *data* και ως όνομα αρχείου η τρέχουσα ημερομηνία και ώρα (συνάρτηση *strftime()* του module *time*). Η επιστροφή της συνάρτησης ανατίθεται στην τοπική μεταβλητή *name*. Για αναγνώριση των Unicode ονομάτων γίνεται χρήση κατάλληλων συναρτήσεων.

```
encoding = sys.getfilesystemencoding()
directory = os.path.dirname(os.path.realpath(__file__)) +os.sep + 'data'
directory=unicode(directory, encoding)
if not os.path.exists(directory):
       os.makedirs(directory)
path=directory+os.sep+time.strftime("%Y%m%d-%H%M%S")+'.txt'
name = QtGui.QFileDialog.getSaveFileName(self, 'Save File',
                                                 path, filter ='txt (*.txt *.)')
```

Στη συνέχεια αξιοποιούνται οι συναρτήσεις της Python για διαχείριση αρχείων: open(), write(), close().

3.5.10 Ο κατασκευαστής της κλάσης DynamicMplCanvas

Στον κατασκευαστή της κλάσης ορίζεται το αντικείμενο *Figure* και στη συνέχεια γίνεται αρχικοποίηση του καμβά (αντικείμενο *FigureCanvas*) με αναφορά στο αντικείμενο *Figure*. Ο καμβάς είναι υπεύθυνος για την απόδοση του αντικειμένου *Figure* σε ένα Qt widget ώστε να ενσωματωθεί σε μια Qt εφαρμογή. Στη συνέχεια τίθεται στη γονική παράμετρο του __init__, το parent του συγκεκριμένου widget.

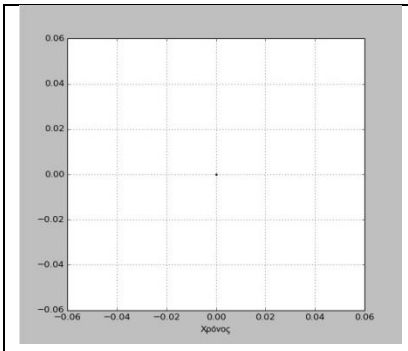
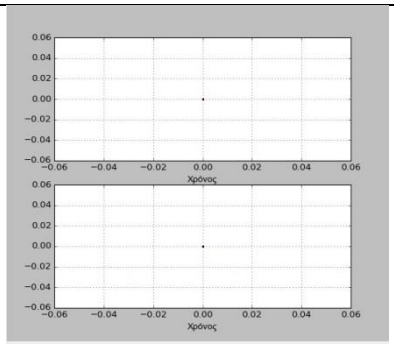
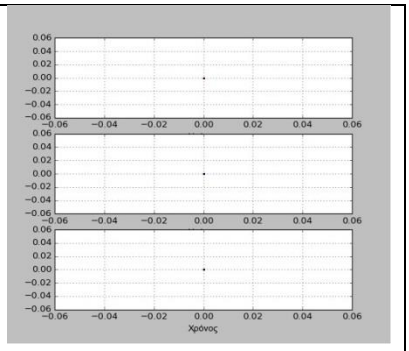
```
def __init__(self, parent=None):
    self.fig = Figure()
    FigureCanvas.__init__(self, self.fig)
    self.setParent(parent)
```

3.5.11 Συνάρτηση initFigure (numplot) της κλάσης DynamicMplCanvas

Στη συνάρτηση *initFigure (numplot)*, γίνεται αρχικοποίηση *numplot* (όπου numplot=1,2 ή 3) αντικειμένων *Axes*, που στο καθένα θα σχεδιάζεται μία γραφική παράσταση. Τα αντικείμενα *Axes* προστίθενται στο *Figure* με τη συνάρτηση

add_subplot(αριθμός_γραμμών, αριθμός_στηλών, fignum)

που δημιουργεί *αριθμός_γραμμών*αριθμός_στηλών* subplots, με fignum το τρέχων subplot που θα σχεδιαστεί η γραφική παράσταση (Σχήμα 3.24).

		
για numplot=1 self.axes[1] = self.fig.add_subplot(1,1, 1)	για numplot=2 self.axes[1] = self.fig.add_subplot(2,1, 1) self.axes[2] = self.fig.add_subplot(2,1, 2)	για numplot=3 self.axes[1] = self.fig.add_subplot(3,1, 1) self.axes[2] = self.fig.add_subplot(3,1, 2) self.axes[3] = self.fig.add_subplot(3,1, 2)

Σχήμα 3.24: Αρχικοποίηση 1,2, ή 3 αντικειμένων Axis

Στη συνέχεια γίνεται αρχικοποίηση *numplot* (όπου numplot=1,2 ή 3) αντικειμένων *Line2D*, με την κλήση της συνάρτησης *plot(x,y)* που σχεδιάζει γραμμές και σημεία στο Axes.

```
def resetFigure(self):
    self.ydata=[[0 for i in range(100)], [0 for i in range(100)], [0 for i in
range(100)]]
    self.t=[0.0]*100
    self.axes={}
    self.line={}

def initFigure(self, numplot):
    colors = ['r', 'b', 'g']
    #Clear the figure
    self.fig.clf(keep_observers=True)

    for i in range(1, numplot+1):
       self.axes[i] = self.fig.add_subplot(numplot,1, i)
       self.line[i], = self.axes[i].plot(self.ydata[i-1], self.t,
                       colors[i-1]+'o-', markersize=2)
       self.axes[i].grid(True)
       self.axes[i].set_xlabel(u'Χρόνος', fontname='Arial' )

       #Redraw the current figure
       self.draw()
```

3.5.12 Συνάρτηση updateFigure (numplot) της κλάσης DynamicMplCanvas

Η συνάρτηση επανασχεδιάζει το figure προσθέτοντας τα νέα δεδομένα των μετρήσεων και ανανεώνει τα γραφήματα εμφανίζοντας τις τελευταίες 100 τιμές. Με αυτό τον τρόπο εξασφαλίζεται η μείωση της καθυστέρησης επανασχεδιασμού. Για να είναι σταθερός ο αριθμός των σημείων (100) που εμφανίζονται σε κάθε γράφημα, στη λίστα δεδομένων για συντεταγμένες x (*self.t*) και y (*self.ydata*) αντίστοιχα, προστίθεται η νέα τιμή και αφαιρείται η πρώτη.

```
self.t.append(float(stime[-1]))
self.t.pop(0)
for i in range(1, numplot+1):
    self.ydata[i-1].append(float(lastReceived[i]))
    self.ydata[i-1].pop(0)
    self.line[i].set_xdata(self.t)
    self.line[i].set_ydata(self.ydata[i-1])
    self.axes[i].relim(); self.axes[i].autoscale_view()
    self.axes[i].set_ylabel(ylabel[i-1],fontname='Arial')
    self.axes[i].ticklabel_format(useOffset=False, style='plain')
self.draw()
```

ΣΥΖΗΤΗΣΗ - ΣΥΜΠΕΡΑΣΜΑΤΑ

Οι κύριοι στόχοι των ανοικτών συστημάτων (υλικού και λογισμικού) αφορούν στη διαλειτουργικότητα, στη φορητότητα και στην επεκτασιμότητα. Η διαλειτουργικότητα διασφαλίζεται με την προσαρμογή σε ένα σύνολο προτύπων, συμβάσεων και προδιαγραφών. Η φορητότητα επιτρέπει τη λειτουργία του συστήματος σε διαφορετικές πλατφόρμες χωρίς αλλαγές, ενώ η δυνατότητα επέκτασης επιτρέπει στον χρήστη να αυξήσει τη λειτουργικότητα του συστήματος. Επίσης τα ανοιχτά συστήματα εξασφαλίζουν τη διαφάνεια της εσωτερικής λειτουργίας.

Στην παρούσα εργασία δημιουργήθηκε περιβάλλον γραφικής διεπαφής χρήστη γενικού σκοπού, σε ένα ανοιχτό, χαμηλού κόστους σύστημα συγχρονικής λήψης και απεικόνισης δεδομένων.

Η πλακέτα Arduino Uno προγραμματίστηκε να λειτουργεί ως συσκευή πρόσκτησης δεδομένων, η οποία λαμβάνει και επεξεργάζεται μετρήσεις από αναλογικούς ή ψηφιακούς αισθητήρες και στη συνέχεια αποστέλλει τα αποτελέσματα στον υπολογιστή μέσω σύνδεσης USB. Η μεγάλη ποικιλία αξιόπιστων και χαμηλού κόστους αναλογικών και ψηφιακών αισθητήρων, που διατίθενται στην αγορά με επαρκή τεκμηρίωση από τις κατασκευάστριες εταιρίες και την κοινότητα χρηστών του Arduino, επεκτείνει τις δυνατότητες του συστήματος για διεξαγωγή μεγάλου εύρους πειραματικών μετρήσεων και κάλυψη οποιασδήποτε σχετικής ανάγκης στο σχολικό εργαστήριο ΦΕ. Το ανεξάρτητο πλατφόρμας ολοκληρωμένο περιβάλλον ανάπτυξης IDE, το οποίο είναι σχεδιασμένο για μη έμπειρους προγραμματιστικά χρήστες, ενσωματώνει και απλοποιεί σημαντικά τα βήματα συγγραφής, επεξεργασίας, μεταγλώττισης και μεταφόρτωσης του αντίστοιχου κώδικα στον μικροεπεξεργαστή, σε συνδυασμό με το πλούσιο υλικό (βιβλιοθήκες, παραδείγματα, εκπαιδευτικό υλικό) που διατίθεται στο Διαδίκτυο.

Στο πλαίσιο της παρούσας εργασίας, δημιουργήθηκαν ενδεικτικά πέντε (5) προγράμματα (sketches) για τους αισθητήρες θερμοκρασίας (TMP36, LM35, DS18B20), πίεσης/θερμοκρασίας (BMP180) και απόστασης (HC-SR04) (https://goo.gl/a4N2AW). Δόθηκε ιδιαίτερη έμφαση στον κώδικα, ώστε τα προγράμματα να ακολουθούν ένα κοινό μοτίβο. Με αυτό τον τρόπο διασφαλίζεται η εύκολη -από έναν αρχάριο προγραμματιστή- τροποποίηση/επέκταση ή δημιουργία εξ' αρχής κώδικα για διαφορετικούς αισθητήρες ή συνδυασμό αισθητήρων.

Στον υπολογιστή, το περιβάλλον γραφικής διεπαφής (εφαρμογή libreMBL-GUI) για την αμφίδρομη επικοινωνία χρήστη-συστήματος, αναπτύχθηκε με τη γλώσσα προγραμματισμού Python (έκδοση 2.7) και δοκιμάστηκε σε λειτουργικά συστήματα Ubuntu Linux και Windows χωρίς αλλαγές. Διατίθεται στη διεύθυνση https://goo.gl/a4N2AW ως λογισμικό ανοιχτού κώδικα, ώστε να μπορεί ελεύθερα να χρησιμοποιηθεί, τροποποιηθεί και επεκταθεί από τους χρήστες του. Επιτρέπει τη ρύθμιση της σειριακής επικοινωνίας, τον έλεγχο έναρξης και λήξης της γραφικής αναπαράστασης των μετρήσεων σε πραγματικό χρόνο, τη διαδραστική πλοήγηση του μαθητή στις γραφικές παραστάσεις και την αποθήκευση δεδομένων/γραφικών παραστάσεων για μεταγενέστερη επεξεργασία.

Ενώ η εφαρμογή μπορεί να υποστηρίξει διαφορετικά είδη Arduino και δύναται να απεικονίσει τις μετρήσεις οποιουδήποτε συμβατού αισθητήρα που συνδέεται στην πλακέτα, έχει τους εξής περιορισμούς:

- Είναι απαραίτητη η δημιουργία και μεταφόρτωση στον μικροελεγκτή του κατάλληλου προγράμματος (sketch), που αφορά στον αισθητήρα ή συνδυασμό αισθητήρων που συνδέονται στην πλακέτα.
- Το πρόγραμμα (sketch) πρέπει να διαβάζει και να επεξεργάζεται περιοδικά τις μετρήσεις του αισθητήρα (ή συνδυασμό αισθητήρων) και να αποστέλλει τα αποτελέσματα της επεξεργασίας στον υπολογιστή, μέσω της σειριακής θύρας.
- Τα αποτελέσματα που διοχετεύονται μέσω της σειριακής θύρας στον υπολογιστή, πρέπει να είναι χωρισμένα με κόμμα (,) ακολουθούμενα στο τέλος από αλλαγή γραμμής (\r\n). Αφορούν αρχικά το ρυθμό δειγματοληψίας (ms) και στη συνέχεια τη μέτρηση για κάθε φυσικό μέγεθος.
- Ο χρήστης δεν έχει δυνατότητα επιλογής του ρυθμού δειγματοληψίας μέσω της εφαρμογής. Ο ρυθμός δειγματοληψίας για κάθε αισθητήρα (ή συνδυασμό αισθητήρων) καθορίζεται στο αντίστοιχο πρόγραμμα (sketch).
- Ο χρήστης δεν έχει δυνατότητα επιλογής διαφορετικής -από αυτής που καθορίζεται στο αντίστοιχο πρόγραμμα (sketch)- μονάδας μέτρησης του φυσικού μεγέθους που ανιχνεύει ο αισθητήρας.
- Η εφαρμογή δίνει τη δυνατότητα γραφικής απεικόνισης της μεταβολής ενός φυσικού μεγέθους σε συνάρτηση με το χρόνο (ξεκινώντας από τη χρονική στιγμή 0) και

δύναται να εμφανίζει ταυτόχρονα τις γραφικές παραστάσεις μέχρι τριών (3) φυσικών μεγεθών.

Στη συνέχεια αναφέρονται συγκεντρωτικά μερικά ειδικά συμπεράσματα, που προέκυψαν από την παρούσα εργασία:

- Ο συνδυασμός της πλατφόρμας Arduino με τη γλώσσα προγραμματισμού Python, αποτελούν ένα φτηνό αλλά ιδιαίτερα αποτελεσματικό εργαλείο για την γρήγορη ανάπτυξη εφαρμογών πρόσκτησης δεδομένων από μετρήσεις, όπως αυτή που παρουσιάστηκε στην παρούσα εργασία.
- Αποδείχθηκε ότι είναι δυνατή η ανάπτυξη λογισμικού διαχείρισης σε χαμηλού κόστους σύστημα συγχρονικής λήψης και απεικόνισης, το οποίο μπορεί να καλύψει τις ανάγκες των εργαστηρίων ΦΕ. Η ανάπτυξη αυτή φάνηκε ότι μπορεί να πραγματοποιηθεί σε μικρό σχετικά χρονικό διάστημα, λόγω της σημαντικής υποστήριξης που υπάρχει στο διαδίκτυο από τις κοινότητες χρηστών του Arduino και της γλώσσας Python.
- Η χρησιμοποίηση εργαλείων (υλικού ή λογισμικού) ανοικτής αρχιτεκτονικής και ανοικτού κώδικα προσφέρει πολύ μεγάλες δυνατότητες για ανάπτυξη σύνθετων εφαρμογών, με μικρό κόστος και μικρό χρόνο ανάπτυξης.
- Ειδικότερα, το ανεξάρτητο πλατφόρμας ολοκληρωμένο περιβάλλον ανάπτυξης IDE, του Arduino ενσωματώνει και απλοποιεί σημαντικά τα βήματα συγγραφής, επεξεργασίας, μεταγλώττισης και μεταφόρτωσης του αντίστοιχου κώδικα στον μικροεπεξεργαστή.
- Η ανάπτυξη ενός συστήματος (που λαμβάνει υπόψη τις ιδιαίτερες ανάγκες των εργαστηρίων ΦΕ), μπορεί να απαιτεί επένδυση σε αρχικό χρόνο εκπαίδευσης και ανάπτυξης, αλλά προσφέρει αρκετά πλεονεκτήματα σε σχέση με τη χρήση εμπορικών προϊόντων ή ακόμα και έτοιμων (μη εμπορικών) ανάλογων εφαρμογών. Τα πλεονεκτήματα αυτά σχετίζονται με το χαμηλό κόστος, την απλότητα, την απουσία ανάγκης υποστήριξης, τη διαλειτουργικότητα/φορητότητα/επεκτασιμότητα και τη δυνατότητα χρήσης από μη εξειδικευμένο προσωπικό.
- Η χρήση της γλώσσας Python σε συνεργασία με τις βιβλιοθήκες PyQt και Matplotlib επέτρεψε να δημιουργηθεί ένα γραφικό περιβάλλον διεπαφής που είναι ανεξάρτητο λειτουργικού συστήματος και μπορεί να χρησιμοποιηθεί τόσο με Windows όσο και με Linux.

Η παρούσα εργασία αποτέλεσε την αφετηρία για ενασχόληση με τις ανοιχτές τεχνολογίες που βασίζονται σε μικροελεγκτές, καθώς και τη γλώσσα προγραμματισμού Python. Τα μελλοντικά βήματα αφορούν:

- Μετατροπή της εφαρμογής «libreMBL-GUI» σε Python 3x.
- Δημιουργία προγραμμάτων (sketches) για αισθητήρες PH, υγρασίας, επιτάχυνσης, δύναμης κ.α.
- Δοκιμή της εφαρμογής, σε συνεργασία με εκπαιδευτικούς ΦΕ, σε πραγματικές εργαστηριακές συνθήκες με χρήση προσυναρμολογημένων συστημάτων (Arduino – αισθητήρες – sketch) και αποτύπωση των αποτελεσμάτων.
- Βελτίωση/επέκταση του κώδικα της εφαρμογής «libreMBL-GUI» για μείωση των περιορισμών της.

ΑΝΑΦΟΡΕΣ

Al-Mamun, A., Sundaraj, K., Ahmed, N., Rahman, M., Rahman, M., Ahamed, N.U. (2013). Design and development of a PC-based automated data logging system for measuring temperature. *ARPN Journal of Engineering and Applied Sciences*, 8(11), 960-968.

Arduino (2017). Ανακτήθηκε στις 9/5/2017 από https://www.arduino.cc.

Artist tutorial matplotlib (2017). Ανακτήθηκε στις 9/5/2017 από https://matplotlib.org/users/artists.html.

Atkin, K. (2016). Construction of a simple low-cost teslameter and its use with Arduino and MakerPlot software. *Phys. Educ.* 51(2), 024001 (7pp). DOI: https://doi.org/10.1088/0031-9120/51/2/024001

Baldini, L., Sgrò, C., Andreoni, E., Angelini, F., Bianchi, A., Bregeon, J., Fidecaro, F., Massai, M. M., Merlin, V., Nespolo, J., Orselli, S. & Pesce-Rollins, M. (2013). Plasduino: an inexpensive, general purpose data acquisition framework for educational experiments (cite arxiv:1312.1805 Comment: 11 pages, 10 figures, presented at the *XCIX conference of the Società Italiana di Fisica)*.

Banzi, M.(2011). *Getting Started with Arduino*, 2nd ed. O'Reilly Media, 2011.

Barton, R. and Rogers, L. (1991), The computer as an aid to practical science—studying motion with a computer. *Journal of Computer Assisted Learning*, 7(2), 104–113. doi:10.1111/j.1365-2729.1991.tb00233.x

Bouquet, F., Bobroff, J. (2017). Project-based physics labs using low-cost open-source hardware. *American Journal of Physics*, 85(3), 216-222. doi: 10.1119/1.4972043

Brasell, Heather. (1987), The Effects of Real-Time Laboratory Graphing on Learning Graphic Representations of Distance and Velocity. *Journal of Research in Science Teaching* 24 (4), 385-395.

Chen, S., Lo, H-C., Lin, J-W. Liang, J-C., Chang, H-Y., Hwang, F-K., Chiou, G-L., Wu, Y-T., Lee, S.W-Y., Wu, H-K., Wang, C-Y., Tsai, C-C.(2012). Development and implications of technology in reform-based physics laboratories. *Phys. Rev. ST Phys. Educ. Res*. 8, 020113. DOI:https://doi.org/10.1103/PhysRevSTPER.8.020113

Chinn, C. A. and Malhotra, B. A. (2002), Epistemologically authentic inquiry in schools: A theoretical framework for evaluating inquiry tasks. *Sci. Ed.*, 86 (2), 175–218. Doi:10.1002/sce.10001

Davitadze, Z., Partenadze, G., Djincharadze, E. (2016), Graphical visualization of data measurement of programmable microcontroller according to ARDUINO-project example. *East-West Design & Test Symposium (EWDTS), 2016 IEEE.* doi: 10.1109/EWDTS.2016.7807629

Desai, P. (2015*) Python Programming for Arduino*, Packt Publishing

Domin. D.S. (1999). A Review of Laboratory Instruction Styles. *J. Chem. Educ*., 76 (4), p 543. Doi: 10.1021/ed076p543

Faugel, H., & Bobkov, V. (2013). Open source hard- and software: using Arduino boards to keep old hardware running. *Fusion Engineering and Design*, 88(6-8), 1276-1279. doi http://dx.doi.org/10.1016/j.fusengdes.2012.12.005.

Gipps, J. (2002). Data Logging and Inquiry Learning in Science. In Proc. WCCE2001 Australian Topics: Selected Papers from the *Seventh World Conference on Computers in Education*, Copenhagen, Denmark. CRPIT, 8. McDougall, A., Murnane, J. and Chambers, D., Eds. ACS. 31-34.

Grinias, J.P., Whitfield, J.T., Guetschow, E.D., Kennedy, R.T. (2016) An Inexpensive, Open-Source USB Arduino Data Acquisition Device for Chemical Instrumentation. *J.Chem. Educ*., 93 (7), 1316–1319. doi: 10.1021/acs.jchemed.6b00262

Hale, P. (2000). Kinematics and graphs: Students' difficulties and CBLs. *Mathematics Teacher*, 93(5), 414-417.

Haugen A., Moore N. (2014). A model for including Arduino microcontroller programming in the introductory physics lab. arXiv:1407.7613v1 [physics.ed-ph].

Hofstein, A., Lunetta, V. N. (2004), The laboratory in science education: Foundations for the twenty-first century. *Sci. Ed.*, 88 (1), 28–54. Doi:10.1002/sce.10106

Hofstein, A., Mamlok-Naaman, R. (2007). The laboratory in science education: the state of the art. *Chem. Educ. Res. Pract.*, 8 (2), 105-107. Doi: 10.1039/B7RP90003A

Hofstein, A., Walberg, H. J. (1995). Instructional strategies. In B. J. Fraser & H. J. Walberg (Eds.), *Improving science education* (pp. 1–20). Chicago: National Society for the Study of Education.

Huang, B. (2015, June), Open-source Hardware – Microcontrollers and Physics Education – Integrating DIY Sensors and Data Acquisition with Arduino Paper presented at *2015 ASEE Annual Conference & Exposition*, Seattle, Washington. Doi: 10.18260/p.24542

Igelsrud, D., & Leonard, W.H. (Eds.). (1988, May). Labs: What research says about biology laboratory instruction. *The American Biology Teacher*, 50(5), 303–306.

Irinoye, J., Bamidele, E.F., Adetunji, A.A., Awodele, B.A. (2014). Relative Effectiveness of Guided Inquiry and Demontration Methods on Students Performance in Practical Chemistry in Secondary School in Osun State, Nigeria. *Advances in Social Sciences Research Journal*, 2(2), 21-30.

Kelly, G.J., Crawford, T.(1996), Students' interaction with computer representations: Analysis of discourse in laboratory groups, *J. Res. Sci. Teach.* 33, 693.

Kennedy, D., Finn, S. (2000). The use of datalogging in teaching physics and chemistry in second-level schools in Ireland. Report submitted to *The National Centre for Technology in Education*.

Koenka, I.J, Sáiz, J., Hauser, P.C. (2014). Instrumentino: An open-source modular Python framework for controlling Arduino based experimental instruments. *Computer Physics Communications*, 185 (10), 2724–2729. doi: 10.1016/j.cpc.2014.06.007

Koenka, I.J, Sáiz, J., Hauser, P.C. (2015). Instrumentino: An Open-Source Software for Scientific Instruments. *Chimia*, 69 (4), 172–175. doi:10.2533/chimia.2015.172

Kopasz, K., Makra, P., Gingl, Z, (2011). Edaq530: a transparent, open-end and open-source measurement solution in natural science education. *Eur. J. Phys.* 32(2) 491–504. doi:10.1088/0143-0807/32/2/020

Krajcik, J. S., & Layman, J. W. (1992). Microcomputer-Based Laboratories in the Science Classroom. *NARST: Publications – Research Matters – to the Science Teacher*. Ανακτήθηκε στις 27/03/2017 από https://www.narst.org/publications/research/microcomputer.cfm

Kuan, WH., Tseng, CH., Chen, S., Wong, CC. (2016). Development of a Computer-Assisted Instrumentation Curriculum for Physics Students: Using LabVIEW and Arduino Platform. *Journal of Science Education and Technology* 25(3), 427-438. doi:10.1007/s10956-016-9603-y

Kushner, D. (2011). The Making of Arduino. *IEEE Spectrum*, http://spectrum.ieee.org/geek-life/hands-on/the-making-of-arduino.

Lapp, D.A., Cyrus, V.F. (2000). Using Data-Collection Devices to Enhance Students' Understanding. *Mathematics Teacher* 93(6), 504-11.

Linn, M. C. (1997). The role of laboratory in science learning. *The Elementary School Journal*, 97(4), 401–417. DOI: 10.1086/461873

Literacy and Numeracy Secretariat (2013). Special Edition # 32. *Capacity building series*. Ontario: Ontario Ministry of Education ISSN: 19138490.

Lunetta, V.N., Hofstein, A., & Clough, M.P. (2005). Learning and Teaching in the School Science Laboratory: An analysis of research, theory, and practice. In S. K. Abell & N. G. Lederman (Eds.*), Handbook of research on science education* (pp. 393-441). NJ: Lawrence Erlbaum Associates.

Mabbott, G.A. (2014). Teaching Electronics and Laboratory Automation Using Microcontroller Boards. *J. Chem. Educ.*, 91 (9), 1458–1463. DOI: 10.1021/ed4006216

Martín, F.J.F., Llopis, M.V., Rodríguez, J.C.C., González, J.R.B., Blanco, J.M. (2014). Low-cost open-source multifunction data acquisition system for accurate measurements. *Measurement*, 55, 265–271.doi: http://doi.org/10.1016/j.measurement.2014.05.010

McFarlane, A., Friedler, Y., Warwick, P., Chaplain, R. (1995). Developing an understanding of the meaning of line graphs in primary science investigations, using portable computers and data logging software. *Journal of Computers in Mathematics and Science Teaching*, 14(4), 461–480.

McKenzie, D. L. and Padilla, M. J. (1986), The construction and validation of the test of graphing in science (togs). *J. Res. Sci. Teach.*, 23(7) 571–579. doi:10.1002/tea.3660230702

Mellis, D.A., Banzi, M., Cuartielles, D., Igoe, T. (2007). Arduino: An Open Electronics Prototyping Platform. *ACM Conference on Human Factors in Computing Systems*, Apr. 2007.

Mikropoulos, T.A., Bellou, J. (2006). The Unique Features of Educational Virtual Environments. Στο P. Isaias, M. McPherson και F. Banister (επιμ.) Proceedings e-society 2006, *International Association for Development of the Information Society* (σελ 122-128). IADIS.

Nath, J.K., Nath, S., Sarmah, K.C. (2014). Microcontroller Based High Resolution Temperature Monitoring Data Acquisition System with Graphical User Interface. *International Journal of Recent Development in Engineering and Technology*, 2(4).

National Instruments (2013). Measurement Fundamentals series, Sensor Terminology. Ανακτήθηκε στις 9/5/2017 από http://www.ni.com/white-paper/14860/en/.

National Research Council (2005). America's Lab Report: Investigations in High School Science. Washington, *DC:The National Academies Press.* Doi: 10.17226/11311 (https://www.nap.edu/read/11311/chapter/3#22)

Naveenkumar, R., Prasad K. (2013). Low Cost Data Acquisition and Control using Arduino Prototyping Platform and LabVIEW. *International Journal of Science and Research (IJSR),* India, 2(2), 366-369.

Nayyar, A., Puri, V. (2016). A review of Arduino board's, Lilypad's & Arduino shields, *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*, New Delhi, 2016, 1485-1492.

Newton, L. R. (1998). Gathering data: does it make sense. *Journal of Technology for Teacher Education*, 7 (3), 379-394.

Nicolaou, C.T., Nicolaidou, I., Zacharia, Z., Constantinou, C.P. (2007). Enhancing Fourth Graders' Ability to Interpret Graphical Representations through the Use of Microcomputer-Based Labs Implemented within an Inquiry-Based Activity Sequence. *Journal of Computers in Mathematics and Science Teaching*, 26(1), 75-99.

OpenLabTools Prtoject (2017). University of Cambridge. Retrieved from http://openlabtools.eng.cam.ac.uk

Pearce, J. (2014). Open-Source Lab. How to Build Your Own Hardware and Reduce Research Costs. Elsevier. ISBN: 978-0-12-410462-4

Petry, C. A., Pacheco, F.S., Lohmann, D., Correa, G.A., Moura, P. (2016). Project teaching beyond Physics: Integrating Arduino to the laboratory. *Technologies Applied to Electronics Teaching (TAEE).* DOI: 10.1109/TAEE.2016.7528376

Processing (2017). Ανακτήθηκε στις 9/5/2017 από https://processing.org/.

Real, G., Raviola, L., Jauré, M.F., Vitali A.O. (2015). Data acquisition system for didactic laboratories based on open-source hardware and free software. *2015 XVI Workshop on Information Processing and Control (RPIC)*. Cordoba, Argentina, 6-9 Oct. 2015. DOI: 10.1109/RPIC.2015.7497130

Reas, C., Make, B.F. (2015). *Getting Started with Processing*, Second Edition, Published September 2015, Maker Media.

Rosenberg, J., Cuff, K. (2012). Low-cost sensing to teach energy for everyone. *Lat. Am. J. Phys. Educ.* 6(1), 39-43.

Russell, D. (2002). Computers in physics instruction: Students' interactions in a constructivist microcomputer-based laboratory. Doctoral thesis, Queensland University of Technology, Australia.

Russell, D.W., Lucas, K.B., & McRobbie, C.J.(2004). Role of the microcomputer-based laboratory display in supporting the construction of new understandings in thermal physics. *Journal of Research in Science Teaching*, 41(2), 165-185.

Saraswati, T., Mani, C. (2015). Smart Real Time Embedded Arduino Based Data Acquisition System. *International Journal of Research in Engineering and Technology*, 4(8), eISSN: 2319-1163 | pISSN: 2321-7308

Sesen, B.A., Tarhan, L. (2013). Inquiry-Based Laboratory Activities in Electrochemistry: High School Students' Achievements and Attitudes. *Res Sci Educ* 43(1), 413-435. doi:10.1007/s11165-011-9275-9

Simić, M. (2014). Design of Monitoring and Data Acquisition System for Environmental Sensors. *X International Symposium on Industrial Electronics INDEL 2014*, Banja Luka, November 06-08, 2014

Szott, A. (2014). Open-ended Laboratory Investigations in a High School Physics Course: The difficulties and rewards of implementing inquiry-based learning in a physics lab. *The Physics Teacher* 52 (1), 17-21. Doi: 10.1119/1.4849147

Tho, S.W., Hussain, B. (2011). The development of a Microcomputer-Based Laboratory (MBL) system for gas pressure law experiment via open source software. *International Journal of Education and Development using Information and Communication Technology* (*IJEDICT)*, 7(1), 42-55.

Thornton, R. K. (1987). Tools for scientific thinking - microcomputer-based laboratories for teaching physics. *Phys Ed*, 22, 230–238.

Thornton, R.K., Sokoloff, D.R., (1990), Learning motion concepts using real-time microcomputer-based laboratory tools, *Am. J. Phys*.,58, 858-867.

Tobin, K. (1990), Research on Science Laboratory Activities: In Pursuit of Better Questions and Answers to Improve Learning. *School Science and Mathematics*, 90(5), 403–418. doi:10.1111/j.1949-8594.1990.tb17229.x

Urban, P.L. (2014). Open-Source Electronics As a Technological Aid in Chemical Education. *J. Chem. Educ.*, 91 (5), 751–752. DOI: 10.1021/ed4009073

Wheat, D. (2011) *Arduino Internals*. Apress, New York, NY, USA.

Zachariadou, K., Yiasemides, K., Trougkakos, N. (2012). A low-cost computer-controlled Arduino-based educational laboratory system for teaching the fundamentals of photovoltaic cells. *Eur. J. Phys.* 33(6), 1599–1610. doi:10.1088/0143-0807/33/6/1599

Zieris, H., Gerstberger, H., Müller, W., (2014). Using Arduino-Based Experiments to Integrate Computer Science Education and Natural Science. *Conference: KEYCIT 2014 – Key Competencies in Informatics and ICT*, 238-243. Potsdam.

Zion, M., Mendelovici, R. (2012). Moving from structured to open inquiry: Challenges and limits. *Science Education International*, 23(4), 383-399.

Βαμβακούσης, Χ., Μακρυωνίτης, Γ. (2003). Σύστημα Συγχρονικής λήψης και απεικόνισης. Ένας χρόνος παρουσίας στα εργαστήρια Φ.Ε. των Ενιαίων Λυκείων. *2ο Συνέδριο στη Σύρο, ΤΠΕ στην Εκπαίδευση.*

Γεωργόπουλος, Κ., Μπέλλου, Ι., Κώτσης, Κ., & Μικρόπουλος, Τ. Α. (2010). Αναπαραστάσεις φυσικών μεγεθών και αντίστοιχα κινηματικά φαινόμενα. *Θέματα Επιστημών και Τεχνολογίας στην Εκπαίδευση*, 3(2), 69-84.

Κώτσης, Κ. (2005). *Διδασκαλία της Φυσικής και Πείραμα*. Πανεπιστήμιο Ιωαννίνων. ISBN: 960-233-167-4.

Μικρόπουλος, Τ.Α. (2006). *Ο υπολογιστής ως γνωστικό εργαλείο*. Αθήνα: Ελληνικά Γράμματα.

Μπισδούνης, Λ. (2009). Τεχνολογία Μετρήσεων, Σημειώσεις Διδασκαλίας. Τ.Ε.Ι. Πάτρας-Τμήμα Ηλεκτρολογίας. Πάτρα 2009

Νούσης, Β., Νούση, Β. (2013). Ο Arduino στο Εργαστήριο Φυσικών Επιστημών.*5ο Συνέδριο «Η Πληροφορική στην Εκπαίδευση (CIE2013)*», Πειραιάς, Οκτώβριος 2012.

Ντούλας Σ., Νίκας Θ. (2017). Ο βαθμός ένταξης του συστήματος συγχρονικής λήψης και απεικόνισης (Multilog) στο σχολικό εργαστήριο Φ.Ε. Έρευνα, Προβληματισμοί, Προτάσεις. *10ο Πανελλήνιο Συνέδριο Διδακτικής των Φυσικών Επιστημών και νέων τεχνολογιών στην εκπαίδευση «Γεφυρώνοντας το χάσμα μεταξύ Φυσικών Επιστημών, Κοινωνίας και Εκπαιδευτικής Πράξης»*, Ρέθυμνο, 7-9 Απριλίου 2017.

Πάλλας, Α., Ορφανάκης, Σ. (2016). Η αξιοποίηση των αισθητήρων του Arduino στις εργαστηριακές και ερευνητικές δραστηριότητες. *4ο Πανελλήνιο Εκπαιδευτικό Συνέδριο Κεντρικής Μακεδονίας «Αξιοποίηση των Τ.Π.Ε. στη Διδακτική Πράξη*».

Ραγιαδάκος, Χ., Γράψας, Ι., Κουμαράς., Π., Μπουρμπουχάκης, Ι., Παπασταματίου, Ν. (2008). Κριτήρια επιμόρφωσης-πιστοποίησης υπευθύνων Εργαστηριακών Κέντρων Φυσικών Επιστημών (ΕΚΦΕ) για την ενεργή αξιοποίηση των εργαστηριακών διδακτικών εργαλείων στο Ενιαίο Λύκειο, ΤΕΕ, Γυμνάσιο και Δημοτικό. Οργανισμός Επιμόρφωσης Εκπαιδευτικών (ΟΕΠΕΚ). Ανακτήθηκε στις 27/03/2017 από http://repository.edulll.gr/1049

ΠΑΡΑΡΤΗΜΑ Α

Προγράμματα (sketches) για τους αισθητήρες

Αναλογικός Αισθητήρας Θερμοκρασίας TMP36

```
/* Αναλογικός αισθητήρας Θερμοκρασίας TMP36
 * https://learn.adafruit.com/tmp36-temperature-sensor/using-a-temp-sensor
 */

// απαιτούμενες βιβλιοθήκες
// https://github.com/PaulStoffregen/MsTimer2
#include <MsTimer2.h>

//δήλωση σταθερών
#define SAMPLES 100
#define INTERVAL 1000
#define arefVoltage 3.3 //συνδέστε 3.3 V στον ακροδέκτη AREF
//ο αναλογικός ακροδέκτης στον οποίο συνδέεται
//ο ακροδέκτης εξόδου τάσης (Vout) του αισθητήρα TMP36
#define sensorPin 0

//δήλωση μεταβλητών
float samples_temp[SAMPLES];
float tempc=0.0;

//η συνάρτηση που καλείται κάθε 1000 ms
void temperature_c()
{
 interrupts(); //ενεργοποίηση των διακοπών
   /* Ελέγχει αν υπάρχουν δεδομένα που έχουν ληφθεί ή
    * έχουν ήδη αποθηκευτεί στο Serial Receive Buffer της σειριακής θύρας.
    */
 if (Serial.available()>0){
   //άδειασμα της Serial Receive Buffer
   serialFlush();

   for (int i=0; i <= SAMPLES-1; i++) //συλλογή 100 μετρήσεων
   {
     /* ο αναλογοψηφιακός μετατροπέας (ADC) διαβάζει
      * την τάση που έρχεται από τον αισθητήρα
      * και τη μετατρέπει σε ψηφιακή με ανάλυση 10 bit (1024 στάθμες)
      */
     int sensorVal = analogRead (sensorPin);

     //μετατροπή της τιμής εξόδου του ADC σε τάση, που βασίζεται στην τάση αναφοράς
     float voltage = (sensorVal / 1023.0) * arefVoltage;

      /* μετατροπή της τάσης σε θερμοκρασία (βαθμούς Κελσίου)
       * ο συντελεστής μετατροπής είναι 10 mV / βαθμό Κελσίου με
       * 500 mV μετατόπιση (offset) ώστε να μετρά και αρνητικές τιμές θερμοκρασίας
       */
     float temperature = (voltage - 0.5) * 100;
```

```
      samples_temp[i]=temperature;
      tempc += samples_temp[i];
      delayMicroseconds(9500); //καθυστέρηση 9500 μs
     }
   tempc /= SAMPLES; //υπολογισμός του ΜΟ για καλύτερη ακρίβεια

/* Οι τιμές που στέλνονται στη σειριακή θύρα χωρισμένες με (,)
 *  η περίοδος που έχει οριστεί στη συνάρτηση set της MsTimer2
 *  ο μέσος όρος με ακρίβεια δύο δεκαδικών ψηφίων
 */
 Serial.print(INTERVAL);
 Serial.print(" , ");
 Serial.println(tempc);

   tempc=0.0;
   }
}
void setup() {
  //για να αλλάξει η τάση αναφοράς ARef σε κάτι διαφορετικό από 5V
  analogReference(EXTERNAL);
  Serial.begin(115200); // Εκκίνηση της σειριακής επικοινωνίας με τον υπολογιστή

  //ρύθμιση της περιόδου και της συνάρτησης που θα καλείται
  MsTimer2::set(INTERVAL, temperature_c);
  MsTimer2::start(); //ενεργοποίηση της διακοπής
}

void loop() {
  }


//συνάρτηση που αδειάζει τη Serial Receive Buffer
void serialFlush(){
  while(Serial.available() > 0)
    Serial.read();
  }
```

Αναλογικός Αισθητήρας Θερμοκρασίας σειράς LM35

```
/* Αναλογικός αισθητήρας Θερμοκρασίας LM35
 * http://playground.arduino.cc/Main/LM35HigherResolution
 */

// απαιτούμενες βιβλιοθήκες
// https://github.com/PaulStoffregen/MsTimer2
#include <MsTimer2.h>

//δήλωση σταθερών
#define SAMPLES 100
#define INTERVAL 1000

//ο αναλογικός ακροδέκτης στον οποίο συνδέεται
//ο ακροδέκτης εξόδου τάσης (Vout) του αισθητήρα TMP36
#define sensorPin 0

//δήλωση μεταβλητών
float samples_temp[SAMPLES];
float tempc=0.0;

//η συνάρτηση που καλείται κάθε 1000 ms
void temperature_c()
{
 interrupts(); //ενεργοποίηση των διακοπών
   /* Ελέγχει αν υπάρχουν δεδομένα που έχουν ληφθεί ή
    * έχουν ήδη αποθηκευτεί στο Serial Receive Buffer της σειριακής θύρας.
    */
 if (Serial.available()>0) {
   //άδειασμα της Serial Receive Buffer
   serialFlush();

   for (int i=0; i < SAMPLES; i++) //συλλογή 100 μετρήσεων
   {
     /* ο αναλογοψηφιακός μετατροπέας (ADC) διαβάζει
      * την τάση που έρχεται από τον αισθητήρα
      * και τη μετατρέπει σε ψηφιακή με ανάλυση 10 bit (1024 στάθμες)
      */
     int sensorVal = analogRead (sensorPin);

     // μετατροπή της τιμής εξόδου του ADC σε τάση, με βάση την τάση αναφοράς 1.1 V
     float voltage = (sensorVal / 1023.0) * 1.1;

      /* μετατροπή της τάσης σε θερμοκρασία (βαθμούς Κελσίου)
       * ο συντελεστής μετατροπής είναι 10 mV / βαθμό Κελσίου
       */
     float temperature = voltage * 100;

     samples_temp[i]=temperature;
     tempc += samples_temp[i];
     delayMicroseconds(9500); //καθυστέρηση 9500 μs
    }
  tempc /= SAMPLES; //υπολογισμός του ΜΟ για καλύτερη ακρίβεια

/* Οι τιμές που στέλνονται στη σειριακή θύρα χωρισμένες με (,)
 *  η περίοδος που έχει οριστεί στη συνάρτηση set της MsTimer2
 *  ο μέσος όρος με ακρίβεια δύο δεκαδικών ψηφίων
 */
 Serial.print(INTERVAL);
 Serial.print(" , ");
```

```
 Serial.println(tempc);

  tempc=0.0;
  }
}
void setup() {
  //αλλαγή της ενσωματωμένης τάσης αναφοράς σε 1.1 V
  analogReference(INTERNAL);
  Serial.begin(115200); // Εκκίνηση της σειριακής επικοινωνίας με τον υπολογιστή

  //ρύθμιση της περιόδου και της συνάρτησης που θα καλείται
  MsTimer2::set(INTERVAL, temperature_c);
  MsTimer2::start(); //ενεργοποίηση της διακοπής
}

void loop() {
  }

//συνάρτηση που αδειάζει τη Serial Receive Buffer
void serialFlush(){
  while(Serial.available() > 0)
    Serial.read();
  }
```

Ψηφιακός Αισθητήρας Θερμοκρασίας 1-wire DS18B20

Για τη σύνδεση πολλαπλών αισθητήρων DS18B20 παράλληλα στον ίδιο 1-Wire δίαυλο, πρέπει να είναι γνωστός ο μοναδικός 64-bit κωδικός του κάθε αισθητήρα DS18B20. Το παρακάτω πρόγραμμα αναζητεί 1-wire συσκευές και εμφανίζει τις διευθύνσεις τους (π.χ. 0x28, 0xC6, 0x6A, 0x87, 0x08, 0x00, 0x00, 0xE5) στη σειριακή οθόνη (Serial Monitor).

```
/* Αυτό το πρόγραμμα αναζητεί 1-wire συσκευές και εμφανίζει τις διευθύνσεις τους
 * στη σειριακή οθόνη ( Serial Monitor) σε format που είναι χρήσιμο στα Arduino
 * sketches.
 * http://www.hacktronics.com/Tutorials/arduino-1-wire-address-finder.html
*/

//απαιτούμενες βιβλιοθήκες
//OneWire - https://github.com/PaulStoffregen/OneWire
#include <OneWire.h>

// Δήλωση σταθερών
#define SENSOR_PIN 2   //συνδέστε την 1-wire συσκευή σας στον ψηφιακό ακροδέκτη 2

//Δήλωση αντικειμένων
// Δημιουργία αντικειμένου 1-Wire
OneWire  ourBus(SENSOR_PIN);

void setup()
{
 Serial.begin(9600);
 discoverOneWireDevices();
}

void loop()
{  }

void discoverOneWireDevices(void) {
 byte i;
 byte present = 0;
 byte data[12];
 byte addr[8];

 Serial.print("Looking for 1-Wire devices...\n\r");
 while(ourBus.search(addr)) {
   Serial.print("\n\r\n\rFound \'1-Wire\' device with address:\n\r");
   for( i = 0; i < 8; i++) {
     Serial.print("0x");
     if (addr[i] < 16) {
       Serial.print('0');
     }
     Serial.print(addr[i], HEX);
     if (i < 7) {
       Serial.print(", ");
     }
   }
   if ( OneWire::crc8( addr, 7) != addr[7]) {
     Serial.print("CRC is not valid!\n\r");
     return;
   }
 }
```

```
 Serial.println();
 Serial.print("Done");
 ourBus.reset_search();
 return;
}
```

Το αποτέλεσμα έχει παρόμοια μορφή:

```
Looking for 1-Wire devices...

Found '1-Wire' device with address:
0x28, 0xC6, 0x6A, 0x87, 0x08, 0x00, 0x00, 0xE5

Found '1-Wire' device with address:
0x28, 0x73, 0x1D, 0x88, 0x08, 0x00, 0x00, 0xAF

Done
```

Μέτρηση της θερμοκρασίας δύο παράλληλων στο ίδιο 1-Wire διάδρομο αισθητήρων:

```
/* Σύνδεση πολλαπλών DS18B20 αισθητήρων θερμοκρασίας
 * παράλληλα στον ίδιο 1-Wire δίαυλο.
 * http://arduino-info.wikispaces.com/Brick-Temperature-DS18B20
 */

// απαιτούμενες βιβλιοθήκες
// https://github.com/PaulStoffregen/OneWire
#include <OneWire.h>
// https://github.com/milesburton/Arduino-Temperature-Control-Library
#include <DallasTemperature.h>
// https://github.com/PaulStoffregen/MsTimer2
#include <MsTimer2.h>

// Δήλωση σταθερών
#define ONE_WIRE_BUS_PIN 2 //συνδέστε τις 1-wire συσκευές σας στον ψηφιακό ακροδέκτη 2
#define INTERVAL 1000

//Δήλωση αντικειμένων
// Δημιουργία στιγμιότυπου oneWire για να επικοινωνήσει με οποιαδήποτε συσκευή OneWire
OneWire oneWire(ONE_WIRE_BUS_PIN);
// Περνάει η αναφορά του στιγμιότυπου oneWire στο Dallas Temperature.
DallasTemperature sensors(&oneWire);

//Δήλωση μεταβλητών
//Δηλώστε τις διευθύνσεις των δικών σας αισθητήρων θερμοκρασίας
DeviceAddress Probe01 = { 0x28, 0xC6, 0x6A, 0x87, 0x08, 0x00, 0x00, 0xE5 };
DeviceAddress Probe02 = { 0x28, 0x73, 0x1D, 0x88, 0x08, 0x00, 0x00, 0xAF };
```

```
void temperature_c()
{
 interrupts(); //ενεργοποίηση των διακοπών
   /* Ελέγχει αν υπάρχουν δεδομένα που έχουν ληφθεί ή
    * έχουν ήδη αποθηκευτεί στο Serial Receive Buffer της σειριακής θύρας.
    */
 if (Serial.available()>0){
  //άδειασμα της Serial Receive Buffer
   serialFlush();

  // Εντολή προς όλες τις συσκευές του διαδρόμου να διαβάσουν τη θερμοκρασία
  sensors.requestTemperatures();

 /* Οι τιμές που στέλνονται στη σειριακή θύρα χωρισμένες με (,)
  * περίοδος (ms) που έχει οριστεί στη συνάρτηση set της MsTimer2
  * τιμή θερμοκρασίας του πρώτου αισθητήρα σε βαθμούς Κελσίου
  * τιμή θερμοκρασίας του δεύτερου αισθητήρα σε βαθμούς Κελσίου
  */
  Serial.print(INTERVAL);
  Serial.print(" , ");
  printTemperature(Probe01);
  Serial.print(" , ");
  printTemperature(Probe02);
  Serial.println();
 }
}

void setup(void)
{
 Serial.begin(115200);
 sensors.begin(); //αρχικοποίηση
 // αν θέλετε να αλλάξετε την ανάλυση του ADC από τα προεπιλεγμένα
 // 12 bit π.χ. σε 10
 //sensors.setResolution(Probe01, 10);
 //sensors.setResolution(Probe02, 10);

  //ρύθμιση της περιόδου που θα καλείται η συνάρτηση temperature_c
  MsTimer2::set(INTERVAL, temperature_c);
  MsTimer2::start(); //ενεργοποίηση της διακοπής
}

void loop()
{}

void printTemperature(DeviceAddress deviceAddress)
{
float tempC = sensors.getTempC(deviceAddress);
  if (tempC != -127.00){
    Serial.print(tempC); //ακρίβεια δύο δεκαδικών ψηφίων
  }}

//συνάρτηση που αδειάζει τη Serial Receive Buffer
void serialFlush(){
  while(Serial.available() > 0)
    Serial.read(); }
```

Ψηφιακός I2C Αισθητήρας Βαρομετρικής πίεσης / θερμοκρασίας BMP180

```
/* Ο  αισθητήρας BMP180 της Bosch είναι ένας
 * ψηφιακός αισθητήρας υψηλής ακρίβειας
 * για τη μέτρηση ατμοσφαιρικής πίεσης και θερμοκρασίας.
 * https://learn.adafruit.com/bmp085/using-the-bmp085-api-v2
 */

//απαιτούμενες βιβλιοθήκες
#include <Wire.h>
//https://github.com/adafruit/Adafruit_Sensor
#include <Adafruit_Sensor.h>
//https://github.com/adafruit/Adafruit_BMP085_Unified
#include <Adafruit_BMP085_U.h>
//https://github.com/PaulStoffregen/MsTimer2
#include <MsTimer2.h>

//Δήλωση αντικειμένου
Adafruit_BMP085_Unified bmp = Adafruit_BMP085_Unified(10085);

//Δήλωση σταθερών
#define SAMPLES 32
#define INTERVAL 1000

//Δήλωση μεταβλητών
float samples_pressure[SAMPLES];
float samples_temp[SAMPLES];
float pressurepha=0.0;
float tempc=0.0;

void pressure_temperature()
{
  interrupts(); //ενεργοποίηση των διακοπών
  /* Ελέγχει αν υπάρχουν δεδομένα που έχουν ληφθεί ή
    * έχουν ήδη αποθηκευτεί στο Serial Receive Buffer της σειριακής θύρας.
    */
  if (Serial.available()>0){
   //άδειασμα της Serial Receive Buffer
   serialFlush();

   for (int i=0; i <= SAMPLES-1; i++)
   {
         /* Νέο συμβάν αισθητήρα */
         sensors_event_t event;
         bmp.getEvent(&event);
         if (event.pressure)
          {
             samples_pressure[i] =event.pressure;
             pressurepha += samples_pressure[i];
             float temperature;
             bmp.getTemperature(&temperature);
             samples_temp[i] =temperature;
             tempc += samples_temp[i];
          }
```

```
    }
    pressurepha /= SAMPLES;
    tempc /= SAMPLES; // καλύτερη ακρίβεια
    Serial.print(INTERVAL);
    Serial.print(" , ");
    Serial.print(pressurepha,1);
    Serial.print(" , ");
    Serial.println(tempc,1);

    pressurepha=0.0;
    tempc=0.0;
   } }

void setup() {
  Serial.begin(115200);
  /* μπορούν να χρησιμοποιηθούν τα εξής mode:
    * BMP085_MODE_ULTRALOWPOWER
    * BMP085_MODE_STANDARD
    * BMP085_MODE_HIGHRES
    * BMP085_MODE_ULTRAHIGHRES
    */
  bmp.begin(BMP085_MODE_HIGHRES); //αρχικοποίηση
  MsTimer2::set(INTERVAL, pressure_temperature);
  MsTimer2::start();
}
void loop() {
    }

//συνάρτηση που αδειάζει τη Serial Receive Buffer
void serialFlush(){
  while(Serial.available() > 0)
    Serial.read();
  }
```

Αισθητήρας Υπερήχων HC-SR04 για τον υπολογισμό απόστασης

```
/* Αισθητήρας Υπερήχων HC-SR04 για τον υπολογισμό απόστασης
 * Υπολογίζει την απόσταση στέλνοντας ένα παλμό και μετρώντας
 * το χρονικό διάστημα που αυτός έκανε για να επιστρέψει.
 */

//απαιτούμενες βιβλιοθήκες
//http://playground.arduino.cc/Code/NewPing
#include <NewPing.h>
//https://github.com/PaulStoffregen/MsTimer2
#include <MsTimer2.h>

// Δήλωση σταθερών
#define INTERVAL 200
//ο ακροδέκτης του Arduino που συνδέεται στον ακροδέκτη
//trigger του αισθητήρα υπερήχων.
#define TRIGGER_PIN 12
//ο ακροδέκτης του Arduino που συνδέεται στον ακροδέκτη
//echo του αισθητήρα υπερήχων.
#define ECHO_PIN    11
#define MAX_DISTANCE 100 //η μέγιστη απόσταση που θέλουμε
//ο αισθητήρας να κάνει ping (σε εκατοστά).
//Η απόσταση που μπορεί να υπολογίσει είναι από 2cm έως 400cm.

//Δήλωση αντικειμένων
NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE);

//Δήλωση μεταβλητών
unsigned long Time0;
unsigned int uS0;

void distance_velocity(){
  interrupts();
    /* Ελέγχει αν υπάρχουν δεδομένα που έχουν ληφθεί ή
    * έχουν ήδη αποθηκευτεί στο Serial Receive Buffer της σειριακής θύρας.
    */
  if (Serial.available()>0){
   //άδειασμα της Serial Receive Buffer
   serialFlush();

   unsigned long Time=micros();//σημειώνεται η χρονική στιγμή της μέτρησης
   unsigned int uS = sonar.ping_cm(); //υπολογισμός της απόστασης σε cm
     //υπολογισμός της στιγμιαίας ταχύτητας σε m/sec
   float dt=(Time-Time0)/1000.0;
   int diafora=(int)(uS-uS0);
   float velocity=(diafora/dt)*10;

   if (uS <= MAX_DISTANCE && uS >= 2)
   {
     Serial.print(INTERVAL);
     Serial.print(", ");
     Serial.print(uS);
     Serial.print(", ");
     Serial.println(velocity);
```

```
     uS0=uS;
    }

Time0=Time;
  }
}

void setup() {
   Serial.begin(115200);//Εκκίνηση της σειριακής επικοινωνίας
   MsTimer2::set(INTERVAL, distance_velocity);
   MsTimer2::start();
   uS0 = sonar.ping_cm(); //υπολογισμός της αρχικής απόστασης σε cm
   Time0=micros(); //σημειώνεται η χρονική στιγμή της μέτρησης
}

void loop() {
   }

//συνάρτηση που αδειάζει τη Serial Receive Buffer
void serialFlush(){
  while(Serial.available() > 0)
    Serial.read();
  }
```

ΠΑΡΑΡΤΗΜΑ Β

Κώδικας της εφαρμογής «libreMBL-GUI»

```
1  #!/usr/bin/env python
2  # -*- coding: utf-8 -*-
3
4  # sys module
5  # https://docs.python.org/2/library/sys.html
6  # for command-line arguments
7  import sys
8
9  # os module
10 # https://docs.python.org/2/library/os.html
11 import os
12
13 # pyserial library
14 # https://pypi.python.org/pypi/pyserial
15 # Copyright (c) 2001-2015 Chris Liechti <cliechti@gmx.net> All Rights Reserved.
16 # license: https://pythonhosted.org/pyserial/appendix.html#license
17 import serial
18
19 # PyQt4 library
20 # https://www.riverbankcomputing.com/software/pyqt/download
21 # Copyright (c) 2015 Riverbank Computing Limited
22 # licensed under GNU GPL v3
23 # http://www.gnu.org/licenses/gpl-3.0.en.html
24 # QtCore and QtGui contains GUI widgets
25 from PyQt4 import QtGui, QtCore
26
27 #Load a Qt Designer .ui file and returns an instance of the user interface
28 from PyQt4.uic import loadUi
29
30 # matplotlib library
31 # Copyright (c) 2012 - 2016 The Matplotlib development team
32 # license: https://matplotlib.org/users/license.html
33 # Matplotlib Figure object
34 from matplotlib.figure import Figure
35
36 # import the Qt4Agg FigureCanvas object, that binds Figure to
37 # Qt4Agg backend. It also inherits from QWidget
38 # import the NavigationToolbar Qt4 widget
39 from matplotlib.backends.backend_qt4agg import (
40 FigureCanvasQTAgg as FigureCanvas,
41 NavigationToolbar2QT as NavigationToolbar)
42
43 # threading module
44 # https://docs.python.org/2/library/threading.html
45 # constructs higher-level threading interfaces
46 # on top of the lower level thread module
47 import threading
48 import time
49
50 # global variables
51 lastReceived=[]
```

```
52 stime=[]
53 ylabel=[' ']*3
54
55 class Main(QtGui.QMainWindow):
56     def __init__(self, ):
57         super(Main, self).__init__()
58         loadUi(os.path.join(os.path.dirname(__file__),'libreMBL_ui.ui'), self)
59         self.setWindowTitle(u'Γραφική Αναπαράσταση Δεδομένων σε Πραγματικό
   Χρόνο')
60         self.initUI()
61
62     def initUI(self):
63         self.dc = DynamicMplCanvas()
64         self.mplvl.addWidget(self.dc)
65         self.toolbar = NavigationToolbar(self.dc,
66                 self.mplwindow, coordinates=True)
67         self.mplvl.addWidget(self.toolbar)
68
69         self.sensorvalues_lbl.setStyleSheet('color: darkred')
70         self.sensorvalues.setStyleSheet('color: darkred')
71         self.timeEdit.setDisplayFormat("hh:mm:ss.zzz")
72
73         self.ok_btn.clicked.connect(self.portBaudIn)
74         self.start_btn.clicked.connect(self.onStartButtonPress)
75         self.reset_btn.clicked.connect(self.onResetButtonPress)
76         self.save_btn.clicked.connect(self.onSaveButtonPress)
77         self.sensors_cBox.currentIndexChanged[int].connect(
   self.onComboBoxCurrentIndexChanged)
78         self.save_btn.hide()
79         # object variables
80         self.running=False
81         self.oldPoints=''
82         self.xyPoints=[]
83         self.clickevent=False
84
85         self.onResetButtonPress()
86
87     def portBaudIn(self):
88         locations=['/dev/ttyACM0','/dev/ttyACM1','/dev/ttyACM2', '/dev/ttyACM3',
89            '/dev/ttyACM4', '/dev/ttyACM5','/dev/ttyUSB0','/dev/ttyUSB1',
90            '/dev/ttyUSB2','/dev/ttyUSB3', '/dev/ttyUSB4', '/dev/ttyUSB5',
91            '/dev/ttyUSB6', '/dev/ttyUSB7', '/dev/ttyUSB8', '/dev/ttyUSB9',
92            '/dev/ttyUSB10','/dev/ttyS0', '/dev/ttyS1', '/dev/ttyS2', 'com2',
93            'com3', 'com4', 'com5', 'com6', 'com7', 'com8', 'com9', 'com10',
94            'com11', 'com12', 'com13', 'com14', 'com15', 'com16', 'com17',
95            'com18', 'com19', 'com20', 'com21', 'com1', 'end']
96         if hasattr(self, 'ser'):
97             try:
98                 self.ser.reset_input_buffer()
99                 self.ser.reset_output_buffer()
100                self.ser.close()
101            except:
102                pass
103        for device in locations:
104            try:
```

```
105                 self.ser = serial.Serial(device,int(self.baud_lineEdit.text()),
    timeout=2)
106                 self.statusbar.showMessage(u'Θύρα επικοινωνίας:
    '+self.ser.name.upper())
107                 self.ser.reset_input_buffer()
108                 break
109             except:
110                 if device == 'end':
111                     warn = QtGui.QMessageBox()
112                     warn.setWindowTitle(u'Προειδοποίηση')
113                     warn.setText(u'Δεν είναι εφικτή η σειριακή επικοινωνία.
    Έχετε συνδέσει το καλώδιο;')
114                     warn.exec_()
115                     return
116         self.sensors_lbl.show()
117         self.sensors_cBox.show()
118         self.sensors_cBox.setEnabled(True)
119 
120     def itemsHide (self):
121         self.sec_lbl.hide()
122         self.radiosec.hide()
123         self.radiomsec.hide()
124         self.start_btn.hide()
125         self.line_1.hide()
126         self.line_3.hide()
127         self.sensorvalues_lbl.hide()
128         self.timeEdit.hide()
129         self.sensorvalues.hide()
130 
131     def itemsShow (self):
132         self.sec_lbl.show()
133         self.radiosec.show()
134         self.radiomsec.show()
135         self.start_btn.show()
136         self.line_1.show()
137         self.line_3.show()
138         self.sensorvalues_lbl.show()
139         self.timeEdit.show()
140         self.sensorvalues.show()
141 
142     def onComboBoxCurrentIndexChanged(self):
143         self.itemsShow()
144         i=int(self.sensors_cBox.currentIndex())
145         self.value1_lbl.hide()
146         self.value1_lineEdit.hide()
147         self.value1_lineEdit.setText('')
148         self.value2_lbl.hide()
149         self.value2_lineEdit.hide()
150         self.value2_lineEdit.setText('')
151         self.value3_lbl.hide()
152         self.value3_lineEdit.hide()
153         self.value3_lineEdit.setText('')
154         if i >= 1:
155             self.value1_lbl.show()
156             self.value1_lineEdit.show()
157         if i >= 2:
```

```
158             self.value2_lbl.show()
159             self.value2_lineEdit.show()
160         if i >= 3:
161             self.value3_lbl.show()
162             self.value3_lineEdit.show()
163         self.dc.initFigure(i)
164 
165     def receiving(self):
166         global lastReceived
167         mybuffer = ''
168 
169         while self.running:
170             self.ser.write('*')
171             while (self.ser.in_waiting==0):
172                 pass #do nothing
173 
174             mybuffer = mybuffer + self.ser.read(self.ser.in_waiting)
175 
176             if '\n' in mybuffer:
177                 # Guaranteed to have at least 2 entries
178                 lines = mybuffer.split('\n')
179                 mybuffer = lines[-1]
180                 lastReceived=lines[-2].split(',')
181                 self.dataToGui()
182 
183     def dataToGui(self):
184         global lastReceived
185 
186         values=','.join(lastReceived[1:])
187         self.sensorvalues.setText(values)
188         ctime=str(QtCore.QTime.currentTime().toString('hh:mm:ss'))
189         self.timeEdit.setTime(QtCore.QTime.currentTime())
190 
191         if self.radiosec.isChecked():
192             multiplier=0.001
193         else:
194             multiplier=1
195         if (stime):
196             stime.append(int(lastReceived[0])*multiplier+stime[-1])
197             stime.pop(0)
198         else:
199             stime.append(int(lastReceived[0])*multiplier)
200             stime.append(int(lastReceived[0])*multiplier-stime[0])
201             stime.pop(0)
202 
203         self.myList.append(','.join((ctime, str(stime[-1]), values)))
204 
205         i=int(self.sensors_cBox.currentIndex())
206         if i >= 1:
207             ylabel[0]=unicode(self.value1_lineEdit.text())
208         if i >= 2:
209             ylabel[1]=unicode(self.value2_lineEdit.text())
210         if i >= 3:
211             ylabel[2]=unicode(self.value3_lineEdit.text())
212         self.dc.updateFigure(i)
213 
```

```
214     def onGraphPointsClick(self, event):
215         if event.button==1 and self.clickevent:
216             if self.toolbar.mode!='':
217                 self.statusbar.showMessage(u'Η Εργαλειοθήκη Διαδραστικής
    Πλοήγησης είναι σε λειτουργία {:s}.'.format(self.toolbar.mode))
218             else:
219                 if (event.xdata and event.ydata):
220                     self.xyPoints.append('('+str('%.1f'%(event.xdata))+" ,
    "+str('%.2f'%(event.ydata))+')')
221                     self.oldPoints=self.oldPoints+"   "+self.xyPoints[-1]
222                     self.statusbar.showMessage(self.oldPoints)
223                 else:
224                     self.oldPoints=''
225                     self.statusbar.showMessage(self.oldPoints)
226 
227     def startPause(self, state):
228         if state and int(self.sensors_cBox.currentIndex()) < 1:
229             warn = QtGui.QMessageBox()
230             warn.setWindowTitle(u'Προειδοποίηση')
231             warn.setText(u'Παρακαλώ επιλέξτε τον αριθμό των φυσικών μεγεθών που
    ανιχνεύουν οι αισθητήρες σας.')
232             warn.exec_()
233             return
234 
235         self.running = state
236         if self.running:
237             self.start_btn.setText(u'Παύση')
238             self.clickevent=False
239             self.comt=threading.Thread(target = self.receiving)
240             self.comt.start()
241         else:
242             if hasattr(self, 'comt'):
243                 self.comt.join(2)
244             self.start_btn.setText(u'Συνέχιση')
245             self.clickevent=True
246             self.dc.mpl_connect('button_press_event', self.onGraphPointsClick)
247 
248             if hasattr(self, 'ser'):
249                 try:
250                     self.ser.reset_input_buffer()
251                     self.ser.reset_output_buffer()
252                 except:
253                     pass
254 
255         self.sensors_cBox.setEnabled(self.start_btn.text == u'Έναρξη')
256         self.save_btn.setEnabled(not self.running)
257         self.reset_btn.setEnabled(not self.running)
258 
259     def onStartButtonPress(self):
260         self.oldPoints=''
261         del self.xyPoints[:]
262 
263         self.statusbar.showMessage(u'Θύρα επικοινωνίας: '+self.ser.name.upper())
264         self.save_btn.show()
265 
266         self.startPause(not self.running)
```

```
267
268
269     def onSaveButtonPress(self):
270         #Return the name of the encoding used to convert Unicode filenames into
    system file names
271         encoding = sys.getfilesystemencoding()
272         directory = os.path.dirname(os.path.realpath(__file__)) +os.sep + 'data'
273         directory=unicode(directory, encoding)
274         if not os.path.exists(directory):
275             os.makedirs(directory)
276
277         path=directory+os.sep+time.strftime("%Y%m%d-%H%M%S")+'.txt'
278         name = QtGui.QFileDialog.getSaveFileName(self, 'Save File', path, filter
    ='txt (*.txt *.)')
279         if (name):
280             myfile = open(unicode(name.toUtf8(), encoding="UTF-8"),'w')
281             myylabel=','.join(ylabel)
282             myheader=','.join((u'Τρέχων Χρόνος', u'Χρονική Στιγμή (sec)' if
    self.radiosec.isChecked() else u'Χρονική Στιγμή (msec)', myylabel))
283             myfile.write(myheader.encode("utf-8")+'\n')
284             for p in self.myList:
285                 myfile.write("%s\n" % p)
286             myfile.close()
287
288
289     def onResetButtonPress(self):
290         ylabel[:]=[' ']*3
291         del stime[:]
292         self.myList=[]
293         del self.xyPoints[:]
294         self.statusbar.showMessage('')
295         self.sensorvalues.setText('')
296
297         self.dc.resetFigure()
298         self.dc.draw()
299
300         self.save_btn.setEnabled(False)
301         self.save_btn.hide()
302         self.sensors_cBox.setEnabled(False)
303         self.value1_lbl.hide()
304         self.value1_lineEdit.hide()
305         self.value2_lbl.hide()
306         self.value2_lineEdit.hide()
307         self.value3_lbl.hide()
308         self.value3_lineEdit.hide()
309         self.sensors_cBox.setCurrentIndex(0)
310         self.timeEdit.setTime(QtCore.QTime(0,0))
311         self.sensors_lbl.hide()
312         self.sensors_cBox.hide()
313         self.itemsHide()
314         self.startPause(False)
315         self.start_btn.setText(u'Εναρξη')
316
317
318     def closeEvent(self, event):
319         print "User has clicked the red x on the main window"
```

```
320         self.onResetButtonPress()
321         event.accept()
322 
323 class DynamicMplCanvas(FigureCanvas):
324 
325     def __init__(self, parent=None):
326         self.fig = Figure()
327         # initialize the canvas where the Figure renders into
328         FigureCanvas.__init__(self, self.fig)
329         # set the parent widget
330         self.setParent(parent)
331         self.resetFigure()
332 
333 
334      def resetFigure(self):
335         self.ydata=[[0 for i in range(100)], [0 for i in range(100)], [0 for i
    in range(100)]]
336         self.t=[0.0]*100
337         self.axes={}
338         self.line={}
339 
340 
341     def initFigure(self, numplot):
342         colors = ['r', 'b', 'g']
343         #Clear the figure
344         self.fig.clf(keep_observers=True)
345 
346         for i in range(1, numplot+1):
347             #The basic Matplotlib Axes initialization
348             self.axes[i] = self.fig.add_subplot(numplot,1, i)
349             self.line[i], = self.axes[i].plot(self.ydata[i-1], self.t, colors[i-
    1]+'o-', markersize=2)
350             self.axes[i].grid(True)
351             self.axes[i].set_xlabel(u'Χρόνος', fontname='Arial' )
352 
353         #Redraw the current figure
354         self.draw()
355 
356 
357     def updateFigure(self, numplot):
358         global lastReceived
359 
360         if len(lastReceived)<4:
361             for i in range(4-len(lastReceived)):
362                 lastReceived.append(0)
363 
364         self.t.append(float(stime[-1]))
365         self.t.pop(0)
366         for i in range(1, numplot+1):
367             self.ydata[i-1].append(float(lastReceived[i]))
368             self.ydata[i-1].pop(0)
369 
370             self.line[i].set_xdata(self.t)
371 
372             self.line[i].set_ydata(self.ydata[i-1])
373 
```

```
374             self.axes[i].set_xlim(min(self.t)-0.01, max(self.t)+0.01)
375 
376                 if (self.ydata[i-1][-1]!=0):
377                     self.axes[i].set_ylim(min([elem for elem in self.ydata[i-1]
    if elem != 0.0])-0.01,max([elem for elem in self.ydata[i-1] if elem !=
    0.0])+0.01)
378                 self.axes[i].relim(); self.axes[i].autoscale_view()
379                 self.axes[i].set_ylabel(ylabel[i-1], fontname='Arial')
380                 self.axes[i].ticklabel_format(useOffset=False, style='plain')
381 
382             self.draw()
383 
384 
385 if __name__ == '__main__':
386 
387     # Create the GUI application
388     app = QtGui.QApplication(sys.argv)
389     # instantiate the main window
390     main = Main()
391     # show the main window
392     main.show()
393     # start the Qt main loop execution, exiting from this script
394     # with the same return code of Qt application
395     sys.exit(app.exec_())
```