ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΗΛΕΚΤΡΙΚΩΝ ΒΙΟΜΗΧΑΝΙΚΩΝ ΔΙΑΤΑΞΕΩΝ
ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ ΑΠΟΦΑΣΕΩΝ

Διάταξη Μετρήσεων Με Χρήση Διασυνδεδεμένων Μικροελεγκτών

Διπλωματική Εργασία

Γεώργιος Δ. Βλάχος

Επιβλέπων: Παναγιώτης Τσαραμπάρης
Λέκτορας Ε.Μ.Π.

Αθήνα, Μάρτιος 2016

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΗΛΕΚΤΡΙΚΩΝ ΒΙΟΜΗΧΑΝΙΚΩΝ ΔΙΑΤΑΞΕΩΝ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ ΑΠΟΦΑΣΕΩΝ

Διάταξη Μετρήσεων Με Χρήση Διασυνδεδεμένων Μικροελεγκτών

Διπλωματική Εργασία

Γεώργιος Δ. Βλάχος

Επιβλέπων: Παναγιώτης Τσαραμπάρης
Λέκτορας Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 23η Μαρτίου 2016.

(Υπογραφή)	(Υπογραφή)	(Υπογραφή)
.....	.....	.....
Παναγιώτης Τσαραμπάρης	Νικόλαος Θεοδώρου	Μαρία-Παρασκευή Ιωαννίδου
Λέκτορας Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Καθηγήτρια Ε.Μ.Π.

Αθήνα, Μάρτιος 2016

.....

Γεώργιος Δ. Βλάχος

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

Περίληψη

Σκοπός της παρούσας διπλωματικής εργασίας είναι η μελέτη και κατασκευή μίας διάταξης μετρήσεων η οποία θα βασίζεται στην πλατφόρμα Arduino. Το σύστημα μέτρησης αποτελείται από μία εφαρμογή διεπαφής με το χρήστη σε περιβάλλον ηλεκτρονικού υπολογιστή και από μικροελεγκτές Arduino σε κεντρικό ρόλο και ρόλο σταθμών μετρήσεων. Η επικοινωνία βασίζεται στο πρωτόκολλο επικοινωνίας RS485. Επιπλέον χρησιμοποιούνται αισθητήρες μέτρησης θερμοκρασίας για την συλλογή δεδομένων.

Στο πρώτο κεφάλαιο γίνεται μία εισαγωγή στις μετρήσεις και στα συστήματα μετρήσεων. Επιπλέον παρουσιάζονται οι μικροελεγκτές, οι αισθητήρες και το πρωτόκολλο επικοινωνίας RS485, η δομή τους και τα χαρακτηριστικά τους. Ιδιαίτερη έμφαση δίνεται στην πλατφόρμα Arduino, όπου γίνεται εκτενής αναφορά στα χαρακτηριστικά της, στην εσωτερική δομή, στους ακροδέκτες του ολοκληρωμένου κυκλώματος και στην επικοινωνία με το υλικό και το λογισμικό.

Στο δεύτερο κεφάλαιο γίνεται η παρουσίαση της διάταξης μετρήσεων και η αναλυτική επεξήγηση των μερών που την αποτελούν. Ιδιαίτερη αναφορά γίνεται στην εφαρμογή διεπαφής με το χρήστη, από όπου είναι δυνατή η επίβλεψη και επεξεργασία των μετρήσεων.

Τέλος εξάγονται τα συμπεράσματα και οι προτάσεις για μελλοντική επέκταση της παρούσας εργασίας, ενώ στο παράρτημα παρουσιάζεται επιγραμματικά ο κώδικας προγραμματισμού των μικροελεγκτών Arduino.

Λέξεις Κλειδιά

Μέτρηση, Διάταξη μετρήσεων, μικροελεγκτής, Arduino, RS485, αισθητήρας, εφαρμογή, σύστημα μέτρησης, master, slave

Abstract

The aim of this thesis is the design and construction of a measuring device, based on the Arduino platform using the communication protocol RS485. The measurement system consists of an interface implemented by the user in a computer environment and Arduino microcontrollers in both central role and as measuring stations. All communication is done through telephone cable with the RS485 communication protocol. In addition temperature sensors are used for data collection.

The first chapter is an introduction to measurement and measurement systems. Furthermore microcontrollers, sensors and communication protocol RS485, their structure and characteristics are presented. Particular emphasis is given to the Arduino platform, where the characteristics, the internal structure of the integrated circuit terminals and the ways of communication with the hardware and software are reported in detailed analysis.

The second chapter is the presentation of the measuring device and a detailed explanation of its parts. Particularly, the user interface application is explained, where the supervision and analysis of the measurements takes place.

Finally conclusions are exported and recommendations for future extension of this work, while in Annex is briefly presented the code of Arduino microcontrollers.

Key Words

Measurement, measuring device, microcontroller, Arduino, RS485, sensor, application, measurement system, master, slave

Ευχαριστίες

Αρχικά θα ήθελα να ευχαριστήσω θερμά τον επιβλέποντα της εργασίας κ. Παναγιώτη Τσαραμπάρη, Λέκτορα Ε.Μ.Π., για την ανάθεση της εργασίας, την εμπιστοσύνη που μου έδειξε κατά την εκπόνηση της και τις καίριες και πολύτιμες συμβουλές του. Επιπλέον ευχαριστώ τα υπόλοιπα μέλη της τριμελούς επιτροπής εξέτασης, την κα. Μαρία-Παρασκευή Ιωαννίδου, Καθηγήτρια Ε.Μ.Π., και τον κ. Νικόλαο Θεοδώρου, Καθηγητή Ε.Μ.Π. .

Τέλος θα ήθελα να ευχαριστήσω τους φίλους μου, όσους ανθρώπους στάθηκαν δίπλα μου, και ιδιαιτέρως την οικογένειά μου για τη αμέριστη στήριξη και συμπαράσταση που μου έδειξαν στην πορεία μου στο Εθνικό Μετσόβιο Πολυτεχνείο.

Γεώργιος Δ. Βλάχος,

Αθήνα, Μάρτιος 2016

Περιεχόμενα

Εικόνες

Πίνακες

«Οι αριθμοί καθορίζουν την τάξη και την αρμονία στο σύμπαν.»

Πυθαγόρας

Κεφάλαιο 1: Εισαγωγή

1.1 Μετρήσεις

1.1.1 Εισαγωγή

Η μέτρηση αποτελεί το θεμέλιο λίθο της επιστήμης και αναπόσπαστο εργαλείο της επιστημονικής μεθόδου. Συχνά, κατά την ιστορία της επιστήμης, μικρές αλλά ουσιώδεις διαφορές ανάμεσα στην θεωρία και στις ακριβείς επιστημονικές μετρήσεις οδήγησαν στην διατύπωση ακριβέστερων και πληρέστερων θεωριών συμβάλλοντας με αυτό τον τρόπο στην εξέλιξή της.

Από τα αρχαία χρόνια οι άνθρωποι χρησιμοποίησαν διάφορα συστήματα μέτρησης, προκειμένου να μετρήσουν τις αποστάσεις, ποσότητες όπως η μάζα και ο όγκος, για σκοπούς εμπορικούς και για καθαρά πρακτικούς λόγους σχετικούς με την καθημερινότητά τους. Υπάρχει το ελληνικό σύστημα, το βαβυλωνιακό, το αιγυπτιακό, το κινέζικο και πλήθος άλλων συστημάτων. Το 1960 στην 11^{η} Γενική Διάσκεψη Μέτρων και Σταθμών υιοθετήθηκε το όνομα Systeme International d'Unites[SI] για το διεθνές σύστημα μονάδων μέτρησης.

Στις οικονομικές, τεχνικές και εμπορικές δραστηριότητες μιας ανεπτυγμένης χώρας δαπανάται για μετρήσεις ποσοστό που ανέρχεται στο 7% του Ακαθάριστου Εθνικού Προϊόντος (ΑΕΠ) της, σύμφωνα με στοιχεία από το Ελληνικό Ινστιτούτο Μετρολογίας. Οι μετρήσεις σχετίζονται με έλεγχο στην ποσότητα των συναλλαγών, στην ποιότητα των προϊόντων, στις διεργασίες παραγωγής, καθώς και με την ασφάλεια της υγείας, της εργασίας και του περιβάλλοντος. Το ποσοστό αυτό αντιστοιχεί σε απασχόληση ανθρώπινου δυναμικού και σε χρήση μετρητικών οργάνων και συσκευών.

Ορισμός της μέτρησης:

- Κλασσικός ορισμός

  Στον κλασσικό ορισμό, ο οποίο είναι κοινός για όλες τις επιστήμες, ως μέτρηση ορίζεται ο προσδιορισμός ή η εκτίμηση του λόγου των ποσοτήτων. Ποσότητα και μέτρηση ορίζονται αμοιβαία: Ποσοτικές ιδιότητες είναι εκείνες που δύνανται να μετρηθούν, τουλάχιστον κατ' αρχήν. Η κλασσική έννοια της ποσότητας μπορεί να αναχθεί στους John Wallis και Isaac Newton, ενώ οι ρίζες της βρίσκονται στα «Στοιχεία» του Ευκλείδη.

- Αναπαραστατική θεωρία

  Στην αναπαραστατική θεωρία ως μέτρηση ορίζεται η συσχέτιση αριθμών με οντότητες που δεν είναι αριθμοί. Η πιο εξελιγμένη τεχνικά μορφή της αναπαραστατικής θεωρίας είναι επίσης γνωστή σαν προσθετική συνδυασμένη μέτρηση. Σε αυτή την μορφή, οι αριθμοί ανατίθενται με βάση τους συσχετισμούς ή τις ομοιότητες μεταξύ της δομής των αριθμητικών

συστημάτων και της δομής των ποσοτικών συστημάτων. Μία ιδιότητα είναι ποσοτικοποιήσιμη εφόσον τέτοιου είδους δομικές ομοιότητες μπορούν να εγκαθιδρυθούν. Σε ασθενέσθερες μορφές της αναπαραστατικής θεωρίας, όπως αυτές που παρουσιάζονται στο έργο του Stanley Smith Stevens, οι αριθμοί μπορούν να ανατίθενται μόνο βάση ενός κανόνα.

Η μέτρηση συχνά παρερμηνεύεται ως η απλή ανάθεση ενός αριθμού, όμως είναι δυνατό να ανατεθεί μία τιμή η οποία να μην καλύπτει τα κριτήρια της προσθετικής συνδυασμένης μέτρησης ώστε να θεωρηθεί μέτρηση. Είναι δυνατό να ανατεθεί μία τιμή για το ύψος ενός ανθρώπου, μέχρι όμως να αποδειχθεί ο συσχετισμός μεταξύ της μέτρησης του ύψους και των τιμών που προκύπτουν εμπειρικά δεν αποτελεί μέτρηση σύμφωνα με τη θεωρία αυτή.

- Θεωρία της πληροφορίας

  Η θεωρία της πληροφορίας αναγνωρίζει όλα τα δεδομένα ως φύσει ανακριβή και στατιστικά. Επομένως η μέτρηση ορίζεται ως μία σειρά από παρατηρήσεις που μειώνουν την αβεβαιότητα και όπου το αποτέλεσμα εκφράζεται ως ποσότητα. Αυτό βρίσκει εφαρμογή στην πράξη των επιστημόνων μετά από μία μέτρηση να αναφέρουν τόσο το μέσο όρο όσο και τα στατιστικά των μετρήσεων. Με πιο απλά λόγια, ο επιστήμονας εκκινεί από μία εκτίμηση σχετικά με την τιμή ενός μεγέθους και στη συνέχεια, χρησιμοποιώντας διάφορες μεθόδους και εργαλεία, προσπαθεί να μειώσει την αβεβαιότητα ανάμεσα σε αυτή την τιμή και την πραγματική τιμή του μεγέθους. Χαρακτηριστικό αυτής της θεωρίας είναι ότι υπάρχει αβεβαιότητα σε κάθε μέτρηση, εκτίμηση και μέτρηση ταυτίζονται, για αυτό το λόγο σε κάθε μέτρηση ανατίθενται ένα σύνολο τιμών αντί μίας ξεχωριστής τιμής.

- Κβαντομηχανική

  Στην κβαντομηχανική ως μέτρηση ορίζεται η πράξη κατά την οποία καθορίζεται μία ιδιότητα του κβαντικού συστήματος. Πριν από τη μέτρηση ένα κβαντικό σύστημα περιγράφεται συγχρόνως από ένα σύνολο τιμών λαμβάνοντας υπόψη την πιθανότητα εμφάνισης των τιμών αυτών σύμφωνα με την κυματοσυνάρτηση του συστήματος. Μετά τη μέτρηση η κυματοσυνάρτηση πιθανότητας του συστήματος "καταρρέει" σε μία μοναδική τιμή. Η ξεκάθαρη ερμηνεία της μέτρησης σε ένα κβαντικό σύστημα αποτελεί θεμελιώδες άλυτο πρόβλημα της κβαντομηχανικής.

Εικόνα 1: Ζυγός μέτρησης βάρους

1.1.2 Μετρολογία

Ως μετρολογία ορίζεται από το Διεθνές Γραφείο Μέτρων και σταθμών (International Bureau of Weights and Measures -IBWM) η επιστήμη της μέτρησης που περιλαμβάνει τόσο πειραματικούς όσο και θεωρητικούς προσδιορισμούς σε κάθε βαθμό αβεβαιότητας σε οποιοδήποτε πεδίο της επιστήμης ή της τεχνολογίας. Περιλαμβάνει τεχνικές και μεθόδους μετρήσεων, την οργανολογία καθώς και τις μονάδες μέτρησης.

Το αντικείμενο της μετρολογίας μπορεί να χωρισθεί σε τρεις επιμέρους δραστηριότητες, αν και όχι με ξεκάθαρο και μη επικαλυπτόμενο τρόπο:

1. Ορισμός διεθνώς αποδεκτών μονάδων μέτρησης.
2. Υλοποίηση των μονάδων αυτών στην πράξη.
3. Σύνδεση των μετρήσεων που λαμβάνουν χώρα με τα πρότυπα αναφοράς.

1.1.3 Συστήματα μονάδων

Κατά καιρούς έχουν χρησιμοποιηθεί διάφορα συστήματα μέτρησης, όπως αναφέρθηκε προηγουμένως, πλέον όμως σε παγκόσμιο επίπεδο αναγνωρίζεται το Διεθνές Σύστημα Μονάδων (SI). Στον πίνακα 1 εμφανίζονται μερικά από τα θεμελιώδη μεγέθη του συστήματος αυτού:

Πίνακας 1: Θεμελιώδη μεγέθη Διεθνούς Συστήματος Μονάδων (SI)

Θεμελιώδες μέγεθος		Θεμελιώδης μονάδα		Χρονιά υιοθέτησης
Όνομα	Σύμβολο	Όνομα	Σύμβολο	
μήκος	*l, x, r, s* κ.λ.π	**μέτρο** metre	**m**	**1983** 17η Σύνοδος
μάζα	*m*	**χιλιόγραμμο** kilogram - kilogramme	**kg**	**1889 και 1901** 1η και 3η Σύνοδος
χρόνος	*t*	**δευτερόλεπτο** second - seconde	**s**	**1967** 13η Σύνοδος
ηλεκτρικό ρεύμα	*I, i*	**αμπέρ** ampere	**A**	**1948** 9η Σύνοδος
Θερμοδυναμική θερμοκρασία	*T*	**κέλβιν** kelvin	**K**	**1967** 13η Σύνοδος
ποσότητα ύλης	*n*	**μολ (γραμμομόριο)** mole	**mol**	**1971** 13η Σύνοδος
φωτεινή ένταση	I_v	**καντήλα** candela	**cd**	**1979** 16η Σύνοδος

Για την καλύτερη χρήση των διαφόρων μεγεθών γίνεται χρήση προθεμάτων ώστε να αποφεύγεται η άσκοπη χρήση υπερβολικά μεγάλων ή μικρών αριθμών. Τα προθέματα αυτά φαίνονται στην εικόνα 2:

Εικόνα 2: Προθέματα μονάδων του Διεθνούς Συστήματος Μονάδων (SI)

1.1.4 Συστήματα Μετρήσεων

Σύστημα μέτρησης καλείται η διάταξη σύμφωνα με την οποία μία ποσοτική έξοδος αντιστοιχίζεται στη μετρούμενη μεταβλητή, μέσω συλλογής των προς μέτρηση μεγεθών και κανόνων που διέπουν τη μεταξύ τους σχέση.

Ιδανικά η τιμή που δίδεται στη μεταβλητή είναι η πραγματική της τιμή, στην πράξη όμως εμφανίζονται αποκλίσεις ανάμεσα στις μετρούμενες και πραγματικές τιμές των μεγεθών. Οι αποκλίσεις αυτές ονομάζονται σφάλματα, επομένως σε κάθε μέτρηση λαμβάνεται η Τιμή±Σφάλμα. Τα σφάλματα διαχωρίζονται σε απόλυτα και σχετικά.

Τα είδη των μετρήσεων διαχωρίζονται σε:

1. Μετρήσεις φυσικών μεγεθών
2. Προσδιορισμός σταθερών
3. Μετρήσεις στατιστικών δεδομένων

1.1.4.1 Βασικά Χαρακτηριστικά

Ένα γενικό σύστημα μετρήσεων αποτελείται από τα εξής επιμέρους στοιχεία:

- Μετρητικά στοιχεία
- Σύστημα προσαρμογής
- Σύστημα μετάδοσης
- Σύστημα επεξεργασίας

Μερικά από τα βασικά χαρακτηριστικά που λαμβάνονται υπόψη είναι τα εξής:

- Ακρίβεια - Accuracy

  Είναι η απόκλιση της τιμής που δίνει το όργανο μετρήσεων από την πραγματική τιμή του προς μέτρηση μεγέθους, η οποία δεν είναι γνωστή, μόνο κατ' εκτίμηση υπολογίζεται, και ορίζεται μέσω κάποιου προτύπου. Σχετίζεται με το μέγιστο εύρος σφαλμάτων στις ενδείξεις του οργάνου. Για να ελαχιστοποιηθεί το σφάλμα είναι απαραίτητο η κλίμακα του οργάνου να είναι κατά το δυνατόν πλησιέστερα στο εύρος του μεγέθους προς μέτρηση.
- Αβεβαιότητα - Uncertainty

  Η αβεβαιότητα είναι ένα μέτρο της αξιοπιστίας των μετρήσεων. Εκφράζει κατά κάποιο τρόπο την αμφιβολία του κατά πόσον το αποτέλεσμα της μέτρησης είναι σωστό. Στη βιβλιογραφία αναφέρεται και ως σφάλμα, χωρίς να ταυτίζεται όμως με την κυριολεκτική έννοια του σφάλματος. Τα σφάλματα κατηγοριοποιούνται σε:
- Τυχαία σφάλματα

  Οφείλονται σε τυχαία μεταβολή παραγόντων και σχετίζονται με την ακρίβεια της μέτρησης. Μπορεί να προκληθούν λόγω της ακρίβειας του οργάνου, λόγω

λαθών που σχετίζονται με τον παρατηρητή ή λόγω εξωτερικών συνθηκών. Είναι αναπόφευκτα και υπολογίζονται με στατιστικές μεθόδους.

- Συστηματικά σφάλματα

  Παραμένουν σταθερά κατά τη λήψη επαναλαμβανόμενων μετρήσεων και σχετίζονται με την αξιοπιστία των μετρήσεων. Τείνουν να μετατοπίζουν τη μέση τιμή των μετρήσεων προς μία συγκεκριμένη διεύθυνση με συστηματικό τρόπο. Ο μόνος τρόπος να διορθωθούν είναι η σύγκριση του χρησιμοποιούμενου οργάνου με το αντίστοιχο πρότυπο. Σε πειράματα μεγάλης έκτασης εξαλείφονται κατά το δυνατόν με τη λήψη επαναλαμβανόμενων μετρήσεων με χρήση διαφορετικών μεθόδων.

  Επομένως για ακρίβεια ο σωστός τρόπος αναγραφής της μετρούμενης τιμής είναι:
  Τιμή ± Σφάλμα ± Συστηματικό σφάλμα

1.1.5 Σήματα

Ως σήμα ορίζεται ένα φυσικό μέγεθος το οποίο μεταβάλλεται σε σχέση με το χρόνο ή το χώρο ή, με οποιαδήποτε άλλη ανεξάρτητη μεταβλητή ή μεταβλητές. Για παράδειγμα η ομιλία μεταδίδεται μέσω σημάτων που παράγονται στις φωνητικές χορδές. Το ηλεκτρικό ρεύμα που διαρρέει τα στοιχεία ενός κυκλώματος αποτελεί ηλεκτρικό σήμα και περιγράφει από τις εξισώσεις κατάστασης του κυκλώματος. Άλλα παραδείγματα αποτελούν τα ιατρικά σήματα(καρδιογράφημα) ή τα σεισμικά. Μαθηματικά το σήμα περιγράφεται ως συνάρτηση μιας ή περισσότερων μεταβλητών. Ανάλογα με το πλήθος των μεταβλητών τα σήματα χαρακτηρίζονται ως μονοδιάστατα, δισδιάστατα ή πολυδιάστατα.

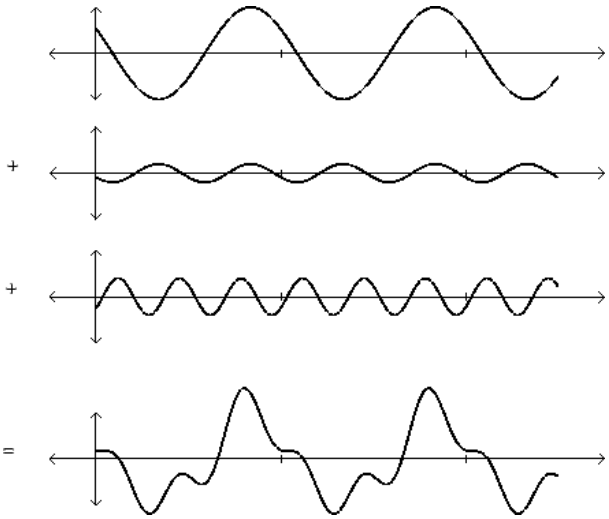
Γενικοί χαρακτηρισμοί των σημάτων:

- Νομοτελειακά - Τυχαία Σήματα

  Είναι τα σήματα εκείνα που είναι δυνατό να περιγραφούν με συναρτησιακές σχέσεις ως προς το χρόνο.
- Αιτιοκρατικά ή Μονόπλευρα Σήματα

  Οι αποκρίσεις των μονόπλευρων σημάτων που οφείλονται στη διέγερση είναι μηδέν μέχρι τη στιγμή που εφαρμόζεται η διέγερση.
- Περιοδικά και μη σήματα

  Περιοδικό είναι ένα σήμα που επαναλαμβάνεται σε τακτά χρονικά διαστήματα τα οποία απέχουν μεταξύ τους σταθερό χρόνο, που ονομάζεται περίοδος του σήματος (βλ. εικόνα 3). Σε αντίθετη περίπτωση είναι μη περιοδικά.



Εικόνα 3: Περιοδικά σήματα

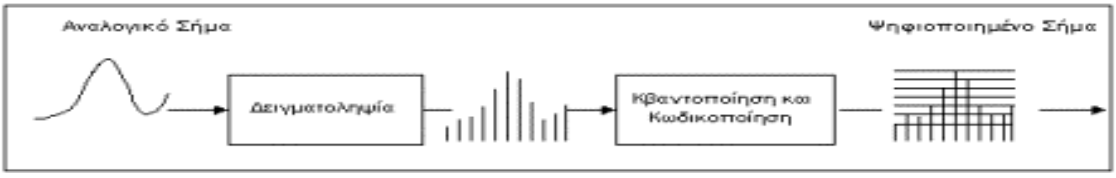
Τα είδη των σημάτων διακρίνονται στα εξής:

- Αναλογικά σήματα

  Αναλογικά ονομάζονται τα σήματα που είναι συνεχή ως προς το χρόνο και ως προς το πλάτος τους. Για παράδειγμα η θερμοκρασία, η πίεση και η ένταση του ήχου είναι αναλογικά μεγέθη. Τα κυκλώματα που επεξεργάζονται μικρά τέτοια σήματα παράγουν στην έξοδο μία τιμή ανάλογη της εισόδου τους και ονομάζονται αναλογικά κυκλώματα. Ένα παράδειγμα τέτοιου κυκλώματος είναι ο τελεστικός ενισχυτής.

- Διακριτά σήματα

  Διακριτά ονομάζονται τα σήματα που είναι διακριτά στο χρόνο και στο πλάτος. Προκύπτουν από το αναλογικό σήμα με τη μέθοδο της δειγματοληψίας(βλ. εικόνα 4).



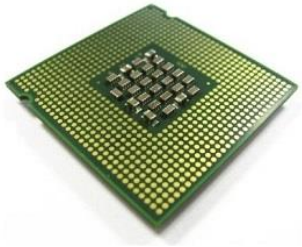
Εικόνα 4: Δειγματοληψία

- Ψηφιακά σήματα

  Ψηφιακά είναι τα σήματα στα οποία τόσο η ανεξάρτητη μεταβλητή όσο και η εξαρτημένη μπορούν να λαμβάνουν μόνο διακριτές τιμές. Σύμφωνα με τη δειγματοληψία που έχουν υποστεί, το πλάτος τους κυμαίνεται ανάλογα με τις

διακριτές κβαντικές στάθμες που έχουν κωδικοποιηθεί με βάση το δυαδικό αριθμητικό σύστημα και τιμές 0 ή 1. Συνήθως η έξοδος των περισσότερων αισθητηρίων είναι σε αναλογική μορφή, όμως για λόγους αξιοπιστίας χρησιμοποιούνται κυρίως οι ψηφιοποιημένες μορφές των σημάτων μέσω ενός ψηφιακού μετατροπέα από αναλογικό σήμα σε ψηφιακό. Τα ψηφιακά συστήματα έχουν τη δυνατότητα επεξεργασίας μεγάλων σημάτων και η λειτουργία τους χαρακτηρίζεται ως μη γραμμική.

1.2 Μικροεπεξεργαστές



Εικόνα 5: Μικροεπεξεργαστής

Από τη πρώτη εμφάνιση του μικροεπεξεργαστή το 1972 μέχρι σήμερα η τεχνολογία των επεξεργαστών έχει παρουσιάσει αλματώδη εξέλιξη. Ενσωματώνοντας εκατομμύρια τρανζίστορς στο ολοκληρωμένο κύκλωμα του μικροεπεξεργαστή καθίσταται πλέον δύσκολος ο σαφής διαχωρισμός ανάμεσα στους μεσαίους υπολογιστές και σε συστήματα που βασίζονται σε μικροεπεξεργαστές.

Ένας μικροεπεξεργαστής περιλαμβάνει τις λειτουργίες μιας κεντρικής μονάδας επεξεργασίας ενός ηλεκτρονικού υπολογιστή σε ένα ενιαίο ολοκληρωμένο κύκλωμα. Ο μικροεπεξεργαστής είναι μία προγραμματιζόμενη συσκευή πολλαπλών χρήσεων η οποία δέχεται ψηφιακά δεδομένα στην είσοδο, τα επεξεργάζεται σύμφωνα με τις οδηγίες που βρίσκονται στην μνήμη της και παρουσιάζει τα αποτελέσματα στην έξοδο. Η ολοκλήρωση μιας κεντρικής μονάδας επεξεργασίας σε ένα μόνο κύκλωμα, καθώς και η δυνατότητα μαζικής παραγωγής, έχουν μειώσει σημαντικά το κόστος με αποτέλεσμα οι μικροεπεξεργαστές να αποτελούν σήμερα αναπόσπαστο κομμάτι της τεχνολογικής προόδου των τελευταίων ετών.

Ένας μικροεπεξεργαστής αποτελείται από τα εξής μέρη:

- Μονάδα αποκωδικοποίησης (Decoding Unit)
  Η μονάδα όπου τα προγράμματα μετατρέπονται σε γλώσσα Assembly.
- Αριθμητική και Λογική Μονάδα (Arithmetical and Logical Unit, ALU)
  Η μονάδα όπου εκτελούνται οι αριθμητικές και λογικές πράξεις που απαιτούνται με βάση τις εντολές του προγράμματος.

- Καταχωρητές (Registers)
  Μικρές μονάδες μνήμης που βρίσκονται στο εσωτερικό του επεξεργαστή, μικρής συνήθως χωρητικότητας και μεγάλης ταχύτητας προσπέλασης, που λειτουργούν επικουρικά στην εκτέλεση των εντολών αποθηκεύοντας προσωρινά τιμές απαραίτητες για την ομαλή διεξαγωγή του προγράμματος.
- Μονάδα Ελέγχου (Control Unit)
  Η μονάδα που ελέγχει τη ροή των δεδομένων από και προς την αριθμητική και λογική μονάδα, τη μνήμη, τους καταχωρητές και τις μονάδες εισόδου - εξόδου.
- Μονάδα Προσκόμισης (Fetch Unit)
  Η μονάδα που μεταφέρει τις εντολές από τη μνήμη στον επεξεργαστή.
- Μονάδα Προστασίας (Protection Unit)
  Η μονάδα που εξασφαλίζει την αποφυγή πράξεων ή εντολών που δύνανται να προκαλέσουν βλάβη στην ομαλή διεξαγωγή του προγράμματος.

1.2.1 Μικροελεγκτές

Ο μικροελεγκτής είναι ένας τύπος μικροεπεξεργαστή που έχει τη δυνατότητα να λειτουργεί με μικρό αριθμό εξωτερικών εξαρτημάτων, εξαιτίας των ενσωματωμένων υποσυστημάτων που διαθέτει. Αποτελείται από ένα ενιαίο ολοκληρωμένο κύκλωμα που περιλαμβάνει έναν πυρήνα επεξεργαστή, μνήμη και προγραμματιζόμενα περιφερειακά εισόδου-εξόδου. Βρίσκει ευρεία εφαρμογή στα ενσωματωμένα συστήματα (embedded systems) ελέγχου χαμηλού και μεσαίου κόστους, όπως αυτά που χρησιμοποιούνται σε αυτοματισμούς, ηλεκτρονικά καταναλωτικά προϊόντα (από ψηφιακές φωτογραφικές μηχανές έως παιχνίδια), ηλεκτρικές συσκευές και κάθε είδους αυτοκινούμενα τροχοφόρα οχήματα. Σε σύγκριση με ένα σύστημα ξεχωριστού μικροεπεξεργαστή, μνήμης και περιφερειακών συσκευών εισόδου-εξόδου, ένας μικροελεγκτής καθιστά δυνατή τη μείωση του μεγέθους και του κόστους με αποτέλεσμα τον ψηφιακό έλεγχο περισσότερων συσκευών και διαδικασιών.

Μικροελεγκτές συγκριτικά με Μικροεπεξεργαστές

Ο μικροελεγκτής αποτελεί μία παραλλαγή επεξεργαστή, όπως προαναφέρθηκε, με σαφείς όμως διαχωριστικές γραμμές και λειτουργία. Στους μικροεπεξεργαστές για μη ενσωματωμένα συστήματα δίνεται μεγαλύτερη έμφαση στην υπολογιστική ισχύ. Αντίθετα οι μικροελεγκτές που εφαρμόζονται στα ενσωματωμένα συστήματα σχεδιάζονται με γνώμονα το χαμηλό κόστος, το μικρό αριθμό απαιτούμενων ολοκληρωμένων κυκλωμάτων και την εξειδίκευση. Ορισμένα πλεονεκτήματα των μικροελεγκτών σε σχέση με τους μικροεπεξεργαστές συνιστώνται στην αυτονομία τους, το χαμηλό κόστος κατασκευής, τη μεγαλύτερη αξιοπιστία και τις μικρότερες απαιτήσεις σε μέγεθος και κατανάλωση ισχύος.

1.2.1.1 Δομή Μικροελεγκτή

Ο μικροελεγκτής είναι ένα ενσωματωμένο σύστημα που περιλαμβάνει επεξεργαστή, μνήμη και περιφερειακά λειτουργώντας αυτόνομα σε ένα μοναδικό ολοκληρωμένο κύκλωμα κατασκευασμένο από πυρίτιο. Η χρήση αυτού του υλικού στην κατασκευή μικροελεγκτών παρέχει υψηλές ταχύτητες μεταφοράς δεδομένων μεταξύ του επεξεργαστή και της μνήμης καθώς και του επεξεργαστή και των περιφερειακών μονάδων.

Επιπλέον οι περισσότεροι μικροελεγκτές έχουν ελάχιστες απαιτήσεις σε μέγεθος μνήμης και προγράμματος, χωρίς την ανάγκη ύπαρξης λειτουργικού συστήματος. Σε πολλά ενσωματωμένα συστήματα λείπουν οι συνήθεις συσκευές που εξυπηρετούν την ανθρώπινη αλληλεπίδραση, όπως πληκτρολόγιο, οθόνη, εκτυπωτές ή άλλες ευρέως χρησιμοποιούμενες συσκευές εισόδου - εξόδου, καθιστώντας τα συστήματα που βασίζονται σε μικροελεγκτές ιδιαίτερα ευέλικτα, οικονομικά και αυτόνομα.

Τα βασικά μέρη ενός μικροελεγκτή είναι τα εξής (βλ. εικόνα 6):

- Η μονάδα επεξεργασίας (CPU)
  Κυμαίνεται από απλούς 4 - bit επεξεργαστές μέχρι πολύπλοκους επεξεργαστές 64 - bit

- Η μνήμη RAM
  Χρησιμοποιείται για την αποθήκευση των δεδομένων.

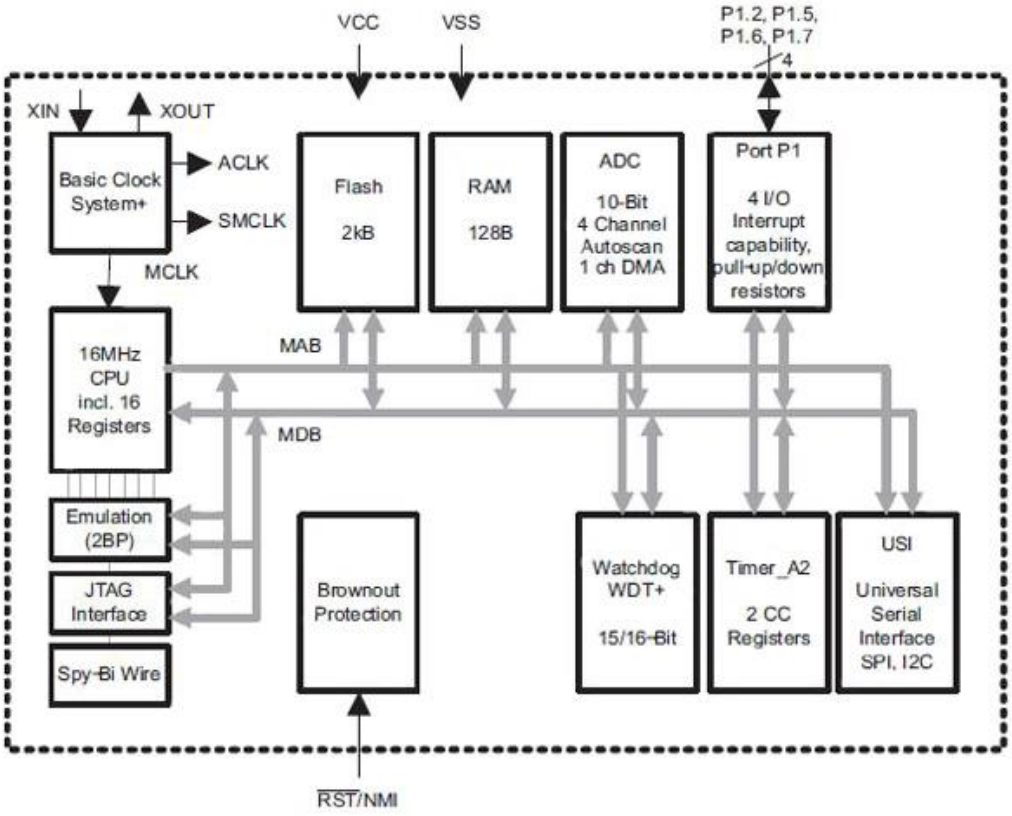
- Η μνήμη προγράμματος
  Μπορεί να είναι ROM, EPROM, EEPROM ή μνήμη Flash και χρησιμοποιείται για την αποθήκευση του προγράμματος και των απαραίτητων παραμέτρων για την ομαλή εκτέλεσή του.

- Σειριακές θύρες εισόδου - εξόδου

- Περιφερειακές μονάδες
  Μπορεί να περιλαμβάνουν περιφερειακά όπως χρονιστές ή γεννήτριες PWM

- Ρολόι συγχρονισμού

- Μετατροπείς από αναλογικό σε ψηφιακό σήμα ή αντίστροφα



Εικόνα 6: Εσωτερικό μικροελεγκτή

Γλώσσα προγραμματισμού μικροελεγκτών

Αρχικά όλοι οι μικροελεγκτές προγραμματίζονταν αποκλειστικά σε γλώσσα Assembly, πλέον όμως ο προγραμματισμός υλοποιείται σε γλώσσα προγραμματισμού C, C++ ή κάποια από τις παραλλαγές τους με τη χρήση μεταγλωττιστών για την μετάφραση της γλώσσας υψηλού επιπέδου σε Assembly που καταλαβαίνει ένας μικροελεγκτής.

Ταξινόμηση μικροελεγκτών

Οι μικροελεγκτές μπορούν να ταξινομηθούν ανάλογα με:

1. Το σετ εντολών (RISC, CISC)
2. Την αρχιτεκτονική κατασκευής (Harvard, Von - Neuman)
3. Τη μνήμη (Ενσωματωμένη, Εξωτερική Μνήμη)
4. Το εύρος των διαύλων επικοινωνίας (4, 8, 16, 32 bit)

1.2.1.2 Κατασκευαστές Μικροελεγκτών

Ορισμένοι από τους πιο γνωστούς κατασκευαστές μικροελεγκτών φαίνονται παρακάτω:

- ARM
- Atmel Corporation
- Texas Instruments
- Microchip

- Silicon
- Renesas Technology Corp
- Intel Corporation
- Analog Devices
- Dallas Semiconductor
- Fujitsu Semiconductor
- National Semiconductor
- STMicroelectronics
- ZiLog
- Freescale Semiconductor
- Infineon Technologies
- NetSilicon
- Rabbit Semiconductor
- Stock Point Electronics
- Western Digital Center
- MicroController Pros Corporation
- Epson
- Hitachi
- Maxim
- NEC
- Toshiba

1.2.1.3 Ενδεικτικές Εφαρμογές μικροελεγκτών

Εφαρμογές μικροελεγκτών σε συσκευές καθημερινής χρήσης:

- Συσκευές ανίχνευσης και ελέγχου φωτισμού
- Συσκευές μέτρησης και ελέγχου θερμοκρασίας
- Πυρανίχνευση
- Αυτοκίνητα
- Αεροπλάνα

Εφαρμογές μικροελεγκτών σε περιβάλλον βιομηχανικού ελέγχου

- Όργανα βιομηχανικού ελέγχου
- Συσκευές ελέγχου βιομηχανικών διεργασιών

Εφαρμογές μικροελεγκτών σε συσκευές μέτρησης

- Βολτόμετρο
- Ψηφιακό μέτρο
- Φορητές συσκευές μέτρησης

1.2.2 Arduino

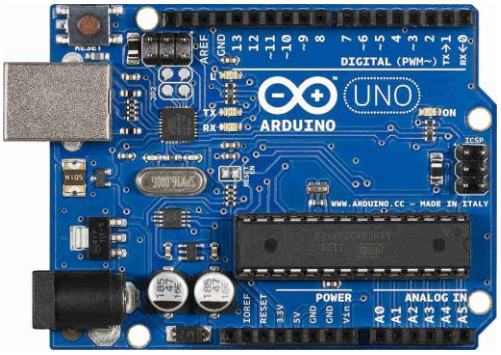
Το arduino είναι μία απλή μητρική πλακέτα ανοιχτού κώδικα που χρησιμοποιείται για την κατασκευή ψηφιακών συσκευών και διαδραστικών αντικειμένων με τον φυσικό κόσμο. Περιλαμβάνει έναν ενσωματωμένο μικροελεγκτή και προγραμματιζόμενες εισόδους-εξόδους. Προγραμματίζεται στη γλώσσα προγραμματισμού Wiring, η οποία είναι στην ουσία η γλώσσα προγραμματισμού C++ και ένα σύνολο από βιβλιοθήκες, υλοποιημένες επίσης στην C++. Η σύνδεση με τον υπολογιστή γίνεται μέσω θύρας USB σε επίπεδο υλικού και μέσω προγραμμάτων όπως τα Processing, Max/MSP, Pure Data, SuperCollider σε επίπεδο λογισμικού. Λειτουργεί σαν βάση για τη δημιουργία διαδραστικών αντικειμένων (interactive objects) που χρησιμοποιούνται για τον έλεγχο και τη λήψη τιμών μέσω αισθητήρων σε πλήθος εφαρμογών. Κοινές τέτοιες εφαρμογές σχετίζονται με ρομποτική, θερμοστάτες, τρισδιάστατους εκτυπωτές, αισθητήρες κίνησης καθώς και χιλιάδες άλλες εφαρμογές, γνωστές ως arduino projects. Οι πληροφορίες κατασκευής του arduino βρίσκονται ελεύθερα διαθέσιμες για όποιον το επιθυμεί, ενώ η πλακέτες διατίθενται προσυναρμολογημένες για όλες τις εκδόσεις. Τέλος μέσω των ψηφιακών και αναλογικών θυρών δίνεται η δυνατότητα χρήσης πλακετών επέκτασης (shields) και άλλων κυκλωμάτων.

Επίσημες εκδόσεις

Η πλατφόρμα Arduino έχει αρκετές επίσημες εκδόσεις όπως τα Mega και Mega2560, με τεχνολογία ATmega1280, το Leonardo, με τεχνολογία ATmega32U4, τα Mini, NG plus και Bluetooth, με τεχνολογία ATmega168, και τα Nano, Lilypad και Duemilanove, με τεχνολογία ATmega328. Η πιο δημοφιλής έκδοση παραμένει το Arduino Uno.

1.2.2.1 Arduino Uno

Το Arduino Uno είναι η πιο παλιά και πιο γνωστή έκδοση Arduino και απεικονίζεται στην εικόνα 7.



Εικόνα 7: Arduino Uno

Χαρακτηριστικά

Τα χαρακτηριστικά στοιχεία του Arduino Uno είναι τα εξής:

- Μικροελεγκτής: ATMega328P
- Τάση λειτουργίας: 5V
- Τάση τροφοδοσίας (προτεινόμενη): 7 - 12V
- Τάση τροφοδοσίας (οριακή): 6 - 20V
- Ψηφιακές θύρες εισόδου - εξόδου (I/O Pins): 14 (εκ των οποίων 6 PWM έξοδοι)
- PWM ψηφιακές θύρες εισόδου - εξόδου: 6
- Αναλογικές θύρες εισόδου: 6
- DC ρεύμα ανά θύρα εισόδου - εξόδου: 20mA
- DC ρεύμα για την θύρα 3.3V: 50mA
- Μνήμη Flash: 32kB
- SRAM: 2kB
- EEPROM: 1kB
- Clock speed: 16MHz
- Μήκος: 68.6mm
- Πλάτος: 53.4mm
- Βάρος: 25g

Πλεονεκτήματα

Τα πλεονεκτήματα του Arduino σε σχέση με την πληθώρα μικροελεγκτών που κυκλοφορεί στο εμπόριο έγκεινται στα εξής:

- Χαμηλό κόστος
  Σε σχέση με τους αντίστοιχους μικροελεγκτές η πλακέτα Arduino παρέχει την πιο οικονομική λύση είτε αγοραστεί έτοιμη, είτε κατασκευαστεί με βάση τα ελεύθερα σχηματικά σχέδια που βρίσκονται διαθέσιμα στο διαδίκτυο.
- Συμβατότητα με διάφορα λειτουργικά συστήματα
  Το Arduino είναι συμβατό με τα λειτουργικά συστήματα Windows, Linux, MacOs ενώ οι περισσότεροι μικροελεγκτές είναι συμβατοί μόνο με την πλατφόρμα των Windows.
- Απλό και ξεκάθαρο προγραμματιστικό περιβάλλον
  Το προγραμματιστικό περιβάλλον του Arduino είναι αρκετά απλό, ώστε να μπορούν να το χρησιμοποιήσουν σχετικά αρχάριοι χρήστες, και ταυτόχρονα αρκετά εξελιγμένο, ώστε να ικανοποιεί περισσότερο έμπειρους και απαιτητικούς χρήστες.
- Βασίζεται σε ανοιχτό λογισμικό
  Το λογισμικό του Arduino διανέμεται σε μορφή ανοιχτού λογισμικού και επιδέχεται βελτιστοποίησης και επέκτασης. Η γλώσσα προγραμματισμού που χρησιμοποιείται για την επέκταση είναι η C++ και οι βιβλιοθήκες της.

- Επιδέχεται επέκταση σε επίπεδο υλικού
  Το Arduino βασίζεται στους επεξεργαστές AT328 και ATMega2560 και επιτρέπεται νομικά η εξέλιξη των υπάρχοντων σχηματικών σχεδίων σε νέες κατασκευές.

Μικροελεγκτής ATMega328P

Το Arduino Uno βασίζεται στον μικροελεγκτή ATMega328P της εταιρείας Atmel. Ο ATMega328P διαθέτει τρεις ομάδες μνήμης:

- Μνήμη SRAM: 2kB
  Η SRAM είναι η ωφέλιμη μνήμη που δεσμεύεται κατά τη διάρκεια εκτέλεσης των προγραμμάτων για την αποθήκευση των απαραίτητων μεταβλητών. Εάν σταματήσει η παροχή ρεύματος ή γίνει επανεκκίνηση η μνήμη χάνει τα δεδομένα της.
- Μνήμη EEPROM: 1kB
  Η EEPROM χρησιμοποιείται από τα προγράμματα για τις εγγραφές ή τις αναγνώσεις δεδομένων. Τα δεδομένα της διατηρούνται στην περίπτωση απώλειας τροφοδοσίας ή επανεκκίνησης.
- Μνήμη Flash: 32kB
  Η μνήμη Flash χρησιμοποιείται για την αποθήκευση των προγραμμάτων, μετά την απαραίτητη μεταγλώττισή τους από τον υπολογιστή. 2kB της μνήμης αυτής είναι μονίμως δεσμευμένα από τον bootloader του Arduino, που είναι αναγκαίος για την εγκατάσταση των προγραμμάτων μέσω της θύρας USB, χωρίς τη χρήση εξωτερικού προγραμματιστή υλικού. Τα δεδομένα και αυτής της μνήμης διατηρούνται στην περίπτωση απώλειας τροφοδοσίας ή επανεκκίνησης.

Ο μικροελεγκτής ATMega328P είναι υψηλής απόδοσης, χαμηλής κατανάλωσης ισχύος, με μήκος διεύθυνσης 8 -bit, σχεδιασμένος στην αρχιτεκτονική Harvard, με σετ εντολών RISC και συχνότητα λειτουργίας στα 16MHz.

1.2.2.2 Ακροδέκτες Arduino

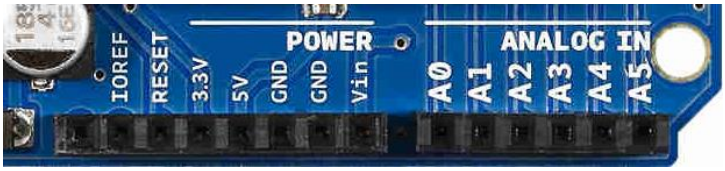
Η πλατφόρμα Arduino Uno διαθέτει 14 ακροδέκτες ψηφιακού σήματος εισόδου - εξόδου γενικού σκοπού (βλ. εικόνα 8). Με χρήση των εντολών - συναρτήσεων pinMode(), digitalWrite() και digitalRead() είναι δυνατόν να τεθούν και να χρησιμοποιηθούν ως ακροδέκτες εισόδου ή ακροδέκτες εξόδου. Κάθε ακροδέκτης λειτουργεί στα 5V και μπορεί να παρέχει ή να λαμβάνει ρεύμα μέγιστης τιμής 40mA.

Εικόνα 8: Ψηφιακές θύρες εισόδου - εξόδου Arduino Uno

Ορισμένοι ψηφιακοί ακροδέκτες επιτελούν και άλλες λειτουργίες εάν προγραμματιστούν κατάλληλα. Αυτές παρουσιάζονται αναλυτικά παρακάτω:

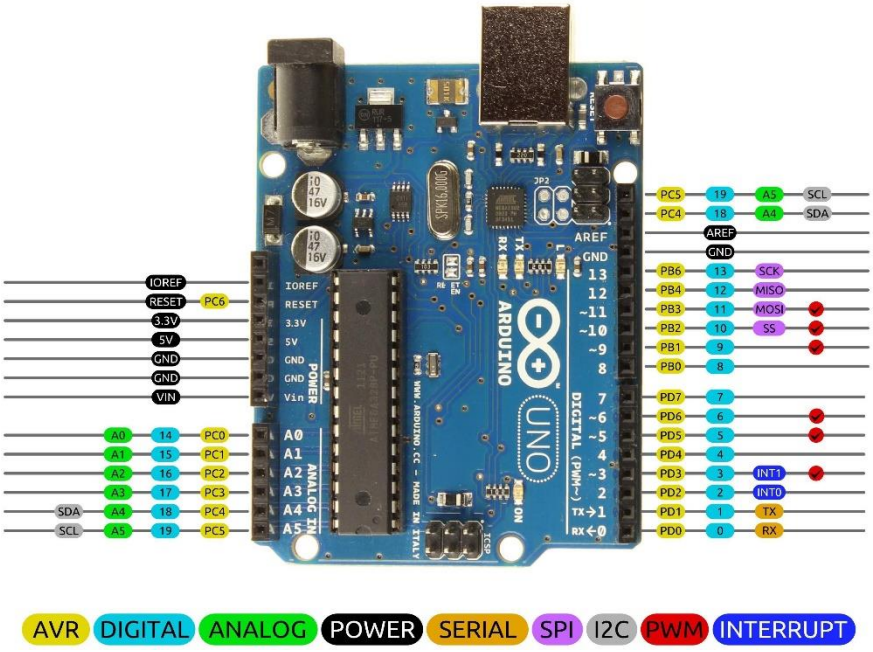
- Σειριακή Λειτουργία
  Οι ακροδέκτες 0 και 1 λειτουργούν ως RX(receive) και TX(transmit) κατά τη διάρκεια της σειριακής επικοινωνίας. Όταν ενεργοποιείται η σειριακή θύρα τα δεδομένα προωθούνται στη θύρα USB μέσω του ελεγκτή Serial-Over-USB και στον ακροδέκτη 0, ώστε να είναι διαθέσιμα προς ανάγνωση ενδεχομένως από μια άλλη συσκευή.
- Interrupts
  Οι ακροδέκτες 2 και 3 λειτουργούν ως είσοδοι για εξωτερικά interrupts. Με χρήση της συνάρτησης attachInterrupt() είναι δυνατό να διακόπτεται η κανονική ροή του προγράμματος και να εκτελείται μία συγκεκριμένη διαδικασία. Αυτό είναι ιδιαίτερα χρήσιμο σε εφαρμογές που απαιτούν συγχρονισμό υψηλής ακρίβειας.
- PWM
  Οι ακροδέκτες 3, 5, 6, 9, 10 και 11 είναι δυνατό, με χρήση της συνάρτησης analogWrite(), να λειτουργήσουν ως κατά συνθήκη αναλογικές έξοδοι με το σύστημα PWM και παρέχουν 8 -bit έξοδο.
- Επικοινωνία Σειριακής Περιφερειακής Διεπαφής (Serial Peripheral Interface Bus - SPI)
  Οι ακροδέκτες 10 (SS), 11 (MOSI), 12 (MISO) και 13 (SCK) επιτρέπουν επικοινωνία SPI με χρήση των αντίστοιχων βιβλιοθηκών. Το πρωτόκολλο SPI επιτρέπει τη σειριακή επικοινωνία σε μικρή απόσταση και έχει κυρίως χρήση σε ολοκληρωμένα συστήματα. Η επικοινωνία μεταξύ των συσκευών λαμβάνει χώρα αμφίδρομα, ενώ οι συσκευές συνδέονται μεταξύ τους με αρχιτεκτονική master - slave με ένα μόνο master.
- Επικοινωνία I2C
  Οι ακροδέκτες 4 (SDA) και 5 (SCL) επιτρέπουν επικοινωνία I2C με χρήση των αντίστοιχων βιβλιοθηκών. Το I2C υλοποιείται με την χρήση δύο καλωδίων διπλής κατεύθυνσης και υποστηρίζει τη σειριακή επικοινωνία μεταξύ ολοκληρωμένων κυκλωμάτων.

Η πλατφόρμα Arduino Uno διαθέτει επιπλέον 6 αναλογικούς ακροδέκτες εισόδου, όπως φαίνεται στην εικόνα 9:



Εικόνα 9: Αναλογικές θύρες εισόδου Arduino Uno

Οι ακροδέκτες αναλογικού σήματος που είναι αριθμημένοι από A0 έως A5 λειτουργούν ως αναλογικές είσοδοι χρησιμοποιώντας τον ADC (Analog to Digital Converter) που είναι ενσωματωμένος στον μικροελεγκτή. Μπορούν να διαβάσουν τάση από 0 έως 5V και να τη μετατρέψουν σε τιμές από 0 έως 1023. Επιπλέον, με τον ανάλογο προγραμματισμό, οι ακροδέκτες αυτοί είναι δυνατό να μετατραπούν σε ψηφιακές εισόδους - εξόδους και ονομάζονται αντίστοιχα ψηφιακές θύρες 14 έως 19.



Εικόνα 10: Ακροδέκτες Arduino

Δίπλα στους αναλογικούς ακροδέκτες βρίσκονται 7 ακόμα ακροδέκτες, όπως φαίνεται στην εικόνα 10 οι οποίοι είναι αναλυτικά:

- Vin
  Ακροδέκτης για μη σταθεροποιημένη τάση. Εδώ συνδέεται συνήθως η εξωτερική πηγή, εάν υπάρχει. Σ' αυτήν την περίπτωση δίνεται η δυνατότητα να τροφοδοτούνται εξαρτήματα με την πλήρη τάση της εξωτερικής τροφοδοσίας.
- GND
  Ακροδέκτες γείωσης

- 5V
  Ακροδέκτης σταθερής τάσης 5V που χρησιμοποιείται για την τροφοδοσία του μικροελεγκτή και άλλων ηλεκτρονικών στοιχείων.
- 3.3V
  Ακροδέκτης σταθερής τάσης 3.3V μέσω του ολοκληρωμένου FTDI που βρίσκεται στην πλατφόρμα του Arduino, με μέγιστο παρεχόμενο ρεύμα 50mA.
- RESET
  Ακροδέκτης που χρησιμεύει για την επανεκκίνηση του Arduino, μετά τη γείωσή του με οποιονδήποτε από τους ακροδέκτες Ground.
- IOREF
  Ακροδέκτης που παρέχει στην πλακέτα την τάση αναφοράς με την οποία λειτουργεί ο μικροελεγκτής.

1.2.2.3 Τροφοδοσία

Η τροφοδοσία του Arduino γίνεται με δύο τρόπους, είτε μέσω της θύρας USB όταν είναι συνδεδεμένο με τον υπολογιστή είτε μέσω εξωτερικής τροφοδοσίας. Η επιλογή της πηγής γίνεται αυτόματα. Επίσης είναι δυνατό να συνδεθεί μπαταρία ανάμεσα στους ακροδέκτες Vin και Gnd. Η τάση εξωτερικής τροφοδοσίας κυμαίνεται μεταξύ 7 και 12V. Ιδανική προτεινόμενη τάση εξωτερικής τροφοδοσίας είναι τα 9V.

1.2.2.4 Περιβάλλον ανάπτυξης (IDE) Arduino

Ο προγραμματισμός του μικροελεγκτή Arduino γίνεται μέσω του ολοκληρωμένου περιβάλλοντος ανάπτυξης Arduino IDE. Το IDE είναι υλοποιημένο σε γλώσσα προγραμματισμού Java και βασίζεται στο περιβάλλον της γλώσσας προγραμματισμού Processing.

Περιλαμβάνει έναν επεξεργαστή κειμένου (text editor) όπου γράφονται τα προγράμματα (sketches), μία περιοχή μηνυμάτων για την ενημέρωση του χρήστη σχετικά με την πρόοδο, την κατάσταση ή τα λάθη των προγραμμάτων, μία γραμμή εργαλείων και ένα παράθυρο προβολής της σειριακής επικοινωνίας(serial monitor), όπου εμφανίζεται η επικοινωνία μεταξύ του Arduino και της θύρας USB. Επίσης διαθέτει μεταγλωττιστή (compiler) για την μεταγλώττιση των προγραμμάτων και πλήθος έτοιμων παραδειγμάτων που μπορούν να χρησιμοποιηθούν ως βάση για μελλοντικό προγραμματισμό.

1.2.2.5 Γλώσσα προγραμματισμού

Η γλώσσα προγραμματισμού του Arduino Uno βασίζεται στη γλώσσα Wiring, μια παραλλαγή C / C++ για μικροελεγκτές αρχιτεκτονικής AVR, όπως ο ATMega και υποστηρίζει

όλες τις βασικές δομές της C και μερικά χαρακτηριστικά της C++. Ο compiler που χρησιμοποιείται είναι ο AVR gcc και ως βασική βιβλιοθήκη C χρησιμοποιείται η βιβλιοθήκη AVRlibc.

Λόγω της καταγωγής της από την C, η γλώσσα του Arduino χρησιμοποιεί τις ίδιες βασικές εντολές και συναρτήσεις, με την ίδια σύνταξη, τους ίδιους τύπους δεδομένων και τους ίδιους τελεστές με τη γλώσσα C. Επιπρόσθετα χρησιμοποιεί κάποιες ειδικές εντολές, συναρτήσεις και σταθερές που έχουν να κάνουν με την εξειδικευμένη διαχείριση του υλικού της πλατφόρμας Arduino.

1.2.2.6 Ολοκληρωμένες πλακέτες επέκτασης (Shields)

Οι shields είναι ολοκληρωμένες πλακέτες που είναι ειδικά σχεδιασμένες ώστε να εφαρμόζουν πάνω στην πλακέτα του Arduino ενισχύοντας τη λειτουργικότητά του. Μερικές από τις πιο δημοφιλείς shields που κυκλοφορούν στο εμπόριο είναι η Ethernet shield, η οποία δίνει τη δυνατότητα δικτύωσης σε ένα τοπικό δίκτυο (LAN) ή ενσύρματα στο Internet, η WiFi shield, η οποία παρέχει τις ίδιες δυνατότητες με την Ethernet shield ασύρματα, η Screen shield, η οποία κυκλοφορεί σε διάφορες εκδοχές και δίνει τη δυνατότητα σύνδεσης μιας ευρείας επιλογής οθονών, η GPS shield, η οποία παρέχει τη δυνατότητα εντοπισμού στίγματος, η Motor shield, η οποία δίνει τη δυνατότητα οδήγησης διάφορων κινητήρων, και τέλος η Data Logging Shield, η οποία παρέχει τη δυνατότητα χρήσης SD μνήμης αποθήκευσης δεδομένων, ενώ επιπλέον διαθέτει Real Time Clock που διατηρεί την ώρα και την ημερομηνία.

1.3 Πρωτόκολλο RS485

Το RS485 είναι ένα πρωτόκολλο σειριακής επικοινωνίας και ορίζεται από την Electronics Industry Assosiation ως EIA/TIA-485. Τα συστήματα επικοινωνιών που βασίζονται στο πρωτόκολλο RS485 στέλνουν ψηφιακές πληροφορίες από τους πομπούς στους δέκτες μέσω συνεστραμμένου ζεύγους καλωδίου. Οι συσκευές μπορούν να έχουν απόσταση μεταξύ τους έως 1220 μέτρα χωρίς την ανάγκη παρεμβολής ενισχυτή σήματος. Σε ένα δίαυλο επικοινωνίας μπορούν να συνδεθούν μέχρι 32 πομποί μέσω RS485. Ο μέγιστος ρυθμός μετάδοσης δεδομένων που μπορεί να υποστηριχθεί φτάνει τα 10Mbps, αν και στην πράξη χρησιμοποιούνται συνήθως μικρότεροι ρυθμοί μετάδοσης.

1.3.1 Χαρακτηριστικά πρωτοκόλλου RS485

Τα χαρακτηριστικά του RS485 φαίνονται παρακάτω:

Διαφορικό: ναι

Μέγιστος αριθμός οδηγών: 32

Μέγιστος αριθμός δεκτών: 32

Τοπολογία δικτύου: Πολλών σημείων

Μέγιστη απόσταση (πρότυπο acc): 1200 m

Μέγιστη ταχύτητα στα 12 m: 35 Mbs

Μέγιστη ταχύτητα στα 1200 m: 100 kbs

Αντίσταση εισόδου δέκτη: ≧ 12 kΩ

Αντίσταση φορτίου οδηγού: 54 Ω

Ευαισθησία δέκτη εισόδου: ±200 mV

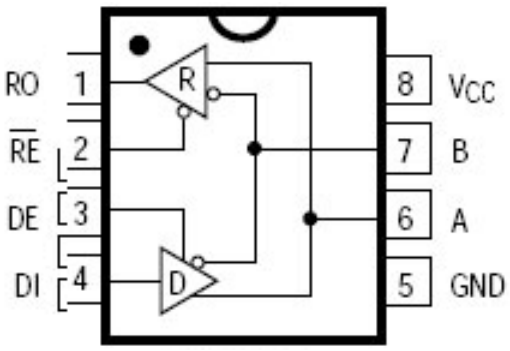
Εύρος δέκτη εισόδου: –7..12 V

Μέγιστη τάση οδηγού εξόδου: –7..12 V

Ελάχιστη τάση οδηγού εξόδου (υπό φορτίο): ±1.5 V

1.3.2 Δομή πομποδέκτη RS485

Η εσωτερική δομή του πομποδέκτη RS485 φαίνεται στην εικόνα 11:



Εικόνα 11: Εσωτερική δομή πομποδέκτη RS485

Στη συνέχεια δίνεται αναλυτική εξήγηση της λειτουργίας της κάθε θύρας:

1. RO (Receive Out)
   Όταν ο πομποδέκτης βρίσκεται σε κατάσταση Receive στη θύρα 1 εμφανίζονται τα δεδομένα του διαύλου επικοινωνίας.
2. RE (Receive Enable)
   Συγκεκριμένα η θύρα 2 υλοποιεί το αντίθετο του RE, επομένως όταν βρίσκεται σε κατάσταση 1 (ON) ο πομποδέκτης βρίσκεται σε λειτουργία Transmit.
3. DE (Data Enable)
   Όταν η θύρα 3 είναι σε κατάσταση 1 (ON) τα δεδομένα είναι έτοιμα προς αποστολή.
4. DI (Data In)
   Όταν ο πομποδέκτης βρίσκεται σε κατάσταση Transmit τα δεδομένα που εμφανίζονται στη θύρα 4 προωθούνται στο δίαυλο επικοινωνίας.

5. GND
   Ακροδέκτης γείωσης
6. A
   Το ένα από τα δύο καλώδια που συνιστούν το δίαυλο επικοινωνίας. Η μεταφορά των δεδομένων του πρωτοκόλλου γίνεται μέσω του διαύλου και τα καλώδια A και B έχουν μεταξύ τους αντίθετες πολικότητες.
7. B
   Το έτερο καλώδιο που συνιστά το δίαυλο επικοινωνίας.
8. Vcc
   Ο ακροδέκτης που συνδέεται η πηγή τάσης.

1.4 Αισθητήρες

Ως αισθητήρας ορίζεται η διάταξη που μετρά ένα φυσικό μέγεθος και το μετατρέπει άμεσα ή έμμεσα σε ηλεκτρικό σήμα, τάση ή ρεύμα, με ή χωρίς τη χρήση κατάλληλου μετατροπέα, που δύναται περαιτέρω επεξεργασίας. Κλασικά παραδείγματα φυσικών μεγεθών που μετρούνται με χρήση αισθητήρων είναι η θερμοκρασία, η υγρασία, η στάθμη υγρών, η θέση και η μετατόπιση ενός αντικειμένου, η ταχύτητα και η επιτάχυνση ενός κινούμενου αντικειμένου, η τάση, το ρεύμα ή η ακτινοβολία. Ιδιαίτερη σημασία έχουν οι αισθητήρες που έχουν χρήση σε συστήματα αυτομάτου ελέγχου και είναι δυνατό να συνδεθούν σε κάποιου είδους ηλεκτρονικό σύστημα μέτρησης.

Ορισμένα χαρακτηριστικά των αισθητήρων παρουσιάζονται παρακάτω:

- Πιστότητα (Accuracy)
  Το κατά πόσον το αποτέλεσμα που δίνει ο αισθητήρας πλησιάζει τη φυσική πραγματικότητα. Δίνεται συνήθως ως ποσοστό επί του εύρους λειτουργίας του αισθητήρα.
- Ακρίβεια (Precision)
  Εκφράζει το βαθμό ελευθερίας του αισθητήρα από τυχαία σφάλματα.
- Ανοχή
  Συνδέεται στενά με την πιστότητα και ορίζει το μέγιστο αναμενόμενο σφάλμα μίας τιμής.
- Εύρος
  Η ελάχιστη και η μέγιστη τιμή του φυσικού μεγέθους που μπορεί να ανιχνευθεί από τον αισθητήρα.
- Συστηματικό σφάλμα
  Ένα σταθερό σφάλμα, για όλο το εύρος του αισθητήρα, το οποίο μπορεί να μηδενιστεί με βαθμονόμηση.

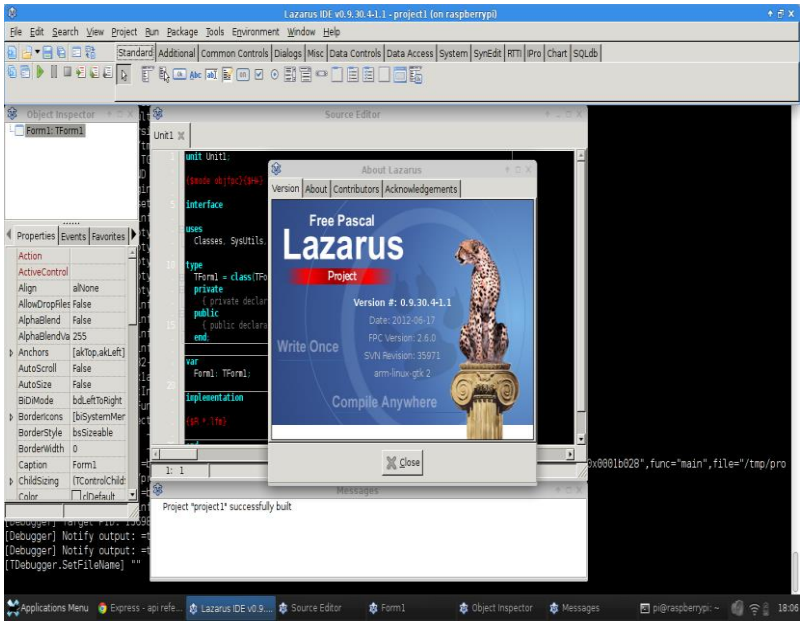
Τα παραπάνω αποτελούν στατικά χαρακτηριστικά των αισθητήρων, τα οποία αναφέρονται στην κατάσταση κατά την οποία έχει επέλθει ισορροπία ανάμεσα στον

αισθητήρα και το μετρούμενο μέγεθος. Αυτό συμβαίνει όταν τα προς μέτρηση μεγέθη είναι σταθερά ή αλλάζουν με πολύ αργό ρυθμό.

Οι αισθητήρες διαθέτουν και δυναμικά χαρακτηριστικά. Αυτά αναφέρονται στη συμπεριφορά του αισθητήρα μεταξύ της στιγμής που το σήμα αρχίζει να μεταβάλλεται μέχρι τη στιγμή που θα σταθεροποιηθεί εκ νέου.

1.5 Λογισμικό δημιουργίας εφαρμογών διεπαφής χρήστη

Η εφαρμογή διεπαφής με το χρήστη υλοποιήθηκε σε γλώσσα προγραμματισμού Pascal μέσω του γραφικού περιβάλλοντος Lazarus (βλ. εικόνα 12). Το Lazarus είναι ένα γραφικό περιβάλλον όπου μπορούν να δημιουργηθούν τα παράθυρα του επιθυμητού προγράμματος, να γραφτεί καθώς και να ελεγχθεί ο κώδικας για τυχόν λάθη, διαδικασία γνωστή ως debugging. Τέτοια περιβάλλοντα είναι γνωστά ως Intergrated Design Environments (IDE). Μέσα στο Lazarus περιέχεται ο μεταγλωττιστής Freepascal, της γλώσσας προγραμματισμού Pascal. Ο μεταγλωττιστής (compiler) είναι ένα πρόγραμμα υπολογιστή το οποίο διαβάζει κώδικα γραμμένο σε γλώσσα προγραμματισμού υψηλού επιπέδου και τον μεταφράζει σε γλώσσα μηχανής, η οποία είναι κατανοητή από τον υπολογιστή. Ο μεταγλωττιστής διαβάζει τις εντολές του προγράμματος, τις αναλύει και δημιουργεί ένα εκτελέσιμο αρχείο, όπου περιέχονται οι απαραίτητες πληροφορίες για την εκτέλεση του προγράμματος. Το Lazarus και ο Freepascal συνδυάζονται αρμονικά ώστε να δημιουργηθεί ένα εύχρηστο περιβάλλον ανάπτυξης εφαρμογών (RAD). Επιπλέον είναι και τα δύο λογισμικά ανοιχτού κώδικα και διατίθενται ελεύθερα στο διαδίκτυο.



Εικόνα 12: Lazarus IDE

Κεφάλαιο 2: Διάταξη μετρήσεων

2.1 Εισαγωγή

Σκοπός της εργασίας είναι η κατασκευή διάταξης μετρήσεων απομακρυσμένης ενσύρματης επικοινωνίας, μέσω του πρωτοκόλλου RS485, βασισμένης στον μικροελεγκτή Arduino και ελεγχόμενης μέσω ηλεκτρονικού υπολογιστή. Η διάταξη θα παίρνει μετρήσεις από 10, εν δυνάμει, διαφορετικά σημεία μέτρησης, μέγιστης απόστασης 1200m και θα στέλνει στον υπολογιστή τα δεδομένα των μετρήσεων, τα οποία στη συνέχεια θα αποθηκεύονται σε ηλεκτρονικό αρχείο της μορφής .txt (έγγραφο σημειωματαρίου). Στην περίπτωση όπου ο υπολογιστής είναι αποσυνδεδεμένος τα δεδομένα θα αποθηκεύονται σε κάρτα μνήμης. Το σύστημα, με μικρές αλλαγές, επιτρέπει τη σύνδεση και άλλων αισθητήρων (τάσης, ρεύματος, υγρασίας κτλ), ανάλογα με τις ανάγκες της εκάστοτε μέτρησης.

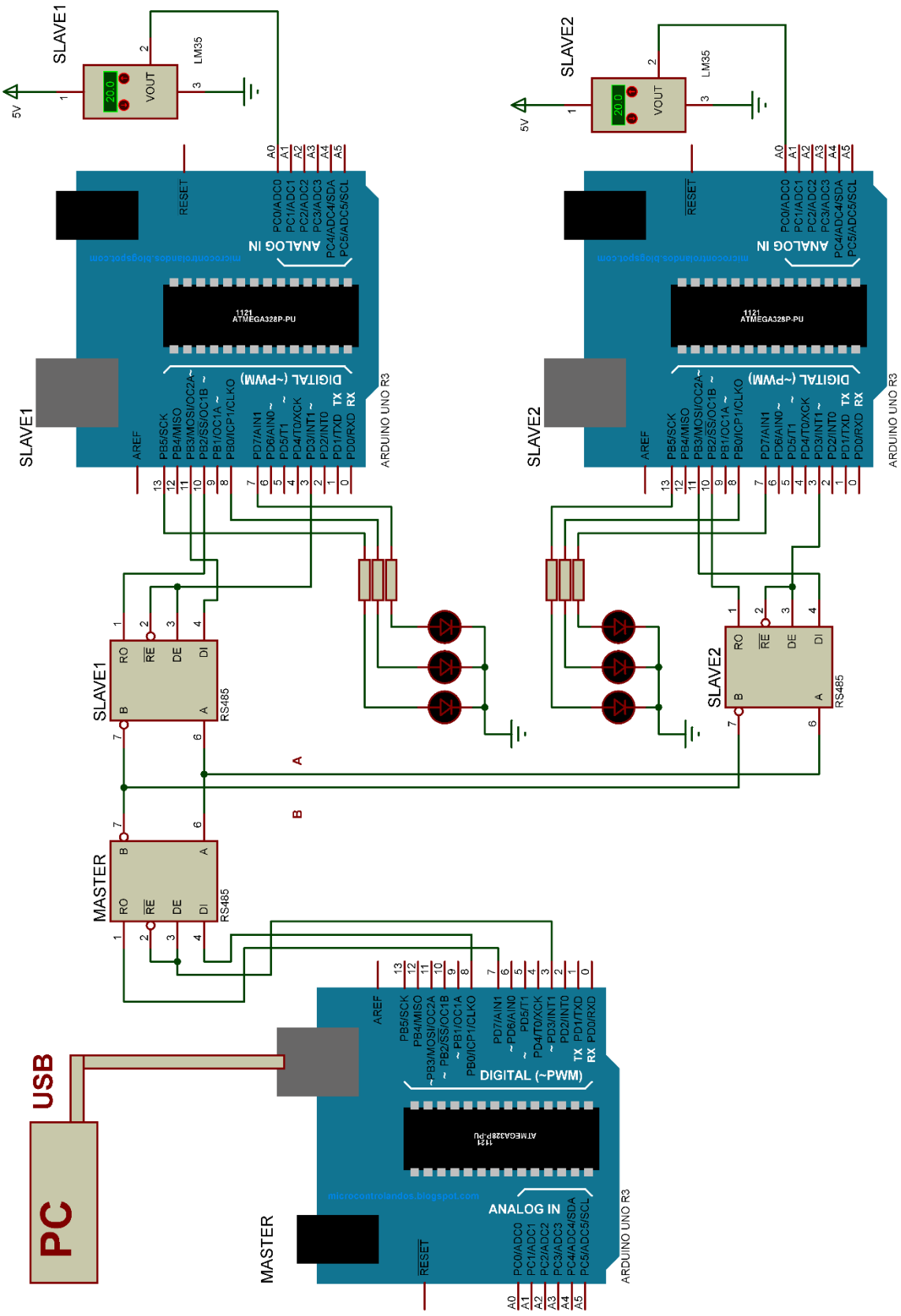
Η υλοποίηση της διάταξης για τους σκοπούς της παρούσας εργασίας περιλαμβάνει τα εξής:

- Μικροελεγκτή Arduino, σε συνδεσμολογία master, για τον έλεγχο των υπόλοιπων μικροελεγκτών και την προώθηση εντολών και μετρήσεων.
- Δύο μικροελεγκτές Arduino, σε συνδεσμολογία slave, για τη λήψη των μετρήσεων
- Δύο πομποδέκτες RS485, συνδεδεμένους με ένα Arduino - slave ο καθένας για την προώθηση των δεδομένων των εντολών ή των μετρήσεων
- Λογισμικό εφαρμογής διεπαφής με τον χρήστη στον ηλεκτρονικό υπολογιστή, όπου λαμβάνει χώρα η δημιουργία των εντολών μέτρησης, η αποστολή τους, η λήψη των δεδομένων μέτρησης και η αποθήκευσή τους.
- Αισθητήρες θερμοκρασίας LM35
- LEDs
- Data Logging Shield για την αποθήκευση των δεδομένων σε κάρτα μνήμης SD, κατά την κατάσταση που ο ηλεκτρονικός υπολογιστής δεν είναι συνδεδεμένος.
- Τηλεφωνικό καλώδιο RJ11 για την υλοποίηση του διαύλου επικοινωνίας του πρωτοκόλλου RS485

2.2 Υλικό (Hardware)

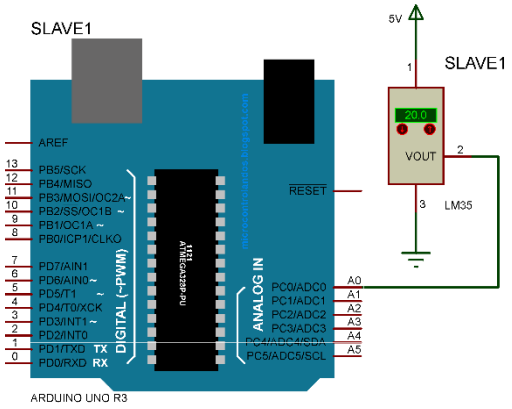
2.2.1 Τοπολογία Διάταξης Μετρήσεων

Στην εικόνα 13 φαίνεται η τοπολογία της διάταξης που υλοποιήθηκε στην παρούσα εργασία. Η υλοποίηση έγινε με δύο Arduino - slaves για λόγους οικονομίας. Η εφαρμογή υποστηρίζει έως 10 Arduino - slaves, ωστόσο η λειτουργία που διέπει την παρούσα υλοποίηση είναι απολύτως ταυτόσημη. Οποιοδήποτε επιπλέον slave είναι επιθυμητό να συνδεθεί στη διάταξη είναι δυνατό να συνδεθεί όπως και τα πρώτα δύο και παράλληλα με αυτά. Η απόσταση ανάμεσα στο master και το τελευταίο slave μπορεί να είναι έως 1200m με τη παρούσα συνδεσμολογία.



Εικόνα 13: Τοπολογία Διάταξης Μετρήσεων

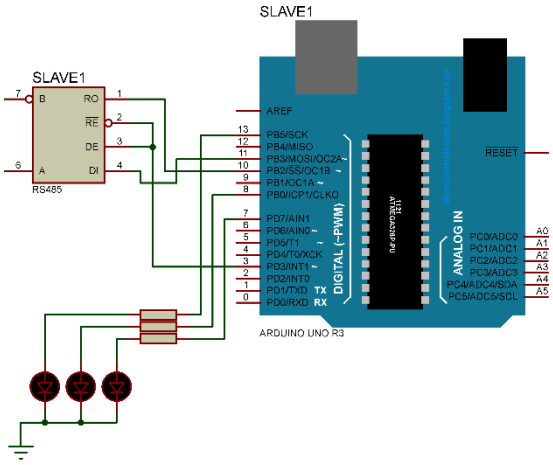
2.2.2 Αισθητήρας θερμοκρασίας LM35



Εικόνα 14: Αισθητήρας θερμοκρασίας LM35

Όπως φαίνεται και στην εικόνα 14 ο αισθητήρας θερμοκρασίας έχει τρεις ακροδέκτες. Ο πρώτος και ο τρίτος ακροδέκτης συνδέονται στην τάση 5V και στη γείωση αντίστοιχα. Ο μεσαίος ακροδέκτης συνδέεται στην αναλογική θύρα του Arduino - slave.

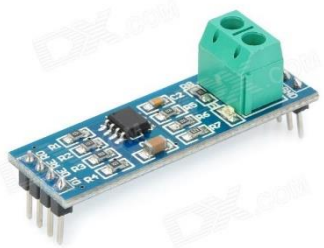
2.2.3 LED



Εικόνα 15: LED συνδεδεμένο σε Arduino

Όπως φαίνεται στην εικόνα 15 τα LEDs συνδέονται με το Arduino - slave με την παραπάνω συνδεσμολογία. Ο θετικός ακροδέκτης συνδέεται με την ψηφιακή θύρα, που καθορίζει την κατάσταση του LED ανάλογα με το αν είναι LOW (OFF) ή HIGH (ON), με την παρεμβολή μιας μικρής αντίστασης για την προστασία του LED της τάξης των 200Ω, ενώ ο αρνητικός ακροδέκτης συνδέεται με τη γείωση.

2.2.4 Πομποδέκτης RS485



Εικόνα 16: Πομποδέκτης RS485

Στην εικόνα 16 φαίνεται ο πομποδέκτης RS485. Η ακριβής συνδεσμολογία του στην πλατφόρμα Arduino αναφέρεται αναλυτικά σε επόμενο κεφάλαιο.

2.2.5 Ολοκληρωμένο Κύκλωμα Επέκτασης Καταγραφής Δεδομένων (Data Logging Shield)

Εικόνα 17: Data Logging Shield

Στην εικόνα 17 παρουσιάζεται μία Data Logging Shield. Η πλακέτα αυτή εφαρμόζει πάνω από το Arduino master και παρέχει τη δυνατότητα αποθήκευσης αρχείων στην κάρτα μνήμης SD σε μορφή .txt (έγγραφο σημειωματαρίου). Επίσης παρέχει τη δυνατότητα διατήρησης της ώρας και της ημερομηνίας, ακόμα και στην περίπτωση αποσύνδεσης από την τροφοδοσία του Arduino, λόγω της μπαταρίας που διαθέτει.

Η Data Logging Shield που χρησιμοποιήθηκε στην παρούσα εργασία είναι της εταιρείας Deek Robot.

2.2.6 Arduino

Ο μικροελεγκτής Arduino που λειτουργεί ως master έχει την εξής συνδεσμολογία:

- Ψηφιακή θύρα 3
  Συνδέεται με τις θύρες 2 (RE) και 3 (DE) του πομποδέκτη RS485

- Ψηφιακή θύρα 7
  Συνδέεται με τη θύρα 1 (RO) του πομποδέκτη RS485
- Ψηφιακή θύρα 8
  Συνδέεται με τη θύρα 4 (DI) του πομποδέκτη RS485
- Ψηφιακές θύρες 10, 11, 12, 13
  Συνδέονται με τη Data Logging Shield

Οι μικροελεγκτές Arduino που λειτουργούν ως slaves έχουν πανομοιότυπη συνδεσμολογία, για λόγους ομοιομορφίας, η οποία είναι η εξής:

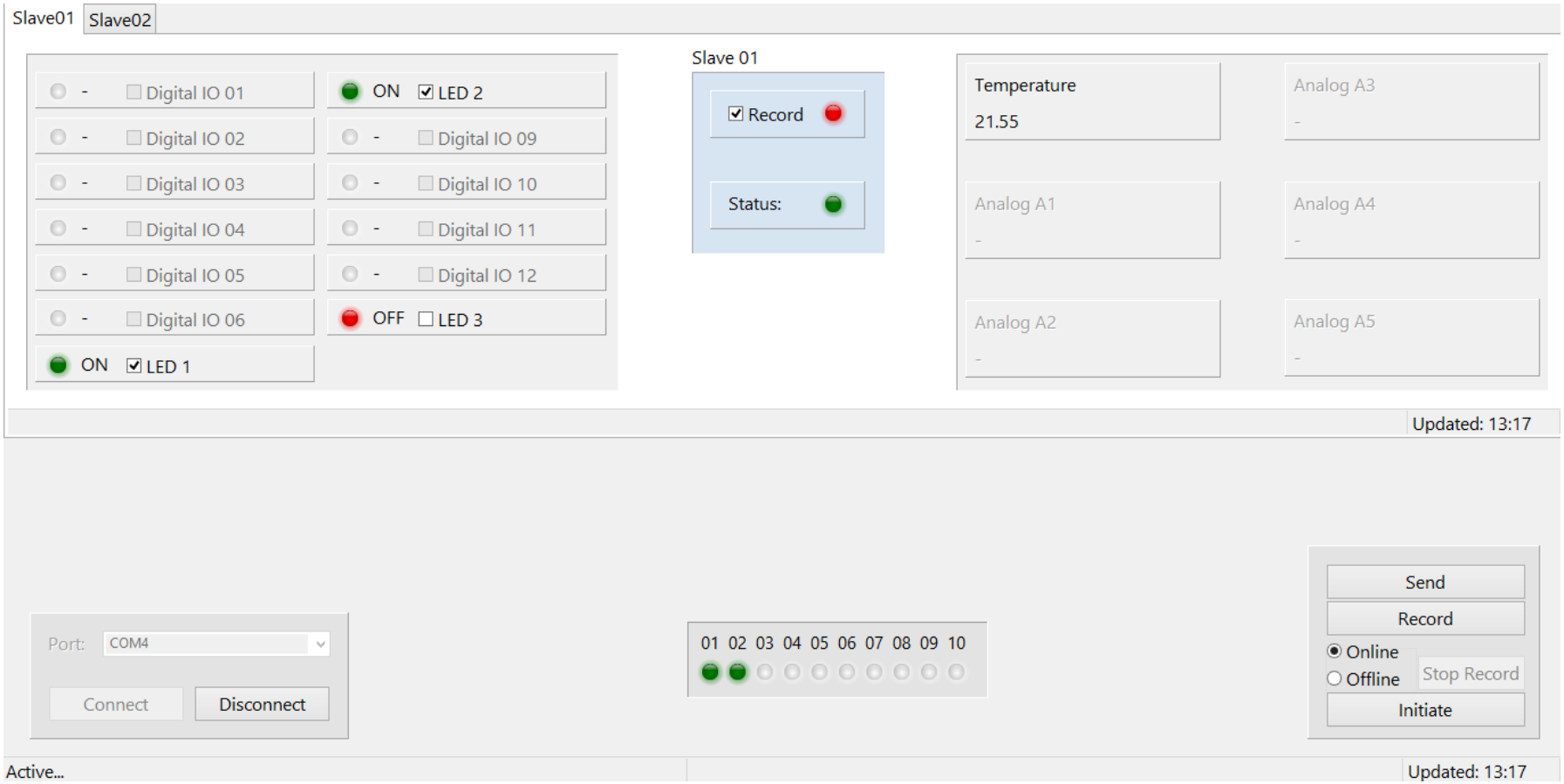
- Αναλογική θύρα A0
  Συνδέεται ο αισθητήρας θερμοκρασίας LM35 με συνδεσμολογία που προαναφέρθηκε.
- Ψηφιακή θύρα 3
  Συνδέεται με τις θύρες 2 (RE) και 3 (DE) του πομποδέκτη RS485
- Ψηφιακή θύρα 7
  Συνδέεται με LED
- Ψηφιακή θύρα 8
  Συνδέεται με LED
- Ψηφιακή θύρα 10
  Συνδέεται με τη θύρα 1 (RO) του πομποδέκτη RS485
- Ψηφιακή θύρα 11
  Συνδέεται με τη θύρα 4 (DI) του πομποδέκτη RS485
- Ψηφιακή θύρα 13
  Συνδέεται με LED

2.3 Λογισμικό (Software)

Η διάταξη μετρήσεων αποτελείται από τρία βασικά μέρη, την εφαρμογή διεπαφής με το χρήστη μέσω ηλεκτρονικού υπολογιστή, το master Arduino και τέλος τα Arduino που λειτουργούν ως slaves.

2.3.1 Εφαρμογή Διεπαφής Χρήστη

Στην εικόνα 18 παρουσιάζεται η εφαρμογή διεπαφής χρήστη.



Εικόνα 18: Εφαρμογή διεπαφής χρήστη

Όπως φαίνεται και στην εικόνα 18 στο πάνω μέρος της εφαρμογής εμφανίζονται οι καρτέλες με τα στοιχεία των μικροελεγκτών που λειτουργούν ως slaves. Ο μέγιστος αριθμός καρτελών που υποστηρίζει η εφαρμογή είναι 10. Οι καρτέλες οι οποίες δεν αντιστοιχούν σε κάποιο ενεργό slave δεν εμφανίζονται για λόγους ευχρηστίας της εφαρμογής. Ενδεικτικά, στην παρούσα εργασία όπου χρησιμοποιούνται δύο slaves εμφανίζονται οι καρτέλες που αντιστοιχούν στα πρώτα δύο slaves. Οι καρτέλες είναι πανομοιότυπες μεταξύ τους. Η κάθε καρτέλα παρέχει στοιχεία σχετικά με τις τιμές από όλες τις αναλογικές θύρες του αντίστοιχου arduino – slave και δίνεται η δυνατότητα να γίνει χειρισμός όλων των ψηφιακών θυρών του. Οι ψηφιακές θύρες που επιλέχθηκαν για την εφαρμογή του πρωτοκόλλου επικοινωνίας RS485 σε κάθε arduino - slave είναι οι θύρες D3, D10 και D11. Οι θύρες αυτές επιλέχθηκαν τυχαία και θα μπορούσαν κάλλιστα να αντικατασταθούν από οποιεσδήποτε άλλες ψηφιακές θύρες στο κάθε arduino - slave, εφόσον βέβαια γίνουν και οι απαραίτητες ρυθμίσεις από την πλευρά της εφαρμογής διεπαφής. Η θύρα D3 χρησιμοποιήθηκε για τη μετάβαση του RS485 από κατάσταση transmit σε κατάσταση receive και αντίστροφα. Η θύρα D10 χρησιμοποιήθηκε για την είσοδο των δεδομένων από το διάδρομο επικοινωνίας των RS485 ενώ η θύρα D11 χρησιμοποιήθηκε αντίστοιχα για την έξοδο των δεδομένων προς το διάδρομο επικοινωνίας. Για λόγους ομοιομορφίας, κρίθηκε σκόπιμο να χρησιμοποιηθούν οι ίδιες θύρες σε κάθε arduino - slave χωρίς αυτό να είναι απαραίτητο. Σε κάθε arduino - slave μπορούν να οριστούν οι θύρες αυτές ανάλογα με τις εκάστοτε ανάγκες. Ο μέγιστος αριθμός ψηφιακών θυρών που είναι δυνατό να γίνει χειρισμός είναι 11, αφού οι υπόλοιπες 3 χρησιμοποιούνται από το πρωτόκολλο επικοινωνίας RS485. Ωστόσο για τις ανάγκες της παρούσας εργασίας κρίθηκε σκόπιμο να γίνει χειρισμός 3 ψηφιακών θυρών σε κάθε arduino – slave και συγκεκριμένα των ψηφιακών θυρών D7, D8 και D13.

Κάτω από τις καρτέλες αυτές βρίσκονται τα πλήκτρα σύνδεσης και αποσύνδεσης, Connect και Disconnect αντίστοιχα, καθώς και το σύνθετο πλαίσιο (combo box) επιλογής θύρας (Port), όπου γίνεται η επιλογή της σειριακής θύρας του υπολογιστή μέσω της οποίας θα λάβει χώρα η επικοινωνία μεταξύ της εφαρμογής διεπαφής και του master arduino. Επίσης το πλήκτρο αποστολής της εντολής Αποστολής (Send) και η συστοιχία επιλογών που αφορά στη λειτουργία Καταγραφής (Record) για τη λήψη επαναληπτικών μετρήσεων, που περιλαμβάνει το ομώνυμο πλήκτρο, το πλήκτρο Διακοπή Καταγραφής (Stop Record) καθώς και την επιλογή εάν οι επαναληπτικές μετρήσεις θα ληφθούν με συνδεδεμένο τον υπολογιστή (Online) ή με αποσυνδεδεμένο τον υπολογιστή (Offline). Κάτω από αυτά βρίσκεται το πλήκτρο Αρχικοποίησης (Initiate) που αντιστοιχεί στη λειτουργία αρχικοποίησης της εφαρμογής. Τέλος η γραμμή κατάστασης (status bar) μέσω της οποίας λαμβάνει χώρα η επικοινωνία της εφαρμογής προς το χρήστη. Αναλυτικά η λειτουργία της εφαρμογής περιγράφεται παρακάτω.

Αρχικά, καθώς ξεκινά η εκτέλεση της εφαρμογής, πρέπει να γίνει η επιλογή της σειριακής θύρας επικοινωνίας μεταξύ της εφαρμογής και του master – arduino. Αυτό καθίσταται δυνατό μέσω του σύνθετου πλαισίου (combo box), Port. Στη συνέχεια, και

εφόσον η σειριακή θύρα επικοινωνίας υπάρχει και αντιστοιχεί στο master - arduino, μέσω του πλήκτρου σύνδεσης, Connect, ενεργοποιείται η διασύνδεση της εφαρμογής και του master – arduino. Σε αντίθετη περίπτωση εμφανίζεται μήνυμα λάθους και η εφαρμογή απαιτεί επανεκκίνηση για την απρόσκοπτη χρήση της. Εμφανίζονται οι καρτέλες των συνδεδεμένων arduino – slaves, στην περίπτωσή μας οι καρτέλες slave01 και slave02, σύμφωνα με τα τελευταία έγκυρα δεδομένα που έλαβε η εφαρμογή. Εάν είναι επιθυμητή η άμεση ενημέρωση σχετικά με τα ενεργοποιημένα arduino - slaves ή εάν έχει γίνει κάποια αλλαγή στον αριθμό τους είναι απαραίτητο να γίνει αρχικοποίηση με το πλήκτρο Initiate.

- Λειτουργία Αποστολής (Send):

Στην παρούσα καρτέλα, η οποία εμφανίζεται στην οθόνη, γίνεται επιλογή της επιθυμητής κατάστασης των ψηφιακών θυρών επιθυμούμε (ON ή OFF) και στη συνέχεια μέσω του πλήκτρου Send ενεργοποιείται η αποστολή της εντολής με τις επιλογές στο αντίστοιχο arduino – slave ενώ επιστρέφονται οι τιμές όλων των αναλογικών θυρών του. Στην προκειμένη περίπτωση θα εμφανιστούν οι τιμές των αναλογικών θυρών που έχουν ενεργοποιηθεί μέσω των προτιμήσεων από το Menu. Επιπλέον επιστρέφονται οι τρέχουσες τιμές των ψηφιακών θυρών που επιλέχθηκαν ώστε να γίνει επαλήθευση πως η εντολή μεταφέρθηκε σωστά. Σε περίπτωση αναντιστοιχίας με τις επιλεγόμενες καταστάσεις εμφανίζεται μήνυμα λάθους στη γραμμή κατάστασης (status bar).

- Λειτουργία Καταγραφής (Record):

Σε κάθε καρτέλα υπάρχει η επιλογή Record, η οποία σχετίζεται με τη λήψη επαναληπτικών μετρήσεων. Αφού γίνει η επιλογή των επιθυμητών arduino - slaves προς λήψη επαναληπτικών μετρήσεων, καθώς και της κατάστασης των ψηφιακών θυρών στις αντίστοιχες καρτέλες, ενεργοποιείται μέσω του πλήκτρου Record η λειτουργία επαναληπτικών μετρήσεων η οποία έχει δύο ενδεχόμενα.

- Συνδεδεμένη Καταγραφή (Online Recording)

Σ' αυτήν την περίπτωση γίνεται αποστολή στο master - arduino των απαραίτητων εντολών και η εφαρμογή μπαίνει σε διαδικασία λήψης των αντίστοιχων δεδομένων από τα arduino - slaves και ενημέρωσης κάθε καρτέλας ξεχωριστά με τις πιο πρόσφατες τιμές των αναλογικών θυρών αυτών. Η λειτουργία αυτή σταματά μόνο με το πλήκτρο Stop Record. Κατά τη λειτουργία αυτή εάν υπάρξει οποιοδήποτε πρόβλημα με τη σωστή λήψη των εντολών από το master - arduino ή οποιαδήποτε αναντιστοιχία μεταξύ της επιθυμητής κατάστασης των ψηφιακών θυρών και της πραγματικής κατάστασής τους η λειτουργία διακόπτεται αυτόματα και εμφανίζεται σχετικό μήνυμα λάθους.

- Αποσυνδεδεμένη Καταγραφή (Offline Recording)

Αντίστοιχα κατά τη λήψη επαναληπτικών μετρήσεων σε κατάσταση αποσύνδεσης γίνεται αποστολή στο master - arduino των ίδιων εντολών, με τη διαφορά ότι στη συνέχεια

η εφαρμογή περνά σε κατάσταση αποσύνδεσης (Disconnect). Εάν υπάρξει κάποιο πρόβλημα κατά την επιτυχή λήψη των εντολών από το master - arduino εμφανίζεται το σχετικό μήνυμα λάθους. Στην συνέχεια οι μετρήσεις αποθηκεύονται στην κάρτα μνήμης της Data Logging Shield.

- Λειτουργία Αρχικοποίησης (Initiate)

Κατά τη λειτουργία αυτή γίνεται αποστολή στο master - arduino της απαραίτητης για αρχικοποίηση εντολής μέσω του πλήκτρου Initiate. Στη συνέχεια γίνεται λήψη των εν ενεργεία arduino - slaves καθώς και οι τιμές των αναλογικών θυρών τους κατά την τρέχουσα χρονική στιγμή. Εάν, σε οποιοδήποτε σημείο της λειτουργίας, υπάρξει κάποιο εμπόδιο στη σωστή μεταφορά των δεδομένων, εμφανίζεται το αντίστοιχο μήνυμα λάθους. Κρίθηκε σκόπιμο η λειτουργία αυτή να εμφανίζεται ξεχωριστά ως επιλογή και να μην ενσωματωθεί στη λειτουργία Connect, καθώς συνιστά μια, εν δυνάμει, αρκετά χρονοβόρα διαδικασία, ανάλογα με τις εκάστοτε συνθήκες, η οποία δεν είναι απαραίτητο να ενεργοποιείται κάθε φορά που γίνεται χρήση της εφαρμογής.

Τέλος, μέσω του πλήκτρου αποσύνδεσης, Disconnect, γίνεται διακοπή της επικοινωνίας μεταξύ της εφαρμογής και του master – arduino και η εφαρμογή επιστρέφει στην αρχική της κατάσταση.

2.3.2 Arduino - master

Κάθε πρόγραμμα του Arduino έχει την εξής δομή:

```
// Δηλώσεις μεταβλητών

void setup() {

// Αρχικοποιήσεις

}

void loop() {

// Κύριο πρόγραμμα

}

//Δηλώσεις υπολοίπων συναρτήσεων (προαιρετικά)
```

Γενικά, υπάρχουν δυο βασικές συναρτήσεις.

Η συνάρτηση setup() εκτελείται στην αρχή του προγράμματος και για μία μόνο φορά. Χρησιμοποιείται για τις αρχικοποιήσεις των μεταβλητών, τις δηλώσεις των ακροδεκτών (αν θα είναι είσοδος ή έξοδος) και τις αρχικοποιήσεις των βιβλιοθηκών.

Η συνάρτηση loop() εκτελείται συνεχώς επαναληπτικά εώς ότου αποσυνδεθεί ή επανεκκινήσει το Arduino. Εδώ γράφεται το κυρίως μέρος του κώδικα το οποίο επαναλαμβάνεται διαρκώς.

Στο Παράρτημα παρουσιάζονται αναλυτικά οι βασικές εντολές και συναρτήσεις που χρησιμοποιούνται καθώς και ο κώδικας των προγραμμάτων.

Στην παρούσα διάταξη το master – arduino είναι αυτό που λειτουργεί σαν ενδιάμεσος ανάμεσα στην εφαρμογή διεπαφής με το χρήστη και τα arduino – slaves, όταν είναι συνδεδεμένος ο υπολογιστής, και σαν master ως προς τα υπόλοιπα arduinos, όταν ο υπολογιστής βρίσκεται σε κατάσταση αποσύνδεσης.

Στη συνάρτηση setup() γίνεται η έναρξη της σειριακής επικοινωνίας μεταξύ master και slaves, καθώς και της σειριακής επικοινωνίας μέσω του πρωτοκόλλου RS485. Στη συνέχεια γίνεται η αρχικοποίηση των μεταβλητών που χρησιμοποιούνται στο κυρίως μέρος του κώδικα, τη συνάρτηση loop().

Η συνάρτηση loop() αποτελείται από τρία επιμέρους κομμάτια κώδικα. Στο πρώτο κομμάτι γίνεται έλεγχος της θύρας σειριακής επικοινωνίας για τυχόν εντολές προερχόμενες από την εφαρμογή διεπαφής χρήστη. Στην περίπτωση που υπάρχουν δεδομένα, αφού εξακριβωθεί η εγκυρότητά τους, γίνεται αποκωδικοποίηση της εντολής.

Εάν η εντολή αναφέρεται στη λειτουργία Send, δηλαδή σε αποστολή επιλογών και αίτηση αναλογικών μετρήσεων γίνεται αποστολή της εντολής και αναμένεται η απάντηση με τα δεδομένα στο τρίτο κομμάτι κώδικα, που θα εξηγηθεί παρακάτω. Εάν η εντολή αναφέρεται στη λειτουργία Online Recording γίνεται αποστολή στο αντίστοιχο arduino - slave και αναμένεται η απάντηση όπως και προηγουμένως. Εάν η εντολή αναφέρεται στην λειτουργία Offline Recording, καταγράφονται οι σταλθείσες από την εφαρμογή πληροφορίες και αποθηκεύονται στην κάρτα μνήμης, ώστε να μην χαθούν σε περίπτωση επανεκκίνησης του arduino. Τότε, και μόνον σε αυτή την περίπτωση, ενεργοποιείται το δεύτερο κομμάτι του κώδικα, το οποίο σχετίζεται με την επιλογή Offline Recording. Σ’ αυτό το σημείο αποστέλλονται επαναληπτικά προς τα arduino – slaves που έχουν επιλεγεί οι τελευταίες επιλογές ψηφιακών θυρών που έγιναν πριν την αποσύνδεση του υπολογιστή και οι λαμβανόμενες τιμές των αναλογικών θυρών καταγράφονται στην κάρτα μνήμης της sd-shield.

Τέλος στο τρίτο μέρος του κώδικα γίνεται έλεγχος της θύρας επικοινωνίας μέσω του πρωτοκόλλου RS485. Στην περίπτωση που υπάρχουν δεδομένα, αφού εξακριβωθεί η εγκυρότητά τους, γίνεται η προώθησή τους στην εφαρμογή, εάν ο υπολογιστής είναι σε κατάσταση σύνδεσης ή στην κάρτα μνήμης, εάν ο υπολογιστής είναι εκτός σύνδεσης.

Όπου είναι απαραίτητη η αποστολή δεδομένων προς τα slaves γίνεται κλήση της συνάρτησης send_command(), που δέχεται σαν όρισμα το string που έχει ληφθεί από την εφαρμογή, κατά την εκτέλεσης της οποίας το master μπαίνει σε κατάσταση transmit,

στέλνει το string που περιέχει την εντολή μέσω του πρωτόκολλου επικοινωνίας RS485 και στη συνέχεια επιστρέφει σε κατάσταση receive, όπου αναμένει την επιστροφή των δεδομένων.

2.3.3 Arduino - Slaves

Τα arduinos που λειτουργούν ως slaves είναι το τρίτο διακριτό μέρος της διάταξης και είναι αυτά που είναι συνδεδεμένοι οι αισθητήρες από τους οποίους λαμβάνονται τα επιθυμητά δεδομένα. Τα arduino – slaves έχουν πανομοιότυπο κώδικα μεταξύ τους για λόγους ομοιομορφίας, με εξαίρεση το κομμάτι που αναφέρεται στην ταυτοποίηση του κάθε slave και κυμαίνεται από 1 έως 10.

Όπως και στον αντίστοιχο κώδικα του master, στη συνάρτηση setup() γίνεται η έναρξη της σειριακής επικοινωνίας μεταξύ master και slaves, καθώς και της σειριακής επικοινωνίας μέσω του πρωτοκόλλου RS485. Στη συνέχεια γίνεται η αρχικοποίηση των μεταβλητών που χρησιμοποιούνται στο κυρίως μέρος του κώδικα, τη συνάρτηση loop().

Η συνάρτηση loop() αποτελείται από ένα βασικό κομμάτι κώδικα, στο οποίο γίνεται έλεγχος της θύρας επικοινωνίας μέσω του πρωτοκόλλου RS485. Εάν υπάρχουν δεδομένα, αφού εξακριβωθεί καταρχήν η εγκυρότητά τους, καθορίζεται το κατά πόσον αναφέρονται στο συγκεκριμένο slave. Εφόσον αυτό είναι αληθές γίνεται κλήση της συνάρτησης define_command(), που έχει σαν όρισμα string, κατά την οποία αποκωδικοποιείται η εντολή. Αρχικά οι ψηφιακές θύρες ρυθμίζονται ανάλογα με τις επιλογές που στάλθηκαν σε κατάσταση HIGH (ON) ή LOW (OFF). Στη συνέχεια μέσω της συνάρτησης digitalRead() ενημερώνεται η τρέχουσα κατάσταση των θυρών αυτών, ώστε να γίνει αργότερα στην εφαρμογή η επαλήθευση για τη σωστή αποστολή των επιλογών. Ακόμη λαμβάνονται οι τιμές από όλες τις αναλογικές θύρες και ενημερώνεται το string που αποστέλλεται προς προώθηση στο master – arduino.

Τέλος καλείται η συνάρτηση send_reply(), που δέχεται σαν όρισμα το προαναφερθέν string, κατά την εκτέλεση της οποίας το arduino – slave μεταβαίνει σε κατάσταση transmit, στέλνει την απάντηση και επιστρέφει σε κατάσταση receive, ώστε να λάβει την επόμενη εντολή από το master.

Κεφάλαιο 3: Συμπεράσματα

3.1 Συμπεράσματα

Κατά την εκπόνηση της παρούσας διπλωματικής εργασίας έγινε γρήγορα αντιληπτό το εύρος των εφαρμογών που δύνανται να βασιστούν σε μικροελεγκτές και ιδιαίτερα στην πλατφόρμα Arduino. Με την παρούσα διάταξη μετρήσεων παρέχεται η δυνατότητα λήψης απομακρυσμένων μετρήσεων με ακρίβεια, αποτελεσματικότητα και ευελιξία. Η αποθήκευση των μετρήσεων στον ηλεκτρονικό υπολογιστή καθιστά ιδιαίτερα εύκολη την περαιτέρω επεξεργασία τους, ακόμα και με μεγάλο πλήθος μετρήσεων.

Εφόσον επιτευχθεί η ομαλή σύνδεση των μερών της διάταξης με τον ηλεκτρονικό υπολογιστή, η επίβλεψη όλης της διαδικασίας γίνεται πλέον από εκεί εύχρηστα και δίχως κόπο. Επιπλέον η βασική εργασία δημιουργίας και αποκωδικοποίησης των εντολών και των μετρήσεων γίνεται μέσω της εφαρμογής εξοικονομώντας πολύτιμους πόρους από τη μνήμη των μικροελεγκτών Arduino και επιτρέποντας την σε μεγάλη κλίμακα επέκταση της εφαρμογής και της διάταξης.

Επιπρόσθετα, με τη λειτουργία του αποσυνδεδεμένου υπολογιστή και αποθήκευσης των δεδομένων στην κάρτα μνήμης του Arduino, παρέχεται η δυνατότητα επαναληπτικών μετρήσεων, με τις επιθυμητές ρυθμίσεις από το χρήστη, και η περαιτέρω επεξεργασία τους στον επιθυμητό χρόνο, αποσυνδέοντας το σύστημα από την ανάγκη συνεχούς επίβλεψης από τον ηλεκτρονικό υπολογιστή, το οποίο μπορεί να φανεί χρήσιμο στην περίπτωση μικρής κλίμακας και μεγάλης χρονικής διάρκειας μετρήσεων.

Τέλος η χρήση του μικροελεγκτή Arduino ως slave, παρέχει πληθώρα επιλογών όσον αφορά στο κομμάτι της χρήσης αισθητήρων και τον πιθανό συνδυασμό με κυκλώματα που απαιτούν χειρισμό, καθιστώντας δυνατή τη χρήση τόσο των υπάρχοντων εφαρμογών, που σχετίζονται με το μικροελεγκτή Arduino, όσο και των εφαρμογών που πρόκειται να υλοποιηθούν στο μέλλον, καθώς ο τομέας είναι δυναμικά και ταχύτατα εξελισσόμενος.

3.2 Μελλοντική εργασία

Η παρούσα εργασία δύναται να επεκταθεί στο μέλλον στους εξής τομείς:

- Την επέκταση των υποστηριζόμενων Arduino - slaves, ως σταθμούς λήψης μετρήσεων, σε 31 που είναι ο μέγιστος αριθμός που υποστηρίζει η τοπολογία του πρωτοκόλλου επικοινωνίας RS485

- Τον εμπλουτισμό της εφαρμογής διεπαφής με το χρήστη στο κομμάτι που αφορά την επεξεργασία των μετρήσεων, παρέχοντας τη δυνατότητα εμφάνισης διαγραμμάτων, στατιστικών και εργαλείων σύγκρισης των μετρήσεων.

- Την εισαγωγή του στοιχείου του αυτομάτου ελέγχου στην εφαρμογή, όπου ανάλογα με τις λαμβανόμενες μετρήσεις το σύστημα των σταθμών μετρήσεων θα ελέγχεται αυτόματα μέσω του ηλεκτρονικού υπολογιστή.

- Τη χρήση των δυνατοτήτων που παρέχουν οι πλακέτες shields του Arduino, όπως είναι η αποστολή μηνυμάτων μέσω SMS και η σύνδεση με το διαδίκτυο.

- Την αύξηση των αναλογικών και ψηφιακών εισόδων με τη χρήση άλλων εκδόσεων του Arduino, όπως το Arduino Mega

Βιβλιογραφία

[1] Ν. Θεοδώρου, «Ηλεκτρικές Μετρήσεις: Κλασσικές, Ηλεκτρονικές και Ψηφιακές», Εκδ. Συμμετρία, 2016

[2] A. Sedra – K. Smith, «Μικροηλεκτρονικά Κυκλώματα, Τόμος 1», Εκδ. Παπασωτηρίου, 2011

[3] Ν. Αβούρης, «Εισαγωγή στην Επικοινωνία Ανθρώπου-Υπολογιστή», Εκδ. Δίαυλος, 2000

[4] Α. Σκορδάς, «Ψηφιακή Επεξεργασία Εικόνων και Σημάτων», Εκδ. ΕΑΠ, 2003

[5] K. James, «PC Interfacing and Data Acquisition: Techniques for Measurement, Instrumentation and Control», Newnes, 1996

[6] J. Webster, «Measurement, Instrumentation, and Sensors Handbook», CRC Press, 1999

[7] M. Margolis, «Arduino Cookbook, Second Edition», O'Reilly Media, 2011

[8] M. Banzi, «Getting Starting With Arduino 3rd Edition», Maker Media Inc, 2014

[9] R. Northrop, «Introduction to Instrumentation and Measurements», CRC Press, 2005

[10] D. Ibrahim, «Microcontroller-Based Temperature Monitoring and Control», Elsevier, 2002

[11] C. Platt, «Encyclopedia of Electronic Components Volume 1», Maker Media Inc, 2012

[12] J. Blum, «Exploring Arduino: Tools and Techniques for Engineering Wizardry», Wiley, 2013

[13] M. Verle, «PIC Microcontrollers - Programming in BASIC, 1st edition», mikroElektronika, 2010

[14] J. Noble, «Programming Interactivity», O'Reilly Media, 2009

[15] Ελληνικό Ινστιτούτο Μετρολογίας, www.eim.gr

[16] Εθνικό Σύστημα Διαπίστευσης, www.esyd.gr

[17] Lazarus Homepage, http://www.lazarus.freepascal.org/

[18] Arduino Official Site, http://www.arduino.cc/

[19] Texas Instruments, http://www.ti.com/

[20] https://arduino-info.wikispaces.com

[21] http://www.physics.ntua.gr/

[22] http://www.lammertbies.nl/comm/info/RS-485.html

[23] AVR120, Characterization and Calibration of the ADC on an AVR, www.atmel.com/images/doc2559.pdf

[24] «High-Performance, Stand-Alone ADCs for a Variety of Embedded Systems Applications, Analog-to-Digital Converter Design Guide», www.microchip.com

[25] RS-422 and RS-485 Applications eBook - «A Practical Guide to Using RS-422 and RS-485 Serial Interfaces», http://www.bb-elec.com/

[26] LM35 Datasheet - Texas Instruments, www.ti.com/lit/ds/symlink/lm35.pdf

Παράρτημα Α: Ο προγραμματισμός της διάταξης

Α.1 Εφαρμογή Διεπαφής Χρήστη

Οι ενέργειες στη διάταξη μετρήσεων ελέγχονται πλήρως από τον χρήστη μέσω της εφαρμογής διεπαφής. Η επικοινωνία μεταξύ εφαρμογής και σταθμών μετρήσεων, μέσω του Arduino master, λαμβάνει χώρα με αποστολή και λήψη strings αυστηρά καθορισμένου μεγέθους και μορφής. Υπάρχουν τρεις βασικές λειτουργίες στο σύστημα, η λειτουργία Send, η λειτουργία Record και η λειτουργία Initiate.

- Λειτουργία Send

Τα string από την εφαρμογή προς τους σταθμούς μέτρησης είναι της μορφής Α00ΨηφιακέςΘύρες*, όπου 00 ο αριθμός του slave που κυμαίνεται από 01 μέχρι 10 και ο χαρακτήρας * λειτουργεί ως τερματικό σύμβολο, για λόγους επαλήθευσης. Η κατάσταση των ψηφιακών θυρών αποστέλλεται στο slave ως εντολή να αλλάξει την κατάσταση των ψηφιακών θυρών του ανάλογα με την σταλθείσα εντολή.

Τα string που επιστρέφουν από τους σταθμούς μέτρησης προς την εφαρμογή είναι της μορφής Β00ΨηφιακέςΘύρεςΑναλογικέςΘύρες*. Εδώ το 00 αντιστοιχεί πάλι στον αριθμό του slave. Η κατάσταση των ψηφιακών θυρών αποστέλλεται για να γίνει η επαλήθευση για την ακριβή άφιξη της εντολής. Οι τιμές των αναλογικών θυρών είναι οι επιθυμητές μετρήσεις. Αποστέλλονται όλες οι τιμές των αναλογικών θυρών και μέσα από την εφαρμογή παρουσιάζονται μόνο οι επιθυμητές, καθότι δεν είναι απαραίτητο ότι σε κάθε χρονική στιγμή θα είναι δεσμευμένες και οι 6 θέσεις αναλογικών θυρών στους σταθμούς μέτρησης.

Ανάλογα με την προηγούμενη λειτουργία και με αντίστοιχο τρόπο τα string για τις άλλες δύο λειτουργίες είναι τα εξής.

- Λειτουργία Record

Τα string από την εφαρμογή προς τους σταθμούς μέτρησης είναι της μορφής R00ΨηφιακέςΘύρες*.

Τα string από τους σταθμούς μέτρησης προς την εφαρμογή είναι της μορφής R00ΨηφιακέςΘύρεςΑναλογικέςΘύρες*.

Εδώ, για λόγους επιπλέον ασφάλειας και επαλήθευσης, έχουμε επικοινωνία ανάμεσα στην εφαρμογή και το master κατά την εκκίνηση και το τέλος της λειτουργίας. Μόλις ενεργοποιηθεί η λειτουργία από την εφαρμογή αποστέλλεται στο master το string εκκίνησης της λειτουργίας R00*. Μόλις η εφαρμογή λάβει πίσω το ίδιο string στέλνει όλα τα strings για τα slaves που πρόκειται να ληφθούν επαναληπτικές μετρήσεις. Στη συνέχεια αποστέλλεται το string ολοκλήρωσης R99* και αφού επιβεβαιωθεί η σωστή άφιξή του στο master ξεκινά η λειτουργία επαναληπτικών μετρήσεων.

- Λειτουργία Initiate

Από την εφαρμογή προς τους σταθμούς μέτρησης αποστέλλεται το string εκκίνησης της λειτουργίας I**.

Από τους σταθμούς μέτρησης προς την εφαρμογή επιστρέφει το string της μορφής I00ΨηφιακέςΘύρεςΑναλογικέςΘύρες*, όπου 00 ο αριθμός του slave, στη συνέχεια η κατάσταση των ψηφιακών θυρών και τέλος οι επιθυμητές μετρήσεις.

A.2 Βασικές Εντολές του Arduino

Οι βασικές εντολές και παράμετροι που χρησιμοποιήθηκαν κατά τον προγραμματισμό της πλατφόρμας Arduino περιγράφονται στον πίνακα που ακολουθεί.

Όρισμα	Είδος	Τύπος	Παράμετροι	Περιγραφή
LOW	Σταθερά	int	-	Έχει την τιμή 0 και είναι αντίστοιχη του λογικού false.
HIGH	Σταθερά	int	-	Έχει την τιμή 1 και είναι αντίστοιχη του λογικού true.
INPUT	Σταθερά	int	-	Έχει την τιμή 0 και είναι αντίστοιχη του λογικού false.
OUTPUT	Σταθερά	int	-	Έχει την τιμή 1 και είναι αντίστοιχη του λογικού true.
pinMode	Εντολή	-	(*pin*, *mode*)	Καθορίζει αν το συγκεκριμένο ψηφιακό *pin* θα είναι pin εισόδου ή pin εξόδου ανάλογα με την τιμή που δίνεται στην παράμετρο *mode* (INPUT ή OUTPUT αντίστοιχα).
digitalWrite	Εντολή	-	(*pin*, *pinstatus*)	Θέτει την κατάσταση *pinstatus* (HIGH ή LOW) στο συγκεκριμένο ψηφιακό *pin*.
digitalRead	Συνάρτηση	int	(*pin*)	Επιστρέφει την κατάσταση του συγκεκριμένου ψηφιακού *pin* (0 για LOW και 1 για HIGH) εφόσον αυτό είναι pin εισόδου.
analogReference	Εντολή	-	(*type*)	Δέχεται τις τιμές DEFAULT, INTERNAL ή EXTERNAL στην παράμετρο *type* για να καθορίσει την τάση αναφοράς (V_{ref}) των αναλογικών εισόδων (5V, 1.1V ή η εξωτερική τάση με την οποία τροφοδοτείται το pin AREF αντίστοιχα)

analogRead	Συνάρτηση	int	(*pin*)	Επιστρέφει έναν ακέραιο από 0 έως 1023, ανάλογα με την τάση που τροφοδοτείται το συγκεκριμένο *pin* αναλογικής εισόδου στην κλίμακα 0 ως Vref.
analogWrite	Εντολή	-	(*pin*, *value*)	Θέτει το συγκεκριμένο ψηφιακό *pin* σε κατάσταση ψευδοαναλογικής εξόδου (PWM). Η παράμετρος *value* καθορίζει το πλάτος του παλμού σε σχέση με την περίοδο του παραγόμενου σήματος στην κλίμακα από 0 ως 255 (π.χ. με *value* 127, το πλάτος του παλμού είναι ίσο με μισή περίοδο).
millis	Συνάρτηση	unsigned long	()	Μετρητής που επιστρέφει το χρονικό διάστημα σε ms από την στιγμή που άρχισε η εκτέλεση του προγράμματος. Λάβετε υπόψη ότι λόγω του τύπου μεταβλητής (unsigned long δηλ. 32bit) θα γίνει overflow σε 2^32ms δηλαδή περίπου σε 50 μέρες, οπότε ο μετρητής θα ξεκινήσει πάλι από το μηδέν.
delay	Εντολή	-	(*time*)	Σταματά προσωρινά την ροή του προγράμματος για *time* ms. Η παράμετρος *time* είναι unsigned long (από 0 ως 2^32). Σημειώστε ότι παρά την προσωρινή παύση, συναρτήσεις των οποίων η εκτέλεση ενεργοποιείται από interrupt θα εκτελεστούν κανονικά κατά την διάρκεια μιας delay.
attachInterrupt	Εντολή	-	(*interrupt*, *function*, *triggermode*)	Θέτει σε λειτουργία το συγκεκριμένο *interrupt*, ώστε να ενεργοποιεί την συνάρτηση *function*, κάθε φορά που ικανοποιείται η συνθήκη που ορίζεται από την παράμετρο *triggermode*: • LOW (ενεργοποίηση όταν η κατάσταση του pin που αντιστοιχεί στο συγκεκριμένο interrupt γίνει LOW)

				• RISING (όταν από LOW γίνει HIGH) • FALLING (όταν από HIGH γίνει LOW) • CHANGE (όταν αλλάξει κατάσταση γενικά)
detachInterrupt	Εντολή	-	(interrupt)	Απενεργοποιεί το συγκεκριμένο *interrupt*.
noInterrupts	Εντολή	-	()	Σταματά προσωρινά την λειτουργία όλων των interrupt
interrupts	Εντολή	-	()	Επαναφέρει την λειτουργία των interrupt που διακόπηκε προσωρινά από μια εντολή noInterrupts.
Serial.begin	Μέθοδος κλάσης	-	(*datarate*)	Θέτει τον ρυθμό μεταφοράς δεδομένων του σειριακού interface (σε baud)
Serial.println	Μέθοδος κλάσης	-	(*data*)	Διοχετεύει τα δεδομένα *data* για αποστολή μέσω του σειριακού interface. Η παράμετρος *data* μπορεί να είναι είτε αριθμός είτε αλφαριθμητικό.

Α.3 Ο κώδικας για το Arduino σε λειτουργία Master

Ο κώδικας προγραμματισμού του Arduino Master παρουσιάζεται παρακάτω.

```
/*-----( Import needed libraries )-----*/
#include <SoftwareSerial.h>
#include <SPI.h>
#include <SD.h>
/*-----( Declare Constants and Pin Numbers )-----*/
#define SSerialRX          7  //Serial Receive pin
#define SSerialTX          8  //Serial Transmit pin
#define SSerialTxControl   3   //RS485 Direction control
#define RS485Transmit        HIGH
#define RS485Receive         LOW
#define num_slaves         10  //number of slaves
#define SS                 10
#define MOSI               11
#define MISO               12
```

```
#define CLK                     13
const int chipSelect = 4;   //can be changed
File myFile;
File SendRecord;
File ReceivedRecord;
/*-----( Declare objects )-----*/
SoftwareSerial RS485Serial(SSerialRX, SSerialTX); // RX, TX
/*-----( Declare Variables )-----*/
…
void setup()
{
  // Open serial communications
  Serial.begin(9600);
  pinMode(SSerialTxControl, OUTPUT);
  digitalWrite(SSerialTxControl, RS485Receive);  // Init Transceiver
  // Start the software serial port, to another device
  RS485Serial.begin(4800);   // set the data rate
  pinMode(SS, OUTPUT);
  //initialize flags and counters
...
  if (str_comp == "*")
  {
   str_RecordOffline = stringReceivedRecord_PC.substring(35, 36);  //RecordOffline
   if (str_RecordOffline == "1")  //Offline
   {
    flag_rec = 1; //initiate Record1
    flag_rec2 = 1;
    flag_RecordOffline = 1; //initiate RecordOffline
   }
  }
}//--(end setup )---

void loop()
{
Start1:
  if (Serial.available() > 0)
  {
   stringReceived_PC = Serial.readString();
   str_comp = "";    //reset string
   str_comp = stringReceived_PC.substring(0, 1);    //split the first char of the string
   if (str_comp == "A")    //check if it is for Send command
   {
    str_comp = stringReceived_PC.substring(16, 17);  //check if the last char of string is '*'
    if (str_comp == "*")
    {
     flag_init = 0;    //release Initiate1
     flag_rec = 0; //release Record1
```

```
 flag_RecordOffline = 0; //release RecordOffline
 flag_RecordOffline_possible = 0; //release RecordOffline_possible
 stringSend_slave = stringReceived_PC;
 str_slave_num = stringSend_slave.substring(1, 3);   //save slave number
 send_command(stringSend_slave);  // Send command to Remote Arduino
 }
}
else if (str_comp == "R") //check if it is for Record command
{
 str_comp = stringReceived_PC.substring(1, 3);
 if (str_comp == "**")   //Record stop
  {
   str_comp = stringReceived_PC.substring(3, 4);  //check if the last char of string is '*'
   if (str_comp == "*")
   {
    flag_rec = 0; //release Record1
    flag_RecordOffline = 0; //release RecordOffline
    flag_RecordOffline_possible = 0; //release RecordOffline_possible
    Serial.flush();
    RS485Serial.flush();
    Serial.print(stringSendRecord_stop_PC);
    Serial.write(termination_char);
   }
  }
 int_slave_num = str_comp.toInt();
 switch (int_slave_num)
 {
  case 0:   //stringReceivedRecord_PC
        str_comp = stringReceived_PC.substring(36, 37);  //check if the last char of string is '*'
        if (str_comp == "*")
        {
         str_RecordOffline = stringReceived_PC.substring(35, 36);  //RecordOffline
         //store str_RecordOffline
         if (str_RecordOffline == "0") //Online
         {
          flag_rec = 0; //release Record1
          flag_RecordOffline = 0; //release RecordOffline
          flag_RecordOffline_possible = 0; //release RecordOffline_possible
         }
         else if (str_RecordOffline == "1")  //Offline possibly
         {
          flag_RecordOffline_possible = 1; //initiate RecordOffline_possible
         }
         flag_rec = 0; //release Record because new record command is arriving
         stringReceivedRecord_PC = "";
         stringSendRecord_slave01 = "";
         stringReceivedRecord_PC = stringReceived_PC;
```

```
          Serial.flush();
          RS485Serial.flush();
          Serial.print(stringSendRecord_begin_PC);
          Serial.write(termination_char);
         }
     break;
   case 1:  //slave 01
        str_comp = stringReceived_PC.substring(16, 17);  //check if the last char of string is '*'
        if (str_comp == "*")
        {
         switch (flag_RecordOffline_possible)
         {
          case 0:   //Online
               flag_init = 0;    //release Initiate1
               flag_rec = 0; //release Record1
               flag_RecordOffline = 0; //release RecordOffline
               stringSend_slave = stringReceived_PC;
               str_slave_num = "01";   //save slave number
               send_command(stringSend_slave);  // Send command to Remote Arduino
            break;
          case 1:  //Offline possibly
               stringSendRecord_slave01 = stringReceived_PC;
            break;
          default:;
         }
        }
     break;
...
   case 99:  //end
...
   }
      flag_init = 1; //initiate Initiate1
      flag_init2 = 1; //initiate Initiate1
      flag_RecordOffline = 0; //release RecordOffline
      flag_RecordOffline_possible = 0; //release RecordOffline_possibly
      counter_init = 1;
      counter_init_delay = 0;
      counter_init_delay_repeat = 1;
      stringSendRecord_slave01 = ""
      Serial.flush();
      Serial.print(stringSendInitiate_begin_PC);
      Serial.write(termination_char);
     }
Initiate1:
  if (flag_init == 1)
  {
   if (counter_init_delay == num_loops_max_init)
```

```
{
 flag_init2 = 1;
 counter_init_delay = 1;
 counter_init_delay_repeat++;
 Serial.flush();
 if (counter_init_delay_repeat == num_repeat_command_max_init + 1)
 {
  counter_init_delay = 0;
  counter_init_delay_repeat = 1;
  counter_init++;
 }
 counter_init_delay++;
 if (flag_init2 == 1)
 {
  switch (counter_init)
  {
   case 1:                          //stringSendInitiate_slave01
        send_command(stringSendInitiate_slave01);
        delay(10);
        flag_init2 = 0;
    break;
   ...
   default:
   {
    counter_init = 1;
    counter_init_delay = 0;
    counter_init_delay_repeat = 1;
    flag_init2 = 1;
 }
Record1:
 if (flag_rec == 1)
 {
  if (counter_rec_delay == num_loops_max)
  {
   flag_rec2 = 1;
   counter_rec_delay = 1;
   counter_rec_delay_repeat++;
   if (counter_rec_delay_repeat == num_repeat_command_max + 1)
   {
    counter_rec_delay = 0;
    counter_rec_delay_repeat = 1;
    flag_rec2 == 1;
    counter_rec++;
    if (counter_rec == num_slaves){
     counter_rec = 1;
  }
  counter_rec_delay++;
```

```
   if (flag_rec2 == 1)
   {
    delay(10);
    switch (counter_rec)
    {
    case 1:                         //stringSendRecord_slave01
          val_rec = stringReceivedRecord_PC.charAt(3);
          if (val_rec == '1')
          {
           flag_rec2 = 0;
           send_command(stringSendRecord_slave01);
           delay(10);
          }
          else
          {
           counter_rec_delay = 0;
           counter_rec_delay_repeat = 1;
           flag_rec2 == 1;
           counter_rec++;
          }
      break;
    ...
    default:
    {
     counter_rec = 1;
     counter_rec_delay = 0;
     counter_rec_delay_repeat = 1;
     flag_rec2 = 1;
  }
Search1:
  if (RS485Serial.available()) //Look for data from other Arduino
  {
   delay(10);
   stringReceived_slave = RS485Serial.readString();    //save incoming string to stringReceived
   delay(10);
   str_comp = "";    //reset string
   str_comp = stringReceived_slave.substring(0, 1);    //split the first char of the string
   if (str_comp == "B")
   {
    str_comp = stringReceived_slave.substring(46, 47);  //check if the last char of string is '*',
else repeat command
    if (str_comp == "*")
    {
     str_slave_num = stringReceived_slave.substring(1, 3);   //save slave number
     Serial.flush();
     delay(10);
     RS485Serial.flush();
```

```
    delay(10);
    store_data();     //store data to sd card
    Serial.print(stringReceived_slave);
    Serial.write(termination_char);
   }
   else
   {
    Serial.flush();
    delay(10);
    RS485Serial.flush();
    delay(10);
    send_command(stringSend);
   }
  }
  else if (str_comp == "R")
  {
   str_comp = stringReceived_slave.substring(46, 47);  //check if the last char of string is '*',
else repeat command
   if (str_comp == "*")
   {
    str_slave_num = stringReceived_slave.substring(1, 3);   //save slave number

    flag_rec2 = 1;
    counter_rec_delay = 0;
    counter_rec_delay_repeat = 1;
    Serial.flush();
    delay(10);
    RS485Serial.flush();
    delay(10)
    switch (flag_RecordOffline)
    {
     case 0:   //Online
          Serial.print(stringReceived_slave);
          Serial.write(termination_char);
       break;
     case 1:  //Offline
          {
           if (flag_sdCard_ready == 1)
           {
            store_ReceivedRecord();
           }
          }
       break;
     default:{  ;
         }
    counter_rec++;
    if (counter_rec == num_slaves + 1)
```

```
    {
     counter_rec = 1;
    }
    else {
     Serial.flush();
     delay(10);
     RS485Serial.flush();
     delay(10);
    }
   else if (str_comp == "I")
   {
    str_comp = stringReceived_slave.substring(46, 47);  //check if the last char of string is '*',
else repeat command
    if (str_comp == "*")
    {
     str_slave_num = stringReceived_slave.substring(1, 3);   //save slave number
     flag_init2 = 1;
     counter_init_delay = 0;
     counter_init_delay_repeat = 1;
     stringSendInitiate_total_PC.setCharAt(counter_init + 2, '1');
     Serial.flush();
     delay(10);
     RS485Serial.flush();
     delay(10);
     store_data();     //store data to sd card
     Serial.print(stringReceived_slave);
     Serial.write(termination_char);
     counter_init++;
    }
    else
    {
     Serial.flush();
     delay(10);
     RS485Serial.flush();
     delay(10);
     send_command(stringSend);
   }
}//--(end main loop )---
/*-----( Declare User-written Functions )-----*/
void send_command(String outStr)  //send command to slave
{
  digitalWrite(SSerialTxControl, RS485Transmit);  // Enable RS485 Transmit
  RS485Serial.print(outStr);
  delay(10);
  RS485Serial.flush();
  delay(10);
  stringSend = outStr; //backup if repeat command is necessary
```

```
  digitalWrite(SSerialTxControl, RS485Receive);  // Disable RS485 Transmit
}
void sdCard_initialize() //initialize sd card
{
…
}
void store_data() //store data to sd card
{
  …
}
//End of program.
```

A.4 Ο κώδικας για τα Arduino σε λειτουργία Slave

Παρακάτω παρουσιάζεται ο κώδικας προγραμματισμού των Arduino Slaves.

```
/*-----( Import needed libraries )-----*/
#include <SoftwareSerial.h>
/*-----( Declare Constants and Pin Numbers )-----*/
#define SSerialRX        10  //Serial Receive pin
#define SSerialTX        11  //Serial Transmit pin
#define SSerialTxControl 3   //RS485 Direction control
#define RS485Transmit    HIGH
#define RS485Receive     LOW
#define Pin7LED          7
#define Pin8LED          8
#define Pin13LED         13
/*-----( Declare objects )-----*/
SoftwareSerial RS485Serial(SSerialRX, SSerialTX); // RX, TX
/*-----( Declare Variables )-----*/
...
void setup()
{
  Serial.begin(9600);
  pinMode(Pin7LED, OUTPUT);
  pinMode(Pin8LED, OUTPUT);
  pinMode(Pin13LED, OUTPUT);
  pinMode(SSerialTxControl, OUTPUT);
  digitalWrite(SSerialTxControl, RS485Receive);  // Init Transceiver
  // Start the software serial port, to another device
  RS485Serial.begin(4800);   // set the data rate
}//--(end setup )---

void loop()
{
Search1:
  if (RS485Serial.available())
```

```
 {
  stringReceived = RS485Serial.readString();   //save incoming string to stringReceived
  str_comp = "";   //reset string
  str_comp = stringReceived.substring(0, 1);     //split the first char of stringReceived
  if (str_comp == "A")
  {
   str_comp = stringReceived.substring(1, 3);
   if (str_comp == str_slaveNumber)
   {
    str_comp = stringReceived.substring(16, 17);   // "*"
    if (str_comp == "*")
    {
     stringSend = "B";  //reset stringSend
     define_command(stringReceived);
     send_reply(stringSend);  //send reply to master arduino after executing the command
  }
  else if (str_comp == "R")
  {
   str_comp = stringReceived.substring(1, 3);
   if (str_comp == str_slaveNumber)
   {
    str_comp = stringReceived.substring(16, 17);   // "*"
    if (str_comp == "*")
    {
     stringSend = "R";  //reset stringSend
     define_command(stringReceived);
     send_reply(stringSend);  //send reply to master arduino after executing the command
  }
  else if (str_comp == "I")
  {
   str_comp = stringReceived.substring(1, 3);
   if (str_comp == str_slaveNumber)
   {
    str_comp = stringReceived.substring(16, 17);   // "*"
    if (str_comp == "*")
    {
     stringSend = "I";  //reset stringSend
     define_command(stringReceived);
     send_reply(stringSend);  //send reply to master arduino after executing the command
  }
 }// End If RS485SerialAvailable
}//--(end main loop )---
/*-----( Declare User-written Functions )-----*/
void define_command(String inStr)  //define the command and execute
{
...
}
```

```
void send_reply(String outStr)  //send reply to master arduino after executing the commands
{
  digitalWrite(SSerialTxControl, RS485Transmit);  // Enable RS485 Transmit
  RS485Serial.print(outStr); // Send reply to master arduino
  digitalWrite(SSerialTxControl, RS485Receive);  // Disable RS485 Transmit
}

//End of program.
```