ΑΛΕΞΑΝΔΡΕΙΟ ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ
ΘΕΣΣΑΛΟΝΙΚΗΣ (Α.Τ.Ε.Ι.Θ.)
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ (ΣΤΕΦ)
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΑΥΤΟΜΑΤΙΣΜΟΥ Τ.Ε.

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

" On-line μέτρηση και καταγραφή κρίσιμων δεδομένων παραγωγικής διαδικασίας κατά την παραγωγή Ελαιολάδου στο Ελαιοτριβείο του ΑΤΕΙΘ "

ΣΑΛΒΑΡΑΣ ΠΑΥΛΟΣ
Αριθμός Μητρώου ΑΜ: 123045

ΕΠΙΒΛΕΠΟΝΤΕΣ ΚΑΘΗΓΗΤΕΣ: ΔΗΜΗΤΡΙΟΣ ΜΠΕΧΤΣΗΣ, ΦΩΤΙΟΣ ΣΤΕΡΓΙΟΠΟΥΛΟΣ

ΘΕΣΣΑΛΟΝΙΚΗ, ΙΟΥΛΙΟΣ 2017

Η παρούσα Πτυχιακή Εργασία και τα συμπεράσματά της, σε οποιαδήποτε μορφή, αποτελούν συνιδιοκτησία του Τμήματος Μηχανικών Αυτοματισμολυ Τ.Ε. του Αλεξάνδρειου ΤΕΙ Θεσσαλονίκης και του φοιτητή. Οι προαναφερόμενοι διατηρούν το δικαίωμα ανεξάρτητης χρήσης και αναπαραγωγής (τμηματικά ή συνολικά) για διδακτικούς και ερευνητικούς σκοπούς. Σε κάθε περίπτωση πρέπει να αναφέρεται ο τίτλος, ο συγγραφέας, ο επιβλέπων και το τμήμα του ΑΤΕΙΘ.
Η έγκριση της παρούσας Πτυχιακής Εργασίας από το τμήμα Μηχανικών Αυτοματισμού Τ.Ε. δεν υποδηλώνει απαραιτήτως και αποδοχή των απόψεων του συγγραφέα εκ μέρους του Τμήματος.

.....

Ο υπογεγραμμένος δηλώνω υπεύθυνα ότι η παρούσα Πτυχιακή Εργασία είναι εξ' ολοκλήρου δικό μου έργο και συγγράφηκε ειδικά για τις απαιτήσεις του προγράμματος σπουδών του Τμήματος Μηχανικών Αυτοματισμού Τ.Ε.

Δηλώνω υπεύθυνα ότι κατά τη συγγραφή ακολούθησα την πρέπουσα ακαδημαϊκή δεοντολογία αποφυγής λογοκλοπής και έχω αποφύγει οποιαδήποτε ενέργεια που συνιστά παράπτωμα λογοκλοπής.

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

" On-line μέτρηση και καταγραφή κρίσιμων δεδομένων παραγωγικής διαδικασίας κατά την παραγωγή Ελαιολάδου στο Ελαιοτριβείο του ΑΤΕΙΘ "

Περίληψη

Σκοπός αυτής της διπλωματικής εργασίας είναι η παρουσίαση του συστήματος τηλεμετρίας παρακολούθησης που έχει υλοποιηθεί και εφαρμοστεί σε εγκατάσταση μικρής βιομηχανικής κλίμακας ελαιολάδου που βρίσκεται στο ΑΤΕΙΘ. Η αρχιτεκτονική σχεδιασμού του συστήματος βασίζεται στο ευρύτερο τεχνολογικό πλαίσιο παρακολούθησης των διαδικασιών παραγωγής στον τομέα των γεωργικών προϊόντων και τροφίμων, ένα πλαίσιο που μπορεί να προσφέρει σημαντική πρόοδο στη ζωή του εξοπλισμού και την ποιότητα του προϊόντος και επιπλέον να βελτιώσει την ιχνηλασιμότητα στο πλαίσιο της αλυσίδας εφοδιασμού . Επιπλέον, η αυτοματοποίηση και ο έλεγχος είναι μεγάλης σημασίας για βιώσιμα και ευέλικτα συστήματα παραγωγής, προκειμένου να ελαχιστοποιηθεί το κόστος και να μειωθεί ο χρόνος παραγωγής. Συνδεμένος με όσα περιγράψαμε, είναι ο σχεδιασμός ενός αποτελεσματικού εργαλείου λογισμικού παρακολούθησης και ελέγχου (MCT) που συγκεντρώνει τα δεδομένα λειτουργίας της εγκατάστασης παραγωγής ελαιολάδου και παρέχει τις αντίστοιχες λειτουργίες charting (διαγράμματα) και πληροφόρησης. Τα αρχικά αποτελέσματα και οι πληροφορίες που συγκεντρώθηκαν συνιστούν μια στέρεη βάση για να βρεθεί μια μελλοντική πλήρους κλίμακας εφαρμογή του προτεινόμενου εργαλείου που μπορεί επίσης να είναι υπεύθυνη όχι μόνο για την απεικόνιση των τιμών αλλά και για τις λειτουργίες ελέγχου σε όλα τα στάδια της παραγωγής.

Abstract

The objective of this bachelor thesis is the presentation of the monitoring - telemetry system which has been implemented and applied in a small industrial scale olive oil facility located at ATEITH, the Alexander Technological Educational Institute of Thessaloniki. The design architecture of the system is based on the wider technological context of monitoring the production processes in the agricultural and food sector, a context that can provide significant advances in equipment life and product quality and furthermore improve visibility and trackability in the digital supply chain context. Additionally, automation and control are of major importance for sustainable and flexible production systems in order to minimize costs and reduce down time. Linked to all the previous is the design of an efficient Monitoring and Control software Tool (MCT) that gathers the operation data of the olive oil production facility and provides the corresponding charting and information functions. Initial results and information obtained constitute a solid basis on which to found a future full scale application of the proposed tool that can also be responsible not only for the charting but also for the control functions to all the stages of the production.

Ευχαριστίες

Στο σημείο αυτό θα ήθελα να ευχαριστήσω όλους όσοι συνέβαλαν στην ολοκλήρωση της διπλωματικής αυτής εργασίας και συγκεκριμένα:

- τους καθηγητές κ.Μπεχτσή Δημήτριο και κ.Στεργιόπουλο Φώτιο για την εμπιστοσύνη στο να μου αναθέσουν την παρούσα πτυχιακή εργασία και την καθοδήγησή τους, καθώς και την καθηγήτρια κ.Καλογιάννη Ελένη για την παραχώρηση του Ελαιουργείου του ΑΤΕΙΘ για τις πρακτικές δοκιμές και για τις πολύτιμες συμβουλές της.
- τους φοιτητές Μενεξέ Ιωάννη και Κατικαρίδη Δημήτριο για την δημιουργία του προγράμματος εποπτείας το οποίο χρησιμοποίησα και επεξεργάστηκα για τις ανάγκες της πτυχιακής εργασίας και για την πολύτιμη βοήθειά τους όπου χρειαζόταν, καθώς και τον φοιτητή Κοντοβό Γεώργιο για την βοήθεια του στο κομμάτι της ιστοσελίδας.
- την οικογένειά μου και τους φίλους/φίλες μου

Θεσσαλονίκη, Ιούλιος 2017

Σαλβαράς Παύλος

ΠΕΡΙΕΧΟΜΕΝΑ

1. Κεφάλαιο: Εισαγωγή 1
2. Κεφάλαιο: Αρχιτεκτονική υλοποίησης της πτυχιακής εργασίας 2
3. Κεφάλαιο: Εξοπλισμός, συνδέσεις και περιγραφή υλικών 8
3.1 Ο μικροελεγκτής Arduino 9
3.2 Συσκευές απεικόνισης και ελέγχου (TLK31) και αισθητήρες θερμοκρασίας (PT100) 15
3.3 RS-485 module και καλώδια 20
3.4 APC220 module και ραδιοκύμματα 26
3.5 Αισθητήρας ροής 34
4. Κεφάλαιο: Ανάλυση λειτουργιών 38
4.1 Κώδικας Arduino (MODBUS RTU protocol) 38
4.2 Ανάλυση λειτουργίας εφαρμογής εποπτείας 50
4.3 Ανάλυση λειτουργίας ιστοσελίδας 65
5. Κεφάλαιο: Τεχνική ανάλυση υλοποίησης 71
5.1 Τεχνική ανάλυση εφαρμογής εποπτείας 71
5.2 Τεχνική ανάλυση ιστοσελίδας 78
6. Επόμενα βήματα 85
7. Βιβλιογραφία 86
Συγγράμματα 86
Ιστοσελίδες 86

ΠΙΝΑΚΑΣ ΕΙΚΟΝΩΝ

Εικόνα 1- Γενική αρχιτεκτονική υλοποίησης 3
Εικόνα 2- Πρώτο και δεύτερο επίπεδο 4
Εικόνα 3- Τρίτο επίπεδο 5
Εικόνα 4 - Τέταρτο επίπεδο 6
Εικόνα 5 - Πέμπτο επίπεδο 7
Εικόνα 6 - Τα πιο γνωστά είδη Arduino: 1 Uno, 2 Nano, 3 Micro, 4 Lily, 5 Leonardo, 6 Mega, 7 Yun ... 10
Εικόνα 7 - Arduino Mega 2560 11
Εικόνα 8 - Περιβάλλον ανάπτυξης κώδικα 13
Εικόνα 9 - Τα πιο γνωστά Shields για Arduino: 1 Relay, 2 RS485, 3 microSD card, 4 MP3, 5, Ethernet, 6 GSM/GPRS, 7 WiFi, 8 Weather, 9 Motor drive 14
Εικόνα 10 - Αισθητήρας PT100 16

1. Κεφάλαιο: Εισαγωγή

Για να γίνουν αντιληπτές οι έννοιες της εποπτείας και της τηλεμετρίας - οι δύο βασικές έννοιες στις οποίες βασίστηκε η πτυχιακή εργασία - μπορούμε να εστιάσουμε στους εξής γενικούς ορισμούς:

Εποπτεία (monitoring) είναι η διαδικασία μη παρεμβατικού ελέγχου και παρακολούθησης ώστε να διαπιστωθεί - εξασφαλιστεί ότι μία ή περισσότερες διαδικασίες λειτουργούν σύμφωνα με συγκεκριμένες προδιαγραφές.

Τηλεμετρία (telemetry) είναι η μέθοδος κατά την οποία συλλέγονται, επεξεργάζονται και στέλνονται από απόσταση όλα τα στοιχεία, τα δεδομένα και οι πληροφορίες που είναι απαραίτητες για να εφαρμοστεί η εποπτεία.

Ο συνδυασμός αυτών των αποτελεί και τον κεντρικό άξονα της πτυχιακής: Εποπτεία και τηλεμετρία είναι αντίστοιχα η διαδικασία και η μέθοδος γύρω από τις οποίες υλοποιήθηκε η εργασία. Αρχικά, να γίνεται έλεγχος και παρακολούθηση τιμών - κρίσιμων δεδομένων που μας ενδιαφέρουν, με τη βοήθεια απεικονιστικών λειτουργιών, τα οποία δεδομένα είναι προϊόν συλλογής, επεξεργασίας και αποστολής από απόσταση, με αποτέλεσμα να μπορεί να έχει πρόσβαση στις λειτουργίες της εποπτείας ένας χρήστης που θα βρίσκεται σε απομακρυσμένο σημείο.

2. Κεφάλαιο: Αρχιτεκτονική υλοποίησης της πτυχιακής εργασίας

Σε αυτό το κεφάλαιο θα αναλυθεί η αρχιτεκτονική υλοποίησης της πτυχιακής εργασίας. Θα παρουσιαστεί το γενικό πλαίσιο μέσα στο οποίο κινήθηκε ο σχεδιασμός του συστήματος της τηλεμετρίας και το γενικό πλάνο πάνω στο οποίο βασίστηκε η υλοποίηση της πτυχιακής εργασίας.

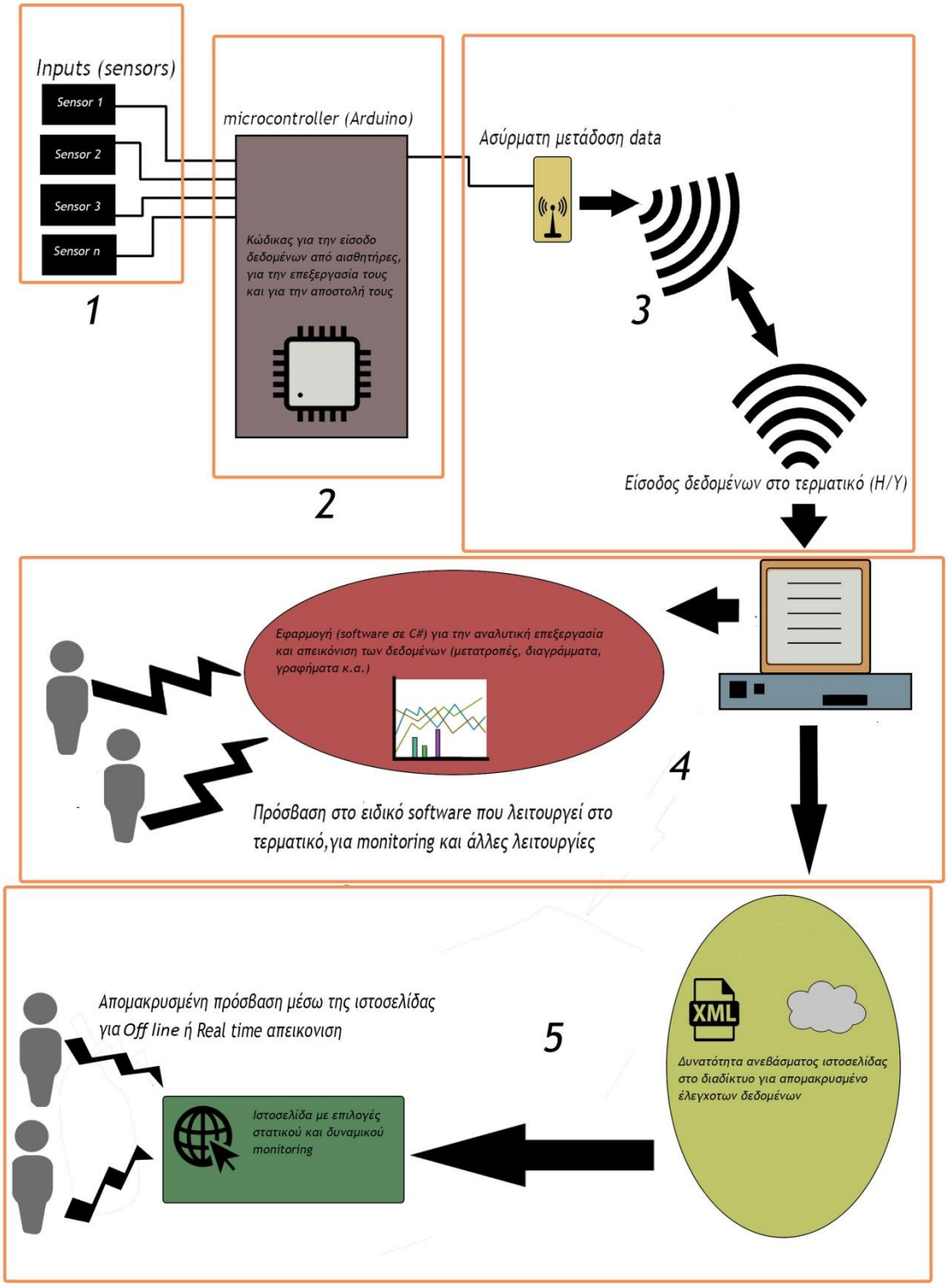
Σκοπός της πτυχιακής ήταν ο σχεδιασμός ενός συστήματος τηλεμετρίας στο ελαιουργείο του ΑΤΕΙΘ, ώστε να είναι δυνατή η παρακολούθηση τιμών θερμοκρασίας και ροής σε ένα τοπικό τερματικό (Η/Υ) αλλά και σε απομακρυσμένα τερματικά μέσω διαδικτύου. Οι τιμές που με διάφορες τεχνικές συλλέγονται από τη μονάδα του ελαιουργείου, στέλνονται στον Η/Υ όπου εκτελείται ειδική εφαρμογή για την επεξεργασία και την παρουσίαση των τιμών και ταυτόχρονα επικοινωνεί με έναν web server , ώστε να υπάρχει πρόσβαση στις μετρήσεις μέσω ιστοσελίδας και από απομακρυσμένο σημείο.

Ο σχεδιασμός της πτυχιακής που περιλαμβάνει την υλοποίηση του συστήματος τηλεμετρίας, χωρίζεται σε πέντε βασικά επίπεδα. Αυτά τα επίπεδα είναι και οι βασικοί άξονες στους οποίους χωρίζεται η φιλοσοφία της υλοποίησης, κάτι που αποτελεί ένα πλαίσιο σχεδιασμού το οποίο μπορεί να εφαρμοστεί σε ένα πλήθος αντίστοιχων projects, εργασιών κλπ. Η αρχιτεκτονική αυτή επομένως δεν υφίσταται μόνο μέσα στα όρια της πτυχιακής, αλλά αποτελεί έναν άξονα γενικότερο, που μπορεί να τροποποιηθεί και να εφαρμοστεί στην παρακολούθηση βιομηχανικών χώρων.

Τα πέντε αυτά βασικά επίπεδα της αρχιτεκτονικής που ακολουθήθηκε είναι τα εξής:

- Το υλικό (πχ αισθητήρες) που παρέχει τα δεδομένα.
- Η συγκέντρωση των δεδομένων (τιμών των αισθητήρων) σε έναν μικροελεγκτή.
- Η μετάδοση αυτών των δεδομένων ασύρματα ή ενσύρματα σε ένα τερματικό (στην πτυχιακή χρησιμοποιήθηκαν κεραίες για ασύρματη μετάδοση σε Η/Υ)
- Η παρακολούθηση (monitoring) και επεξεργασία των δεδομένων στο τερματικό σε πραγματικό χρόνο, μέσω ειδικής εφαρμογής και η χρήση τους για offline επεξεργασία
- Η online εποπτεία (monitoring) των δεδομένων, μέσω διαδικτυακής διεπαφής.

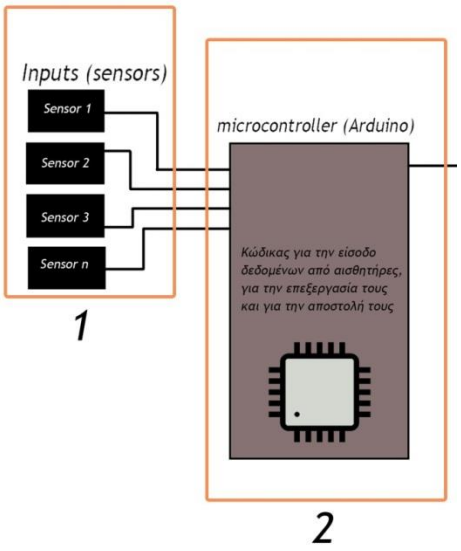
Πιο αναλυτικά μπορούμε να δούμε την ανάλυση της αρχιτεκτονικής αυτής και των πέντε βασικών αξόνων - επιπέδων στην παρακάτω εικόνα. Έπειτα θα αναλυθεί το κάθε στάδιο καθώς και οι απαιτήσεις του ξεχωριστά.



Εικόνα 1- Γενική αρχιτεκτονική υλοποίησης

Επίπεδο 1

Στο πρώτο επίπεδο έχουμε τους αισθητήρες. Οι αισθητήρες αποτελούν τη βάση ενός συστήματος τηλεμετρίας αφού πάνω σε αυτούς κτίζεται ολόκληρο το σύστημα. Είναι συσκευές που τοποθετούνται σε ένα σύστημα ώστε να ανιχνεύσουν την παρουσία και την τιμή ενός φυσικού μεγέθους και παράγουν μία έξοδο η οποία είναι μετρήσιμη και ουσιαστικά αποτελούν τον συνδετικό κρίκο μεταξύ των φυσικών μεγεθών και του ανθρώπινου παράγοντα. Είναι απαραίτητη η σωστή επιλογή αισθητηρίων ανάλογα με τις ανάγκες και τις περιστάσεις του έργου βάσει διαφόρων χαρακτηριστικών όπως η ανοχή τους, το εύρος τους, το σφάλμα, η ακρίβεια, η ευαισθησία, η γραμμικότητα κ.α. Η έξοδος των αισθητηρίων, είναι η είσοδος μας στο σύστημα τηλεμετρίας. ΄Ο,τι μετράνε, εμείς το παίρνουμε ως είσοδο σε μια πρώιμη μορφή δεδομένων και αυτό αποτελεί την βασική ύλη πάνω στην οποία θα εφαρμόσουμε επεξεργασία ώστε να λάβουμε την αντίστοιχη πληροφορία και τέλος θα την εμφανίσουμε σε επίπεδο παρουσίασης. Στην παρούσα εργασία θα ελέγχουμε τιμές θερμοκρασίας και ροής. Αναλυτικότερα οι αισθητήρες που χρησιμοποιήθηκαν είναι 4 ήδη εγκατεστημένοι αισθητήρες θερμοκρασίας (PT 100) οι οποίοι είναι ήδη συνδεμένοι με 4 display/controller (TLK 31), καθώς επίσης και αισθητήρες ροής για την εποπτεία της ροής του νερού.



Εικόνα 2- Πρώτο και δεύτερο επίπεδο

Επίπεδο 2

Στο δεύτερο επίπεδο υπάρχει η συγκέντρωση των εξόδων των αισθητήρων, δηλαδή των τιμών που αποτελούν τη βάση των δεδομένων μας, σε έναν μικροελεγκτή (πχ Arduino) και η πρώτη φάση επεξεργασίας τους. Σε γενικότερο πλαίσιο, πραγματοποιείται συγκέντρωση των εξόδων των αισθητηρίων και είσοδός τους σε μία μονάδα ελέγχου (έναν μικροελεγκτή, έναν Προγραμματιζόμενο Λογικό Ελεγκτή-PLCκλπ).

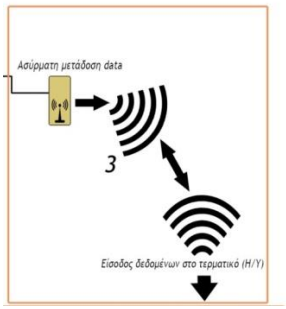
Σε αυτό το επίπεδο δίνεται βάση καταρχήν στον τρόπο μετάδοσης των τιμών από τους αισθητήρες προς τον μικροελεγκτή. Η επιλογή των κατάλληλων μέσων (όπως για παράδειγμα καλωδίων) που θα πληρούν τις εκάστοτε ανάγκες και τις ιδιαιτερότητες κάθε έργου είναι ιδιαίτερα σημαντική. Πέρα από τα μέσα μεταφοράς των δεδομένων, εδώ εντάσσονται και διάφοροι τρόποι – μέθοδοι για την μεταφορά αυτή όπως για παράδειγμα οι ιδιαιτερότητες που μπορεί να έχει μία συσκευή ή ένας αισθητήρας από τον οποίο θέλουμε να διαβάσουμε την τιμή και να την εισάγουμε στον μικροελεγκτή και άρα οι τεχνικές που θα εφαρμόσουμε για να γίνει αυτό. Σε δεύτερη φάση – όπως βλέπουμε και στην εικόνα στο επίπεδο 2 – είναι ο ειδικός κώδικας που θα υλοποιήσουμε ώστε να γίνει αφενός η είσοδος των δεδομένων από τους αισθητήρες, αλλά και αφετέρου μια πρώτη βασική επεξεργασία αυτών των δεδομένων, ώστε να είναι στην κατάλληλη μορφή για να αποσταλούν στα επόμενα επίπεδα.

Στην εργασία αυτή επειδή οι αισθητήρες θερμοκρασίας είναι ήδη συνδεμένοι σε αντίστοιχα στον αριθμό display/controller (TLK 31), για να γίνει με επιτυχία η ανάγνωση των τιμών τους χωρίς να υπάρξει κάποια αλλαγή στις υπάρχουσες συνδέσεις ή πρόβλημα, επιλέχθηκε η επικοινωνία RS 485 με ειδικό module ώστε να διαβάσουμε στο Arduino τις τιμές των αισθητήρων που έχουν στη μνήμη τους τα TLK 31, εφαρμόζοντας το πρωτόκολο MODBUS RTU μέσω κώδικα. Αυτή η λύση θεωρήθηκε από τη μελέτη που προηγήθηκε ως η πιο αποτελεσματική για το διάβασμα των αισθητήρων θερμοκρασίας. Λεπτομέρειες της διαδικασίας αναλύονται στα επόμενα κεφάλαια.

Επίπεδο 3

Το τρίτο επίπεδο είναι η διαδικασία της αποστολής των δεδομένων που προηγουμένως εισάγαμε στον μικροελεγκτή, προς ένα τοπικό τερματικό. Το τερματικό αυτό στην περίπτωση της πτυχιακής αυτής είναι ένας Ηλεκτρονικός Υπολογιστής.

Σε αυτή τη φάση μας απασχολεί το ζήτημα της επιτυχούς και ασφαλούς μετάδοσης των δεδομένων ώστε να αποστέλλονται σωστά και χωρίς απώλειες. Εδώ γίνεται πάλι έρευνα για τις πιο αποδοτικές μεθόδους επικοινωνίας με βάση το περιβάλλον στο οποίο υλοποιείται το έργο και έπειτα ακολουθούν οι απαραίτητες ρυθμίσεις σε επίπεδο διασύνδεσης ώστε να υπάρξει αποτελεσματική επικοινωνία πομπού – δέκτη. Η αποστολή των δεδομένων στην εργασια έγινε με τη χρήση ραδιοκυμάτων μέσω ενός ζεύγους πομποδέκτη (APC 220).



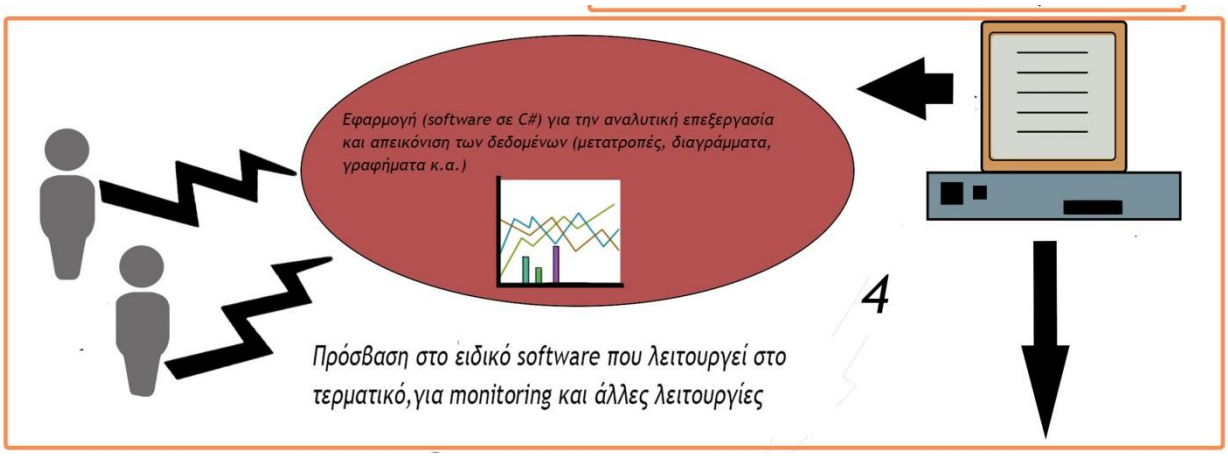
Εικόνα 3- Τρίτο επίπεδο

Επίπεδο 4

Το τέταρτο επίπεδο αποτελεί μεγάλο τμήμα του συστήματος τηλεμετρίας. Εδώ γίνεται η είσοδος των δεδομένων μέσω του πομπού στο τερματικό (Η/Υ), η υλοποίηση ειδικού λογισμικού (software) ώστε να γίνει η κατάλληλη επεξεργασία τους και να μετατραπούν σε μορφή κατάλληλη για τις ανάγκες μας και τέλος η απεικόνισή τους σε διαγράμματα που θα μας παρέχουν πλήρη εποπτεία και έλεγχο πάνω στην εγκατάσταση πάλι μέσω της ειδικής εφαρμογής (software). Επιπροσθέτως, εντάσσονται και επιπλέον λειτουργίες στα πλαίσια του software που πέρα από την ικανοποιητική λήψη, επεξεργασία και παρουσίαση των τιμών/δεδομένων, παρέχουν ειδικότερες μορφές επεξεργασίας τους για άλλες χρήσεις.

Σε πρώτη φάση - για την παρούσα εργασία - υλοποιείται το ειδικό λογισμικό (software) με κατάλληλη γλώσσα προγραμματισμού (C#). Με βάση το λογισμικό αυτό, το σύστημα τηλεμετριας θα υλοποιήσει την λήψη από τον πομπό, θα επεξεργαστεί τα δεδομένα, αλλά και τελικώς θα τα παρουσιάσει ώστε να πραγματοποιηθεί επιτυχώς η παρακολούθηση του συστήματος. Η υλοποίηση του ειδικού αυτού προγράμματος πρέπει να παίρνει υπόψιν της πολλούς παράγοντες που έχουν να κάνουν με το κομμάτι της λήψης των δεδομένων από το προηγούμενο επίπεδο, με τη μορφή στην οποία έρχονται αυτά τα δεδομένα, με το πως θα πρέπει να υποστούν επεξεργασία, με την παρουσίασή τους, με την περαιτέρω επεξεργασία τους κλπ.

Η επιτυχής υλοποίηση αυτής της ειδικής εφαρμογής στον Η/Υ αποτελεί και εγγύηση οτι έχει ολοκληρωθεί ο βασικός άξονας τηλεμετρίας και οτι μπορεί ο χρήστης να έχει πρόσβαση στο συγκεκριμένο τερματικό και να ελέγχει – εποπτεύει τις τιμές των αισθητήρων που έχουν εγκατασταθεί από απόσταση, με το πλεονέκτημα της φιλικής προς τον χρήστη παρουσίασής τους αλλά και της δυνατότητας για παραπέρα επεξεργασία των δεδομένων αυτών για άλλες χρήσεις (πχ μετατροπή ομαδοποιημένων μετρήσεων σε μορφές αρχείων όπως .xml, .xlsx, .txt κ.α.)



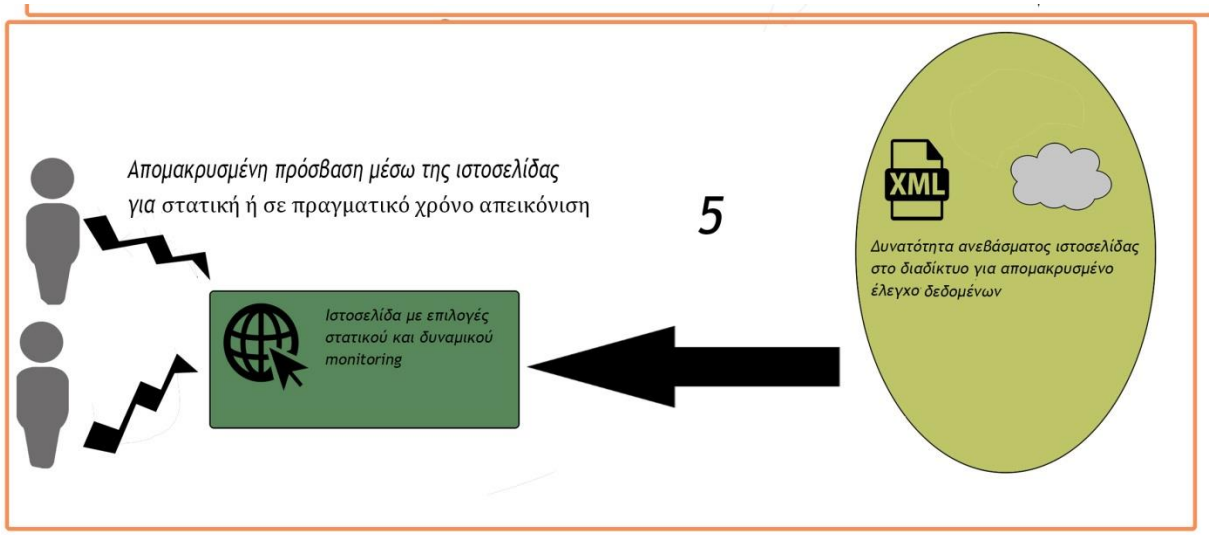
Εικόνα 4 - Τέταρτο επίπεδο

Επίπεδο 5

Στο τελευταίο επίπεδο έχουμε την διαδικτυακή διεπαφή με το χρήστη. Πρόκειται για την υλοποίηση ενός δεύτερου υποσυστήματος τηλεμετρίας που θα μπορεί να παρέχει μια μορφή βασικής εποπτείας από απομακρυσμένη θέση – από οπουδήποτε– αφού θα βασίζεται στο διαδίκτυο. Εδώ η φιλοσοφία υλοποίησης είναι να μεταφορτώνονται οι πληροφορίες από το τοπικό τερματικό που λειτουργεί και ως εξυπηρετητής, στον εξυπηρετητή ιστού (Apache web server), όπου κατάλληλος κώδικας δημιουργίας σελίδων θα επιτρέπει σε απομακρυσμένους χρήστες να εποπτεύσουν το σύστημα από οποιοδήποτε σημείο με πρόσβαση στο διαδίκτυο. Εδώ έχουμε μια επέκταση του συστήματος της τηλεμετρίας λόγω ακριβώς του οτι η απόσταση, δεν αποτελεί πλέον εμπόδιο προς την εποπτεία του συστήματος. Με τον τρόπο αυτό δίνεται η ευελιξία χειρισμών καθώς δεν απαιτείται φυσική παρουσία στο χώρο παραγωγής για την επισκόπηση λειτουργίας της μονάδας.

Η ιστοσελίδα αναπτύχθηκε σε γλώσσες προγραμματισμού τόσο server side (PHP, HTML), όσο και client side (Javascript) για κατασκευή ιστοσελίδων και μεταφορτώνεται μέσω του εξυπηρετητή στο διαδίκτυο. Σημαντικό σημείο σε αυτό το επίπεδο είναι η επιτυχής είσοδος των δεδομένων/τιμών από το ειδικό λογισμικό (software) του τερματικού, στην ιστοσελίδα (website) που θα μεταφορτώνεται. Αυτό μπορεί να υλοποιηθεί με διάφορες τεχνικές (πχ εξαγωγή αρχείου .xml με τα δεδομένα προς απεικόνιση από το πρόγραμμα και η είσοδός του αρχείου στην ιστοσελίδα).

Τέλος, υπάρχουν μέσα στο περιεχόμενο της ιστοσελίδας φιλικά προς τον χρήστη διαγράμματα που ικανοποιεί την ανάγκη για απλή, εύχρηστη και ευέλικτη εποπτεία των τιμών των αισθητήρων, είτε σε πραγματικό χρόνο (real time), είτε σε μεταγενέστερο στάδιο με χρήση παλαιότερων καταγραφών, αυτή τη φορά μέσω του internet.



Εικόνα 5 - Πέμπτο επίπεδο

3. Κεφάλαιο: Εξοπλισμός, συνδέσεις και περιγραφή υλικών

Σε αυτό το κεφάλαιο παρουσιάζονται τα υλικά και ο εξοπλισμός που χρησιμοποιήθηκε στην υλοποίηση της εργασίας, οι συνδέσεις που έγιναν καθώς και τα ιδιαίτερα χαρακτηριστικά τους.

Όπως περιγράψαμε πιο πάνω στην αρχιτεκτονική υλοποίησης, χρησιμοποιήθηκαν ήδη υπάρχοντες αισθητήρες του ελαιουργείου (για τη θερμοκρασία) αλλά και κάποιοι επιπλέον (για τη ροή). Με διαφορετικές τεχνικές στον καθένα υλοποιήθηκε η διαδικασία συγκέντρωσης των τιμών τους στον κεντρικό μικροελεγκτή (Arduino) και απο εκεί με ειδικό εξοπλισμό (module) ασύρματης επικοινωνίας στέλνονται οι τιμές σε Η/Υ στον οποίο εκτελείται η εφαρμογή εποπτείας, το πρόγραμμα δηλαδή που τις παρουσιάζει και τις επεξεργάζεται. Ταυτοχρόνως, ο Η/Υ μπορεί εν δυνάμει να καταστεί και εξυπηρετητής (server) μεταφορτώνοντας τις πληροφορίες στην ιστοσελίδα για την παρουσίαση σε απομακρυσμένους χρήστες

Στις επόμενες σελίδες αυτού του κεφαλαίου λοιπόν, θα παρουσιαστούν όλα τα υλικά και οι συνδέσεις που χρησιμοποιήθηκαν στην υλοποίηση της εφαρμογής.

3.1 Ο μικροελεγκτής Arduino

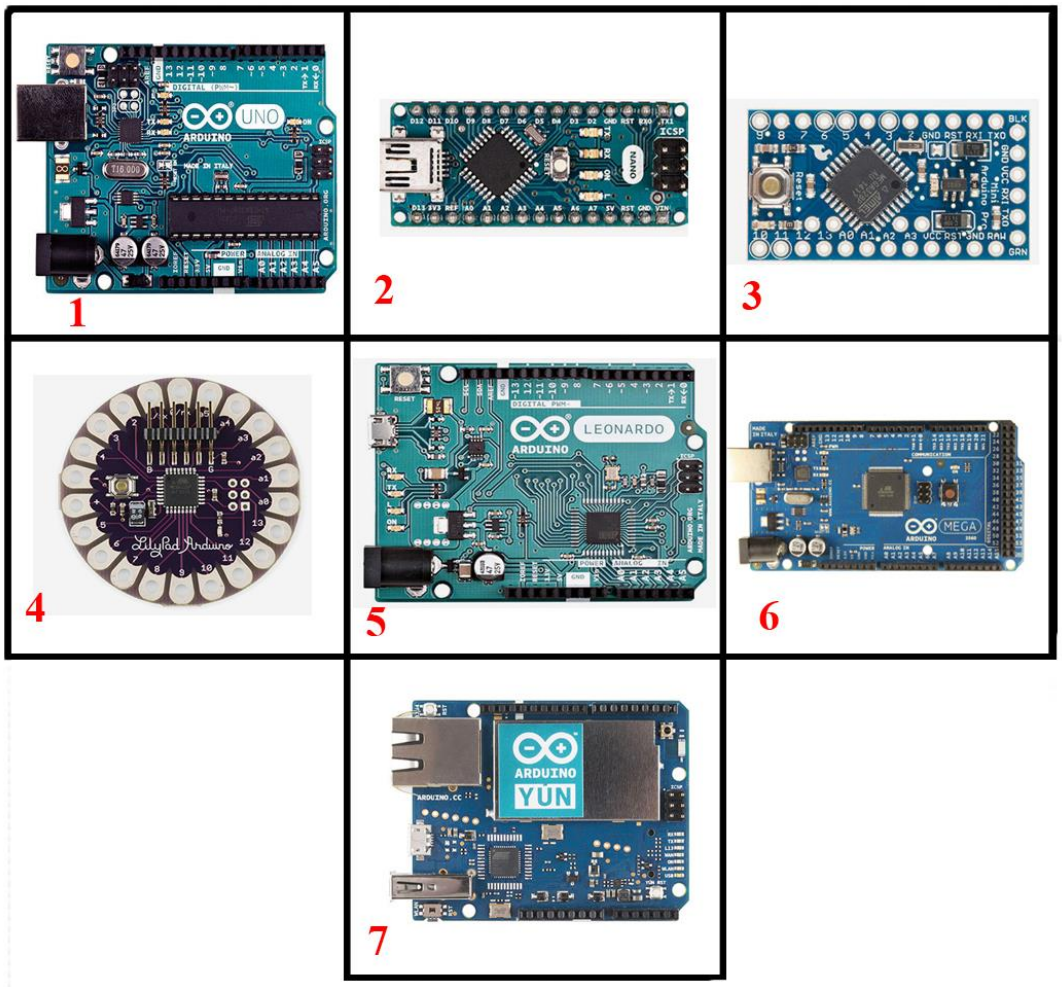
Το Arduino είναι υλικό ανοιχτού κώδικα με ενσωματωμένη πλατφόρμα ανάπτυξης και περιλαμβάνει μία πλακέτα μικροελεγκτή, μαζί με περιβάλλον ανάπτυξης προγραμμάτων για τον προγραμματισμό του. Προέρχεται από την Ιβρέα της Ιταλίας όπου αναπτύχθηκε πρώτη φορά το 2003 ως Wiring Project από έναν φοιτητή στα πλαίσια εργασίας του. Οι περισσότερες εκδόσεις Arduino χρησιμοποιούν μικροελεγκτές Atmel AVR και σήμερα υπάρχουν πολυάριθμες διαφορετικές εκδόσεις (πλακέτες) για κάθε είδους εφαρμογή με πιο πολλές ή πιο λίγες δυνατότητες. Για να προγραμματιστεί ο μικροελεγκτής χρησιμοποιείται η γλώσσα προγραμματισμού Wiring C, μία παραλλαγή της γλώσσας C++. Μεγάλο πλεονέκτημα είναι η υποστήριξη προσθήκης βιβλιοθηκών οι οποίες είναι ανοιχτές στον καθένα και έτσι διαμορφώνεται ένα πλαίσιο πολλών δυνατοτήτων για εφαρμογές με χρήση του Arduino.

Στην εργασία αυτή, το Arduino κατείχε τον ρόλο του κεντρικού μικροελεγκτή, υπεύθυνο για την συγκέντρωση των τιμών των αισθητήρων και την μεταφορά τους προς το τερματικό ηλεκτρονικό υπολογιστή.

Το Arduino έγινε γρήγορα γνωστό και εξαπλώθηκε ταχύτατα σε όλο τον κόσμο κυρίως λόγω τεσσάρων βασικών χαρακτηριστικών του:

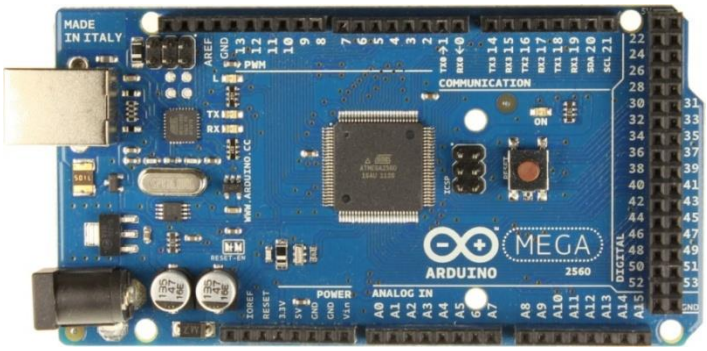
- Χαμηλό κόστος.
- Πολύ απλό ως περιβάλλον για ανάπτυξη προγράμματος.
- Συμβατότητα με όλα τα γνωστά λογισμικά (Windows, Linux, Macintosh)
- Είναι ανοιχτού κώδικα και επεκτάσιμο, που σημαίνει πως ο καθένας μπορεί να ερευνήσει, μελετήσει και υλοποιήσει οτιδήποτε ώστε να συνεισφέρει στην προσπάθεια γιγάντωσης και ανάπτυξης του Arduino, αλλά και να χρησιμοποιήσει ήδη υπάρχοντα εργαλεία (π.χ. βιβλιοθήκες) φτιαγμένα από άλλους χρήστες, δωρεάν.

Υπάρχουν πάρα πολλές διαφορετικές εκδόσεις Arduino με κάποιες από τις πιο γνωστές να είναι οι εξής 7 που φαίνονται στην παρακάτω εικόνα:



Εικόνα 6 - Τα πιο γνωστά είδη Arduino: 1 Uno, 2 Nano, 3 Micro, 4 Lily, 5 Leonardo, 6 Mega, 7 Yun

Από όλες τις εκδόσεις θα αναλυθεί η έκδοση που χρησιμοποιήθηκε και στην υλοποίηση της πτυχιακής εργασίας που είναι το Arduino Mega 2560 (Εικόνα 7).



Εικόνα 7 - Arduino Mega 2560

Η πλακέτα του Arduino Mega 2560 βασίζεται στον επεξεργαστή ATmega1280 και η χρήση του τα τελευταία χρόνια έχει αυξηθεί και έχει ξεπεράσει το πλαίσιο των μικροκατασκευών και της ρομποτικής φτάνοντας στο σημείο να χρησιμοποιείται ακόμα και σε βιομηχανίες μικρής κλίμακας και σε περιβάλλοντα τα οποία δεν ήταν προορισμένο εξαρχής να δουλεύει. Αυτό συμβαίνει κυρίως λόγω της απλότητας στη διαδικασία προγραμματισμού - εγκατάστασης και επέκτασης όπου κάνει τους μηχανικούς να γλιτώνουν χρόνο και χρήματα.

Η πλακέτα του Mega έχει 54 pin για ψηφιακή είσοδο/έξοδο, 16 pin αναλογικής εισόδου, 4 θύρες UART για σειριακή επικοινωνία (η έκδοση UNO έχει μόνο 1 τέτοια θύρα) και συνδέεται με USB σε Η/Υ και με jack στην τροφοδοσία. Μια πιο αναλυτική περιγραφή των χαρακτηριστικών του, δίνεται παρακάτω:

- Τάση λειτουργίας: 5V
- Τάση εισόδου: 7-12V
- Μικροελεγκτής: ATmega1280
- Ψηφιακά I/O pins: 54
- PWM pins: 15
- Αναλογικά Input pins: 16
- Ρεύμα ανά pin: 40mA (max)
- Flash μνήμη: 128KB
- SRAM: 8KB
- EEPROM: 4KB
- Ταχύτητα ρολογιού: 16MHz

Όπως παρατηρείται το Arduino Mega έχει 8KB SRAM μνήμη για την αποθήκευση μεταβλητών και άλλων στοιχείων που χρησιμοποιεί το πρόγραμμα και η οποία χάνει ό,τι έχει γραμμένο μέσα της όταν το Arduino σταματήσει να τροφοδοτείται με ρεύμα. Η μνήμη Flash είναι 128KB και λειτουργία της είναι η αποθήκευση των μεταγλωτισμένων προγραμμάτων και, όπως και η μνήμη EEPROM που στον Mega είναι 4KB και χρησιμοποιείται για την ανάγνωση ή εγγραφή δεδομένων από τα προγράμματα κατά το runtime, δεν χάνει τα περιεχόμενά της.

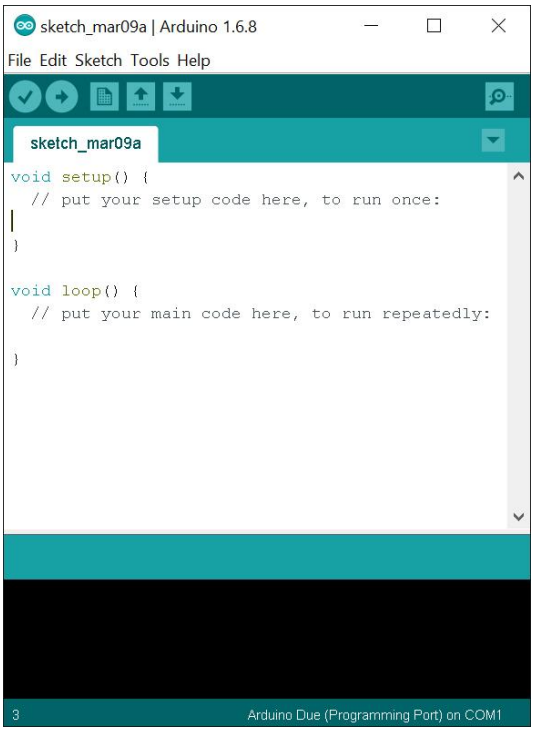
α) Ακροδέκτες

Κοιτώντας κανείς την εικόνα 7 με την πλακέτα μπορεί να διακρίνει την ιδιότητα του κάθε pin (ακροδέκτη). Όπως βλέπουμε, υπάρχουν 4 διαφορετικά ζεύγη TXD/RXD pins, που είναι και οι 4 UART serial θύρες επικοινωνίας που έχει η πλακέτα. Αυτό σημαίνει πως μπορεί να επικοινωνεί αμφίδρομα με 4 διαφορετικούς τρόπους/συσκευές ταυτόχρονα. Ακόμα, 15 από τα digital pins μπορούν να έχουν και λειτουργία PWM (ψευδοαναλογική είσοδος με ανάλυση 8bit, 0 έως 255 καταστάσεις). Τα pins 2, 3, 18, 19, 20 και 21 στο Arduino Mega λειτουργούν και ως εξωτερικοί interrupt ακροδέκτες, ώστε να μπορούμε όταν γίνεται μία εξωτερική αλλαγή, να διακόπτεται η κανονική ροή του προγράμματος και να εκτελείται συγκεκριμένη διαδικασία που θα ορίζει ο χρήστης. Τέλος βλέπει κανείς πέρα από τα Analog pins εισόδου και τα Digital pins εισόδου/εξόδου και τους ακροδέκτες Gnd όπου είναι η γείωση, Vin που είναι η τάση εισόδου όταν γίνεται χρήση εξωτερικής τροφοδοσίας, 5V που είναι έξοδος 5V τα οποία βγάζει η πλακέτα, αλλά και το AREF pin που είναι η τάση αναφοράς για τις αναλογικές εισόδους. Αυτή είναι η τάση αναφοράς με την οποία συγκρίνει ο μικροελεγκτής ένα αναλογικό σήμα κατά την ανάγνωσή του ώστε να γίνεται ρύθμιση της ακρίβειας μέτρησης (resolution).

Όσον αφορά την τροφοδοσία, όπως είδαμε πιο πάνω η πλακέτα μπορεί να τροφοδοτηθεί είτε από εξωτερική τροφοδοσία μέσω ενός μετασχηματιστή πχ των 9 volt (η τάση να είναι μέσα στα όρια που δίνει ο κατασκευαστής, άλλα οπωσδήποτε πάνω από 7 volt), ή και από τη θύρα USB, ή ακόμα και από μπαταρία.

β) Το περιβάλλον ανάπτυξης του προγράμματος

Η πλακέτα προγραμματίζεται με τη βοήθεια της γλώσσας προγραμματισμού Wiring C, μιας γλώσσας που βασίζεται στη C++. Η πλατφόρμα του Arduino παρέχει ένα ειδικό περιβάλλον ανάπτυξης (Arduino IDE) για τη γραφή και μεταγλώτιση του προγράμματος. Το software αυτό είναι κατασκευασμένο στη γλώσσα Java και είναι ευέλικτο και εύκολο στη χρήση. Υποστηρίζει την προσθήκη βιβλιοθηκών τις οποίες βρίσκει κανείς στο διαδύκτιο ή δημιουργούνται από τον χρήστη και οι οποίες παρέχουν απεριόριστα εργαλεία για να επεκτείνουμε ή να υλοποιήσουμε την εκάστοτε εργασία. Στην επίσημη ιστοσελίδα http://www.arduino.cc μπορεί εύκολα να βρει κανείς την τελευταία έκδοση του software αυτού για την εγκατάσταση σε Η/Υ.

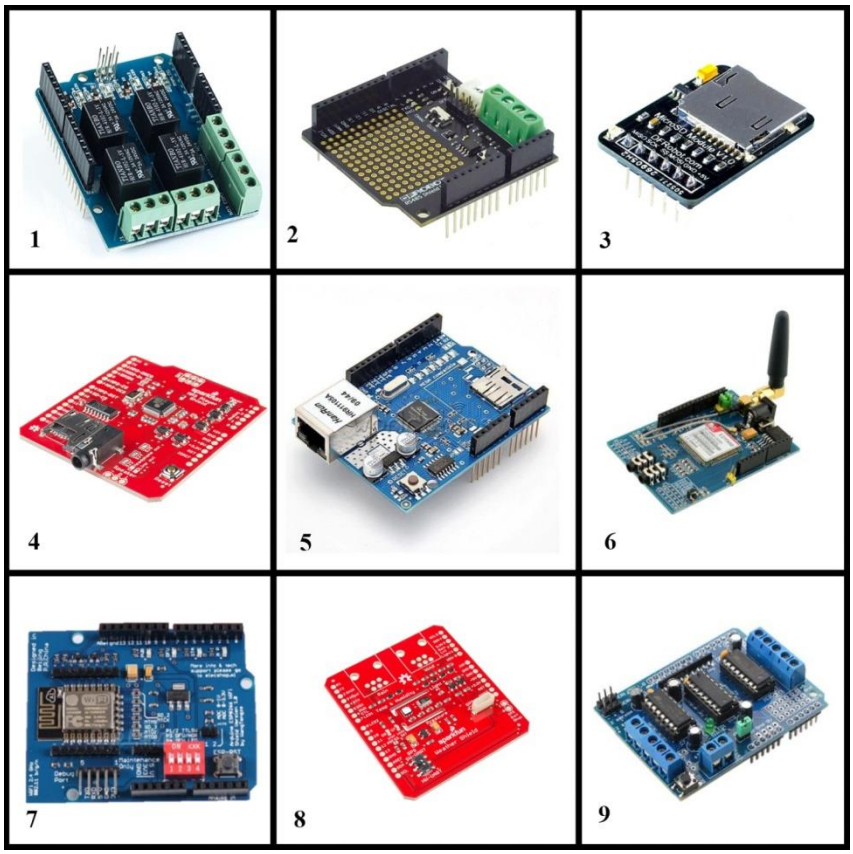


Εικόνα 8 - Περιβάλλον ανάπτυξης κώδικα

Στην εικόνα 8 βλέπει κανείς την αρχική εμφάνιση του Arduino IDE όταν ανοίγει. Η χαρακτηριστική φιλοσοφία πίσω από τον προγραμματισμό της πλακέτας είναι οτι χωρίζεται σε δύο μέρη. Όπως φαίνεται και στην εικόνα υπάρχουν δύο βασικές συναρτήσεις, η setup και η loop. Η setup καλείται μόνο μία φορά κατά την αρχή του προγράμματος (στο Arduino τα προγράμματα λέγονται sketches) και η loop έπειτα, τρέχει συνεχώς όσο η πλακέτα έχει ρεύμα. Μέσα σε αυτό το πλαίσιο ίσως να εμφανίζεται ως περιοριστική η όλη φιλοσοφία ως προς την προγραμματιστική διαδικασία, αλλά αντιθέτως το σύνολο τους προγραμματισμού μετατρέπεται σε μια ευέλικτη διαδικασία. Με τη λογική των δύο συναρτήσεων (setup και loop) που η μία καλείται μία φορά και η άλλη συνεχώς και αδιάκοπα, μπορούμε είτε να εκμεταλλευτούμε την επαναληπτικότητα της κλήσης της loop και να προγραμματίσουμε σε αυτό το πλαίσιο, είτε να κάνουμε χρήση ειδικών τεχνικών και να χρησιμοποιήσουμε κλάσεις, συναρτήσεις κλπ σε αντιστοιχία όπως αντικειμενοστραφείς γλώσσες προγραμματισμού (πχ C++).

γ) Πλακέτες επέκτασης (Shields) και μονάδες (modules)

Στην αγορά υπάρχει ένα εξαιρετικά μεγάλο πλήθος από πλακέτες επέκτασης. Αυτές οι πλακέτες ονομάζονται shields και αποτελούν υλικό (hardware) με έτοιμες κολλήσεις (συχνά χρειάζεται επιπλέον δουλειά κολλήσεων) και υλικό, οι οποίες επεκτείνουν τις λειτουργίες του Arduino σε διάφορους τομείς. Παρακάτω φαίνεται μια εικόνα με 9 πολύ γνωστές πλακέτες επέκτασης:



Εικόνα 9 - Τα πιο γνωστά Shields για Arduino: 1 Relay, 2 RS485, 3 microSD card, 4 MP3, 5, Ethernet, 6 GSM/GPRS, 7 WiFi, 8 Weather, 9 Motor drive

Όλα τα shields συνδέονται στην πλακέτα του Arduino και συνήθως δεσμεύουν αρκετά pins. Γιαυτό το λόγο υπάρχουν οι μονάδες (modules). Ένα module μπορεί να είναι είτε μια μικρή και λιτή έκδοση ενός shield, είτε μία αυτόνομη μονάδα συγκεκριμένης λειτουργίας από μόνο του (πχ αισθητήρας). Το πλήθος των modules ειναι εξαιρετικά μεγάλο και οι τιμές τους αρχίζουν ακόμα και από λίγα cents.

Χαρακτηριστικά παραδείγματα module για Arduino είναι πάσης φύσεως αισθητήρες όπως υγρασίας, θερμοκρασίας, πίεσης, αερίων, καπνού, νερού, γυροσκόπια, απόστασης με υπερήχους, φωτεινότητας, ήχου, μικρόφωνα, κίνησης, βάρους, επιτάχυνσης, αλλά και μονάδες αυτόνομες όπως πομποδέκτης για ασύρματη επικοινωνία ή όπως αναφέρθηκε μικρότερες εκδόσεις των shields. Τα modules συνδέονται όχι απαραίτητα πάνω στην πλακέτα του Arduino αλλά είτε σε breadboard, είτε σε πλακέτα που φτιάχνει ο εκάστοτε χρήστης κολλώντας τα απαραίτητα pins και καλώδια, συνεπώς ένα module πιάνει λιγότερο χώρο και μπορεί να τοποθετηθεί οπουδήποτε. Σημαντική εξέλιξη επίσης είναι τα τελευταία χρόνια, η δημιουργία δικτύων ασύρματων αισθητήρων - module δηλαδή - τα οποία βρίσκονται διάσπαρτα σε αποστάσεις και επικοινωνούν μεταξύ τους ή/και με το Arduino.

3.2 Συσκευές απεικόνισης και ελέγχου (TLK31) και αισθητήρες θερμοκρασίας (PT100)

Βασικό κομμάτι πάνω στο οποίο βασίστηκε η πτυχιακή εργασία είναι η εποπτεία (monitoring) των τιμών τεσσάρων αισθητήρων θερμοκρασίας. Τα δεδομένα μέτρησης των τεσσάρων αισθητήρων που βρίσκονται σε 4 διαφορετικά σημεία του ελαιουργείου αποτελούν το υλικό πάνω στο οποίο θα δομηθεί το σύστημα τηλεμετρίας. Σκοπός του συστήματος τηλεμετρίας είναι η απεικόνιση από απόσταση των τιμών σε διαγράμματα, η επεξεργασία τους, η ομαδοποίησή τους σε μορφές αρχείων κατάλληλες για επιπλέον επεξεργασία και άλλες λειτουργίες που συμπληρώνουν ένα συνολικό και ολοκληρωμένο σύστημα τηλεμετρίας - εποπτείας.

Οι τέσσερις αυτοί αισθητήρες ήταν ήδη εγκατεστημένοι στο ελαιουργείο συνεπώς θα έπρεπε να διαβάζεται η τιμή τους, μη επεμβατικά, χωρίς να διαταραχθεί κάποιο άλλο κύκλωμα. Κάτι τέτοιο αποτέλεσε ένα πρόβλημα καθώς οι αισθητήρες ήταν ήδη συνδεμένοι με τέσσερις αντίστοιχες συσκευές απεικόνισης - ελέγχου (TLK 31). Επομένως σύμφωνα με αυτά που έχουν περιγραφεί και στο κεφάλαιο της αρχιτεκτονικής υλοποίησης της εργασίας, έπρεπε να ερευνηθούν και να υλοποιηθούν οι κατάλληλες τεχνικές ώστε να γίνεται το διάβασμα των τιμών χωρίς να δημιουργούνται προβλήματα στην λειτουργία - σύνδεση αισθητήρων και display συσκευών.

Παρακάτω θα αναλυθούν ξεχωριστά οι αισθητήρες και οι συσκευές display/controller και θα γίνει μια πρώτη αναφορά για τον τρόπο ανάγνωσης των τιμών τους. Ωστόσο σε αυτό το κεφάλαιο θα γίνουν αναφορές μόνο στην περιγραφή του hardware (υλικού). Οι τελικοί τρόποι υλοποίησης με κώδικα, αναφέρονται στα αντίστοιχα κεφάλαια. Τέλος όσον αφορά την ανάγνωση των τιμών με την επικοινωνία RS 485 υπάρχει στο παρακάτω κεφάλαιο αντίστοιχη αναφορά.

α) Οι αισθητήρες θερμοκρασίας PT100

Οι τέσσερις αισθητήρες που υπάρχουν εγκατεστημένοι στο ελαιουργείο είναι αισθητήρες τύπου PT 100. Αυτοί οι αισθητήρες είναι τύπου RTD, δηλαδή όργανα μετρήσεων που μετράνε θερμοκρασία μέσω αντίστασης. Όσο αυξάνεται η θερμοκρασία, η αντίσταση του μετάλλου του αισθητήρα μεγαλώνει, συνεπώς αν εφαρμοστεί τάση πάνω σε αυτή την αντίσταση, μπορούμε μετρώντας την αντίσταση κάθε φορά, να ξέρουμε και την θερμοκρασία. Ο συγκεκριμένος αισθητήρας (PT 100) είναι κατασκευασμένος από πλατίνα η οποία είναι η καλύτερη επιλογή σε σχέση με άλλα υλικά λόγω συγκεκριμένων ιδιοτήτων όπως είναι η ανθεκτικότητα σε οξείδωση και σε άλλες μεταβολές, το μεγάλο εύρος θερμοκρασιών που μπορεί να καλύψει, η χημική αδράνειά της κ.α. Η ακρίβεια ενός αισθητήρα με πλατίνα όπως του PT 100

είναι της τάξης του +/- 0.2%, πολύ καλή σε σχέση με άλλους αισθητήρες κάτι που κάνει τα συγκεκριμένα όργανα ιδανικά για βιομηχανικές εφαρμογές. Στην εικόνα που ακολουθεί βλέπουμε τη μορφή ενός τέτοιου αισθητήρα:

Εικόνα 10 - Αισθητήρας PT100

Ο αισθητήρας ονομάζεται PT 100 διότι στους 0 βαθμούς Κελσίου, παρουσιάζει αντίσταση 100 Ω. Οι PT 100 γενικά έχουν ένα εύρος θερμοκρασιών που μπορούν να μετρήσουν από -200 έως +850 βαθμούς Κελσίου. Απο εκεί και πέρα κάθε εταιρεία έχει PT 100 με διαφορετικό εύρος (μέσα στο μέγιστο που αναφέραμε) και διαφορετικό σφάλμα/ακρίβεια και επομένως υπάρχουν PT 100 με διαφορετικές τιμές αγοράς.

Η σχέση μεταξύ θερμοκρασίας και αντίστασης δεν είναι ακριβώς γραμμική αλλά τείνει προς τη γραμμικότητα, κυρίως σε μικρό εύρος θερμοκρασιών. Όσο μεγαλώνει το εύρος, μεγαλώνει και το σφάλμα. Για να υπάρχει όσο το δυνατόν μεγαλύτερη ακρίβεια στη μέτρηση θα πρέπει, αφού έχουμε πρώτα αντιμετωπίσει άλλους παράγοντες όπως αυτοθέρμανση, συνθήκες περιβάλλοντος κ.α., να γραμμικοποιήσουμε την αντίσταση. Τα μαθηματικά παρέχουν έναν τύπο γραμμικοποίησης ο οποίος είναι και σύμφωνα με το διεθνές στάνταρ για τις θερμοκρασίες (ITS-90) ο πιο ακριβής μέχρι στιγμής. Ο τύπος αυτός είναι ο:

Rt = R0 * (1 + A* t + B*t2 + C*(t-100)* t3)

όπου Rt = η αντίσταση σε θερμοκρασία t
R0 = η θερμοκρασία στους 0 βαθμούς Κελσίου
A = 3.9083 E-3
B = -5.775 E-7
C = 0 (για πάνω από 0 βαθμούς) ή -4.183 E-12 για κάτω από τους μηδέν

Με βάση τον παραπάνω τύπο για κάθε 1 βαθμό κελσίου η αντίσταση αλλάζει κατά 0.384 Ω.

Για να αποφύγουμε το σφάλμα του φαινομένου της αυτοθέρμανσης, δηλαδή του φαινομένου εκείνου που ο αισθητήρας μετράει παραπάνω από την εξωτερική θερμοκρασία γιατί τα καλώδιά του έχουν αυξημένη θερμοκρασία, θα πρέπει να περιορίζεται το ρεύμα που διαρρέει τους αγωγούς του. Όσο πιο λίγο είναι το ρεύμα τόσο καλύτερο, με συνηθισμένες τιμές γύρω στο 1mA. Για να επιτευχθεί αυτό συνήθως περιορίζουμε με διαιρέτες τάσης το ρεύμα και έπειτα ενισχύουμε με τελεστικούς ενισχυτές την έξοδο του αισθητήρα ώστε να είναι αρκετή για να διαβαστεί απο μικροϋπολογιστές, μικροελεγκτές κ.α. συσκευές.

Ο αισθητήρας έχει τρία καλώδια, ενώ υπάρχουν και εκδόσεις του ίδιου τύπου αισθητήρα με 4 καλώδια. Θα πρέπει και τα τρία καλώδια να έχουν το ίδιο μήκος ώστε να έχουν την ίδια αντίσταση υλικού, ώστε να μην επηρεαστεί η μέτρηση της ηλεκτρικής αντίστασης. Στην κάτω εικόνα βλέπουμε πως είναι η δομή ενός αισθητήρα τριών καλωδίων όπως αυτοί που υπάρχουν εγκατεστημένοι ήδη στο ελαιουργείο.

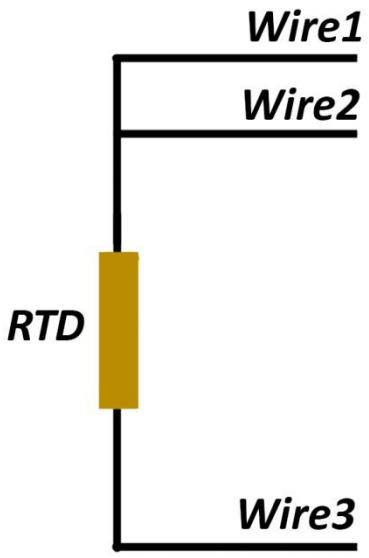


Figure 1 - Διάγραμμα αισθητήρα PT100

β) Το display/controller TLK31 και η σύνδεση με τον PT100

Ο TLK 31 είναι ψηφιακός ελεγκτής βασισμένος σε μικροεπεξεργαστή, με οθόνη. Εφαρμόζει PID έλεγχο με ειδικό αλγόριθμο που έχει στο εσωτερικό του και παρέχει έλεγχο ακριβείας κυρίως σε βιομηχανικές

εφαρμογές. Για επικοινωνία, το TLK 31 παρέχει RS 485 σειριακή επικοινωνία βασισμένη στο πρωτόκολλο επικοινωνίας MODBUS - RTU (πληροφορίες σχετικά με την επικοινωνία RS 485 και το πρωτόκολλο MODBUS - RTU στα αντίστοιχα κεφάλαια).

Στον κεντρικό πίνακα ελέγχου του ελαιουργείου υπάρχουν εγκατεστημένες τέσσερις συσκευές TLK31 στις οποίες συνδέονται αντίστοιχα οι τέσσερις αισθητήρες PT100, ώστε οι συσκευές να πληροφορούν τον χρήστη μέσω της οθόνης τους για την τιμή θερμοκρασίας κάθε αισθητήρα. Συνεπώς για κάθε έναν από τους αισθητήρες, παρόλο που βρίσκεται σε διαφορετικό σημείο της γραμμής παραγωγής λαδιού, καταλήγουν τα τρία καλώδιά του στον κεντρικό πίνακα πάνω σε ένα TLK31.

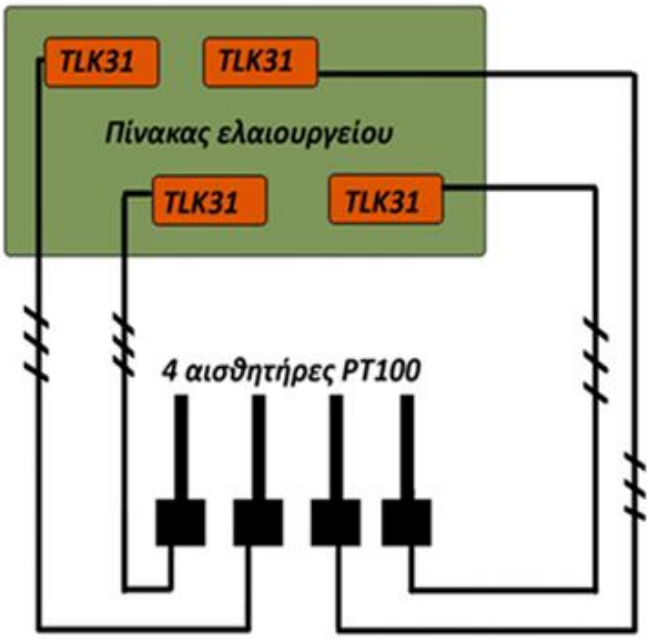


Figure 2 - Τα TLK31 στο ελαιουργείο

Figure 3 - Το TLK31 εξωτερικά

Στην εικόνα 13 φαίνεται η μορφή που έχει μία τέτοια συσκευή. Εξωτερικά αποτελείται από τέσσερα πλήκτρα και από την οθόνη στην οποία και αναγράφεται η τιμή θερμοκρασίας του αισθητήρα που είναι συνδεμένος. Με τα πλήκτρα γίνονται οι ειδικές ρυθμίσεις που υποστηρίζει η συσκευή. Στη συνέχεια δίνεται το ηλεκτρικό διάγραμμα ενός TLK 31.

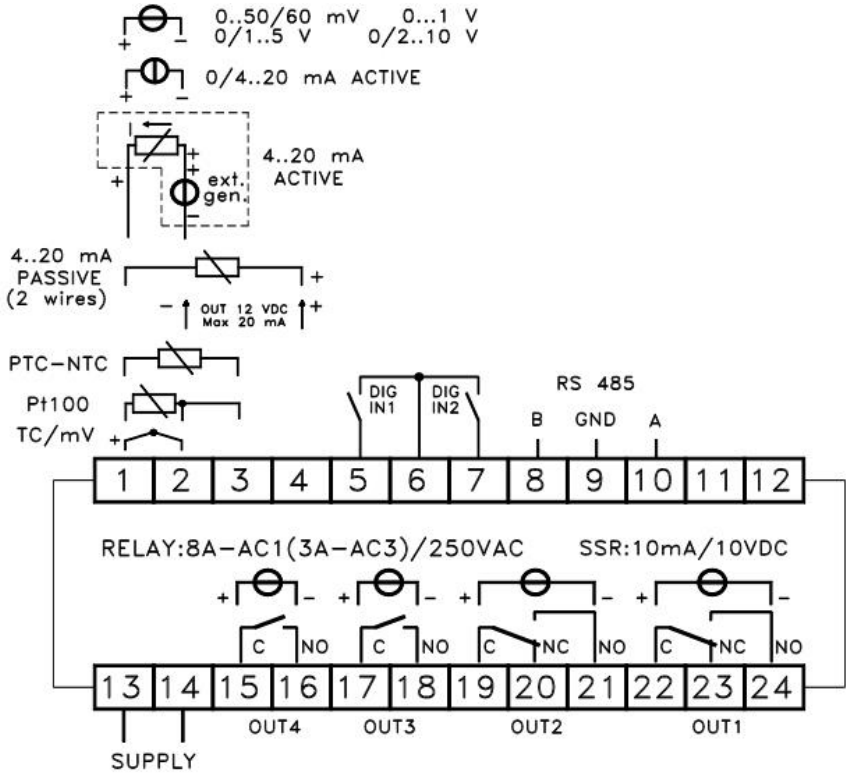


Figure 4 - Ηλεκτρικό διάγραμμα του TLK31

Όπως παρατηρούμε έχει 3 pins (1, 2 και 3) για τη σύνδεση αισθητήρα PT 100. Η συγκεκριμένη συσκευή όχι μόνο είναι συμβατή με αυτούς τους αισθητήρες αλλά εφαρμόζει ειδικές λειτουργίες ώστε να τους διαβάζει με μεγάλη ακρίβεια χωρίς σφάλματα και αποκλίσεις.
Η συσκευή παίρνει τροφοδοσία απο τα pins 13 και 14 και οι αισθητήρες PT 100 στο ελαιουργείο, ήταν συνδεμένοι ήδη με τα TLK 31 στα αντίστοιχα pins (1, 2 και 3).

Αφού υφίσταται υπάρχουσα σύνδεση του αισθητήρα με το controller/display, θα ήταν εσφαλμένο να εφαρμόζαμε παράλληλα με την σύνδεση αυτή, στα καλώδια του αισθητήρα, παρεμβατικές ηλεκτρολογικές μεθόδους (πχ γέφυρα Wheatstone) για να τον διαβάσουμε. Κάτι τέτοιο θα προκαλούσε πρόβλημα αφού το TLK 31 εφαρμόζει ήδη τάση και ειδικό κύκλωμα για να διαβάζει τον αισθητήρα PT 100, επομένως η παράλληλη σύνδεση ενός επιπλέον κυκλώματος πάνω στο υπάρχον θα έδινε εντελώς εσφαλμένες μετρήσεις.

Η λύση που επιλέχθηκε για να διαβαστούν οι αισθητήρες θερμοκρασίας ήταν να γίνει χρήση των RS 485 pins για σειριακή επικοινωνία (8, 9 και 10) που παρέχει το TLK 31. Η φιλοσοφία είναι να μετατραπούν τα TLK 31 σε Slaves και το Arduino σε Master ο οποίος θα ζητάει ανά χρονικό διάστημα τις τιμές των αισθητήρων που βρίσκονται αποθηκευμένες στις μνήμες των slaves. Στην επομένη εικόνα παρουσιάζεται το διάγραμμα του TLK 31 για τέτοιου είδους συνδέσεις, η ανάλυση του οποίου υπάρχει στην επόμενη ενότητα για την σύνδεση RS-485:

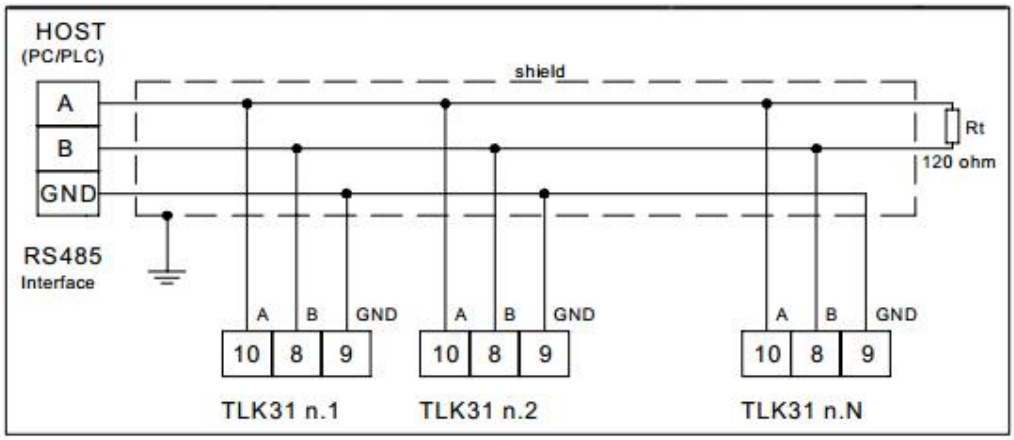


Figure 5 - Διάγραμμα σύνδεσης επικοινωνίας TLK31 (RS-485 Master/Slave)

3.3 RS-485 module και καλώδια

Για να μεταφερθούν τα δεδομένα από το TLK-31 στο Arduino θα πρέπει όπως αναφέρθηκε στο κεφάλαιο για το TLK-31, να χρησιμοποιηθεί η θύρα σειριακής επικοινωνίας RS-485 που έχει αυτό το controller/display. Ωστόσο για να επιτευχθεί η διασύνδεση θα πρέπει να γίνουν οι απαραίτητες ενέργειες ώστε να μπορεί και το Arduino να χρησιμοποιεί την διασύνδεση (τυποποίηση) RS-485. Αυτό έγινε χρησιμοποιώντας ειδικό module RS-485 για Arduino.

α) Τι είναι η επικοινωνία με RS-485

Η RS-485 είναι μία τυποποίηση διασύνδεσης για σειριακή επικοινωνία. Χαρακτηριστικά της είναι η πραγματική σύνδεση πολλαπλών χρηστών (multidrop) πάνω στην ίδια γραμμή (μπορούν να συνδεθούν μέχρι 32 εκπομποί και 32 δέκτες), οι γρήγορες ταχύτητες (πάνω απο 10 Mbit/s σε συγκεκριμένες συνθήκες), οι μεγάλες αποστάσεις (πάνω από 1.200 μέτρα) καθώς και το γεγονός οτι έχει επιτυχή χρήση σε περιβάλλοντα με ηλεκτρικό - ηλεκτρομαγνητικό θόρυβο. Συνεπώς είναι ένας τύπος διασύνδεσης που χρησιμοποιείται ιδιαίτερα στην βιομηχανία μιας και παρέχει όλα τα παραπάνω με τη χρήση απλού καλωδίου συνεστραμένου ζεύγους. Το οτι μπορεί και υποστηρίζει multidrop τοπολογίες στην ίδια γραμμή οφείλεται στο οτι εν αντιθέσει με την διασυνδεση RS-422, η RS-485 έχει κύκλωμα εκπομπού tri state όπου εκτός από τις δύο καταστάσεις μπορεί να μπαίνει και στην κατάσταση υψηλής αντίστασης. Με αυτή την διασύνδεση μπορούμε να υλοποιήσουμε είτε full-duplex, είτε half-duplex σύνδεση, δηλαδή είτε με 4 είτε με 2 καλώδια. Πολύ σύνηθες είναι να χρησιμοποιείται σε

συνδέσεις τύπου Master - Slave, όπου υπάρχει ένας Master και πολλοί Slaves συνδεσμολογία που υλοποιήθηκε και στην εργασία.

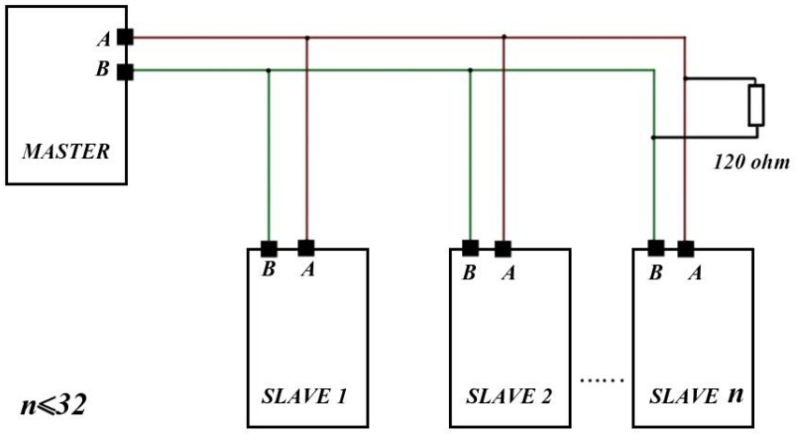


Figure 6 - Σύνδεση RS-485 Master Slave με αντίσταση εξισορρόπησης 120 Ω

Μία γραμμή RS-485 δύο καλωδίων έχει δύο σήματα: Το A ή "+" και το B ή "-". Τα καλώδια που συνήθως χρησιμοποιούνται είναι τύπου UTP, FTP, STP, συνεστραμμένα δηλαδή ζεύγη, αναλόγως τη θωράκιση και το επίπεδο προστασίας που επιθυμούμε σε κάθε εφαρμογή. Για την ιδανική αντιστοίχιση της σύνθετης αντίστασης της γραμμής μετάδοσης με την σύνθετη αντίσταση του υλικού (hardware) είναι απαραίτητη μία αντίσταση 120 ohm στο τέλος.

β) Το module RS-485 για Arduino

Το συγκεκριμένο module είναι μια μικρή πλακέτα βασισμένη στο ολοκληρωμένο κύκλωμα (τσιπ) MAX485 και ως σκοπό έχει τη δημιουργία διεπαφής RS-485 για Arduino ή άλλους μικροϋπολογιστές ώστε να μπορούν να υποστηρίξουν τέτοιες συνδέσεις. Το μικροτσιπ MAX485 είναι χαμηλής ισχύος πομποδέκτης που χρησιμοποιείται για RS-485 και RS-422 επικοινωνίες. Η πλακέτα του module δεν είναι τίποτα άλλο παρά το τσιπ MAX485 με έτοιμες τις συνδέσεις των αντιστάσεων, των γειώσεων, της τροφοδοσίας καθώς και ειδικές κλέμες για την σύνδεση των καλωδίων. Να σημειωθεί οτι περιλαμβάνει και την αντίσταση 120 ohm που χρειάζονται οι περισσότερες RS-485 συνδέσεις. Παρακάτω βλέπουμε το module καθώς και το διάγραμμα συνδέσεών του.

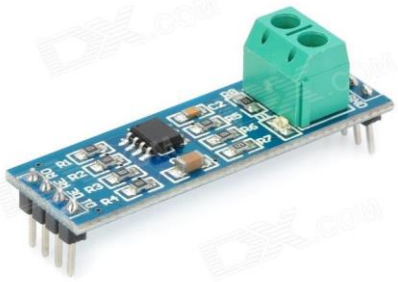


Figure 7 - Το module RS-485 για Arduino

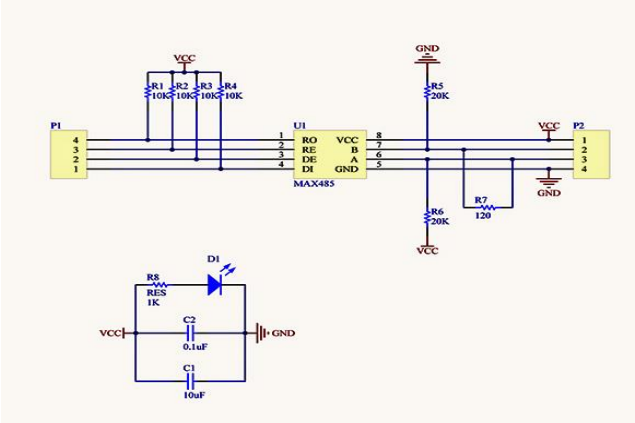


Figure 8 - Διάγραμμα της πλακέτας του module RS-485

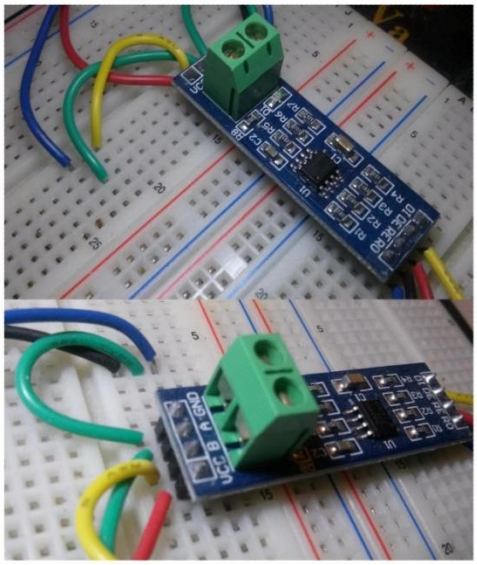


Figure 9 - Το module RS-485 κατά τη διαδικασία δοκιμών

Το pinout του RS-485 module το βλέπουμε στην εικόνα που ακολουθεί:

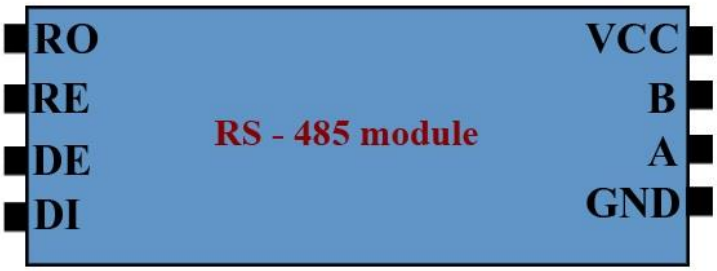


Figure 10 - Οι ακροδέκτες του RS-485 module

Το module έχει τους παρακάτω ακροδέκτες:

- RO - Receiver Out
- RE - Receiver Enable (Low)
- DE - Driver Enable (High)
- DI - Driver In
- VCC - Power (+5 volt)
- B - Data pin B
- A - Data pin A
- GND - Ground

γ) Σύνδεση του module RS-485

Η χρήση του module αυτού είναι σημαντική μιας και μέσω αυτού γίνεται η επικοινωνία μεταξύ των TLK 31 και του Arduino. Με τη βοήθειά του δηλαδή, καταφέρνουμε να καταστήσουμε τον Arduino Master και τα TLK 31 Slaves. Έπειτα με κατάλληλο κώδικα που τρέχει στο Arduino και με την υλοποίηση του πρωτόκολλου MODBUS RTU που θα δούμε στο ανάλογο κεφάλαιο, γίνεται η ανάγνωση των δεδομένων, των τιμών δηλαδή των αισθητήρων, που έχουν στη μνήμη τους τα TLK 31, στο Arduino.

Πριν όμως δείξουμε το γενικό σχέδιο σύνδεσης Arduino - module RS 485 και TLK 31, ας δούμε πιο αναλυτικά τη σύνδεση με το module. Το module όπως είδαμε πιο πάνω έχει 8 συνολικά pins. Τα pin Vcc, Gnd, A και B, συνδέθηκαν αντίστοιχα, με την τάση 5 volt που δίνει το Arduino, την γείωση του Arduino και με τις ανάλογες κλέμες data (A και B) που παρέχει ο πίνακας του ελαιουργείου μέσω ενός RS 485 Terminal Bridge Connector που υπάρχει στη βάση του κουτιού του πίνακα ελέγχου. Το συγκεκριμένο Terminal Connector βγάζει 3 κλέμες για χρήση RS 485. Η μία είναι η γείωση και οι

άλλες δύο η γραμμή A και η γραμμή B που έρχονται από τα TLK 31. Το συγκεκριμένο Terminal ουσιαστικά, είναι η κατάληξη των RS 485 pins των TLK 31.

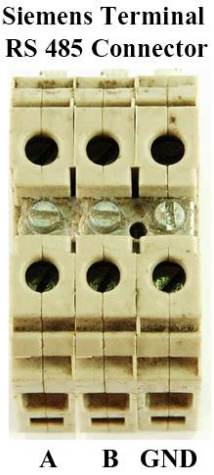


Figure 11 - Bridge Terminal για την RS-485 σύνδεση (Ελαιουργείο)

Τα υπόλοιπα 4 pins του module, δηλαδή τα RO, RE, DE και DI που είδαμε στο pinout συνδέθηκαν όπως δείχνει η κάτω εικόνα:

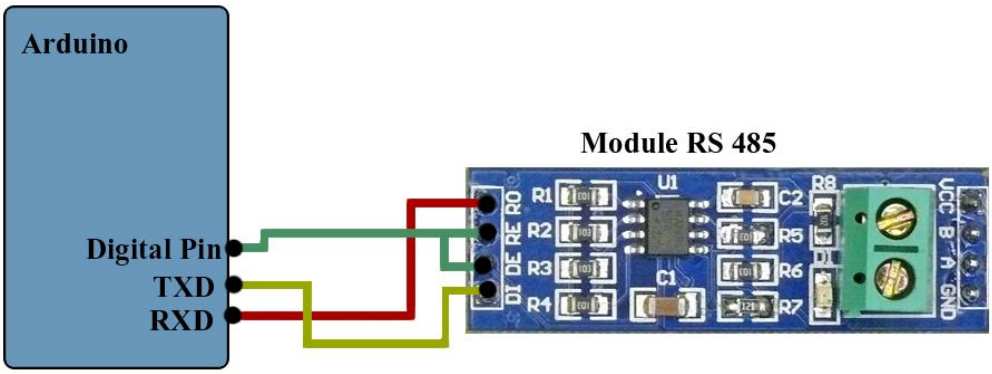


Figure 12 - Σύνδεση Arduino - RS485 module

Το pin DI συνδέθηκε σε ένα από τα TXD pins του Arduino (ο Arduino Mega έχει 3), αντίστοιχα και το RO συνδέθηκε με ένα από τα RXD pins. Τέλος τα DE και RE συνδέονται στο ίδιο σημείο, σε ένα συγκεκριμένο Digital pin της επιλογής μας στο Arduino το οποίο θα είναι και το pin ελέγχου του module μέσα στον κώδικα.

δ) Σχέδιο γενικής σύνδεσης:

Η συνολική σύνδεση Arduino - module RS 485 - TLK 31 δίνεται στο σχέδιο που ακολουθεί:

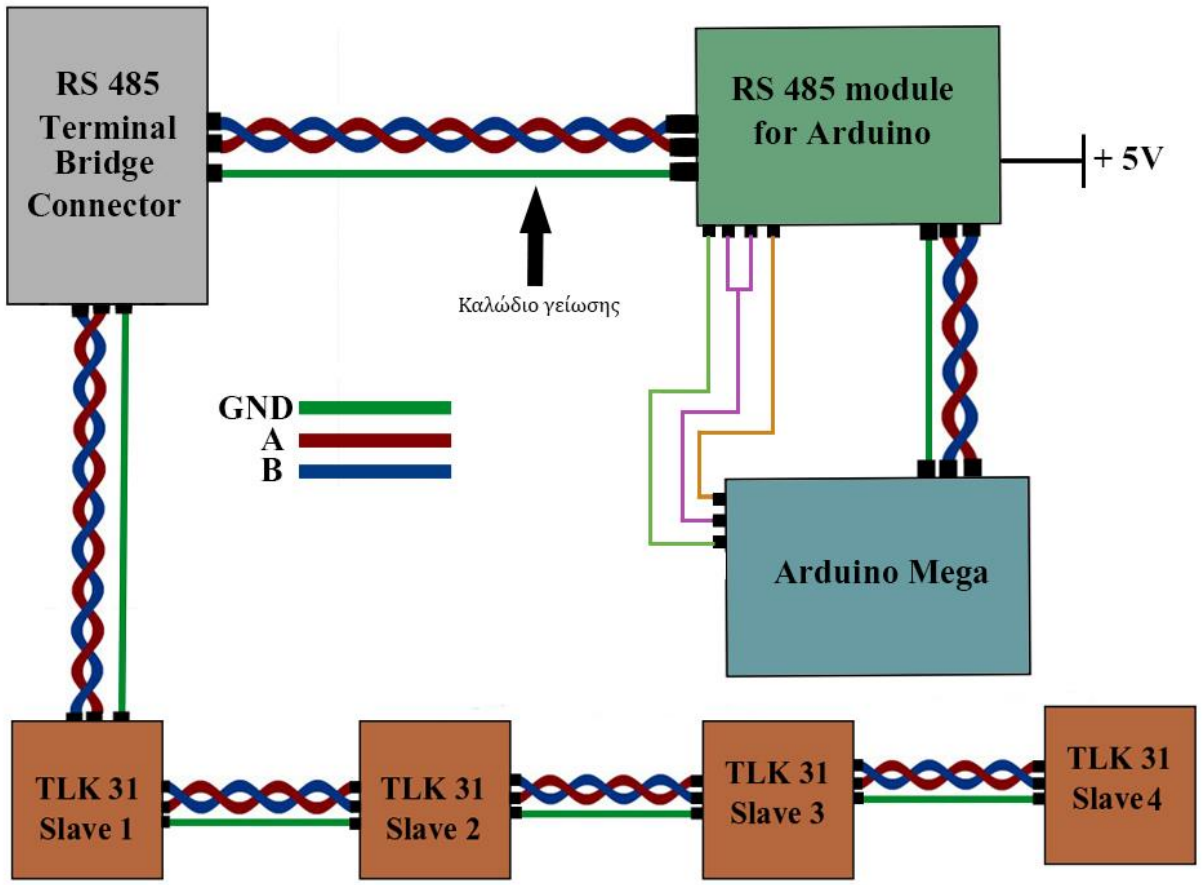


Figure 13 - Σχέδιο σύνδεσης TLK31 - Bridge - module - Arduino στο Ελαιουργείο

Σε πρώτη φάση τα TLK 31 συνδέθηκαν μεταξύ τους - όπως έχουμε δει και στο ανάλογο κεφάλαιο για αυτά. Μέσα τους έχουν αποθηκευμένες σε θέσεις μνήμης τις τιμές των αισθητήρων τις οποίες εμείς θα διαβάζουμε. Η μεταξύ τους σύνδεση ακολουθεί την τοπολογία Multipoint που έχουμε περιγράψει ήδη στις επικοινωνίες RS 485. Το καλώδιο που επιλέχθηκε είναι συνεστραμμένο ζεύγος FTP συν άλλο ένα καλώδιο για την γείωση. Το καλώδιο FTP λόγω του ειδικού shield που το περιβάλλει μας παρέχει επιπλέον προστασία από παρεμβολές. Από τα TLK 31 φεύγει καλώδιο το οποίο κατεβαίνει στη βάση από το κουτί ελέγχου του ελαιουργείου όπου και βγαίνουν 3 κλέμες (A, B και Gnd) μέσω του Terminal Connector που δείξαμε. Εδώ θα γίνει και η σύνδεση στα ανάλογα pins του module (A, B και Gnd) πάλι με καλώδιο FTP για προστασία από τυχόν παρεμβολές. Στο module έπειτα, έχουμε από την άλλη μεριά τις συνδέσεις με το Arduino που δείξαμε σε αναλυτική εικόνα. Τέλος το module συνδέεται και στα ανάλογα pins γείωσης και τροφοδοσίας (5v) του Arduino.

Οι συνδέσεις των TLK 31 μεταξύ τους, αλλά και του Terminal Connector με το module έγιναν με καλώδιο FTP Cat6. Πρόκειται για καλώδιο υψηλής θωράκισης κατάλληλο για βιομηχανικές εφαρμογές. Αποτελείται από τέσσερα ζεύγη συνεστραμμένων χάλκινων αγωγών με μονωτικό περίβλημα. Γύρω τους υπάρχει ειδικός μανδύας PVC και γύρω όλων μαζί υπάρχει η θωράκιση που είναι επιμεταλλωμένο φύλο. Τέλος όλο αυτό περικλείεται από την εξωτερική προστασία (συνήθως πλαστικό, PVC κλπ). Τα καλώδια έχουν επιπλέον ένα σύρμα το οποίο χρησιμεύει σαν γείωση καθώς και ειδικό πλαστικό σταυρό ώστε να κρατάει τα τέσσερα ζεύγη χωρισμένα.

Αυτό το βασικό σχέδιο σύνδεσης των TLK 31 μέσω του module με το Arduino είναι το hardware μέρος. Για να υλοποιηθεί ολοκληρωτικά η επικοινωνία στην πράξη, θα γραφτεί ειδικός κώδικας από τη μεριά του Arduino στη γλώσσα wiring C και θα εφαρμοστεί το πρωτόκολλο MODBUS RTU (η ανάλυση του στο ανάλογο κεφάλαιο). Ο συνδυασμός hardware (η παραπάνω διαδικασία) και software (το πρωτόκολλο MODBUS RTU) θα δώσει το αποτέλεσμα του διαβάσματος των τιμών των αισθητήρων θερμοκρασίας μέσω των TLK 31.

Κατά την σύνδεση των τεσσάρων TLK 31 μεταξύ τους έπρεπε να πραγματοποιηθούν κατάλληλες ρυθμίσεις ώστε να γίνει προετοιμασία για την εφαρμογή του πρωτοκόλλου επικοινωνίας MODBUS RTU. Έτσι λοιπόν με τη βοήθεια του αντίστοιχου manual του κατασκευαστή, ρυθμίζονται ώστε να έχουν έναν μοναδικό αριθμό ID με τη σειρά (1, 2, 3, 4) καθώς επίσης και το επιθυμητό Baud Rate, που στην περίπτωσή μας είναι 9.600 bit/sec.

3.4 APC220 module και ραδιοκύμματα

Πριν αναλυθεί το τι είναι, πως ρυθμίζεται και πως λειτουργεί το APC 220 (το module πομποδέκτη το οποίο χρησιμοποιήθηκε στην εργασία) θα εξηγήσουμε πρώτα κάποια βασικά για τα ραδιοκύματα και τη διαμόρφωση σήματος, ώστε να μπορούμε έπειτα να εξηγήσουμε την λειτουργία του APC 220. Με το APC220 υλοποιήθηκε η επικοινωνία Arduino - τερματικού Η/Υ ώστε να μεταφέρονται οι τιμές των αισθητήρων.

α) Τα ραδιοκύμματα

Τα ραδιοκύματα είναι τα ηλεκτρομαγνητικά κύματα που έχουν συχνότητα από 3Hz έως 300Ghz και χρησιμοποιούνται στις τηλεπικοινωνίες. Το εύρος μέσα στο οποίο λειτουργεί το APC 220 (418 - 455 Mhz) ανήκει στο εύρος (300 - 3000 Mhz), δηλαδή στα UHF (Ultra High Frequency κύματα) όπου το μήκος κύματος είναι 10 - 100 εκατοστά και χρησιμοποιούνται για κινητά τηλέφωνα, ασύρματα δίκτυα Η/Υ, κλειδαριές αυτοκινήτων και γκαράζ, φούρνους μικροκυμάτων, τηλεοπτικές εκπομπές, στρατιωτικές υπηρεσίες, ραντάρ κ.α.

Τα ραδιοκύματα είναι ουσιαστικά η κίνηση ηλεκτρικών φορτίων μεταξύ κεραιών και όπως και όλα τα ηλεκτρομαγνητικά κύματα διαδίδονται με την ταχύτητα του φωτός (3x10^8) ευθύγραμμα. Όταν παρεμβάλλεται κάποιο υλικό, τότε η ταχύτητά τους μικραίνει και εξαρτάται από συγκεκριμένα χαρακτηριστικά όπως η διηλεκτρική σταθερά, η μαγνητική διαπερατότητα και η αγωγιμότητα τους. Η εμβέλεια του κύματος κατά την πορεία από τον πομπό στον δέκτη, δηλαδή η μέγιστη απόσταση που

μπορεί να διανύσει εξαρτάται από τη συχνότητα του κύματος, την ισχύ του πομπού καθώς και από χαρακτηριστικά του περιβάλλοντος (ατμόσφαιρα, έδαφος κ.α.). Οι συγκεκριμένες συχνότητες στις οποίες εκπέμπει το APC 220 (418-455Mhz) που χρησιμοποιήσαμε, είναι μέσα στις επιτρεπόμενες συχνότητες εκπομπής από το Ελληνικό κράτος, ωστόσο οι νομοθεσίες για τις εκπομπές ραδιοκυμάτων διαφέρουν από χώρα σε χώρα.

Τα κύματα αυτά είναι μη ιοντίζουσες ακτινοβολίες, δεν είναι δηλαδή ικανά να διασπούν χημικούς δεσμούς ή να προκαλούν ιοντισμό της ύλης και η μόνη επίδρασή τους στην υγεία σύμφωνα με τις μέχρι τώρα επιστημονικές έρευνες είναι η αύξηση της θερμοκρασίας των ιστών που εκτίθενται σε αυτά.

Παρακάτω βλέπουμε μία χρήσιμη εικόνα από την Εθνική Επιτροπή Τηλεπικοινωνιών και Ταχυδρομείων (ΕΕΤΤ) όπου μπορούμε να δούμε τις συχνότητες των κυμάτων και το πως αυτά χωρίζονται βάση το μήκος κύματός του και της συχνότητάς τους.

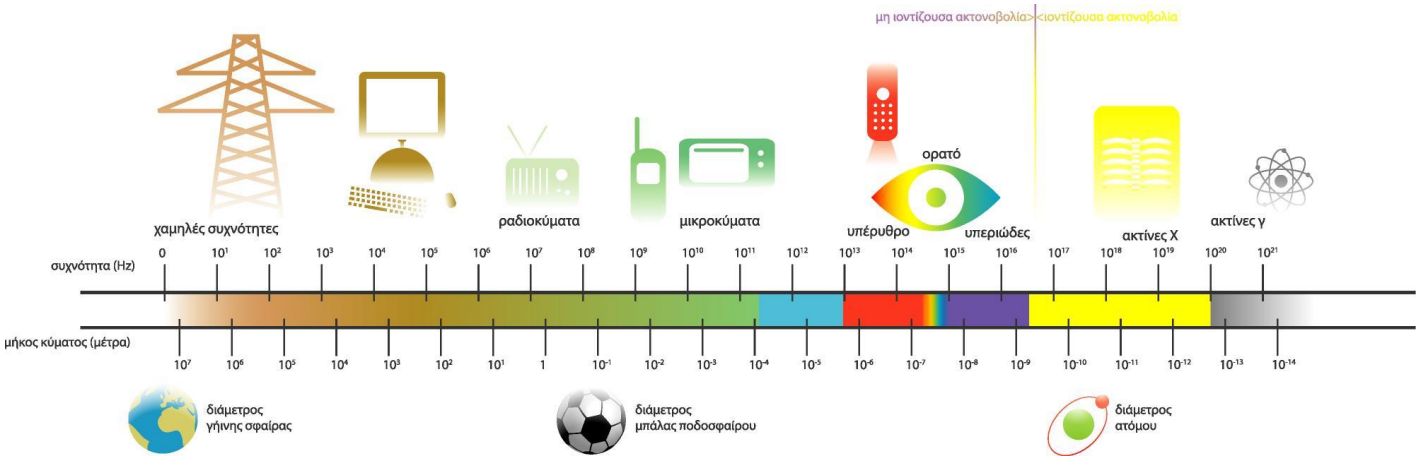


Figure 14 - H/M κύματα και συχνότητες

β) Η κεραία

Για να μεταδοθεί με επιτυχία το κύμα από τον πομπό στον δέκτη μέσω των κεραιών θα πρέπει να υπάρχει η λεγόμενη line of sight (LOS)- οπτική επαφή μεταξύ τους. Αν δεν υπάρχει αυτή, τότε το κύμα μεταδίδεται με μηχανισμούς διάδοσης όπως η ανάκλαση, η διάθλαση κ.α.
Οι κεραίες αυτές πρέπει να είναι συγκεκριμένου τύπου ώστε να έχουν το ιδανικό αποτέλεσμα. Στην περίπτωσή της εργασίας μας οι κεραίες είναι παντοκατευθυντικές κεραίες τύπου rubber ducky. Αυτές οι κεραίες χρησιμοποιούνται αντί για τις υπόλοιπες κεραίες UHF γιατί λόγω κατασκευής έχουν πολύ μικρότερο μήκος. Μία τέτοια κεραία αποτελείται από ένα σύρμα το οποίο έχει ελικοειδή μορφή και προστατεύεται από πλαστικό κάλυμμα. Συνήθως έχει το 4% με 15% του μήκους κύματος και άρα είναι αρκετά μικρότερη από τις κλασσικές μονοπολικές κεραίες που έχουν μήκος 1/4 του μήκους κύματος λ. (όπου λ=v/f, μήκος κύματος με f = συχνότητα και v=ταχύτητα του φωτός).
Ωστόσο παρουσιάζει όπως είναι λογικό χαμηλότερο κέρδος και άρα έχει περιορισμένη εμβέλεια, για

αυτό και χρησιμοποιείται σε εργασίες που δεν απαιτούν πολύ μεγάλες αποστάσεις. Η κεραία που χρησιμοποιήθηκε παρουσιάζεται στην παρακάτω εικόνα.

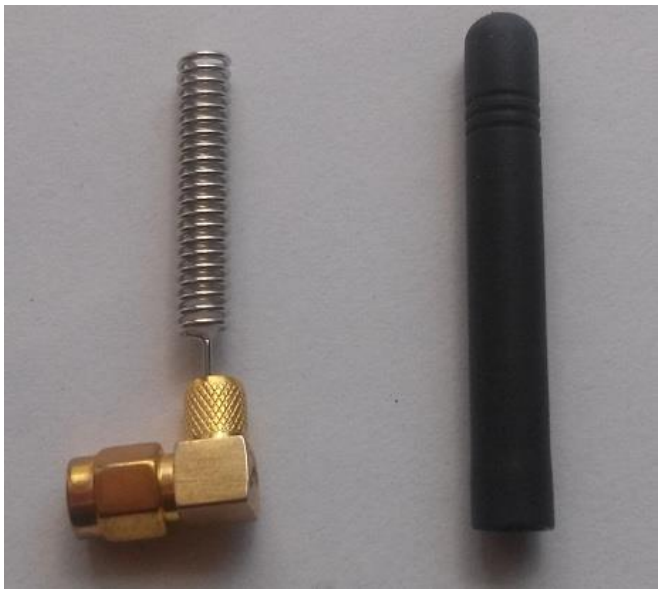


Figure 15 - Η κεραία που χρησιμοποιήθηκε

γ) Διαμόρφωση και μετάδοση του σήματος

Για να φτάσει το σήμα από τον πομπό στον δέκτη θα πρέπει να υποστεί διαμόρφωση. Με τη διαμόρφωση, η πληροφορία που πρέπει να μεταδοθεί μεταβάλλει ένα υψίσυχνο σήμα ώστε να μπορέσει να μεταδοθεί η ίδια. Αυτό το υψίσυχνο σήμα λέγεται φέρον και η διαδικασία αυτή γίνεται διότι αλλιώς θα ήταν αδύνατη η κατευθείαν μετατροπή της πληροφορίας (σήματος) σε ηλεκτρομαγνητική ακτινοβολία. Έτσι με τη χρήση του φέροντος καταφέρνουμε καλύτερη ποιότητα εκπομπής με ελάχιστες παρεμβολές από άλλες συχνότητες και συσκευές. Επίσης λόγω του τύπου για το μήκος κύματος (λ=v/f) και το ότι οι κεραίες είναι υποπολλαπλάσια του λ, με το υψίσυχνο φέρον μειώνουμε το μήκος της κεραίας.

Υπάρχουν διάφορα είδη διαμόρφωσης όπως κατά πλάτος (AM), συχνότητας (FM) καθώς και ψηφιακές διαμορφώσεις όπως FSK, ASK, PSK. Στην AM διαμόρφωση, για να μεταδοθεί με επιτυχία η πληροφορία υπάρχει ο ταλαντωτής, ένα ειδικό κύκλωμα που παράγει το φέρον σε υψηλές συχνότητες. Το φέρον αυτό, ενισχύεται και έπειτα διαμορφώνεται με το σήμα που θέλουμε να μεταδώσουμε. Τέλος αφού ενισχυθούν ξανά ώστε το τελικό σήμα να έχει την απαιτούμενη ισχύ, το συνολικό σήμα οδηγείται στην κεραία η οποία πλέον διαρρέεται από ρεύμα και άρα δημιουργεί ηλεκτρομαγνητικό κύμα όπως φαίνεται στην παρακάτω εικόνα.

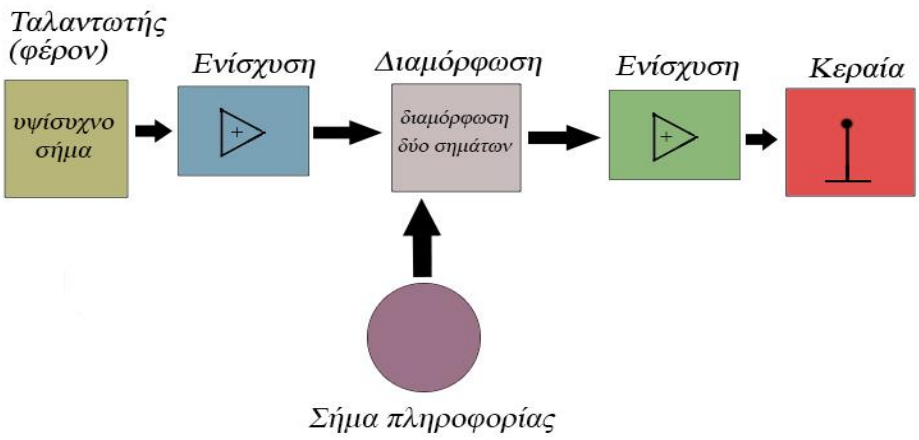


Figure 16 - Διαμόρφωση AM

Στην περίπτωση του πομποδέκτη που χρησιμοποιήσαμε για την εργασία (APC 220 Rf module), αυτός εφαρμόζει διαμόρφωση GFSK. Η διαμόρφωση αυτή είναι ψηφιακή και αυτό που κάνει είναι να παρουσιάζει τα ψηφιακά δεδομένα που είναι προς αποστολή ως αλλαγές στη συχνότητα του φέροντος. Επιπροσθέτως χρησιμοποιεί ένα γκαουσιανό φίλτρο (Gaussian filter) για καλυτερα αποτελέσματα και ποιότητα. Παράδειγμα της GFSK διαμόρφωσης μπορούμε να δούμε στην εικόνα που ακολουθεί. Στο πρώτο κομμάτι φαίνονται τα ψηφιακά δεδομένα που θέλουμε να στείλουμε. Ακριβώς απο κάτω φαίνεται το φέρον το οποίο έχει υψηλή συχνότητα και τέλος φαίνεται πως γίνεται η συγκεκριμένη διαμόρφωση αλλάζοντας τη συχνότητα του φέροντος ανάλογα με το πλάτος των παλμών των δεδομένων που θέλουμε να μεταδώσουμε. Ενδεικτικά, η συχνότητα αυξάνει για τη μετάδοση δεδομένων λογικού '1' και μειώνεται για τη μετάδοση δεδομένων λογικού '0'.

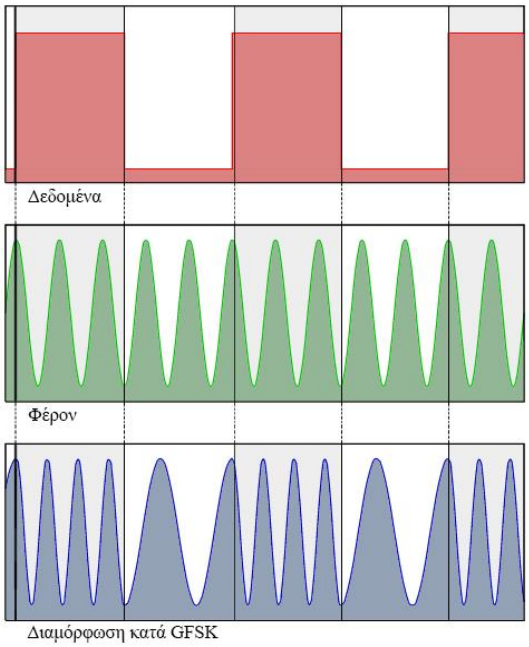


Figure 17 - Αρχή λειτουργίας διαμόρφωσης GFSK

δ) Ο πομποδέκτης APC 220 rf module

Για την εργασία χρησιμοποιήσαμε τον πομποδέκτη APC 220 RF module. Πρόκειται για έναν semi-duplex πομποδέκτη χαμηλής ισχύος. Μεταξύ των υπόλοιπων module που υπάρχουν στην αγορά, υπερισχύει σε σχέση με το γνωστό Xbee λόγω του χαμηλότερου κόστους, αλλά και του nRF24l01 λόγω της ευελιξίας και της απλότητας στον προγραμματισμό του και τις ρυθμίσεις του. Είναι απλός στην εφαρμογή του και παρέχει πάνω από 100 διαφορετικά κανάλια σε εμβέλεια 1.200 μέτρα line of sight (οπτική επαφή).

Figure 18 - APC220 Rf module

Τα κυριότερα τεχνικά χαρακτηριστικά δίνονται παρακάτω:

- Εύρος συχνοτήτων λειτουργίας: 418 - 478 MHz
- Εμβέλεια: 1.200 μέτρα LOS
- Κανάλια: πάνω από 100
- Διαμόρφωση: GFSK
- Interface: UART/TTL
- Baud rate: 1200 - 19200 bps

- Spacing καναλιών: 0.2 MHz
- Data buffer: 256 Bytes
- Power supply: +3.3 - 5.5 V
- Κατανάλωση: Rx current: 28mA, Tx current: 35mA
- Θερμοκρασίες λειτουργίας: -30 - 85 °C
- Μέγεθος: 37.5 x 18.3 x 7.0 (mm)
- Βάρος: 30 gr

Το σύστημα αποτελείται από δύο πλακέτες, δύο κεραίες και έναν αντάπτορα TTL to USB Converter για σύνδεση με Η/Υ. Το μεγάλο του πλεονέκτημα είναι ότι συνδέεται σε θύρα USB και έχει ενσωματωμένη λειτουργία για την δημιουργία σειριακής μετάδοσης χωρίς να χρειάζεται κάποιος ιδιαίτερος προγραμματισμός. Ακόμα, χρησιμοποιεί την μέθοδο FEC (Forward Error Correction) ώστε να παρέχει μετάδοση χωρίς σφάλματα αλλά και διαμόρφωση GFSK για ελαχιστοποίηση παρεμβολών από άλλες συσκευές και κοντινές συχνότητες. Στην παρακάτω εικόνα βλέπουμε το pinout της πλακέτας:

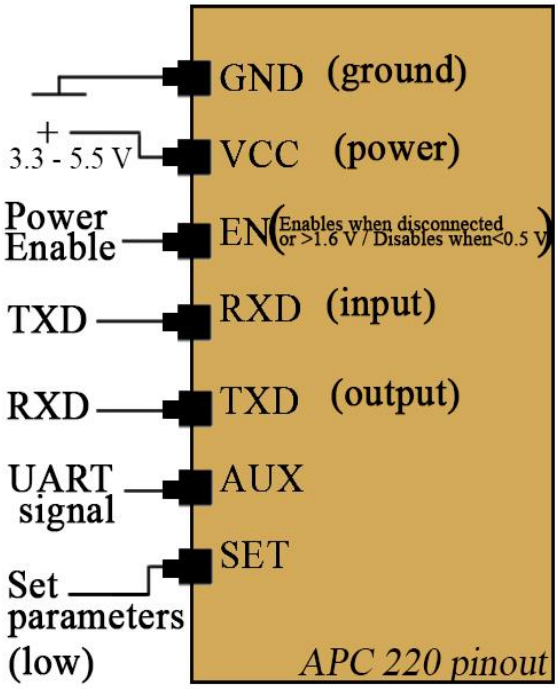


Figure 19 - Pinout πλακέτας APC 220 rf module

Στο GND και VCC συνδέεται αντίστοιχα γείωση και 5 volt από το Arduino. Το EN χρησιμεύει για να απενεργοποιεί ή να ενεργοποιεί το APC220. Τα pins TXD και RXD είναι τα δύο pins εισόδου εξόδου αντίστοιχα με τα οποία θα γίνεται η μετάδοση των δεδομένων. Το AUX pin χρησιμεύει στο να θεωρείται το εκάστοτε module ως μεταδότης (high) ή λήπτης (low). Τέλος το pin SET χρησιμοποιείται για να μπαίνει η πλακέτα σε κατάσταση ρύθμισης (low). Το πακέτο συνοδεύεται επίσης από software ώστε να γίνονται οι ρυθμίσεις σε κάθε module. Πιο κάτω θα δούμε πως ακριβώς συνδέουμε στο Arduino αλλά και στον Η/Υ τα module και πως τα ρυθμίζουμε.

ε) Ρυθμίσεις APC220

Αρχικά θα πρέπει να ρυθμίσουμε σε κάθε module τις παραμέτρους μέσω του software της εταιρείας. Το software το βρίσκουμε διαθέσιμο δωρεάν στη σελίδα ***https://www.dfrobot.com*** και ονομάζεται ***RF-magic***. Έπειτα από ***το http://www.silabs.com*** θα εγκαταστήσουμε τους ειδικούς drivers για Η/Υ με όνομα ***Silicon Labs CP210x USB to UART Bridge***. Είναι απαραίτητο να εγκαταστήσουμε τους drivers αυτούς στον Η/Υ ώστε να μπορεί ο υπολογιστής να εντοπίζει τον UART/TTL αντάπτορα στον οποίο θα συνδέσουμε τον δέκτη. Αφού εγκατασταθούν, θα πρέπει από την διαχείριση συσκευών των Windows να ελέγξουμε την σειριακή θύρα στην οποία εγκαταστάθηκαν (στην περίπτωσή μας COM 5). Αφού έχουμε εγκατεστημένο και το software RF-magic, προχωράμε στις ρυθμίσεις. Αφού συνδέσουμε το πρώτο module στον αντάπτορα και τον αντάπτορα στην θύρα USB του Η/Υ, γίνονται οι απαραίτητες ρυθμίσεις στο ειδικό software (RF-magic). Η εικόνα κάτω μας δείχνει το περιβάλλον της εφαρμογής.

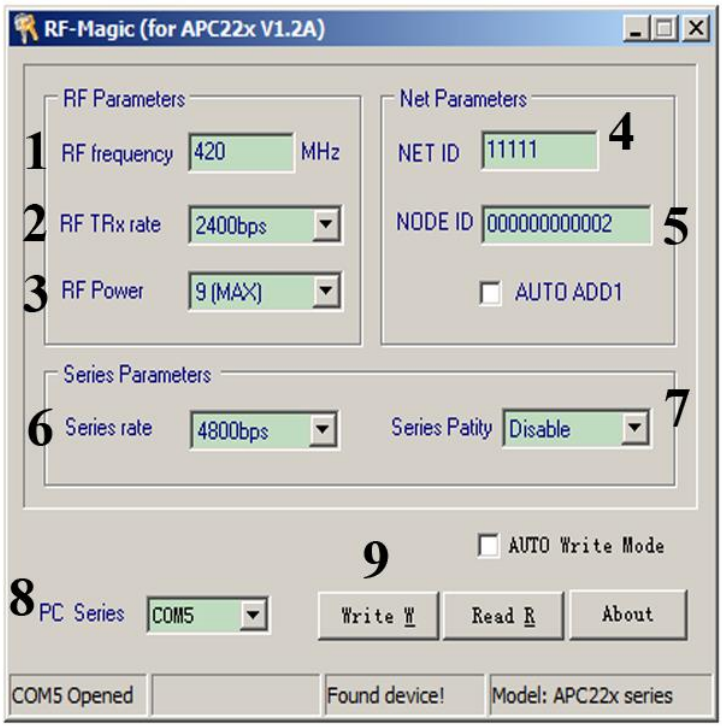


Figure 20 - Ρυθμίσεις APC220 μέσω RF-Magic

Τα βήματα που ακολουθούνται είναι τα παρακάτω:

Βήμα 1: Επιλέγεται η επιθυμητή συχνότητα εκπομπής (420 MHz στην περίπτωσή μας - η επιλογή της συγκεκριμένης συχνότητας έγινε τυχαία, θα μπορούσε να είναι κάποια άλλη που υποστηρίζεται από το module, ωστόσο πρέπει να είναι ίδια και στα δύο)
Βήμα 2: Επιλέγεται το TRx rate
Βήμα 3: Ρύθμιση ισχύος. Αφήνουμε το 9 (MAX) ώστε να έχουμε την μέγιστη ισχύ.
Βήμα 4: Ορίζεται το ID του δικτύου μας με έναν 5ψήφιο αριθμό
Βήμα 5: Ορίζεται το ID του συγκεκριμένου κόμβου (της πλακέτας)
Βήμα 6: Επιλέγεται το Serial rate που θα πρέπει να είναι ίδιο με την σειριακή θύρα στον H/Y στην οποία εγκαταστήσαμε τους drivers. Οπότε ό,τι έχει επιλεγεί το βάζουμε από τη διαχείριση συσκευών και στον H/Y.
Βήμα 7: Απενεργοποίηση του Parity bit
Βήμα 8: Επιλογή της σειριακής θύρας στην οποία έχουμε εγκαταστήσει τους drivers
Βήμα 9: Πατάμε το πλήκτρο Write ώστε να εγγραφούν τα δεδομένα στο πρώτο module

Την ίδια διαδικασία κάνουμε και για το άλλο module, αλλάζοντας απλά το Node ID του. Στην περίπτωση που δε δουλέψει, μπορούμε να βάλουμε ίδια Node ID. Πρέπει ωστόσο να προσέξουμε όλα τα άλλα να είναι ακριβώς ίδια και στα δύο modules. Με τις ρυθμίσεις έτοιμες και στα δυο modules, η υλοποίηση της διασύνδεσης των APC 220 φαίνετια στο παρακάτω σχέδιο:

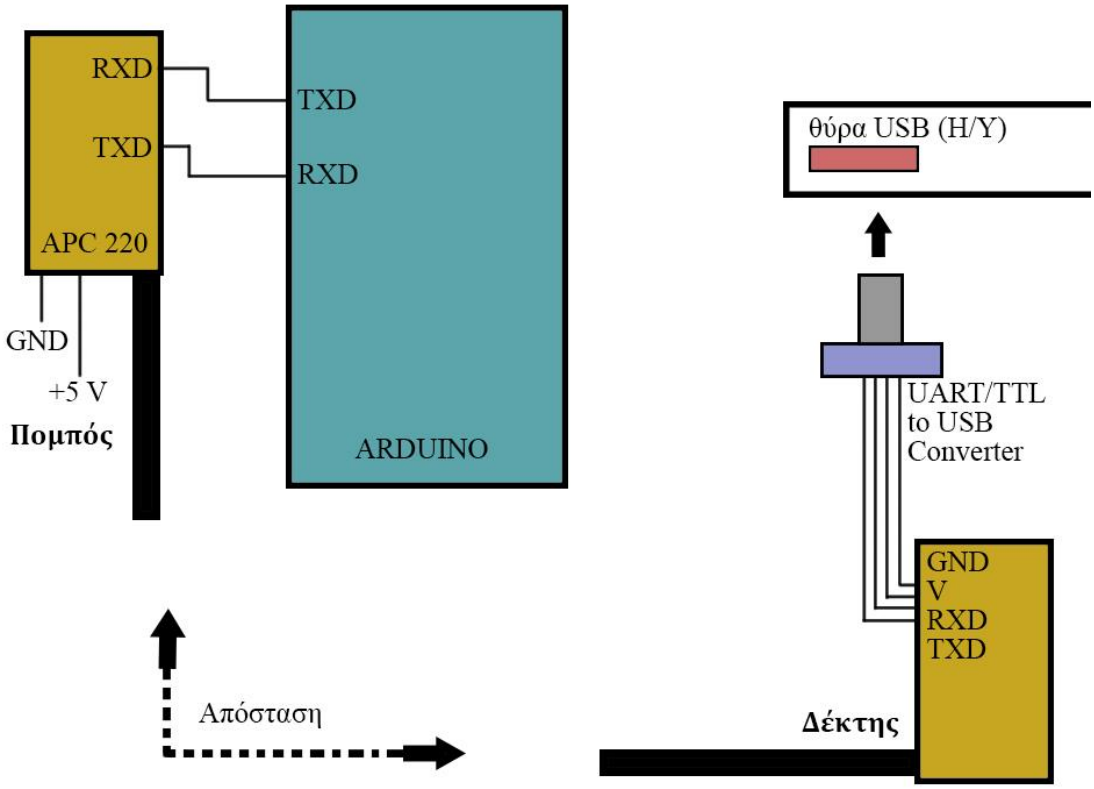


Figure 21 - Διασύνδεση APC 220 rf module

Όπως φαίνεται το RXD pin του πομπού θα συνδεθεί με το TXD pin του Arduino και το TXD pin του πομπού με το RXD του Arduino. Ο πομπός παίρνει τροφοδοσία και γειώνεται από τα αντίστοιχα pins του Arduino. Στον δέκτη θα πρέπει να συνδέσουμε όλα τα pins εκτός των EN, AUX και SET στον αντάπτορα και έπειτα να συνδέσουμε τον αντάπτορα στην θύρα USB του H/Y. Είναι σημαντικό να απομονωθούν με κάποιο τρόπο αυτά τα pins, διαφορετικά δε θα λειτουργεί ο δέκτης. Στην παρακάτω εικόνα μπορούμε να δούμε τον τρόπο σύνδεσης που επιλέχθηκε μεταξύ δέκτη και αντάπτορα ώστε να είναι απομονωμένα τα pins SET, EN και AUX.

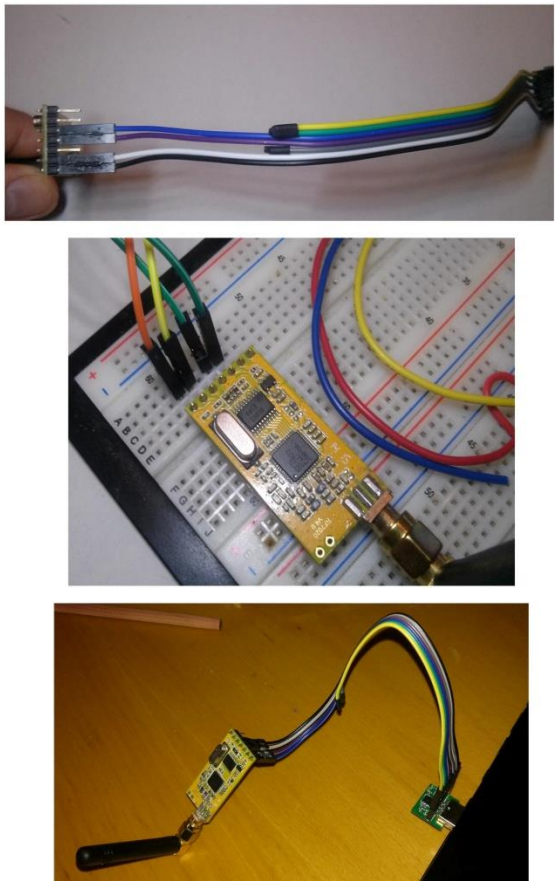


Figure 22 - Το APC 220 rf module σε δοκιμές

3.5 Αισθητήρας ροής

Για την μέτρηση ροής νερού στο ελαιουργείο χρησιμοποιήθηκαν αισθητήρες ροής νερού με έξοδο τετραγωνικό παλμό. Οι συγκεκριμένοι αισθητήρες εγκαταστάθηκαν σε σημεία όπου υπήρχε ανάγκη για έλεγχο της ροής νερού και με τη βοήθειά τους μπορούμε να βλέπουμε είτε την ροή σε πραγματικό

χρόνο, είτε τη συνολική ποσότητα νερού που πέρασε. Στην κάτω εικόνα φαίνεται ο τύπος του αισθητήρα που χρησιμοποιήθηκε.

Figure 23 - Αισθητήρας ροής

Ο αισθητήρας έχει τρία καλώδια, ένα κίτρινο, ένα μαύρο και ένα κόκκινο. Το μαύρο και το κόκκινο είναι αντίστοιχα για Gnd και +5v, ενώ το κίτρινο δίνει τους παλμούς εξόδου. Στο επόμενο σχέδιο φαίνεται η λειτουργία του αισθητήρα ροής:

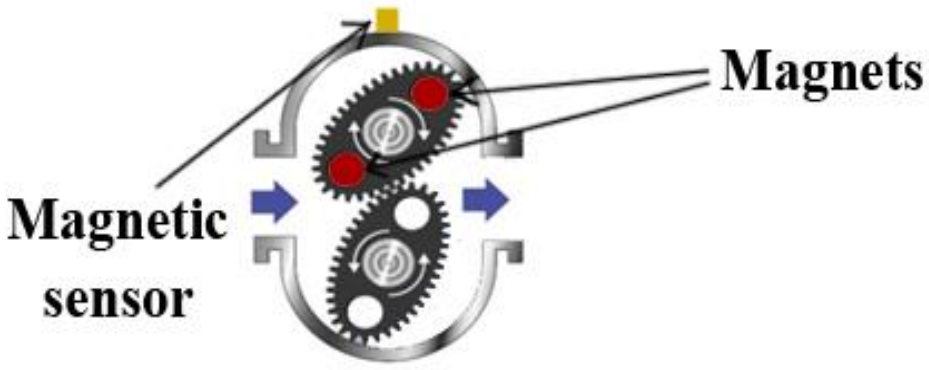


Figure 24 - Εσωτερικό του αισθητήρα ροής

Κάθε φορά που περνάει νερό, τα δύο γρανάζια κινούνται. Στο πάνω γρανάζι υπάρχουν δύο μαγνήτες. Κάθε φορά που περνάνε οι μαγνήτες αυτοί κατά την κίνηση του γραναζιού, κάτω από τον μόνιμο σταθερό μαγνητικό αισθητήρα, το ροόμετρο στέλνει παλμούς στο καλώδιο εξόδου του. Η απλή αυτή λειτουργία το καθιστά ευέλικτο για οποιαδήποτε τύπου εφαρμογή αφού μπορεί να προσαρμοστεί σε διάφορες εργασίες που απαιτούν την απλή εποπτεία του νερού αλλά και τον έλεγχό του. Στην επόμενη εικόνα φαίνονται τα στάδια λειτουργίας του αισθητήρα κατά την είσοδο νερού σε αυτόν:

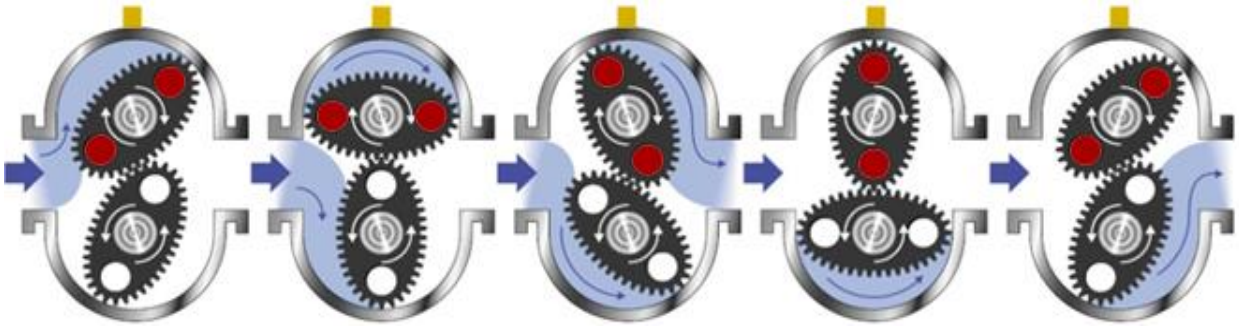


Figure 25 - Στάδια λειτουργίας αισθητήρα ροής

Παρατηρούμε πως η είσοδος και η έξοδος του νερού γίνονται σε δύο φάσεις, οι οποίες στο άθροισμά τους αποτελούν έναν κύκλο εργασίας του αισθητήρα. Αρχικά μπαίνει νερό το οποίο κατευθύνεται προς μία από τις δύο μεριές (πάνω ή κάτω). Όσο η δύναμη του νερού σπρώχνει το γρανάζι να κινηθεί, αυτό με τη σειρά του κινεί το δεύτερο γρανάζι το οποίο υποχρεώνει το υπόλοιπο νερό να περάσει και από την άλλη μεριά. Στη συνέχεια το νερό που είχε εισέλθει στην αρχή βγαίνει, και με την κίνησή του γραναζιού όπως και στην αρχή ακολουθεί η έξοδος του δεύτερου μέρους του νερού. Όσες φορές πέρασε μαγνήτης από τον μαγνητικό αισθητήρα κατά τη διάρκεια της παραπάνω διαδικασίας, τόσους παλμούς θα δώσει στην έξοδό του το ροόμετρο. Πιο γενικά, το συγκεκριμένο ροόμετρο εκμεταλλεύεται την πίεση εισόδου του νερού με τη βοήθεια των γραναζιών. Όσο περνάει νερό τα γρανάζια γυρνάνε. Αν η ροή του νερού σταματήσει, τότε και τα γρανάζια δεν γυρνάνε και άρα δεν υπάρχει κανένας παλμός στην έξοδο. Συνεπώς αν δεν υπάρχει ύπαρξη νερού, δεν υπάρχει και κίνδυνος να γυρίσουν μόνα τους τα γρανάζια και να έχουμε ανεπιθύμητους παλμούς. Καλό είναι ακόμα, να γίνεται η εγκατάσταση του συγκεκριμένου αισθητήρα ροής οριζόντια και όχι κάθετα.

Τα τεχνικά χαρακτηριστικά του ροομέτρου είναι ως παρακάτω:

- Σύνδεση: G1/2''
- Flow range: 1-30 L/min
- Τάση λειτουργίας: 3-12V (DC)
- Accuracy: +/- 0.5%
- Max πίεση: 0.5 MPa στους 20 βαθμούς Κελσίου
- Θερμοκρασία περιβάλλοντος για ορθή λειτουργία: -10 με 70 βαθμούς Κελσίου.

Όσον αφορά την μονάδα μέτρησης ώστε να γίνεται υπολογισμός των ml ή λίτρων, ακολουθήθηκε η μέθοδος της βαθμονόμησης (calibrating). Μετά από επαναλαμβανόμενες δοκιμές με συγκεκριμένη ποσότητα νερού, διαπιστώθηκε πως κάθε παλμός αντιστοιχεί σε 6.94 ml νερού για τον συγκεκριμένο αισθητήρα ροής. Αυτό είναι κάτι που θα αναλυθεί στο κεφάλαιο για τον κώδικα στο Arduino και στην εφαρμογή, όπου και θα εξηγηθεί το πως υπολογίζεται μέσω του κώδικα η ροή του νερού.

Το καλώδιο του αισθητήρα με τα τρία καλώδια (+5V, Gnd και Παλμός - Κόκκινο, Μαύρο και Κίτρινο αντίστοιχα) φαίνεται στην κάτω εικόνα:

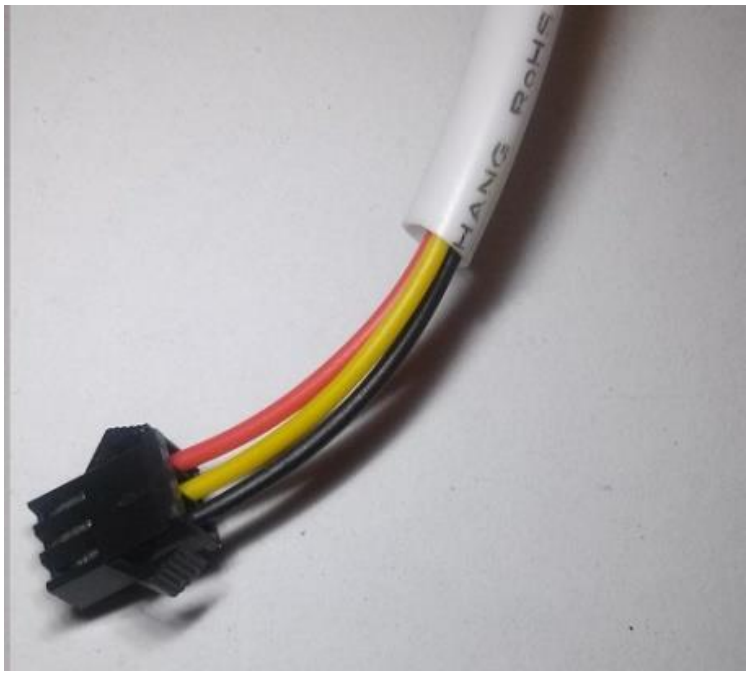


Figure 26 - Καλώδιο αισθητήρα ροής (R - Y - B)

4. Κεφάλαιο: Ανάλυση λειτουργιών

Μέχρι τώρα έχει περιγραφεί ο τρόπος συνδέσεων ώστε να λαμβάνονται οι τιμές από τους αισθητήρες στο Arduino και από εκεί να στέλνονται στον Η/Υ. Τώρα θα παρουσιαστεί αρχικά ο κώδικας που εκτελείται στο Arduino και έπειτα οι λειτουργίες των βασικών απεικονιστικών εφαρμογών που λαμβάνουν μέρος στον τερματικό Η/Υ. Αυτές είναι η βασική εφαρμογή εποπτείας, λογισμικό που εκτελείται στον υπολογιστή και παρέχει λειτουργίες απεικόνισης, αποθήκευσης, επεξεργασίας κ.α., καθώς και η ιστοσελίδα που μετα-φορτώνεται για να υπάρχει και απομακρυσμένη πρόσβαση στις τιμές των αισθητήρων. Σε αυτό το κεφάλαιο θα παρουσιαστεί η χρήση τους από την πλευρά του χρήστη.

4.1 Κώδικας Arduino (MODBUS RTU protocol)

Ο κώδικας στον Arduino είναι κρίσιμος για τη λειτουργία του συστήματος. Είναι το πρωταρχικό στάδιο συγκέντρωσης και επεξεργασίας των δεδομένων και από τον κώδικά αυτό εξαρτάται το αν τα δεδομένα θα φτάνουν σωστά και χωρίς προβλήματα στο τερματικό Η/Υ. Στην εργασία αυτή, ο κώδικας που εκτελείται στο Arduino:

1. Είναι υπεύθυνος για την επικοινωνία με τις συσκευές TLK 31 και για τη λήψη των δεδομένων των αισθητήρων θερμοκρασίας
2. Είναι υπεύθυνος για τη σωστή λήψη των δεδομένων των αισθητήρων ροής
3. Είναι υπεύθυνος για την ορθή αποστολή των δεδομένων

α) Πρωτόκολλο MODBUS RTU και TLK31

Όπως αναφέραμε και σε προηγούμενο κεφάλαιο η ανάγνωση των τιμών των αισθητήρων θερμοκρασίας γίνεται μέσω συγκεκριμένης τεχνικής. Αναφέρθηκε οτι επειδή οι αισθητήρες είναι συνδεμένοι ήδη σε 4 αντίστοιχες συσκευές controller - display (**TLK-31**), δεν είναι δυνατή η ανάγνωσή των τιμών τους απευθείας από τα καλώδιά τους. Κάτι τέτοιο θα μας υποχρέωνε να εφαρμόσουμε τάση στους αισθητήρες ώστε να διαβάζουμε έπειτα την μεταβολή της τάσης μιας και στο Arduino δεν μπορούμε να διαβάσουμε αντίσταση, αλλά τάση την οποία έπειτα θα την μεταφράζαμε σε αντίσταση. Ωστόσο εαν γινόταν αυτό, θα υπήρχε πρόβλημα καθώς στα καλώδια των αισθητήρων εφαρμόζεται ήδη μέθοδος ανάγνωσης (μέσω κυκλώματος) από τις συσκευές TLK-31 οι οποίες διαβάζουν τους αισθητήρες και δείχνουν στις οθόνες τους τις τιμές. Επομένως θα πρέπει η ανάγνωση των τιμών στο Arduino να γίνει με άλλη μέθοδο χωρίς να επέμβουμε στη σύνδεση PT-100 και TLK-31 (αισθητήρα και συσκευής displaying). Αυτό όπως περιγράψαμε γίνεται αν εκμεταλλευτούμε τα 3 pins που έχει η συσκευή TLK-31 αφιερωμένα σε σειριακή σύνδεση τύπου RS-485. Πάνω σε αυτού του τύπου την σύνδεση θα εφαρμοστεί το πρωτόκολλο επικοινωνίας ***MODBUS - RTU*** το οποίο αφού περιγραφεί, θα αναλυθεί η υλοποίησή του στο Arduino.

Το ***MODBUS*** είναι ένα πρωτόκολλο επικοινωνίας που αναπτύχθηκε στα τέλη της δεκαετίας του '70 από την Modicon και αυτή τη στιγμή αποτελεί το πιο διαδεδομένο πρωτόκολλο επικοινωνίας στις

βιομηχανίες. Σύμφωνα με το μοντέλο OSI, ανήκει στο επίπεδο Εφαρμογής και χρησιμοποιείται ακόμα και σήμερα μετά από δεκαετίες σε συστήματα αυτοματισμού, SCADA κ.α. Το MODBUS έχει κάποια χαρακτηριστικά που το κατατάσσουν σε πλεονεκτική θέση έναντι άλλων μεθόδων πιο σύγχρονων, παρόλη την ηλικία του. Καταρχήν είναι αρκετά απλό στην υλοποίησή του. Έπειτα σχεδιάστηκε για βιομηχανική χρήση. Ακόμα, η χρήση του είναι ελεύθερη και τέλος το βασικότερο πλεονέκτημά του είναι ο έλεγχος σφαλμάτων που γίνονται μέσω διάφορων τεχνικών (CRC, parity check κ.α.) και το καθιστούν εξαιρετικά αξιόπιστο.

Η φιλοσοφία του είναι η ύπαρξη ενός Master και πολλών (ακόμα και εκατοντάδων) Slaves σε μία γραμμή που μπορεί να ξεπερνάει και το 1 χιλιόμετρο. Ο Master στέλνει ερωτήσεις και οι Slaves απαντάνε στον Master.Το μήνυμα που μεταφέρεται μπορεί να είναι είτε ερώτηση, είτε απάντηση, είτε μήνυμα σφάλματος. Οι τύποι δεδομένων του πρωτοκόλλου αυτού είναι 4:

Το ***Coil*** που είναι για εγγραφή και ανάγνωση,
το ***Input Discrete*** που είναι μόνο για ανάγνωση,
το ***Holding Register*** που είναι για εγγραφή και ανάγνωση και
το ***Input Register*** που είναι μόνο για ανάγνωση.

Υπάρχουν τρία βασικά είδη πρωτοκόλλων MODBUS: Το ***RTU***, το ***ASCII*** και το ***TCP***. Για τις ανάγκες της εργασίας χρησιμοποιήθηκε το RTU. Αυτό υλοποιείται σε επικοινωνία RS-485, RS-232, RS-422. Όπως είδαμε στο κεφάλαιο του RS-485, χρησιμοποιήθηκε ειδικό module RS-485 για Arduino ώστε μέσου αυτού να καταστεί το Arduino ικανό να υποστηρίξει την σύνδεση αυτή και το πρωτόκολλο MODBUS - RTU. Στην εργασία, Master έγινε το Arduino και Slaves οι 4 συσκευές TLK-31.

Στην περίπτωση της συσκευής TLK-31 ο κατασκευαστής μας πληροφορεί ότι υποστηρίζει το MODBUS RTU και πως υπάρχουν δύο βασικές συναρτήσεις που καλούνται μέσω του πρωτοκόλλου, η συνάρτηση 3 (***function 3***) που είναι για ανάγνωση τιμής από τη μνήμη της συσκευής και η συνάρτηση 6 (***function 6***) που είναι για εγγραφή. Στην περίπτωσή μας, γίνεται χρήση της συνάρτησης 3 αφού θέλουμε ο Master (Arduino) να διαβάζει τις τιμές των αισθητήρων που είναι αποθηκευμένες στις μνήμες των TLK-31. Παρακάτω βλέπουμε το πλαίσιο ενός ερωτήματος του Master καθώς και την απάντηση του Slave:

Πλαίσιο ερωτήματος Master (Arduino Mega)

slave number	3	first word address MSB	LSB	number of words MSB	LSB	CRC LSB	MSB
byte 0	byte 1	byte 2	byte 3	byte 4	byte 5	byte 6	byte 7

Πλαίσιο απάντησης Slave (TLK-31)

slave number	3	NB number of read bytes	value of first word MSB	LSB	following words	CRC LSB	MSB
byte 0	byte 1	byte 2	byte 3	byte 4	byte 5	byte NB + 2	byte NB + 3

Figure 27 - Πλαίσιο ερώτησης και απάντησης

Όπως βλέπουμε, το πλαίσιο χωρίζεται σε 5 κομμάτια. Στο ερώτημα του Master, πρώτο είναι ο μοναδικός αριθμός που έχει κάθε Slave (πχ 1,2,3 κλπ). Δεύτερο είναι το ποιά συνάρτηση καλούμε (η 3 για ανάγνωση ή η 6 για εγγραφή). Μετά ακολουθεί η διεύθυνση (σε δεκαεξαδικό σύστημα) της θέσης μνήμης που θέλουμε να διαβάσουμε. Έπειτα το πόσες θέσεις μνήμης θέλουμε να διαβάσουμε και τέλος ο έλεγχος CRC. Η απάντηση του Slave φέρει αρχικά τον μοναδικό αριθμό του, τον αριθμό της συνάρτησης που κλήθηκε, τον αριθμό των bytes που διαβάστηκαν και μετά το τι περιέχουν οι θέσεις μνήμης που θέλαμε να διαβάσουμε με τον έλεγχο σφάλματος CRC στο τέλος. Ο έλεγχος CRC είναι ειδικό τεστ που ελέγχει αν όλα είναι σωστά ως προς την μετάδοση του μηνύματος. Το TLK-31 εφαρμόζει από μόνο του έναν τέτοιον έλεγχο.

Όσον αφορά την θέση μνήμης που θέλουμε να διαβάσουμε, πληροφορούμαστε από το manual του κατασκευαστή των TLK-31 πως η θέση μνήμης ***0200 (HEX),*** είναι μεταξύ των άλλων θέσεων αυτή που κρατάει την τιμή του μετρούμενου μεγέθους. Συνεπώς εάν καταφέρουμε και υλοποιήσουμε σωστά το πρωτόκολλο MODBUS RTU στο Arduino, τότε ζητώντας να αναγνώσουμε την θέση 0200 (HEX) της μνήμης του TLK-31, θα έχουμε απευθείας την τιμή του αισθητήρα σε βαθμούς Κελσίου.

β) Βιβλιοθήκη MODBUS για Arduino και κώδικας

Για την υλοποίηση του πρωτοκόλλου χρησιμοποιήθηκε η βιβλιοθήκη ***SimpleModbusMaster*** για Arduino, αναγνωρισμένη από την επίσημη σελίδα Arduino και με συνοδευτικό manual. Η βιβλιοθήκη έχει 3 βασικές συναρτήσεις:

1. Την ***modbus_configure***
2. Την ***modbus_construct***
3. Την ***modbus_update***

Για να υλοποιηθεί η επικοινωνία πρέπει να δομούμε πακέτα (Blocks) τα οποία και θα στέλνουμε από το Arduino στο κάθε TLK-31. Μέσω της modbus_construct γίνεται η αρχικοποίηση των πακέτων, με την modbus_configure γίνεται η διαμόρφωση και με την modbus_update η ανανέωση κάθε ορισμένο χρονικό διάστημα.

Για να υλοποιήσουμε έναν πίνακα με πακέτα (στην περίπτωσή μας 4 πακέτα, όσα και τα TLK-31 στα οποία θα απευθυνόμαστε) γράφουμε το παρακάτω:

```
enum {
  PACKET1,
  PACKET2,
  PACKET3,
  PACKET4,
  TOTAL_NO_OF_PACKETS
};
Packet packets[TOTAL_NO_OF_PACKETS];
```

Κάθε πακέτο αντιστοιχεί σε ένα request του Master προς κάθε Slave, συνεπώς αφού οι Slaves είναι τέσσερις, άρα τέσσερα θα είναι και τα πακέτα. Έπειτα θα πρέπει να αρχικοποιήσουμε το μέγεθος των καταχωρητών του Master (Arduino) όπου θα αποθηκεύονται οι τιμές (ορίζουμε πχ μέγεθος 8):

```
#define TOTAL_NO_OF_REGISTERS 8
unsigned int MB_regs[TOTAL_NO_OF_REGISTERS];
```

Στη συνέχεια έρχεται η σειρά της συνάρτησης modbus_construct ώστε να αρχικοποιήσουμε τις πληροφορίες:

```
modbus_construct(&packets[PACKET1], 1, READ_HOLDING_REGISTERS, 0x0200, 2, 0);
modbus_construct(&packets[PACKET2], 2, READ_HOLDING_REGISTERS, 0x0200, 2, 2);
modbus_construct(&packets[PACKET3], 3, READ_HOLDING_REGISTERS, 0x0200, 2, 4);
modbus_construct(&packets[PACKET4], 4, READ_HOLDING_REGISTERS, 0x0200, 2, 6);
```

Το πρώτο όρισμα που βλέπουμε στην construct είναι ***ο δείκτης πίνακα*** που δείχνει στην διεύθυνση του πακέτου μέσα στον πίνακα πακέτων που φτιάξαμε ***(&packets[PACKET1]).***

Το δεύτερο όρισμα είναι ο μοναδικός αριθμός Slave (***ID number***) που για κάθε TLK-31 είναι ξεχωριστός (1, 2, 3 και 4). Αυτό προϋποθέτει οτι τα TLK-31 έχουν ρυθμιστεί χειροκίνητα από πριν ώστε να έχουν αυτές τις τιμές ID, κάτι που γίνεται έυκολα από τα πλήκτρα που έχουν πάνω τους με τη βοήθεια του manual χρήσης τους το οποίο έχει οδηγίες ώστε να ρυθμιστεί κάθε συσκευή με τον κατάλληλο μοναδικό αριθμό.

Το τρίτο όρισμα είναι η ειδική συνάρτηση που θέλουμε να καλέσουμε. Όπως είπαμε πιο πάνω, τα TLK-31 έχουν δύο συναρτήσεις για ανάγνωση ή εγγραφή, η βιβλιοθήκη SimpleModbusMaster όμως υποστηρίζει 8 διαφορετικές συναρτήσεις. Συνεπώς θα πρέπει να επιλέξουμε την κατάλληλη. Επειδή εμείς θέλουμε να αναγνώσουμε τιμές από τους Slaves, θα επιλέξουμε την ***READ_HOLDING_REGISTERS ή αλλιώς function 3 για την συγκεκριμένη βιβλιοθήκη***, με την οποία διαβάζουμε τα δυαδικά περιεχόμενα του καταχωρητή που θέλουμε.

Το τέταρτο όρισμα είναι η διεύθυνση σε δεκαεξαδική μορφή (HEX), στην οποία βρίσκεται ο καταχωρητής που έχει την τιμή που μας ενδιαφέρει (τιμή του αισθητήρα θερμοκρασίας), μέσα στη μνήμη του TLK-31. Η διεύθυνση αυτή είναι η ***0x0200***.

Το πέμπτο όρισμα είναι το ***πόσες θέσεις μνήμης θέλουμε να διαβάσουμε*** από τον καταχωρητή και μετά. Εμείς επιλέγουμε ***2 θέσεις μνήμης***, δηλαδή την θέση 0x0200 και την επόμενή της, ώστε να συμπεριλάβουμε την σπάνια περίπτωση που το δεδομένο είναι καταχωρημένο σε 2 καταχωρητές (πχ επειδή δε χωράει στον πρώτο).

Το έκτο και τελευταίο όρισμα της modbus_construct είναι ***σε ποιά θέση του πίνακα θέλουμε να αποθηκεύσουμε το δεδομένο***. Πίνακα εδώ, εννοούμε τον ***MB_regs[]*** που είχαμε φτιάξει πιο πάνω με σκοπό την αποθήκευση των τιμών που θα διαβάζονται.

Επόμενο βήμα είναι η εντολή modbus_configure:

```
modbus_configure(&Serial1, baud, SERIAL_8N1, timeout, polling, retry_count, TxEnablePin,
packets, TOTAL_NO_OF_PACKETS, MB_regs);
```

Το πρώτο όρισμα σε αυτή την συνάρτηση είναι η ***διεύθυνση του σειριακού αντικειμένου*** με το οποίο θέλουμε να επικοινωνήσουμε. Με τη χρήση της ***Serial1*** (το & δηλώνει διεύθυνση), σημαίνει οτι χρησιμοποιούμε την δεύτερη από τις 4 σειριακές θύρες του Arduino Mega (Serial0, Serial1, Serial2, Serial3) και άρα ότι έχουμε συνδέσει το RS-485 Module (την πλακέτα την οποία αναλύσαμε στο αντίστοιχο κεφάλαιο) στα pins TX01 και RX01 και όχι στα TX0, RX0 ή στα υπόλοιπα. Αυτό το κάνουμε διότι θα χρησιμοποιήσουμε την πρώτη θύρα (Serial0: pins TX0 και RX0) για επικοινωνία του Arduino με τον τερματικό Η/Υ μέσω των module ασύρματης επικοινωνίας ραδιοκυμμάτων (APC 220). Προφανώς και δεν γίνεται να χρησιμοποιήσουμε την ίδια θύρα και για το APC 220 module και για το RS-485 module, αλλά ***κάθε module πρέπει να μπεί σε ξεχωριστή θύρα***.

Το δεύτερο όρισμα είναι το ***baud rate (ρυθμός μετάδοσης).*** Με την εντολή ***#define baud 9600*** στην αρχή του προγράμματος, επιλέγουμε την τιμή 9600. Εδώ πρέπει να σημειωθεί πως θα πρέπει να ρυθμιστεί και το baud rate των TLK-31 στην ίδια τιμή (9600) μέσω των κουμπιών που έχουν χειροκίνητα και με τη βοήθεια του manual τους, όπως είχε γίνει και για τον αριθμό ID τους. Διαφορετική τιμή baud rate στους Slaves και στον κώδικα, σημαίνει πως δε θα λειτουργήσει η επικοινωνία.

Το τρίτο όρισμα είναι το λεγόμενο ***Byte Format ασύγχρονης επικοινωνίας***. Αυτό περιλαμβάνει τις εξής επιλογές:

SERIAL_8N2:	1 start bit, 8 data bits και 2 stop bits
SERIAL_8E1:	1 start bit, 8 data bits, 1 Even parity bit και 1 stop bit
SERIAL_8O1:	1 start bit, 8 data bits, 1 Odd parity bit και 1 stop bit
SERIAL_8N1:	1 start bit, 8 data bits και 1 stop bit

Στην εργασία χρησιμοποιήθηκε το τελευταίο format (***SERIAL_8N1***) το οποίο δεν έχει bit ελέγχου ισοτιμίας, ωστόσο είναι γνωστό ότι συνεργάζεται χωρίς προβλήματα στην σύνδεση ρυθμού μετάδοσης 9600 καθώς επίσης είναι και το πιο διαδεδομένο format επικοινωνίας με Η/Υ μέχρι σήμερα.

Το τέταρτο όρισμα είναι η τιμή ***timeoute***, ***η οποία εκφράζει τον χρόνο που δίνουμε σε έναν Slave να ανταποκριθεί***. Κοινές τιμές είναι από 1 δευτερόλεπτο μέχρι 5 δευτερόλεπτα. Εμείς επιλέγουμε την τιμή του ενός δευτερολέπτου ***(#define timeout 1000***).

Το πέμπτο όρισμα είναι η τιμή ***polling delay***. ***Πρόκειται για την περίοδο κενού μεταξύ αιτημάτων του Master ώστε να μπει ο Slave σε κατάσταση IDLE***. Είναι ιδιαίτερα σημαντική τιμή και δεν πρέπει να ρυθμίζεται τυχαία, αφού είναι σημαντικό ο Slave να μπορέσει να μπει σε κατάσταση IDLE, αλλιώς δε θα είναι σε θέση να απαντήσει στις ερωτήσεις (αιτήσεις) του Master. Κάποιοι Slaves είναι πολύ γρήγοροι και γυρνάνε στην κατάσταση IDLE μέσα σε ελάχιστα milisecond (πχ 10ms), ενώ άλλοι πιο αργοί μπορεί να κάνουν μέχρι και 200 ή και 300 milisecond. Για αυτό το λόγο ρυθμίσαμε την τιμή polling στο μισό δευτερόλεπτο ***(#define polling 500***) ώστε να δώσουμε τον απαραίτητο χρόνο στους Slaves (TLK-31) να μπουν στην κατάσταση IDLE καλύπτοντας ένα σχετικά μεγάλο εύρος, χωρίς όμως να καθυστερούμε πολύ την όλη διαδικασία. Η τιμή αυτή μπορεί να μειωθεί όσο οι Slaves αυξάνονται διότι σε αυτήν την περίπτωση λόγω φόρτωσης της γραμμής μετάδοσης, υπάρχει ήδη καθυστέρηση (delay).

Το έκτο όρισμα είναι η ***τιμή retry count***. Για να μην καθυστερεί η γραμμή σε περίπτωση που κάποιος Slave αργεί να απαντήσει (λόγω κάποιας βλάβης κλπ), το πρωτόκολλο MODBUS - RTU περιλαμβάνει μία διαδικασία κατά την οποία μετράει αποτυχημένες προσπάθειες, προσπάθειες δηλαδή που δεν κατάφεραν να μεταδώσουν μία αίτηση ή μία απάντηση. Όταν αυτές οι προσπάθειες φτάσουν την τιμή retry count, τότε το πακέτο που αναφέρεται στον συγκεκριμένο Slave απενεργοποιείται ώστε να μην καθυστερεί τους υπόλοιπους που λειτουργούν ομαλά. Τότε, μεταδίδει μονίμως την τιμή 0 και χρειάζεται η επέμβαση τεχνικού-προγραμματιστή ώστε χειροκίνητα να ξανα ενεργοποιήσει (αφού διορθώσει το τυχόν πρόβλημα) το πακέτο αυτό. Για την εργασία, ορίσαμε την τιμή αυτή στις 3 προσπάθειες ***(#define retry_count 3).***

Το έβδομο όρισμα είναι το ***TxEnablePin και αναφέρεται στο pin το οποίο ελέγχει το ολοκληρωμένο κύκλωμα του RS-485 Module*** ώστε να είναι σε κατάσταση που δέχεται ή που αποστέλλει δεδομένα. Στην προκειμένη περίπτωση το συγκεκριμένο pin του RS-485 Module το έχουμε συνδέσει στο pin 3 του Arduino (#define TxEnablePin 3).

Το όγδοο όρισμα είναι η ***διεύθυνση του πίνακα πακέτων (packets)*** και το ένατο το ***μέγεθος*** αυτού του πίνακα.

Δέκατο και τελευταίο όρισμα της modbus_configure είναι η ***διεύθυνση στην οποία θα αποθηκεύει ο Master τα δεδομένα που διαβάζει (ο πίνακας MB_regs)***.

Στη συνέχεια, η modbus_update καλείται ανά συγκεκριμένο χρονικό διάστημα μέσα στη συνάρτηση loop() που επαναλαμβάνεται στην εκτέλεση του κώδικα του Arduino. Η συνάρτηση modbus_update χρειάζεται να κληθεί τουλάχιστον μία φορά υποχρεωτικά.

Όσον αφορά τη διαδικασία που ακολουθείται για την modbus_construct, στην επόμενη σελίδα υπάρχει μια εικόνα που εξηγεί πιο αναλυτικά τη λειτουργία του κάθε ορίσματός της ώστε να κατανοηθεί καλύτερα η λειτουργία της συνάρτησης. Μπορούμε να δούμε 4 μπλόκ και την εντολή modbus_construct. Κάθε όρισμα της συνάρτησης συνδέεται με κόκκινα βέλη που δείχνουν στον αντίστοιχο πίνακα που αναφέρονται (πίνακας πακέτων, πίνακας καταχωρητών του Master, πίνακας καταχωρητών του Slave) και με πράσινα βέλη που δείχνουν σε ποιό στοιχείο του πακέτου που αποστέλλεται, αναφέρονται:

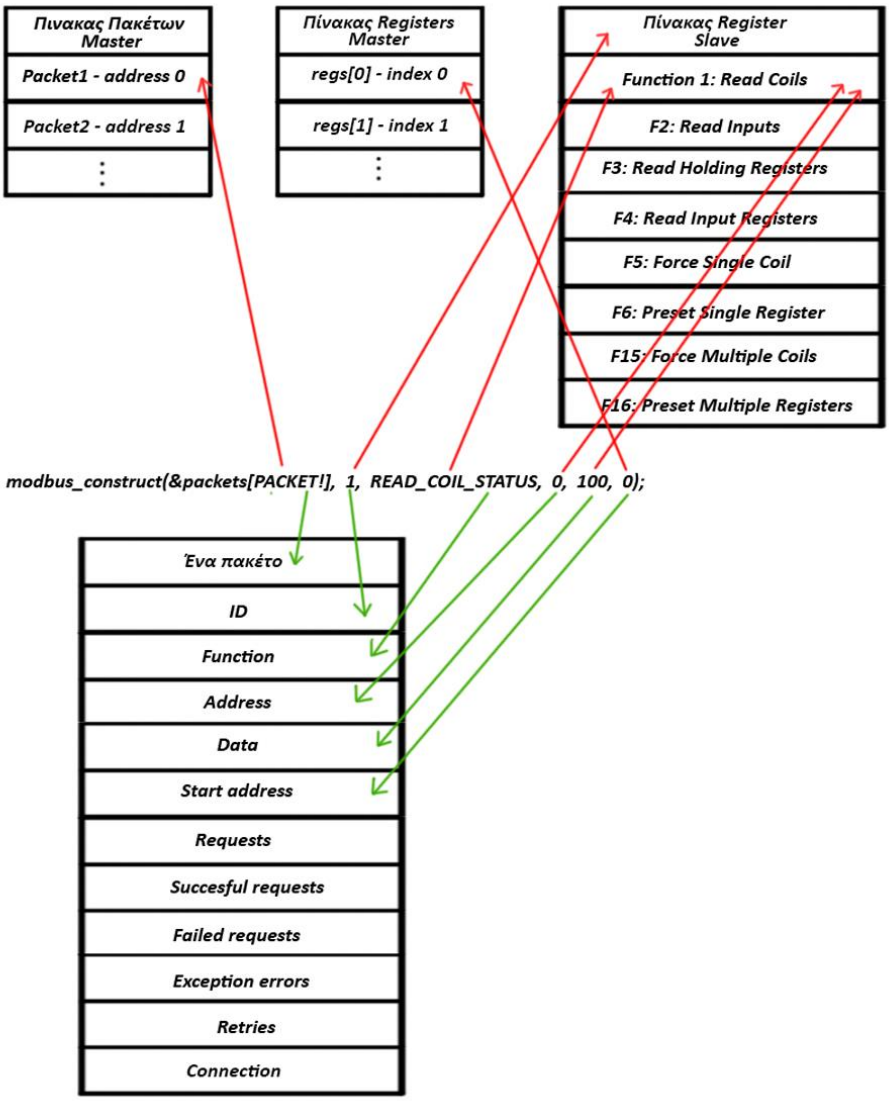


Figure 28 - Εξήγηση της modbus construct

γ) Κώδικας για τη ροή

Όπως είδαμε στο κεφάλαιο του αισθητήρα ροής υπάρχουν τρία καλώδια, μαύρο-κόκκινο για γείωση (gnd)-τάση αντίστοιχα και κίτρινο για τον παλμό εξόδου του αισθητήρα. Ο παλμός της εξόδου δίνεται κάθε φορά που γυρνάει ο εσωτερικός τροχός του αισθητήρα και σύμφωνα με δοκιμές για τις ανάγκες της βαθμονόμησης (callibrating) που έγιναν, ένας παλμός αντιστοιχεί σχεδόν σε 6.94ml νερού. Όσον αφορά τη σύνδεση του αισθητήρα στο Arduino, συνδέουμε το κόκκινο καλώδιο στα +5volt (τάση που εξάγει η πλακέτα), το μαύρο καλώδιο στο Gnd και το κίτρινο σε ένα digital Input pin που να υποστηρίζει Interrupt (πχ το pin 2). Στην κάτω φωτογραφία βλέπουμε το καλώδιο του αισθητήρα κατά τη διάρκεια των δοκιμών.

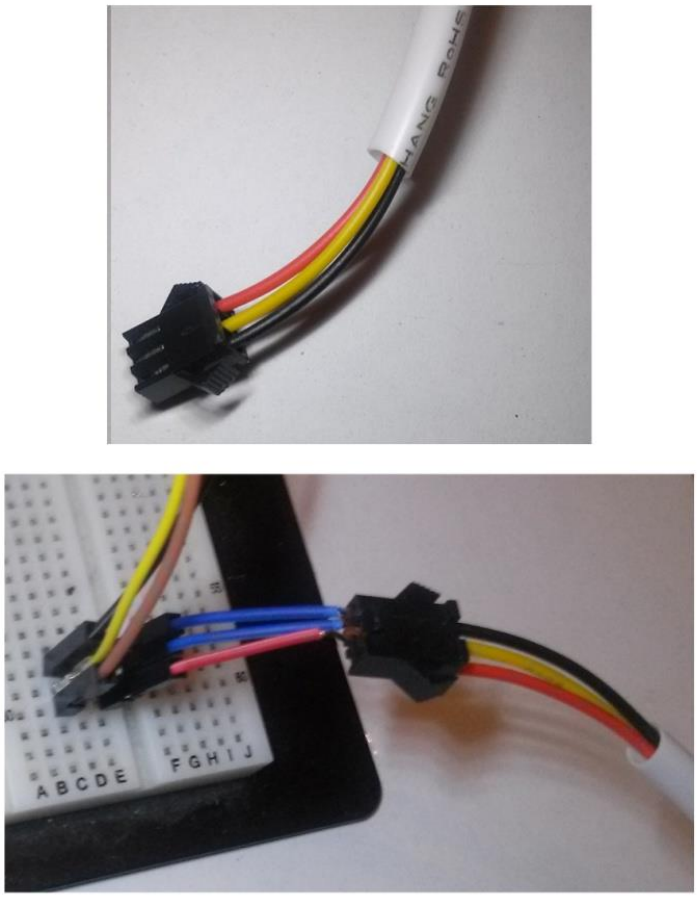


Figure 29 - Καλώδιο αισθητήρα ροής κατά τη διάρκεια δοκιμών

Η τεχνική που θα χρησιμοποιήσουμε ώστε να γίνεται ορθή καταγραφή της ροής είναι αυτή των Interrupts (διακοπών). Όπως είπαμε, συνδέουμε το καλώδιο που δίνει παλμούς σε ένα pin εισόδου του Arduino. Αν εποπτεύαμε ανά τακτά χρονικά διαστήματα το pin αυτό (πχ κάθε μισό δευτερόλεπτο ή και πιο συχνά ή αραιά) ώστε να δούμε εάν υπάρχει ροή, τότε θα χάναμε τυχόν δεδομένα που θα μας έστελνε στον νεκρό χρόνο (χρόνο μεταξύ δύο check του ακροδέκτη) και επιπροσθέτως θα φορτώναμε - επιβαρύναμε το πρόγραμμα και την εκτελέσή του σε μεγάλο βαθμό, αφού θα το υποχρεώναμε σε μια αδιάκοπη ενέργεια εποπτείας του ακροδέκτη. Για να λυθούν αυτά τα προβλήματα και κυρίως το πρώτο - το να χάνονται δηλαδή δεδομένα - χρησιμοποιούμε Interrupts.

Οι διακοπές (Interrupts) είναι καταστάσεις οι οποίες όταν ενεργοποιηθούν προκαλούν την προσωρινή παύση της ροής λειτουργίας του μικροεπεξεργαστή από αυτό που έκανε και την εκτέλεση κάποιων άλλων συγκεκριμένων λειτουργιών. Όταν τελειώσουν αυτές, ο μικροεπεξεργαστής συνεχίζει την λειτουργία του από εκεί που είχε μείνει κανονικά, μέχρι να έρθει (ίσως) κάποια άλλη διακοπή. Οι διακοπές ενεργοποιούνται απο εσωτερικά ή εξωτερικά συμβάντα. Τα εσωτερικά συμβάντα ή software interrupts, σχετίζονται με κάτι που συμβαίνει μέσα στο πρόγραμμα/μικροελεγκτή, ενώ τα εξωτερικά ή

hardware αφορούν αιτίες που έχουν να κάνουν με το εξωτερικό περιβάλλον όπως σήματα αισθητήρων, κουμπιών κλπ.

Στο Arduino Mega οι διακοπές υποστηρίζονται απο συγκεκριμένα Digital pins (ψηφιακούς ακροδέκτες εισόδου) που είναι τα: 2, 3, 18, 19, 20 και 21 και άρα μπορεί να υποστηρίξει μέχρι 6 διαφορετικούς σκανδαλισμούς διακοπών (με κατάλληλες τεχνικές μπορούν να γίνουν και περισσότερα. Για τις διακοπές, υπεύθυνες είναι οι συναρτήσεις ISR (Interrupt Service Routines). Μία ISR πρέπει να είναι πολύ σύντομη όσον αφορά το χρόνο και δεν μπορεί να εκτελείται παράλληλα με άλλη. Με λίγα για να εκτελεστεί μία διακοπή θα πρέπει να μην εκτελείται άλλη εκείνη την στιγμή.

Η εντολή στο Arduino (γλώσσα Wiring C) για τις διακοπές είναι η: ***attachInterrupt(int, func, mode)***

Το πρώτο όρισμα είναι ο αριθμός της διακοπής ο οποίος αναφέρεται σε συγκεκριμένο pin. Για το Arduino Mega, εαν χρησιμοποιούμε το pin 2 καλείται η διακοπή νούμερο 0.

Το δεύτερο όρισμα είναι η συνάρτηση που θέλουμε να εκτελείται όταν κληθεί η διακοπή και σταματήσει η κανονική ροή του προγράμματος.

Το τρίτο όρισμα (mode) είναι η κατάσταση του παλμού σκανδαλισμού, πότε δηλαδή θέλουμε να ενεργοποιείται η διακοπή. Μπορεί να πάρει τις εξής επιλογές:

LOW: ενεργοποίηση διακοπής όταν ο παλμός στον ακροδέκτη είναι σε κατάσταση LOW (μηδέν)

CHANGE: ενεργοποίηση διακοπής όταν ο πλαμός στον ακροδέκτη αλλάζει κατάσταση

RISING: ενεργοποίηση διακοπής όταν ο παλμός στον ακροδέκτη πηγαίνει από την κατάσταση LOW στην κατάσταση HIGH (πχ από 0 σε 5 volt κλπ)

HIGH: ενεργοποίηση διακοπής όταν ο παλμός στον ακροδέκτη είναι σε κατάσταση HIGH

Τέλος, είναι απαραίτητη η ενεργοποίηση των διακοπών με την κλήση της sei().

Σύμφωνα με το σχέδιο που θα εφαρμόσουμε για τον υπολογισμό της ροής, όταν θα γίνεται ο ακροδέκτης στον οποίο έχουμε συνδεμένο τον αισθητήρα από LOW - HIGH, τότε θα καλούμε μία συνάρτηση (flow()) η οποία μέσα της θα αυξάνει έναν μετρητή +1. Ο συγκεκριμένος μετρητής θα μηδενίζεται ανά συγκεκριμένο χρονικό διάστημα ώστε να μην συσσωρεύει τους παλμούς από την αρχή, αλλά να έχει κάθε φορά τους παλμούς της τελευταίας διακοπής. Ο κώδικας της ροής θα είναι όπως παρακάτω (για έναν αισθητήρα ροής):

```
volatile int flow_frequency;
unsigned int ml;
unsigned char flowsensor =2;
unsigned long ml_sum;

void flow()
```

```
{
        flow_frequency++;
}

pinMode(flowsensor, INPUT);
digitalWrite(flowsensor, HIGH);

attachInterrupt(0, flow, RISING);
sei();
ml_sum=0;
```

'Ενα άλλο σημαντικό θέμα, είναι οτι δεν θα ήταν ορθό να χρησιμοποιήσουμε την συνάρτηση delay() για να στέλνουμε δεδομένα από το Arduino κάθε συγκεκριμένο χρονικό διάστημα (πχ κάθε 1 δευτερόλεπτο). Το πρόγραμμα στο Arduino έχει την συνάρτηση loop() μέσα στην οποία εκτελείται συνεχώς και αδιάκοπα (όσο έχει τροφοδοσία η πλακέτα) οτιδήποτε περιλαμβάνεται μέσα σε αυτή τη συνάρτηση. Αν εμείς υπολογίζουμε τη ροή μέσα στη loop() και θέλουμε να την στέλνουμε (μαζί με τις τιμές θερμοκρασίας), κάθε 1 δευτερόλεπτο, τότε με χρήση της συνάρτησης delay() το Arduino θα σταματούσε οτιδήποτε έκανε γα 1 δευτερόλεπτο και θα έμενε σε κατάσταση freeze. Έπειτα, μετά από 1 δευτερόλεπτο, θα ακολουθούσε εκ νέου την διαδικασία. Ωστόσο αυτός ο νεκρός χρόνος κατά τον οποίο παγώνει το Arduino είναι προφανώς μη θεμιτός. Το ιδανικό είναι να στέλνουμε κάθε 1 δευτερόλεπτο, χωρίς όμως να σταματάει η λειτουργία του Arduino.

Για τη λειτουργία αυτή θα χρησιμοποιήσουμε τη συνάρτηση millis(). Αυτή η συνάρτηση επιστρέφει τον αριθμό των milliseconds που πέρασαν από τότε που ξεκίνησε η εκτέλεση του προγράμματος και υπερχειλίζει (δηλαδή γίνεται reset και ξαναμηδενίζει) μετά από 50 περίπου μέρες. Συνεπώς εάν ελέγχουμε συνεχώς με μια συνθήκη if αν έχει περάσει κάθε φορά 1 δευτερόλεπτο από τη στιγμή που θέλουμε, μπορούμε να στέλνουμε τα δεδομένα τότε, χωρίς να σταματάει το πρόγραμμα. Αυτό γίνεται ως εξής:

```
uint32_t lastScan=0;
```
//πρώτα δηλώνουμε μία 32bit μεταβλητή που θα ξεκινάει από 0 και θα αποθηκεύει κάθε φορά την στιγμή που συμπληρώνεται 1 δευτερόλεπτο

```
const uint32_t periodScan=1000;
```
//δηλώνουμε μία 32bit σταθερά με τιμή 1 δευτερόλεπτο

```
if (millis() - lastScan > periodScan)
{
        lastScan=millis();
        . . .
}
```

//μέσα στην loop() ελέγχουμε συνεχώς αν η διαφορά από τότε που ξεκίνησε το πρόγραμμα μέχρι την τιμή lastScan είναι μεγαλύτερη από 1 δευτερόλεπτο. Όταν είναι, τότε η lastScan ενημερώνεται παίρνοντας την τιμή της millis() εκείνη τη στιγμή και εκτελείται ο υπόλοιπος κώδικας. Όσο όμως δεν

είναι η διαφορά αυτή μεγαλύτερη από το 1 δευτερόλεπτο, το Arduino συνεχίζει και λειτουργεί κανονικά συλλέγοντας τις τιμές θερμοκρασίας μέσως του MODBUS - RTU και τις τιμές ροής μέσω των διακοπών και δεν μένει σε αδράνεια όπως θα γινόταν με την συνάρτηση delay().

δ) Συνολικός κώδικας και σχόλια

Παρακάτω θα δούμε όλο τον κώδικα που εκτελείται στο Arduino (MODBUS RTU για 4 συσκευές Slave και ροή για 1 αισθητήρα ροής) και σχολιασμό του:

```
#include <SimpleModbusMaster.h> //εισαγωγή βιβλιοθήκης modbus

#define baud 9600 //τιμή για baud rate (modbus)
#define timeout 1000 //τιμή για timeout (modbus)
#define polling 500 //τιμή για polling (modbus)
#define retry_count 3 //τιμή για retry_count (modbus)
#define TxEnablePin 3 //τιμή για pin που συνδέθηκε το RS-485 module (modbus)

volatile int flow_frequency; // μετράει τους παλμούς της ροής
unsigned int ml; // αποθηκεύει τα ml ροής
unsigned char flowsensor = 2; // pin που συνδέθηκε ο αισθητήρας ροής

void flow () // η συνάρτηση που καλείται κατά την διακοπή (Interrupt function
{
  flow_frequency++; //αυξάνεται κατά 1 ο μετρητής παλμών
}

uint32_t lastScan = 0;      //αποθηκεύει την τελευταία φορά που έγινε scan
const uint32_t periodScan = 1000;      //κάθε πότε θα γίνεται στέλνονται τα δεδομένα

enum {          // δημιουργία απαριθμητή για τα πακέτα modbus
  PACKET1,
  PACKET2,
  PACKET3,
 PACKET4,
  TOTAL_NO_OF_PACKETS
};

Packet packets[TOTAL_NO_OF_PACKETS]; //πίνακας πακέτων (όσα και τα πακέτα)

#define TOTAL_NO_OF_REGISTERS 8 //8 καταχωρητές για αποθήκευση των δεδομένων που λαμβάνονται απο τους Slaves (modbus)

unsigned int MB_regs[TOTAL_NO_OF_REGISTERS]; //πίνακας για την αποθήκευση των δεδομένων που λαμβάνονται από τους Slaves (modbus)
```

```
void setup()     //η πρώτη βασική συνάρτηση του Arduino
{
pinMode(flowsensor, INPUT);  //Οριζουμε το pin που τοποθετήθηκε ο αισθητήρας ροής, ως εισόδου
digitalWrite(flowsensor, HIGH); // Δημιουργία εσωτερικής Pull-up αντίστασης για το Pin του
αισθητήρα ροής
Serial.begin(9600); //Εναρξη σειριακής επικοινωνίας στα 9600 bps (rate)
attachInterrupt(0, flow, RISING); // Setup της Interrupt (διακοπής για ροή)

sei();  //ενεργοποίηση διακοπών

//Οι επόμενες τρεις γραμμές δομούν τα 4 πακέτα που θα στέλνονται στους slaves
modbus_construct(&packets[PACKET1], 1, READ_HOLDING_REGISTERS, 0x0200, 2, 0);
modbus_construct(&packets[PACKET2], 2, READ_HOLDING_REGISTERS, 0x0200, 2, 2);
modbus_construct(&packets[PACKET3], 3, READ_HOLDING_REGISTERS, 0x0200, 2, 4);
modbus_construct(&packets[PACKET4], 4, READ_HOLDING_REGISTERS, 0x0200, 2, 6);

//Η κάτω γραμμή διαμορφώνει την επικοινωνία για το πρωτόκολλο MODBUS - RTU
modbus_configure(&Serial1, baud, SERIAL_8N1, timeout, polling, retry_count, TxEnablePin,
packets, TOTAL_NO_OF_PACKETS, MB_regs);
}

void loop()  // δεύτερη βασική συνάρτηση του Arduino που εκτελείται συνεχώς
{
  if (millis() - lastScan > periodScan) {   // εκτέλεση κώδικα ανά 1 δευτερόλεπτο
    lastScan = millis();

      ml = flow_frequency*6.94;  //μετατροπή παλμών ροής σε ml
      flow_frequency = 0; // Reset του μετρητή παλμών ροής

      Serial.print(MB_regs[0]); //αποστολή σε μία γραμμή, των δεδομένων με 'n' μεταξύ τους
      Serial.print('n');
      Serial.print(MB_regs[2]);
      Serial.print('n');
      Serial.print(MB_regs[4]);
      Serial.print('n');
      Serial.print(MB_regs[6]);
      Serial.print('n');
      Serial.print(ml);  //τα ml ροής
      Serial.print('n');
      Serial.print(random(14,26)); //τυχαίες τιμές
      Serial.print('n');
      Serial.print(random(2,13));
      Serial.print('n');
      Serial.print(random(4,15));
```

```
    Serial.print('n');
    Serial.println(random(4,15)); //αποστολή extra δεδομένου (9ου) για αποφυγή σφαλμάτων στην εφαρμογή εποπτείας του Η/Υ και αλλαγή σειράς
    Serial.flush(); //εντολή για να περιμένει το πρόγραμμα να γίνει αποστολή της γραμμής πριν συνεχιστεί ο κώδικας, αυτό γίνεται για αποφυγή σφαλμάτων - για ορθή αποστολή
  }
 modbus_update();  //κλήση της modbus_update
}
```

4.2 Ανάλυση λειτουργίας εφαρμογής εποπτείας

Η εφαρμογή εποπτείας είναι ο βασικός άξονας γύρω από τον οποίο αναπτύχθηκε η φιλοσοφία της παρούσας πτυχιακής εργασίας. ***Σχεδιάστηκε και υλοποιήθηκε από τον φοιτητή Μενεξέ Ιωάννη, με την βοήθεια του Κατικαρίδη Δημήτριου και επεξεργάστηκε για τις ανάγκες της πτυχιακής εργασίας από τον γράφοντα***. Όπως είδαμε και στην περιγραφή υλικού, συνδέσεων, αισθητήρων κλπ, ο απώτερος σκοπός των πρώτων επιπέδων σχεδιασμού ήταν να φτάσουν τα δεδομένα στο τερματικό Η/Υ όπου και εκτελείται η ειδική εφαρμογή που έχει σχεδιαστεί για την εποπτεία των αισθητήρων. Η εφαρμογή αυτή:

1. Δέχεται τα δεδομένα
2. Τα επεξεργάζεται
3. Τα παρουσιάζει σε γραφήματα πραγματικού χρόνου
4. Τα αποθηκεύει
5. Τα συγκρίνει σε στατικά γραφήματα
6. Τα μετατρέπει σε άλλες μορφές αποθήκευσης
7. Εξάγει τα απαραίτητα αρχεία στον φάκελο του server ώστε να λειτουργεί και η ιστοσελίδα

Αυτή η ειδική εφαρμογή υλοποιήθηκε στη γλώσσα προγραμματισμού C# . Στο παρών κεφάλαιο θα γίνει μια παρουσίαση της λειτουργίας του ειδικού αυτού software σε επίπεδο UI (user interface), δηλαδή στο τι περιλαμβάνει και πως χειρίζεται από έναν απλό χρήστη. Η φιλοσοφία δομής της εφαρμογής που έχει να κάνει με τον κώδικα και τις τεχνικές υλοποίησης περιγράφεται στο ανάλογο κεφάλαιο.

Αρχικά θα παρουσιαστεί το περιβάλλον λειτουργίας της εφαρμογής από την πλευρά της πρώτης επαφής που έχει κάποιος που θα το χρησιμοποιήσει. Αφού περιγραφούν τα απαραίτητα κομμάτια, θα υπάρξει περιγραφή της λειτουργίας του προγράμματος η οποία θα επικεντρωθεί γύρω από τις 3 βασικές λειτουργίες του:

1. την εποπτεία (monitoring)
2. την σύγκριση τιμών αισθητήρων (compare) και
3. την μετατροπή των δεδομένων σε άλλες μορφές.

α) Γνωριμία με το περιβάλλον της εφαρμογής

Στην παρακάτω εικόνα βλέπουμε την πρώτη άποψη της εφαρμογής κατά την εκτέλεσή της για πρώτη φορά απο έναν χρήστη.

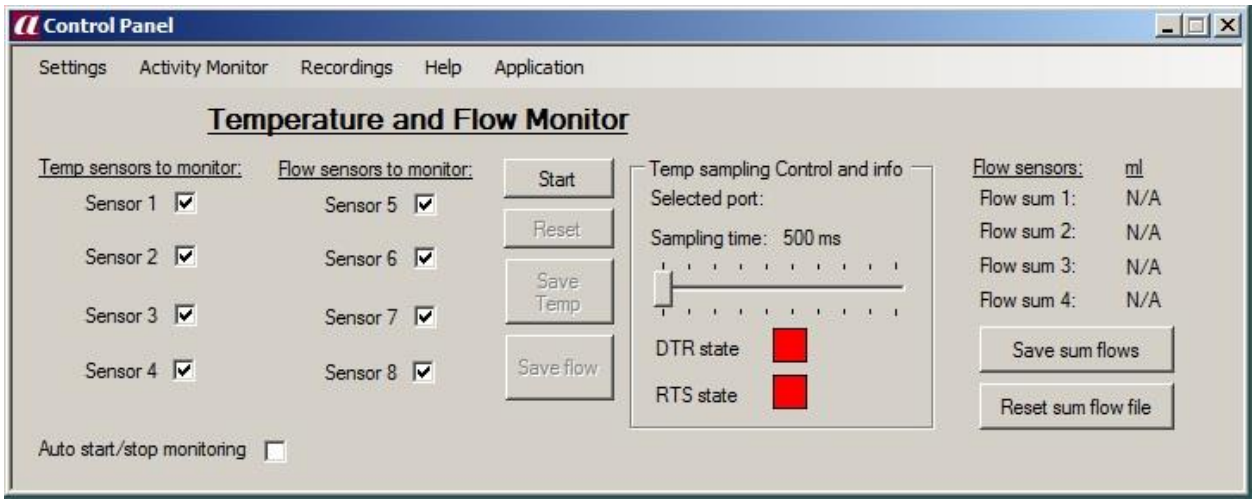


Figure 30 - Κεντρικό πάνελ

Το κεντρικό αυτό παράθυρο του προγράμματος που ανοίγει αρχικά, μπορεί να χωριστεί σε 5 τμήματα.

1. Το τμήμα του βασικού menu λειτουργιών που βρίσκεται σε οριζόντια στοίχιση στο πάνω μέρος
2. Το τμήμα των chech boxes των αισθητήρων και της αυτόματης λειτουργίας (αριστερά)
3. Το τμήμα των βασικών κουμπιών λειτουργίας (στο κέντρο)
4. Το τμήμα του χρόνου δειγματοληψίας (δεξιά από τα κουμπιά), με τη μορφή κυλιόμενης μπάρας
5. Ένα συμπληρωματικό τμήμα που αφορά τη ροή (δεξιά)

Στο πρώτο κομμάτι, οριζόντια στο πάνω τμήμα του παραθύρου, υπάρχει το βασικού menu το οποίο περιλαμβάνει τις επιλογές ***Settings***, όπου είναι οι επιλογές σειριακής επικοινωνίας, ***Activity Monitor*** όπου είναι οι επιλογές για εποπτεία πραγματικού χρόνου, ***Recordings*** όπου γίνονται οι στατικές συγκρίσεις και οι μετατροπές και τέλος ***Help*** για πληροφορίες και ***Application*** για έξοδο ή επανεκκίνηση.

Τα checkboxes στα αριστερά αποτελούν τις επιλογές των αισθητήρων που θέλει ο χρήστης να εποπτεύσει. Χωρίζονται σε δύο κατηγορίες, στους 4 αισθητήρες θερμοκρασίας που αναφέρονται στους αισθητήρες PT100 του ελαιουργείου και στους 4 αισθητήρες ροής που υπάρχουν/μπορούν να υπάρξουν εκεί. Αν είναι επιλεγμένο το check box, τότε ο αντίστοιχος αισθητήρας θα περιλαμβάνεται στη διαδικασία της εποπτείας. Ακριβώς από κάτω υπάρχει ένα επιπλέον checkbox για τον προγραμματισμό αυτόματης λειτουργίας.

Δεξιά βλέπουμε τα 4 βασικά κουμπιά της εφαρμογής. Με το ***Start*** η εφαρμογή ξεκινάει την διαδικασία υποδοχής δεδομένων από τη θύρα USB, το διάβασμά τους, την επεξεργασία τους και την παρουσίασή τους σε διαγράμματα. Την ίδια φιλοσοφία έχει και η επιλογή ***Stop*** για την περίπτωση που θέλουμε προσωρινή διακοπή (Pause) της όλης διαδικασίας. Από κάτω υπάρχει η επιλογή ***Reset*** με το οποίο γίνεται αφενός πλήρης - οριστική διακοπή της διαδικασίας όταν αυτή είναι σε κατάσταση προσωρινής διακοπής (stoped), αφετέρου επαναφέρει όλα τα απαραίτητα στοιχεία του προγράμματος στην αρχική κατάσταση ώστε να είναι έτοιμο για μια εκ νέου μέτρηση χωρίς να συνεχίσει από εκεί που έμεινε. Τα κουμπιά ***Save temp*** και ***Save flow*** είναι ιδιαίτερα σημαντικά. Με αυτά τα δύο κουμπιά μπορεί ο χρήστης να αποθηκεύσει σε αρχείο τύπου .xml είτε τις μετρήσεις των αισθητήρων θερμοκρασίας, είτε αντίστοιχα των αισθητήρων ροής που έχουν προηγηθεί. Συνεπώς κάθε κουμπί δίνει την δυνατότητα για ξεχωριστή αποθήκευση ενός ξεχωριστού .xml αρχείου με το ιστορικό των μετρήσεων.

Αμέσως δεξιότερα υπάρχει ένα πλαίσιο με μία κινούμενη μπάρα. Εδώ γίνεται η επιλογή του χρόνου δειγματοληψίας και περιλαμβάνει ένα εύρος από 500ms έως 5s. Οι κόκκινοι κύβοι DTR και RTS λειτουργούν ως "λυχνίες" (κόκκινο/πράσινο) σχετικά με το Reset στο Arduino (απο το οποίο η εφαρμογή παίρνει τις πληροφορίες - δεδομένα) και με το πλαίσιο RTS ασύρματης επικοινωνίας.

Τέλος υπάρχουν κάποια συμπληρωματικά στοιχεία για την ροή. Επειδή το να γίνεται εποπτεία στην ροή ανά κάποια δευτερόλεπτα δεν είναι στοιχείο που είναι πάντα χρήσιμο, έχει προστεθεί η λειτουργία που δείχνει την συνολική ροή νερού που έχει περάσει απο κάθε αισθητήρα. Με τα αντίστοιχα κουμπιά από κάτω μπορεί ο χρήστης είτε να αποθηκεύσει την συνολική ροή σε ένα αρχείο .txt, είτε να κάνει reset στα στοιχεία που σχετίζονται με την συνολική ροή.

Όπως παρατηρείται, το περιβάλλον της εφαρμογής έχει διακριτά και ομαδοποιημένα στοιχεία για ευκολότερη, ευέλικτη και απλή χρήση ακόμη και από μη εξειδικευμένους χρήστες. Στις επόμενες παραγράφους θα φανεί βήμα - βήμα το πως γίνεται η εποπτική λειτουργία καθώς και οι άλλες λειτουργίες που παρέχει το πρόγραμμα και μέσα από αυτή τη παρουσίαση θα φανεί κάθε πτυχή των διαδικασιών χρήσης που υποστηρίζει η εφαρμογή. Αρχικά θα δούμε την εποπτική λειτουργία πραγματικού χρόνου και το πως γίνεται εξ αρχής, έπειτα την λειτουργία σύγκρισης μετρήσεων - αισθητήρων σε στατικό πλαίσιο και τέλος τις μετατροπές που μπορούμε να κάνουμε στα αρχεία μετρήσεων που θέλουμε να αποθηκεύσουμε.

β) Η λειτουργία εποπτείας (monitoring)

Για να ξεκινήσει η διαδικασία βασικός παράγοντας είναι να είναι συνδεμένος στη θύρα USB του Η/Υ στον οποίο θα εκτελεστεί η εφαρμογή, το module του δέκτη (APC 220) ασύρματης μετάδοσης μέσω του ειδικού μετατροπέα του για USB. Με αυτό συνδεμένο κάνοντας κλικ στην επιλογή Settings του Menu, επιλέγουμε το Port όπου με ένα Refresh θα δούμε την αντίστοιχη θύρα (πχ. COM 4). Αφού την επιλέξουμε, κάνουμε κλικ στο κουμπί με το όνομά της για να επικυρωθεί η επιλογή της, όπως βλέπουμε στην παρακάτω φωτογραφία.

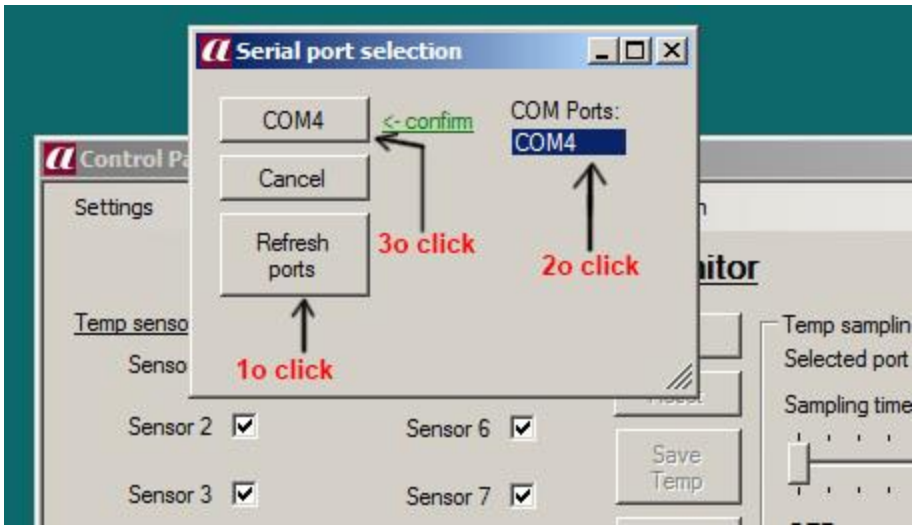


Figure 31 - Επιλογή σειριακής θύρας για επικοινωνία με το rf module

Έπειτα επιλέγουμε τον χρόνο δειγματοληψίας ο οποίος μπορεί να ρυθμιστεί μέσω της μπάρας η οποία μετακινείται με το ποντίκι από μισό δευτερόλεπτο έως 5 δευτερόλεπτα. Αν επιθυμούμε να εποπτεύσουμε ροή θα πρέπει να επιλέξουμε αναγκαστικά 1 δευτερόλεπτο για χρόνο δειγματοληψίας. Αυτό είναι ιδιαίτερα σημαντικό καθώς η ροή στέλνεται από το Arduino κάθε 1 δευτερόλεπτο, συνεπώς μεγαλύτερα κενά στη δειγματοληψία από τη μεριά της εφαρμογής θα προκαλέσουν απώλεια πληροφοριών. Μετά από την επιλογή του χρόνου δειγματοληψίας (sampling time), επιλέγουμε όπως βλέπουμε και στην εικόνα 42 το πόσους/ποιούς αισθητήρες θέλουμε να ενεργοποιήσουμε ώστε να κάνουμε εποπτεία (Monitoring). Αυτό γίνεται επιλέγοντας τα ανάλογα κουτιά (η προεπιλογή είναι όλα τα κουτιά επιλεγμένα). Τέλος, όλα είναι έτοιμα για την εκκίνηση της διαδικασίας με το κουμπί Start.

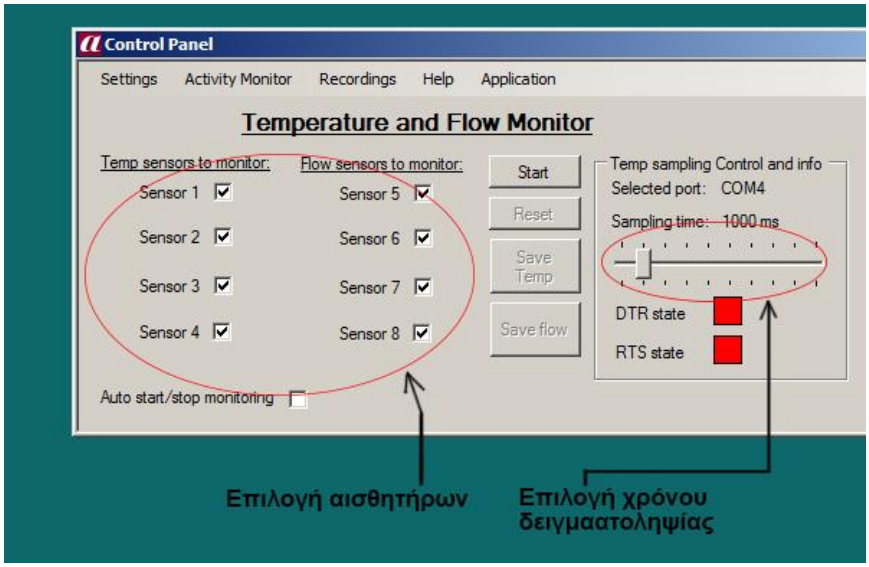


Figure 32 Επιλογή αισθητήρων και χρόνου δειγματοληψίας

Μετά το πάτημα του κουμπιού Start (θεωρούμε στο παράδειγμα πως επιλέξαμε δειγματοληψία 1 δευτερόλεπτο και όλους τους αισθητήρες) αρχίζει η διαδικασία. Ήδη η εφαρμογή έχει αρχίσει και δέχεται δεδομένα τα οποία διαβάζει απο την θύρα USB σειριακά, όπου υπάρχει συνδεμένος ο δέκτης APC 220. Αρχικά δίνονται κάποια στοιχεία για τη συνολική ροή στα δεξιά του κεντρικού πάνελ. Αυτά τα στοιχεία είναι απλώς η ροή σε σύνολο που έχει περάσει απο τον αντίστοιχο αισθητήρα.

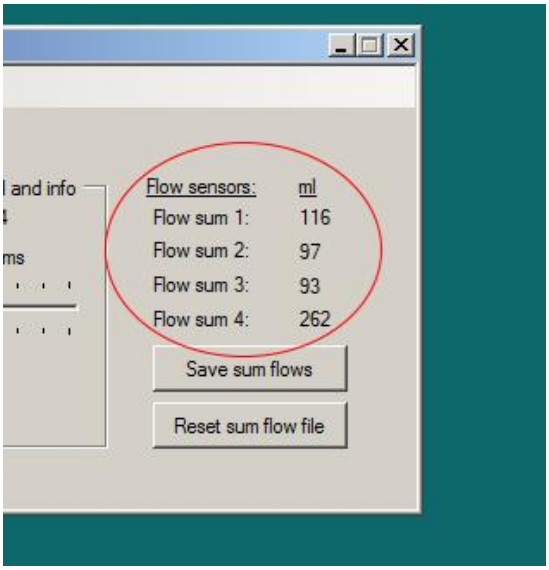


Figure 33 - Συνολική ροή νερού

Για να δούμε τις βασικές λειτουργίες της εφαρμογής πρέπει να επιλέξουμε στο πάνω Menu την καρτέλα ***Activity Monitor***. Αμέσως θα ανοίξουν από κάτω οι δύο βασικές επιλογές, ***Real time values*** και ***Real time graphs***. Στην πρώτη επιλογή, την Real time values θα δούμε μια εικόνα όπως η επόμενη.

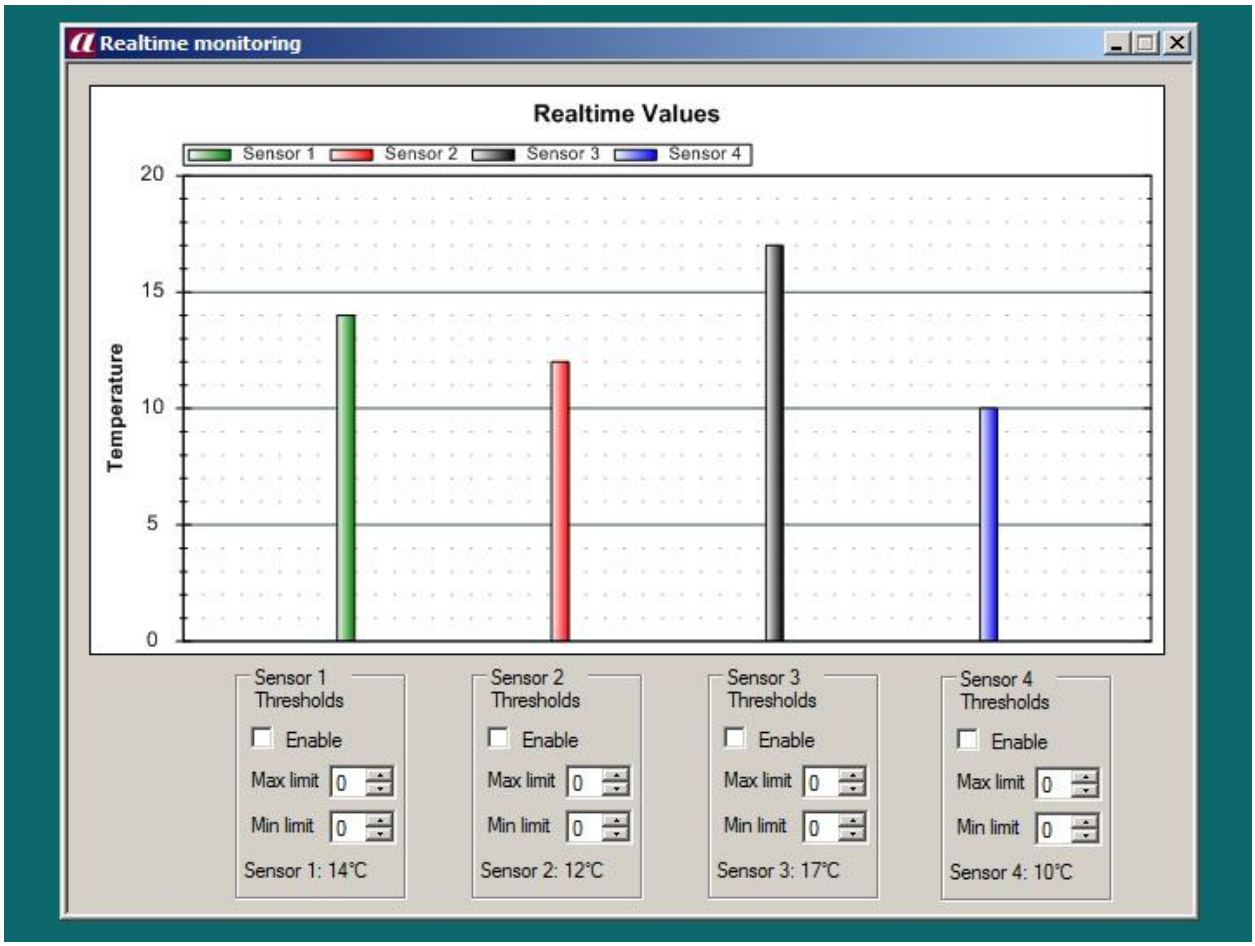


Figure 34 - Διάγραμμα Τιμών πραγματικού χρόνου (Real time values)

Σε αυτό το παράθυρο που ανοίγει φαίνονται οι τιμές από τους αισθητήρες θερμοκρασίας σε πραγματικό χρόνο. Κάθε ένα δευτερόλεπτο (όσος και ο χρόνος δειγματοληψίας που επιλέχθηκε) οι μπάρες ανανεώνονται και δείχνουν την θερμοκρασία για τον κάθε ένα από τους τέσσερις αισθητήρες θερμοκρασίας. Υπενθυμίζεται ότι αυτά τα δεδομένα έρχονται σε πραγματικό χρόνο ασύρματα από το Arduino μέσω ενός ζεύγους πομποδέκτη ραδιοκυμάτων.

Στη συγκεκριμένη λειτουργία υπάρχει η επιλογή να ρυθμιστεί ένα ανώτερο και ένα κατώτερο όριο από τα αντίστοιχα βελάκια κάτω από κάθε αισθητήρα και έπειτα από το check box Enable να ενεργοποιηθεί ο έλεγχος για το αν ξεπεράστηκαν αυτά τα δύο όρια. Σε περίπτωση που γίνει αυτό, κοκκινίζει το αντίστοιχο limit box και έτσι ειδοποιούμαστε για τυχόν παραβιάσεις θερμοκρασιακών ορίων που μπορεί να υπάρξουν μέσα στο ελαιουργείο κατά τη διάρκεια της διαδικασίας της εποπτείας. Επιπροσθέτως μπορούμε να κάνουμε πάνω στο γράφημα είτε zoom με το ποντίκι, είτε δεξί κλικ και να αποθηκεύσουμε όποιο στιγμιότυπο επιθυμούμε.

Στη συνέχεια κλείνουμε το παράθυρο ***Real time values*** και ενώ συνεχίζει να είναι σε κατάσταση εκτέλεσης η εφαρμογή (πατημένο το κουμπί Start), ανοίγουμε πάλι από το ***Activity Monitor***, αυτή τη φορά την καρτέλα ***Real time graphs (γραφήματα σε πραγματικό χρόνο)***.

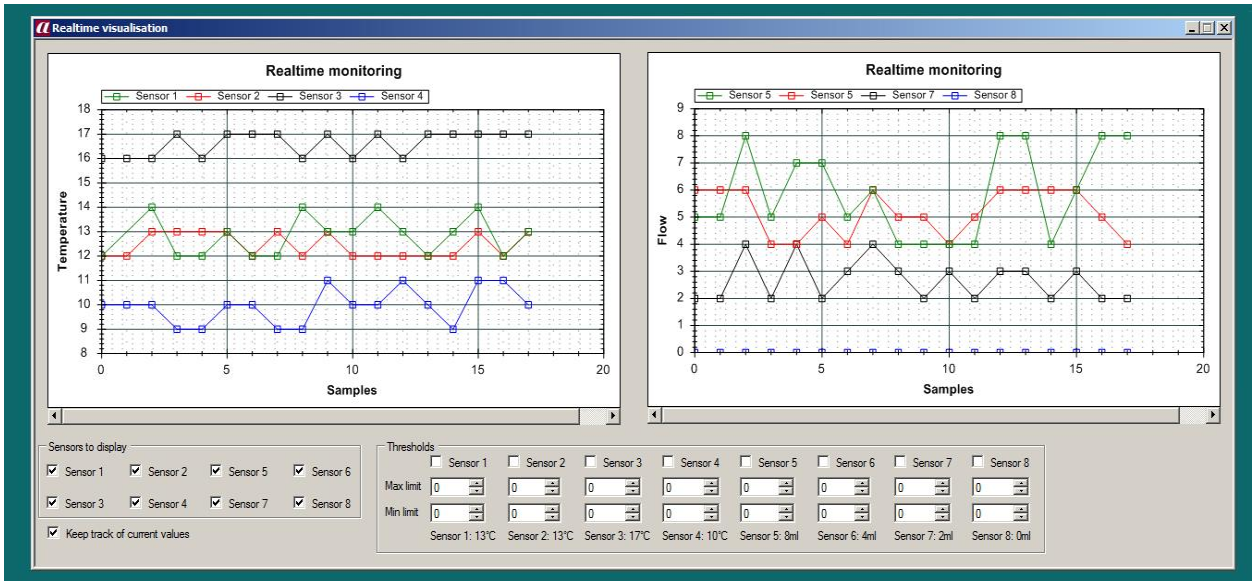


Figure 35 - Διαγράμμα πραγματικού χρόνου (Real time graphs)

Σε αυτή την λειτουργία εμφανίζονται δύο διαγράμματα. Το αριστερό διάγραμμα όπως γράφει και ο άξονας Υ του αναφέρεται στους αισθητήρες θερμοκρασίας και το δεξί διάγραμμα αναφέρεται στους αισθητήρες ροής (σύνολο 8 αισθητήρες, 4 και 4). Εδώ, κάθε δευτερόλεπτο (ή κάθε μονάδα sampling που έχουμε ορίσει εμείς στην αρχή από την ειδική μπάρα) εμφανίζεται μία κουκίδα στο επίπεδο για κάθε επιλεγμένο αισθητήρα, η οποία έχει μία τιμή Υ (τιμή βαθμών Κελσίου ή ml) και μία τιμή Χ (τιμή δείγματος) . Η επόμενη κουκίδα για τον κάθε αισθητήρα θα τοποθετηθεί ένα δείγμα δεξιότερα του προηγούμενου και επιπροσθέτως θα ενωθεί με μία ίσια γραμμή - ανάλογου χρώματος για τον κάθε αισθητήρα - με την προηγούμενη κουκίδα κ.ο.κ. Με αυτόν τον τρόπο δείγμα το δείγμα, σχηματίζεται μία γραμμή δομημένη από το σύνολο αυτών των γραμμών/κουκίδων. Και εδώ όπως και στη λειτουργία Real time values, μπορούμε να κάνουμε zoom με το ποντίκι σε κάθε διάγραμμα. Για να γίνει σωστά το zoom θα πρέπει πρώτα να γίνεται ένα κλικ πάνω στο επιθυμητό διάγραμμα (θερμοκρασίας ή ροής) ώστε να επιλεχθεί και έπειτα μπορεί να γίνει το zoom in και zoom out με τη βοήθεια της ροδέλας του ποντικιού.

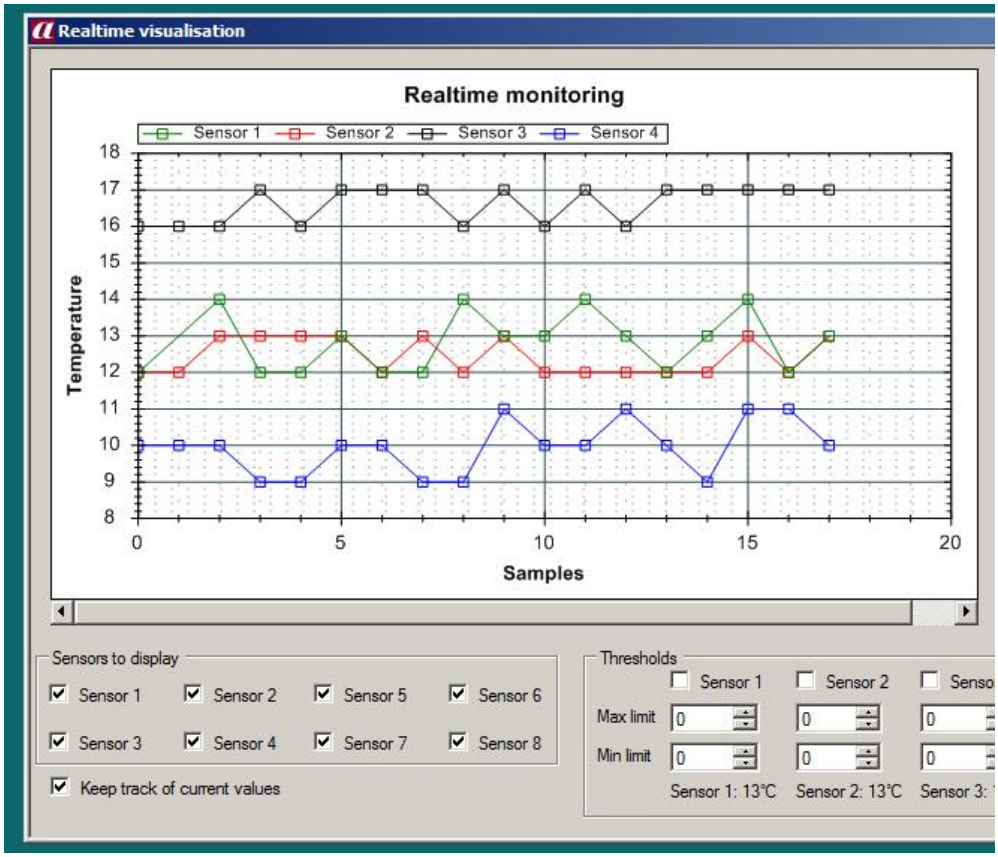


Figure 36 - Λεπτομέρεια των διαγραμμάτων πραγματικού χρόνου

Όπως παρατηρούμε στην επάνω εικόνα, μπορούμε να επιλέξουμε απενεργοποιώντας / ενεργοποιώντας τα κατάλληλα check boxes, το ποιούς αισθητήρες θέλουμε να βλέπουμε και ποιούς όχι. Στο πάνω μέρος του παραθύρου η εφαρμογή μας πληροφορεί για το ποιό χρώμα αντιστοιχεί σε ποιόν αισθητήρα, συνεπώς αν απενεργοποιήσουμε το αντίστοιχο check box, θα εξαφανιστεί και η αντίστοιχη γραμμή στο διάγραμμα. Αυτό ωστόσο δε σημαίνει ότι το πρόγραμμα σταματάει να δέχεται δεδομένα για τον συγκεκριμένο αισθητήρα. Αντιθέτως, δέχεται κανονικά τις πληροφορίες, ωστόσο ο αισθητήρας (η γραμμή του στο διάγραμμα) είναι μη ορατός. Εάν ενεργοποιήσουμε το check box, η γραμμή θα επανέλθει. Το check box που βρίσκεται στο κάτω μέρος του παραθύρου είναι το keep track. Όσο είναι ενεργοποιημένο, τα διαγράμματα ακολουθούνε - μετατοπίζονται, όσο προχωράνε τα δείγματα και έτσι σιγά-σιγά αφήνουν πίσω τα παλιά δείγματα. Εάν όμως απενεργοποιηθεί το συγκεκριμένο check box, τότε τα διαγράμματα σταματάνε στο συγκεκριμένο σημείο δειγμάτων και μπορούμε με τις οριζόντιες μπάρες που βρίσκονται στο κάτω μέρος τους, να μετακινηθούμε και να πάμε σε οποιοδήποτε σημείο του ιστορικού δειγμάτων επιθυμούμε. Και σε αυτή την περίπτωση η εφαρμογή συνεχίζει να δέχεται και να προβάλει κανονικά τα δεδομένα, απλώς βάλαμε τα διαγράμματα σε freeze mode. Όταν ενεργοποιήσουμε εκ νέου το check box, τότε τα διαγράμματα συνεχίζουν να ακολουθούνε τα δείγματα και να μετατοπίζονται ανάλογα με αυτά.

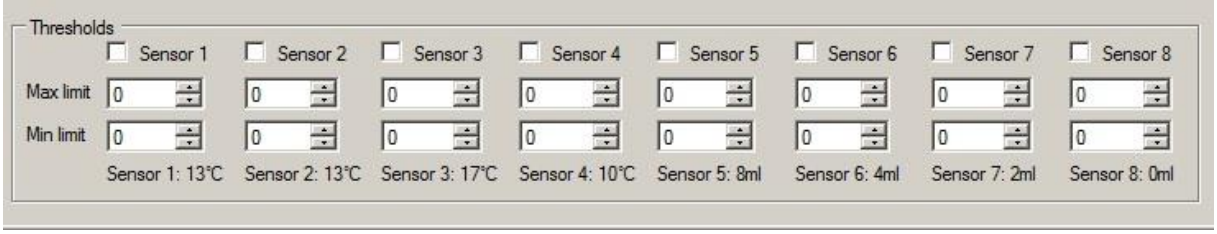


Figure 37 - Λεπτομέρεια που δείχνει τα όρια

Τέλος υπάρχει όπως και στο Real time values, ξεχωριστό περιβάλλον για τα Thresholds. Εδώ μπορούμε πάλι να ορίσουμε κατώτερο - ανώτερο όριο, να ενεργοποιήσουμε το επίπεδο συναγερμού (alarm) και να επιβλέπουμε την ακριβή τιμή κάθε αισθητήρα (πάνω εικόνα). Αν τώρα κλείσουμε το παράθυρο του Real time graphs θα δούμε πως το κεντρικό πάνελ έχει την παρακάτω μορφή:

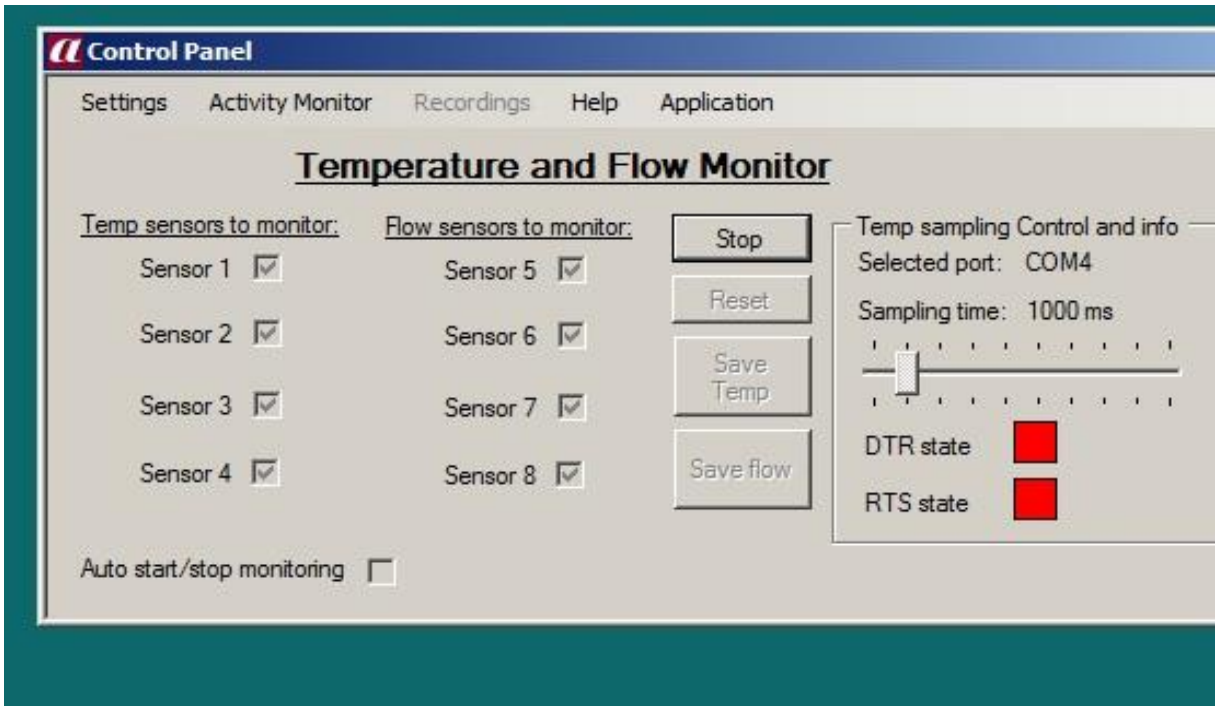


Figure 38 - Κεντρική οθόνη κατά τη φάση εκτέλεσης του προγράμματος

Με λίγα λόγια είναι απενεργοποιημένα όλα τα πλήκτρα και τα check boxes των αισθητήρων, εκτός από το πλήκτρο ***Stop***. Εάν όμως πατήσουμε το πλήκτρο Stop θα δούμε να ενεργοποιούνται τα πλήκτρα Save και Reset και το κουμπί Stop να γίνεται αυτή τη φορά όχι Start, αλλά Resume. Σε αυτό το σημείο μπορούμε σε πρώτη φάση να αποθηκεύσουμε σε ειδικά αρχεία τις μέχρι τώρα μετρήσεις που έκανε το πρόγραμμα. Έχοντας σταματήσει λοιπόν την εκτέλεση (απλά έχοντας πατήσει το Stop), επιλέγουμε το κουμπί ***Save Temp***. Θα ανοίξει κατευθείαν παράθυρο για αποθήκευση σε οποιοδήποτε directory του

σκληρού δίσκου ορίσουμε εμείς, ενος .xml αρχείου των μετρήσεων θερμοκρασίας. Ίδια ακριβώς λειτουργία έχει και το κουμπί ***Save Flow***, με τη διαφορά οτι αποθηκεύει τις τιμές της ροής.

Στο δεξί μέρος του κεντρικού πάνελ υπάρχει αντίστοιχο κουμπί ώστε να αποθηκεύονται οι συνολικές ροές σε ένα αρχείο. txt. (Η αποθήκευση των συνολικών ροών είναι ίσως πιο σημαντικό δεδομένο για την εποπτική λειτουργία απ'ότι είναι οι τιμές ροής σε πραγματικό χρόνο). Από πάνω του βλέπουμε τις τιμές της συνολικής ροής νερού ανά αισθητήρα ροής και από κάτω του υπάρχει το κουμπί ***Reset sum flow file.*** Αυτό το πλήκτρο υπάρχει ώστε να μηδενίζει/καθαρίζει και ο μετρητής συνολικής ροής, αλλά και το αρχείο συνολικής ροής από το οποίο αντιγράφεται δεύτερο αρχείο κάθε φορά που πατάμε το κουμπί ***Save sum flows***. Εδώ να επισημανθεί ότι εάν δεν κάνουμε reset το αρχείο συνολικής ροής, όταν θα κάνουμε νέες μετρήσεις ροής και τις αποθηκεύσουμε, αυτές θα αποθηκεύονται κάτω από τις προηγούμενες και έτσι δε θα χάνουμε παλαιότερες πληροφορίες.

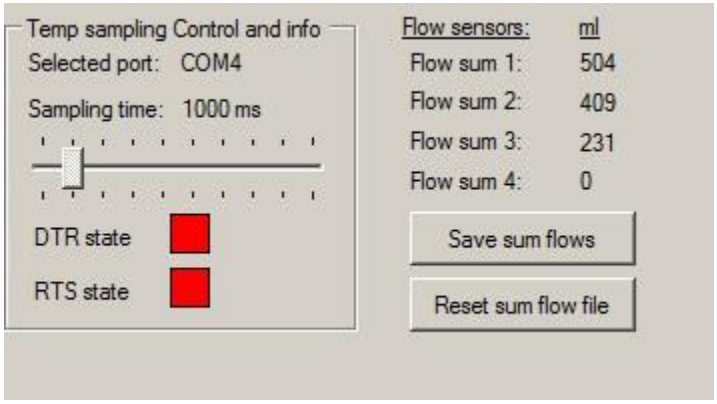


Figure 39 - Λεπτομέρεια μετρητών ροής

Αφού κάνουμε όλα τα παραπάνω, μπορούμε πατώντας το κουμπί Resume να συνεχίσουμε την διαδικασία ελέγχου από εκεί που είχε μείνει. Πρέπει να τονιστεί σε αυτό το σημείο το γεγονός οτι κάθε φορά γίνεται εποπτεία για εκείνη την στιγμή και όχι για προηγούμενες. Κάθε τιμή από αισθητήρα που έρχεται στην εφαρμογή αντιπροσωπεύει την κατάσταση του αισθητήρα την στιγμή που έρχεται και όχι προηγούμενες χρονικές στιγμές. Συνεπώς η προσωρινή διακοπή της λειτουργίας θα επιφέρει ένα λογικό κενό δεδομένων.

γ) Αρχεία .xml και συγκρίσεις μετρήσεων

Όταν αποθηκεύουμε ένα αρχείο .xml θερμοκρασίας ή ροής, αυτό το αρχείο είναι αντιγραφή ενός προσωρινού αρχείου το οποίο εξάγει η εφαρμογή όση ώρα είναι σε λειτουργία η διαδικασία εποπτείας (monitoring), δηλαδή όση ώρα είναι πατημένο το πλήκτρο Start. Το αρχείο αυτό εξαφανίζεται με το που πατηθεί το Reset. Ακόμα, αυτό το προσωρινό .xml αρχείο από το οποίο θα προέλθει το αρχείο που θα αποθηκεύσουμε εμείς στον Η/Υ εξάγεται μέσα στον φάκελο του sevrer της ιστοσελίδας για τον λόγο του οτι θέλουμε να διαβάζεται και από την ιστοσελίδα. Συνεπώς αν θέλουμε να αλλάξουμε το που εξάγεται το προσωρινό αυτό αρχείο, θα πρέπει να επέμβουμε στον κώδικα, οπότε η εγκατάσταση του προγράμματος αυτού καλό θα είναι να γίνεται από άτομο που γνωρίζει τον κώδικα της εφαρμογής ώστε να αλλάξει το συγκεκριμένο path όσες φορές χρειάζεται ανάλογα με το που είναι το αρχείο του server. Περισσότερες πληροφορίες για αυτό δίνονται στο κεφάλαιο ανάλυσης του κώδικα της εφαρμογής.

Το .xml αρχείο είναι ίδιο στη δομή του είτε πρόκειται για αισθητήρες θερμοκρασίας, είτε για αισθητήρες ροής. Η βασική διαφορά τους είναι ότι στο μεν ένα οι αισθητήρες είναι Sensor 1, 2 και 3 ενώ στο άλλο είναι Sensor 4, 5 και 6. Η μορφή του .xml είναι η παρακάτω:

```
<!-- 5/5/2017 -->

<!-- 14:54:26 -->
<!-- 1 s -->
<!--
Sensor1: ON /  Sensor2: ON /  Sensor3: ON /  Sensor4: ON
-->
<Values>
<Sample1>
<Sensor1 value="12" Time="14:54:27"/>
<Sensor2 value="12" Time="14:54:27"/>
<Sensor3 value="16" Time="14:54:27"/>
<Sensor4 value="9" Time="14:54:27"/>
</Sample1>
<Sample2>
<Sensor1 value="13" Time="14:54:28"/>
<Sensor2 value="13" Time="14:54:28"/>
<Sensor3 value="17" Time="14:54:28"/>
<Sensor4 value="9" Time="14:54:28"/>
</Sample2>
<Sample3>
<Sensor1 value="14" Time="14:54:29"/>
<Sensor2 value="13" Time="14:54:29"/>
<Sensor3 value="17" Time="14:54:29"/>
<Sensor4 value="9" Time="14:54:29"/>
</Sample3>
</Values>
```

Η εφαρμογή υποστηρίζει την σύγκριση μετρήσεων μέσω διαγραμμάτων. Αυτό γίνεται κάνοντας εισαγωγή αρχείων .xml που έχουν την μορφή της παραπάνω δομής και τα οποία αποθηκεύτηκαν μέσα από την εφαρμογή. Για να δούμε πως λειτουργεί η διαδικασία της σύγκρισης επιλέγουμε από το πάνω menu την επιλογή ***Recordings*** και μετά την υποεπιλογή ***Compare Recordings***. Το παράθυρο που θα ανοίξει φαίνεται στην παρακάτω εικόνα.

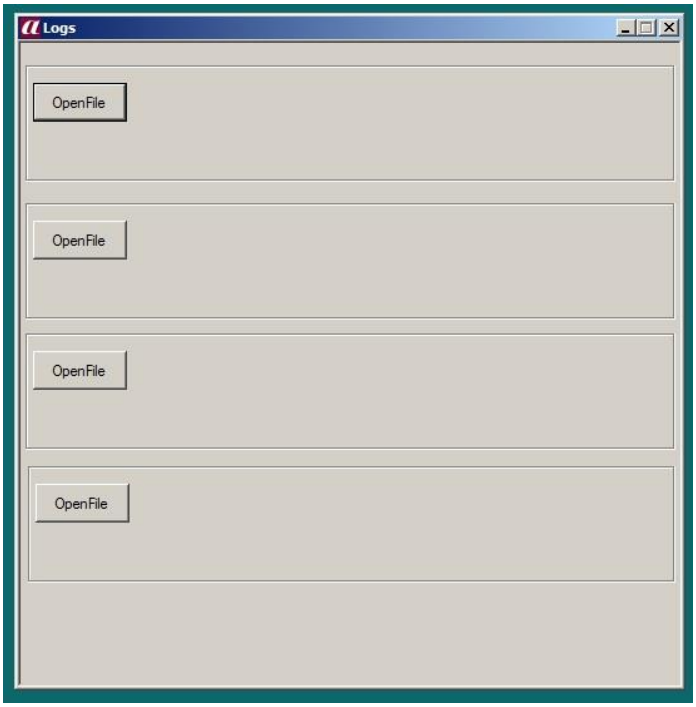


Figure 40 - Η φόρμα Compare Recordings

Από τα κουμπιά ***Open File*** μπορούν να εισαχθούν μέχρι και 4 αρχεία μετρήσεων της μορφής .xml, ωστόσο όπως πληροφορεί και το παράθυρο που θα βγει αυτομάτως, δεν πρέπει να επιχειρήσουμε σύγκριση δύο αρχείων διαφορετικών στοιχείων (ροής με θερμοκρασίας) γιατί θα υπάρξει πρόβλημα στην ομαλή λειτουργία του προγράμματος. Θα πρέπει να εισάγονται από 1 μέχρι 4 αρχεία είτε μόνο ροής, είτε μόνο θερμοκρασίας. Ωστόσο δίνεται η δυνατότητα στο χρήστη εξετάζοντας ξεχωριστά δεδομένα θερμοκρασίας και ροής να εξάγει συμπεράσματα για τον τρόπο λειτουργίας της παραγωγικής διαδικασίας, την ανίχνευση αιτιών σφαλμάτων κτλ.

Όταν πατήσουμε το κουμπί για εισαγωγή ενός αρχείου μετρήσεων θα ανοίξει ένα παράθυρο μέσω του οποίου μπορεί ο χρήστης να ψάξει στο αρχεία (directory) του σκληρού δίσκου του Η/Υ και να επιλέξει το αρχείο που επιθυμεί. Αφού εισάγουμε για τις ανάγκες του παραδείγματος δύο αρχεία, το πάνελ σύγκρισης θα έχουν την παρακάτω μορφή:

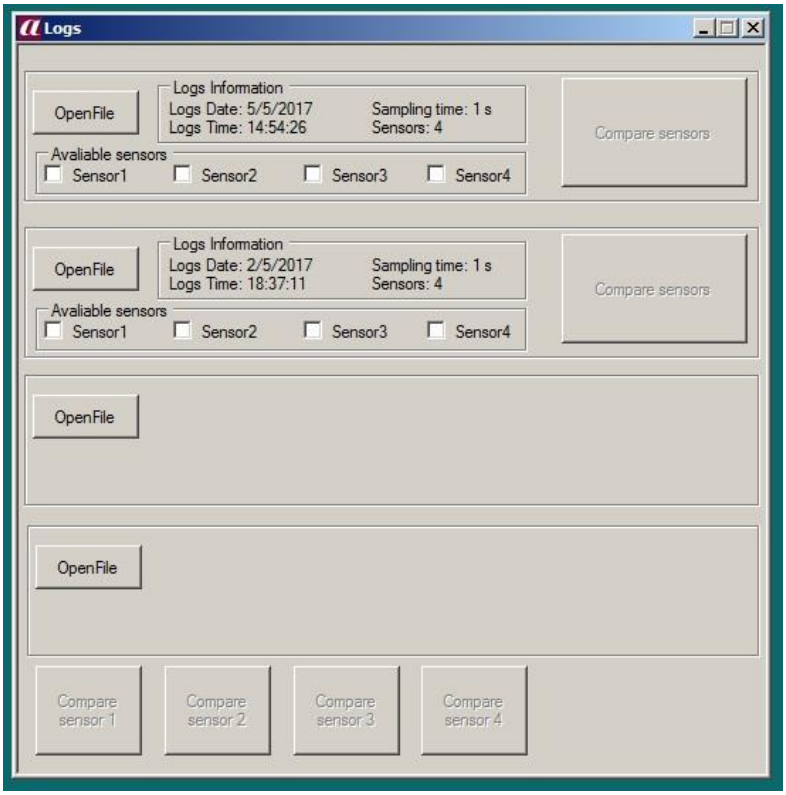


Figure 41 - Η φόρμα Compare Recordings αφού εισάγουμε αρχεία για σύγκριση

Όπως παρατηρούμε από την εικόνα τα κουμπιά Compare sensors και Compare sensor 1, 2, 3 κλπ είναι απενεργοποιημένα. Θα ενεργοποιηθούν μόνο αφού επιλέξουμε τα check boxes των αισθητήρων που επιθυμούμε να δούμε. Έστω οτι επιλέγουμε όλους τους αισθητήρες και επομένως επιλέγουμε και τα 8 check boxes. Οι επιλογές που μας παρέχει η εφαρμογή είναι:

1. Η σύγκριση των ίδιων αισθητήρων διαφορετικών αρχείων (κάθετη σύγκριση - κουμπί Compare Sensor 1, 2, 3 και 4)
2. Η σύγκριση των αισθητήρων ίδιου αρχείου μεταξύ τους (οριζόντια συκγριση - κουμπί Compare sensors)

Στην σύγκριση ίδιων αισθητήρων (πχ μόνο των Sensor 2) διαφορετικών αρχείων, πρέπει να πατήσουμε το κουμπί ***Compare sensor 2*** που βρίσκεται κάτω. Αμέσως θα ανοίξει το παρακάτω παράθυρο:

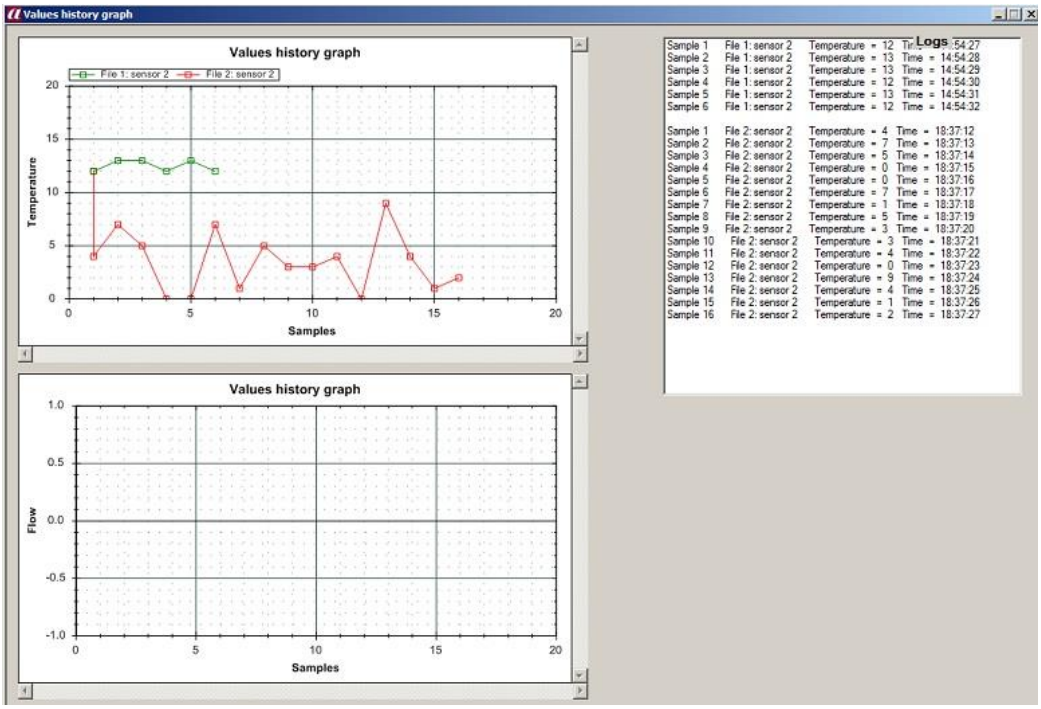


Figure 42 - Διαγράμματα σύγκρισης

Το παράθυρο αυτό αποτελείται από τρία μέρη. Το κάτω μέρος είναι το διάγραμμα για σύγκριση ροής το οποίο και είναι κενό διότι στην προκειμένη περίπτωση συγκρίνουμε θερμοκρασία. Το πρόγραμμα καταλαβαίνει μόνο του αυτόματα το αν πρόκειται για ροή ή για θερμοκρασία. Το πάνω διάγραμμα είναι το διάγραμμα σύγκρισης θερμοκρασίας που σε αυτή την περίπτωση μας ενδιαφέρει και δεξιά υπάρχει ειδικό πλαίσιο με τα δείγματα, τις τιμές τους και την ώρα καταγραφής τους.

Όπως βλέπουμε με πράσινο χρώμα είναι η γραμμή και τα δείγματα του Sensor 2 του πρώτου αρχείου και με κόκκινο χρώμα τα δείγματα και η γραμμή του Sensor 2 του δεύτερου αρχείου. Όπως και στην εποπτεία πραγματικού χρόνου (real time monitoring) έτσι και εδώ, ισχύει το zoom in/out με το ποντίκι πάνω στο διάγραμμα το οποίο μπορεί να γίνει και σε συγκεκριμένο σημείο κρατώντας πατημένο το αριστερό κλικ και σχεδιάζοντας ένα παραλληλόγραμμο, αλλά και το δεξί κλικ το οποίο παρέχει λειτουργίες εκτύπωσης, αντιγραφής κ.α. Η πιο σημαντική όμως λειτουργία του συγκεκριμένου παραθύρου σύγκρισης είναι η δυνατότητα που παρέχει, να επιλέγεται αυτόματα το αντίστοιχο δείγμα στο δεξί πλαίσιο όταν πηγαίνουμε τον δείκτη του ποντικιού πάνω από τα δείγματα. Αυτό είναι εξαιρετικά χρήσιμο σε αρχεία καταγραφής που αποτελούνται από πάρα πολλά δείγματα. Συνεπώς πηγαίνοντας όπου επιθυμούμε στο διάγραμμα και περνώντας το ποντίκι απλά πάνω απο τα δείγματα, επιλέγεται αυτόματα η πληροφορία του αντίστοιχου δείγματος στο διπλανό πίνακα που περιλαμβάνει τον αριθμό του δείγματος, την τιμή του, καθώς και την ώρα καταγραφής του.

Αν κλείσουμε αυτό το παράθυρο θα επανέλθουμε στο πάνελ εισαγωγής αρχείων για σύγκριση. Πατώντας τώρα κάποιο από τα κουμπιά Compare sensors, επιλέγουμε να συγκρίνουμε τους επιλεγμένους αισθητήρες ενός μόνο αρχείου μεταξύ τους. (κάτω εικόνα).

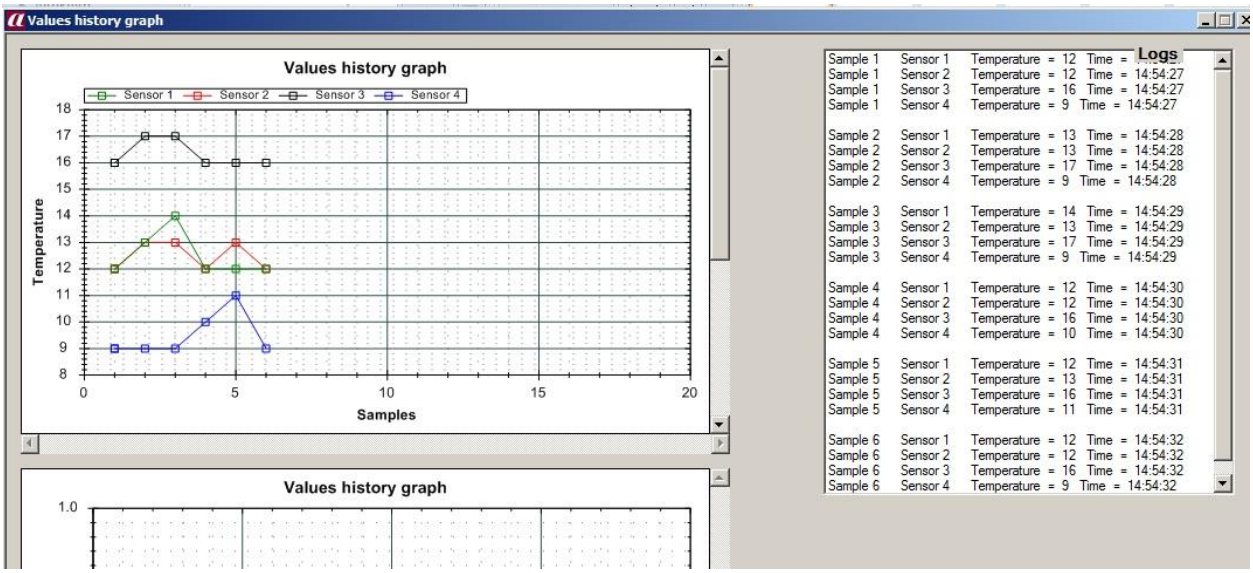


Figure 43 - Σύγκριση αισθητήρων με δεδομένα σε κοινό αρχείο

Και σε αυτή την περίπτωση ισχύουν τα ίδια ακριβώς με την προηγούμενη όσον αφορά την αυτόματη επιλογή στον δεξί πίνακα όταν περνάμε τον δείκτη πάνω από τα δείγματα, το zoom κλπ. Η μόνη διαφορά είναι ότι συγκρίνουμε τους αισθητήρες ενός αρχείου μεταξύ τους για αυτό και βλέπουμε στον πίνακα δίπλα ότι οι πληροφορίες είναι δομημένες ανα μπλοκ αισθητήρων για κάθε δείγμα. Στο διάγραμμα, κάθε χρώμα αντιπροσωπεύει τον αντίστοιχο αισθητήρα του αρχείου που εξετάζουμε.

Το παραπάνω παράδειγμα έγινε με αρχείο καταγραφής θερμοκρασίας. Ίδια διαδικασία είναι και με το αρχείο ροής με τη διαφορά ότι η σύγκριση γίνεται στο κάτω διάγραμμα όπως γράφει και ο άξονας Y του. Τονίζεται ότι το πρόγραμμα καταλαβαίνει αυτόματα αν το αρχείο είναι ροής ή θερμοκρασίας και το τοποθετεί στο κατάλληλο διάγραμμα κάθε φορά.

Τέλος, υπάρχουν λειτουργίες για την μετατροπή των αρχείων .xml σε αρχεί .txt και .xlsx (Excel). Με κλικ στο Recordings από την επιλογή του menu, επιλέγουμε αυτή τη φορά το Compatibility (συμβατότητα) και αντίστοιχα .txt ή .xlsx αναλόγως σε τι θέλουμε να το μετατρέψουμε. Θα ανοίξει παράθυρο για επιλογή του xml αρχείου και έπειτα το που θέλουμε να αποθηκεύσουμε το .txt ή το .xlsx.

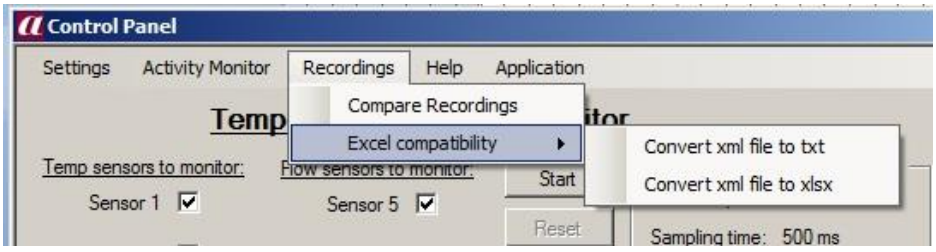


Figure 44 - Επιλογή για μετατροπές αρχείων σε διάφορες μορφές

Οι επιλογές Help και Application παρέχουν πληροφορίες χρήσης της εφαρμογής, πληροφορίες για την δημιουργία της και επιλογές για την επαννεκίνησή της ή τον τερματισμό της.

4.3 Ανάλυση λειτουργίας ιστοσελίδας

Το τελικό στάδιο της εργασίας αποτελείται από μία ιστοσελίδα η οποία φιλοξενεί κάποιες λειτουργίες που καθιστούν κατάλληλη την εποπτεία τιμών από το ελαιουργείο σε οποιοδήποτε μέρος, μιας και η ιστοσελίδα μπορεί είτε να μεταφορτώνεται στο Internet είτε σε τοπικό δίκτυο του Ιδρύματος. Αυτό λειτουργεί συμπληρωματικά στην βασική εφαρμογή εποπτείας που περιγράφηκε σε προηγούμενα κεφάλαια, σε περιπτώσεις που ο χρήστης θέλει να μάθει πληροφορίες για τις τιμές των αισθητήρων και δε μπορεί να βρίσκεται εκείνη τη στιγμή στον χώρο του Ελαιουργείου ώστε να κάνει χρήση του ειδικού software που εκτελείται στον Η/Υ. Για αυτό το λόγο, από τον ίδιο τερματικό Η/Υ που βρίσκεται στον χώρο του ελαιουργείο, που εισέρχονται οι τιμές μέσω της κεραίας και που εκτελείται το πρόγραμμα εποπτείας, θα μεταφορτώνεται ένα website στο δίκτυο, καθιστώντας τον συγκεκριμένο υπολογιστή και εξυπηρετητή (server).

Με την πρώτη επίσκεψη στην ιστοσελίδα βλέπουμε την αρχική σελίδα υποδοχής στην οποία όπως βλέπουμε και στην εικόνα υπάρχουν στο πάνω μέρος οι επιλογές του menu.
Η επιλογή Realtime Temp που αναφέρεται στην εποπτεία σε πραγματικό χρόνο της θερμοκρασίας (των τεσσάρων αισθητήρων θερμοκρασίας), η επιλογή Real time total flow όπου μπορούμε να δούμε πάλι σε πραγματικό χρόνο την συνολική ροή από τους τέσσερις αισθητήρες ροής, η Offline Tables όπου γίνεται σε μη πραγματικό χρόνο σύγκριση παλαιότερων μετρήσεων, η Realtime both όπου σε πραγματικό χρόνο βλέπουμε και τη ροή και την θερμοκρασία και τέλος η επιλογή Contact για επικοινωνία με τον διαχειριστή.

Figure 45 - Πάνω μέρος ιστοσελίδας

α) Εποπτεία Real time temp

Όπως έχει αναφερθεί, η ιστοσελίδα είναι σχεδιασμένη ώστε να είναι εξαιρετικά εύκολη η εποπτεία των τιμών των αισθητήρων. Κάνοντας κλικ στην επιλογή Real Time Temp, ανοίγει μία σελίδα όπως η παρακάτω.

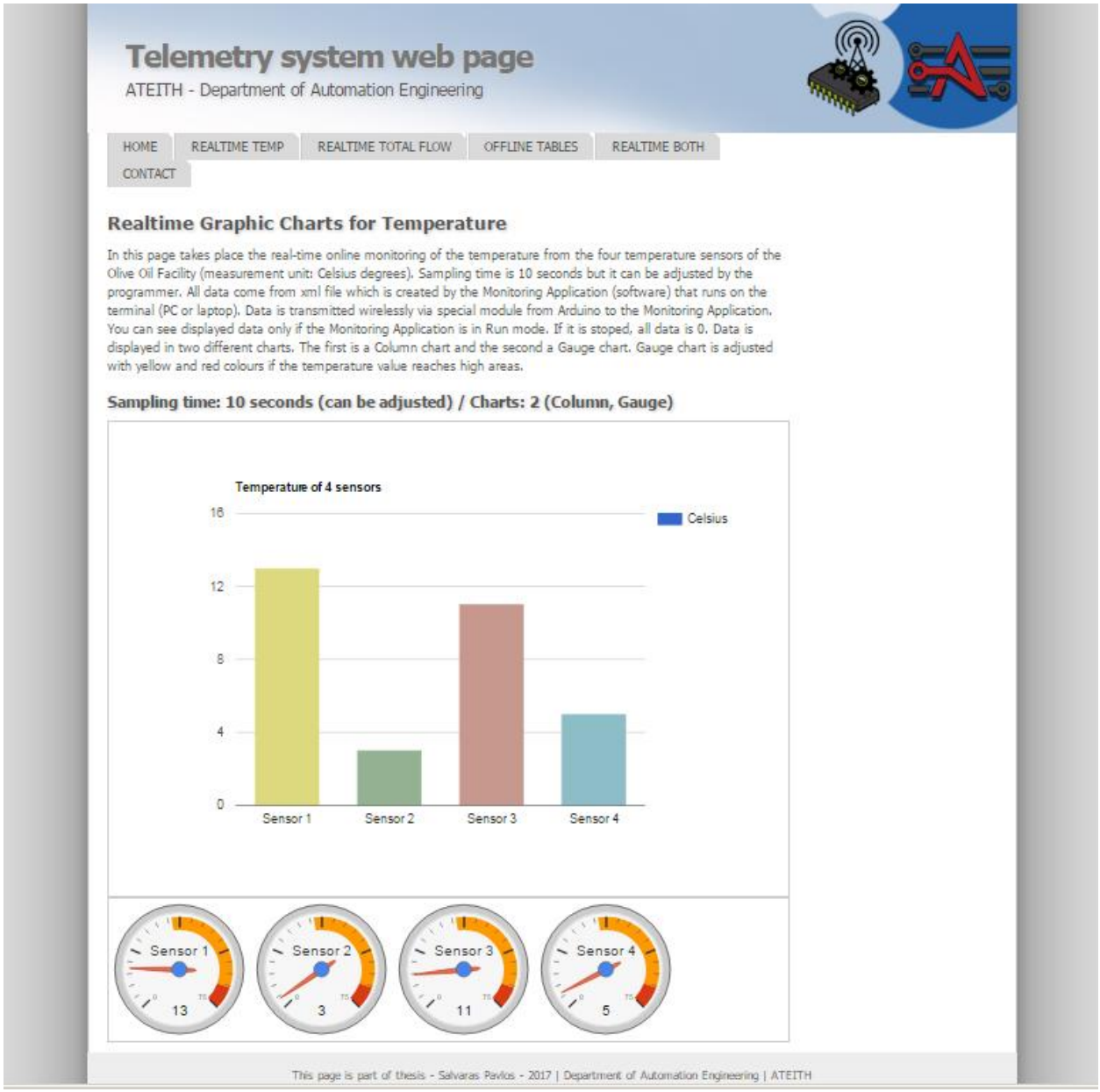


Figure 46 - Λειτουργία Real Time Temperature

Η σελίδα αυτή αποτελείται από δύο είδη γραφημάτων. Τις μπάρες που βλέπουμε στην πάνω εικόνα και ακριβώς κάτω τα ρολόγια μέτρησης. Και τα δύο παρέχουν απλή και ουσιαστική απεικόνιση των τιμών θερμοκρασίας. Εάν δεν είναι ανοιχτό το σύστημα από το ελαιουργείο, εάν το Arduino δεν δουλεύει, εάν η εφαρμογή που εξάγει τα xml δεν είναι σε λειτουργία ή εάν υπάρχει οποιοδήποτε πρόβλημα, τότε τα παραπάνω δύο γραφήματα θα δείχνουν απλά την τιμή 0 παντού. Αν όμως όλα λειτουργούν όπως πρέπει και είναι σε λειτουργία (Ελαιουργείο - TLK 31 - αισθητήρες PT 100 - Modbus - Arduino - Κεραίες - εφαρμογή στον Η/Υ), τότε θα βλέπουμε κανονικά τις τιμές που στέλνονται. Η σελίδα αυτή έχει προγραμματισθεί ώστε να κάνει αυτόματο refresh κάθε 10 δευτερόλεπτα ώστε να μην γίνονται συνεχώς

άσκοπα refresh, να προλαβαίνει ο χρήστης να δει - σημειώσει τις τιμές που επιθυμεί, αλλά και να μην χάνονται πολλές ενδιάμεσες τιμές κατά την διαδικασία λειτουργίας του Ελαιουργείο. Όσον αφορά τα γραφήματα, βάζοντας τον κέρσορα του ποντικιού πάνω από κάθε μπάρα μπορούμε να δούμε κατευθείαν την αριθμιτική τιμή της θερμοκρασίας κάθε αισθητήρα, η οποία ωστόσο φαίνεται και από τα γραφήματα των δεικτών. Αυτά μάλιστα έχουν σχεδιαστεί ώστε να αλλάζει η κλίμακά τους αυτόματα όσο ανεβαίνει η θερμοκρασία. Για παράδειγμα αρχικά αν οι θερμοκρασίες είναι για παράδειγμα στους 20 βαθμους Κελσίου οι μετρητές έχουν μέγιστη τιμή τους 50 βαθμούς Κελσίου. Εάν όμως οι θερμοκρασίες ξεπεράσουν τους 50 βαθμούς, τότε και οι μετρητές αυτόματα αλλάζουν την τιμή max στους 100 βαθμούς Κελσίου.

β) Εποπτεία Real time total flow

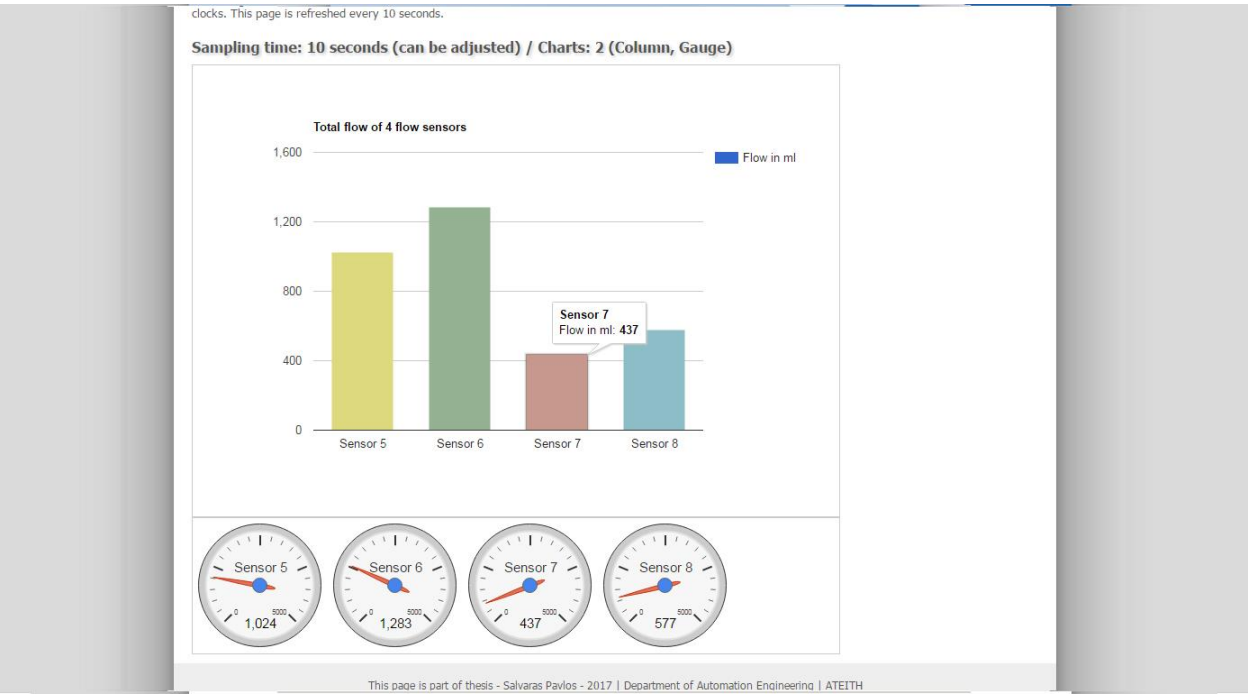


Figure 47 - Λειτουργία Real time total flow

Και εδώ η λειτουργία είναι ακριβώς ίδια με την προηγούμενη καρτέλα μόνο που εδώ εποπτεύονται οι αισθητήρες ροής. Να σημειωθεί πως και εδώ οι μέγιστες τιμές των δεικτών του δεύτερου διαγράμματος όπως και στην θερμοκρασία, αλλάζουν αυτόματα και προσαρμόζονται στην ροή. Όσο μεγαλώνει η ροή, τόσο αυξάνεται η max τιμή των μετρητών. Για παράδειγμα αρχίζει με 1 λίτρο ως max, αλλά εάν ξεπεραστεί η ροή του ενός λίτρου, τότε οι μετρητές αποκτούν ως max τιμή τα 5 λίτρα. Εάν ξεπεραστούν και τα 5 λίτρα σε ροή που έχει περάσει από κάποιον αισθητήρα, τότε αλλάζει και πάλι η max τιμή σε 10 λίτρα κ.ο.κ. Και εδώ τα δείγματα ανανεώνονται κάθε 10 δευτερόλεπτα, ωστόσο αυτό δεν επηρεάζει την μέτρησή μας καθώς γίνεται μέτρηση της συνολικής ροής του κάθε αισθητήρα και όχι της στιγμιαίας (κάτι τέτοιο δε θα είχε νόημα στην περίπτωση της ροής).

γ) Εποπτεία Real time both

Εάν κάνουμε κλικ στην επιλογή Real time both, τότε μπορούμε να εποπτεύουμε ανά 10 δευτερόλεπτα και τους 8 αισθητήρες του Ελαιουργείου. Όπως φαίνεται και στην εικόνα, υπάρχουν δύο γραφήματα με πλάγιες μπάρες, με το κάθε ένα να έχει από τέσσερις αισθητήρες. Οι 4 πρώτοι είναι της θερμοκρασίας και οι 4 κάτω της ροής. Και εδώ ισχύει ο,τι έχει περιγραφεί πιο πάνω, με τη διαφορά οτι εδώ γίνεται ταυτόχρονη εποπτεία όλων των αισθητήρων.

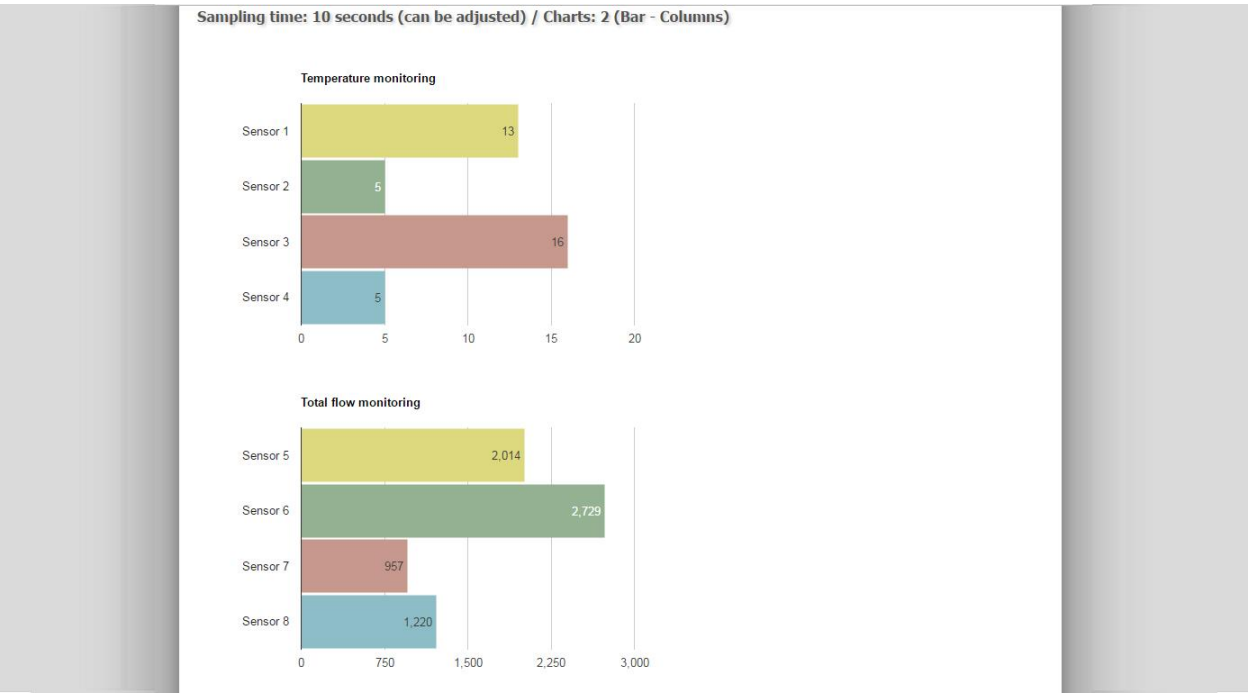


Figure 48 - Λειτουργία online παρακολούθησης για θερμοκρασία και ροή

δ) Εποπτεία αρχείων μη πραγματικού χρόνου (Offline tables)

Σε αυτή την σελίδα μπορεί ο χρήστης να δει σε μη πραγματικό χρόνο, στατικά δηλαδή και όχι δυναμικώς μεταβαλλόμενα, παλαιότερα αρχεία .xml μετρήσεων και να συγκρίνει τις τιμές τους δείγμα - δείγμα. Στην εικόνα που ακολουθεί μπορούμε να δούμε οτι υπάρχουν έξι κουμπιά. Τα τρία πρώτα είναι για παλιές μετρήσεις θερμοκρασίας, τα επόμενα τρία για μετρήσεις ροής ανά δευτερόλεπτο, και οι επόμενες τρείς για μετρήσεις συνολικής ροής. Τα συγκεκριμένα αρχεία ωστόσο θα πρέπει πρώτον να έχουν εισαχθεί στον ειδικό φάκελο recordings που βρίσκεται μέσα στον φάκελο της ιστοσελίδας και δεύτερον να έχουν υποστεί ειδική επεξεργασία με κώδικα σε γλώσσα PHP και να υπάρχει για κάθε ένα τέτοιο αρχείο μέτρησης και ένα αντίστοιχο αρχείο PHP στον ίδιο φάκελο (recordings). Το δεύτερο είναι υποχρεωτικό ώστε να εμφανίζονται τα αρχεία στην ιστοσελίδα και η όλη διαδικασία που θα πρέπει να γίνεται για κάθε αρχείο μέτρησης θα αναλυθεί στο κεφάλαιο της ανάλυσης του κώδικα της ιστοσελίδας.

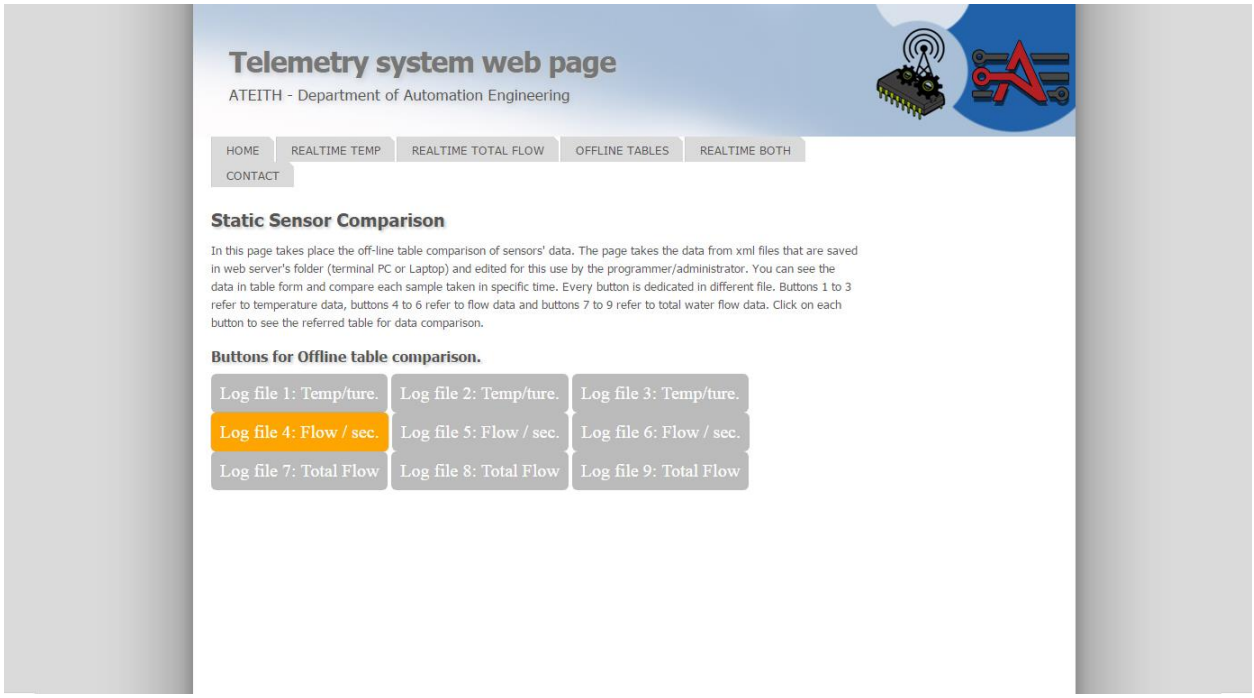


Figure 49 - Λειτουργία για off line σύγκριση

Έτσι εάν γίνει κλικ σε ένα κουμπί θα ανοίξει ο αντίστοιχος πίνακας ιστορικού μέτρησης όπως στην παρακάτω εικόνα:

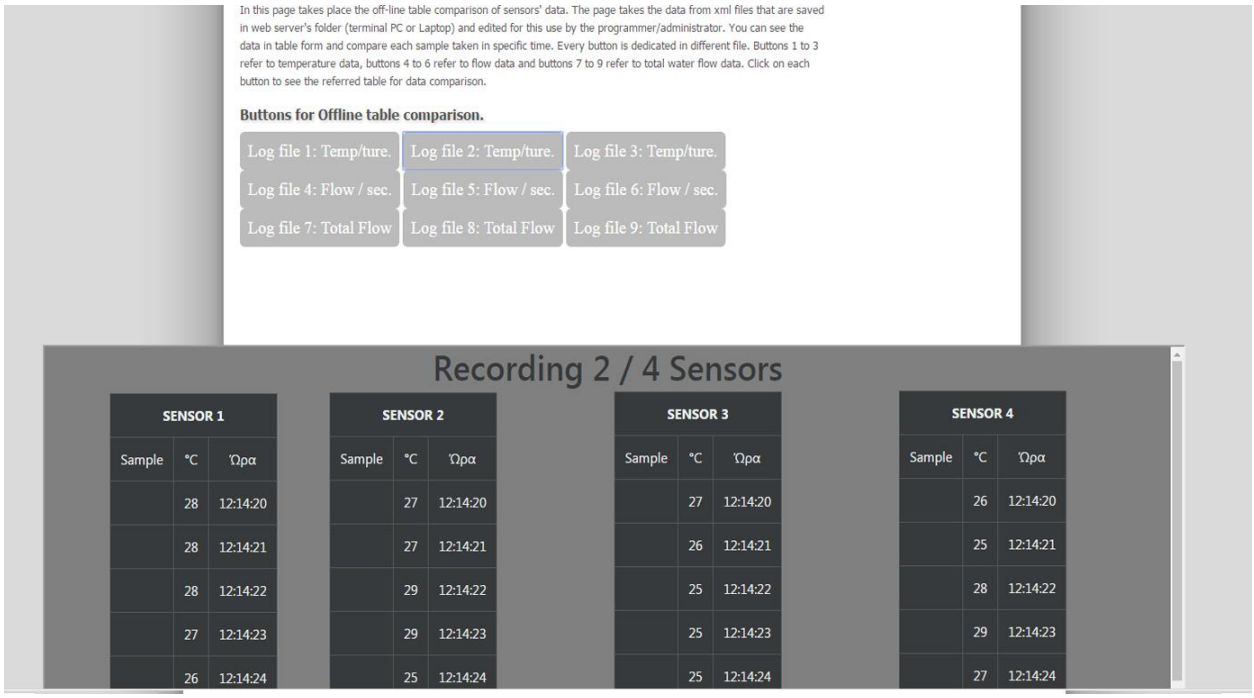


Figure 50 - Πίνακας σύγκρισης

Ο πίνακας ανοίγει ως αναδυόμενο παράθυρο στην ίδια σελίδα και περιλαμβάνει 4 στήλες (όσοι και οι αισθητήρες που περιλαμβάνονται στο αρχείο). Εδώ παρέχεται ακριβής εποπτεία δείγμα προς δείγμα

και δευτερόλεπτο προς δευτερόλεπτο. Κάθε δείγμα συνοδεύεται και από την ώρα την οποία μετρήθηκε και έτσι μπορεί ο χρήστης να βλέπει τις τιμές που επιθυμεί στην ακριβή ώρα που θέλει. Ακόμα το παράθυρο του πίνακα σύγκρισης έχει στα δεξιά και συρόμενη μπάρα σε περίπτωση που οι μετρήσεις είναι πολλές και δε χωράνε. Αν πατηθεί κάποιο άλλο κουμπί, ο πίνακας εξαφανίζεται και τη θέση του παίρνει ο νέος πίνακας που ανοίξαμε.

5. Κεφάλαιο: Τεχνική ανάλυση υλοποίησης

Σε αυτό το κεφάλαιο θα παρουσιαστεί η γενική αρχιτεκτονική δόμησης της εφαρμογής αλλά και της ιστοσελίδας. Ο κώδικας της εφαρμογής αριθμεί κοντά στις 4.000 γραμμές και συνεπώς μία αναλυτική παρουσίασή του είναι αδύνατη. Ωστόσο θα παρουσιαστεί ο τρόπος σκέψης πίσω από τις λειτουργίες που εκτελεί η εφαρμογή, οι συναρτήσεις που εκτελούνται και η ομαδοποίηση των λειτουργιών της κατά τον σχεδιασμό και τον προγραμματισμό της εφαρμογής αυτής.

Αντίστοιχα για την ιστοσελίδα, η αναλυτική παρουσίαση του κώδικα δημιουργίας και μορφοποίησης της θα ήταν κάτι περιττό. Για αυτό το σκοπό το τμήμα που αναφέρεται στην τεχνική ανάλυση της σελίδας επικεντρώθηκε στις τεχνικές συλλογης και παρουσίασης των δεδομένων.

5.1 Τεχνική ανάλυση εφαρμογής εποπτείας

Στο κεφάλαιο 4.2 αναλύθηκε το πως λειτουργεί το software του έργου το οποίο αποτελεί και τον κεντρικό άξονα γύρω από τον οποίο δομήθηκε η όλη εργασία. Το software αυτό υλοποιήθηκε με την γλώσσα C# στο Microsoft Visual Studio. Η C# είναι μία αντικειμενοστραφής γλώσσα προγραμματισμού η οποία εμφανίστηκε το 2000 από την Microsoft μέσω της πλατφόρμας .NET. Αυτή τη στιγμή (2017) έχει κυκλοφορήσει η έκδοση 7.0 η οποία είναι και η πιο πρόσφατη μέχρι στιγμής, με πιο πρόσφατη έκδοση της πλατφόρμας ανάπτυξης εφαρμογών Framework .NET, την 4.6.2 και πιο πρόσφατη έκδοση IDE, το Microsoft Visual Studio 2017. Η σύνταξη της C# είναι πολύ παρόμοια με γλώσσες όπως η C++, η C και η Java που μοιράζονται πολλά κοινά ως προς τον τρόπο γραφής τους και επομένως είναι σχετικά εύκολο το να προσαρμοστεί ένας προγραμματιστής που γνωρίζει κάποια από τις υπόλοιπες γλώσσες, στο περιβάλλον και τον τρόπο σύνταξης και λειτουργίας της C#.

Παρακάτω θα γίνει περιγραφή της γενικής φιλοσοφίας αρχιτεκτονικής της εφαρμογής, πάνω στην οποία δομήθηκε το πρόγραμμα και θα τονιστούν κάποια πιο σημαντικά σημεία. Η εφαρμογή αυτή αποτελείται από περίπου 4.000 γραμμές κώδικα συνεπώς η περίληψη της λειτουργίας πρέπει να εστιαστεί στα σημαντικά σημεία. Η περιληπτική περιγραφή της λειτουργίας του κώδικα και της τεχνοτροπίας που εφαρμόστηκε σε αυτόν είναι ιδιαίτερα χρήσιμη για να κατανοήσει κάποιος το πως λειτουργεί ο κώδικας. Σε όσα σημεία είναι απαραίτητο θα υπάρχουν μικρές παραθέσεις από κώδικα για την καλύτερη κατανόηση των όσων παρουσιάζονται.

Το πρόγραμμα αποτελείται απο 14 φόρμες:

1. ***Form1*** -> Η βασική αρχική φόρμα που εμφανίζεται κατά την εκτέλεση στην αρχή

2. ***Form1_functions*** -> Όλες οι συναρτήσεις που σχετίζονται με τη βασική φόρμα και λειτουργία

3. ***Open_logs_form*** -> Η φόρμα εισαγωγής αρχείων xml για σύγκριση

4. ***Open_logs_form_functions*** -> Οι συναρτήσεις που σχετίζονται με την open_logs_form

5***. Saved_logs_display*** -> Η φόρμα των συγκρίσεων

6. ***Saved_logs_display_functions*** -> Οι συναρτήσεις που σχετίζονται με την saved_logs_display

7***. Realtime*** -> Η φόρμα των γραφημάτων real time values

8. ***Realtime_functions*** -> Οι συναρτήσεις που σχετίζονται με την Realtime

9. ***Realtimecharts*** -> Η φόρμα των γραφημάτων real time graphs

10. ***Realtimecharts_functions*** -> Οι συναρτήσεις που σχετίζονται με την realtimecharts

11. ***Port*** -> Η φόρμα της επιλογής σειριακής θύρας

12. ***Manual*** -> Η φόρμα του manual

13***. Progress*** -> Η φόρμα της progress bar που εμφανίζεται κατα τις μετατροπές αρχείων

14. ***About*** -> Η φόρμα της επιλογής about

α) Εκκίνηση και ασύγχρονη λειτουργία

Με την εκκίνηση του προγράματος εκτελείται η φόρτωση των απαραίτητων αντικειμένων και στοιχείων που αποτελούν την αρχική φόρμα που παρουσιάζεται στον χρήστη κατά το ξεκίνημα. Δύο μετρητές του χρόνου timers οι οποίοι θα χρησιμοποιηθούν στη δειγματοληψία παίρνουν τιμή σύμφωνα με την επιλογή της ειδικής track bar και την θέση της.

Με το πάτημα του κουμπιού start καλείται και η αντίστοιχη συνάρτηση Start() η οποία είναι μια από τις βασικότερες συναρτήσεις της πρώτης φόρμας. Μέσα σε αυτή τη συνάρτηση και εάν έχει ήδη εντοπιστεί και επιλεχθεί σειριακή θύρα, αποθηκεύονται σε πρώτη φάση οι καταστάσεις των check boxes των αισθητήρων σε πίνακα τύπου bool, ώστε να γνωρίζει η εφαρμογή ποιούς αισθητήρες έχει επιλέξει ο χρήστης και ποιούς όχι. Έπειτα δημιουργούνται δύο αντικείμενα, το ***charts*** και το ***_f2*** τα οποία συσχετίζονται με τις φόρμες ***realtimecharts*** και ***realtime***. Τα δύο αυτά αντικείμενα είναι εξαιρετικά σημαντικά αφού είναι υπεύθυνα για τη μεταβίβαση των πληροφοριών από την κεντρική φόρμα (Form1) στις αντίστοιχες δύο φόρμες των γραφημάτων. Κάτι τέτοιο γίνεται αμέσως μετά με την μεταφορά της κατάστασης των check boxes των αισθητήρων σε αυτές τις δύο φόρμες γραφημάτων μέσω ειδικής συνάρτησης.

Αν λοιπόν έχει πατηθεί το κουμπί Start (εκκίνησης), τότε μεταβιβάζονται στις δύο φόρμες γραφημάτων (realtimecharts και realtime) και άλλες σημαντικές πληροφορίες όπως π.χ. ο χρόνος δειγματοληψίας που έχει επιλεχθεί από την track bar. Αμέσως μετά ανοίγει ο ***receiver***, ένα αντικείμενο που ανήκει στην κλάση ***Serial Port***, υπεύθυνο για το διάβασμα των τιμών από την θύρα USB του Η/Υ στον οποίο έχει συνδεθεί ο δέκτης (module APC 220). Αφού διαβάσει ο δέκτης (receiver) την θύρα USB ανά γραμμή και ενεργοποιήσει έναν ειδικό timer (timer1) ο οποίος καθαρίζει τον διάδρομο δεδομένων εισόδου (DiscardInBuffer) ανά ένα δευτερόλεπτο, καλείται η συνάρτηση ***InitCurveValues***. Με αυτή τη συνάρτηση γίνεται μία ασύγχρονη πρώτη ανάγνωση της σειριακής θύρας πριν ξεκινήσει η συγχρονισμένη ανάγνωση βάση του χρόνου δειγματοληψίας. Ελέγχεται μέσα σε αυτήν αν η τιμή του

receiver πληροί κάποια κριτήρια μεγέθους ώστε να μην προκύψει σφάλμα και έπειτα διαχωρίζει τις 8 τιμές που στέλνει το Arduino με το διαχωριστικό 'n' και τις καταχωρεί σε αντίστοιχες θέσεις πίνακα. Παρακάτω βλέπουμε ένα μικρό κομμάτι από αυτή τη διαδικασία με τον απαραίτητο σχολιασμό απο δίπλα ώστε να γίνει κατανοητός ο διαχωρισμός:

```
char[] delim = { 'n' }; //αναγνώριση του 'n' ως στοιχείο διαχωρισμού

string[] _tempValues;  // δημιουργία πίνακα προσωρινών τιμών

_tempValues = input.Split(delim, 8);   //"σπάσιμο" της string σειράς που στέλνει το Arduino σε 8 κομμάτια, αναγνωρίζοντας το διαχωριστικό χαρακτήρα 'n'

 if (_tempValues[0] != "")
  if (cb_state[0])
   b[0] = Convert.ToDouble(_tempValues[0]);  //αποθήκευση της πρώτης τιμής του αισθητήρα στην πρώτη θέση του πίνακας b

    if (_tempValues[1] != "")
     if (cb_state[1])
     b[1] = Convert.ToDouble(_tempValues[1]);  //αποθήκευση της δεύτερης τιμής του αισθητήρα στην δεύτερη θέση του πίνακα b κ.ο.κ.
```

Παράλληλα ξεκινάει μέσα από την InitCurveValues και ένας ακόμα timer (timer5) υπεύθυνος για τον υπολογισμό της συνολικής ροής νερού για τους 4 τελευταίους αισθητήρες.

Αφού σταλθούν όλες οι τιμές που έχουν αποθηκευθεί στον πίνακα b (οι τιμές δηλαδή όλων των αισθητήρων) ασύγχρονα, στις δύο φόρμες γραφημάτων μέσω των αντικειμένων _f2 και charts, τότε ξεκινάει η συγχρονισμένη διαδικασία λειτουργίας ενεργοποιώντας τους 2 timers: arduino_sampler και logger με χρονισμό την τιμή που έχει επιλεχθεί από την track bar για επιθυμητή δειγματοληψία.

β) Σύγχρονη λειτουργία

Ο timer arduino_sampler που έχει χρονιστεί με την τιμή δειγματοληψίας που έχει επιλεχθεί, καλεί την συνάρτηση ***receiveData***. Η συγκεκριμένη συνάρτηση ξαναδιαβάζει την είσοδο της σειριακής θύρας και καλεί την ***dataProcessing*** η οποία με την σειρά της επαναλαμβάνει τη διαδικασία διαχωρισμού της input τιμής από τον receiver, το σπάσιμο της δηλαδή σε 8 κομμάτια και καλεί την συνάρτηση ***values_generator.*** Η values_generator αφού κάνει κάποιους ελέγχους για το μέγεθος της input τιμής που ήρθε από το Arduino και διαβάστηκε από τον receiver, επαναλαμβάνει τη διαδικασία τοποθέτησης των διαχωρισμένων πλέον τιμών των αισθητήρων στο πίνακα b όπως ακριβώς είχε γίνει την πρώτη φορά. Η διαφορά όμως τώρα είναι οτι η διαδικασία είναι πλέον συγχρονισμένη με βάση την δειγματοληψία που έχει επιλεχθεί και έτσι θα επαναλαμβάνεται συνεχώς όσο η κατάσταση της εφαρμογής είναι εν λειτουργία. Τέλος η τελευταία αυτή συνάρτηση αφού μεταφέρει και πάλι τις τιμές με τη βοήθεια των αντικειμένων _f2 και charts στις αντίστοιχες φόρμες γραφημάτων, καλεί την συνάρτηση ***write_to_xml.*** Η συγκεκριμένη συνάρτηση είναι υπεύθυνη για την εγγραφή στα προσωρινά xml αρχεία, των τιμών των επιλεγμένων αισθητήρων. Πρώτα δημιουργεί τα δύο αυτά αρχεία στο directory του Apache server ώστε να ανεβαίνουν και στην ιστοσελίδα σε περίπτωση που αυτή λειτουργεί:

```
XDocument doc = XDocument.Load(@"C:\xampp\htdocs\teliko_site\_temp.xml");

XDocument doc2 = XDocument.Load(@"C:\xampp\htdocs\teliko_site\_flow.xml");
```

Στη συνέχεια ακολουθεί μία διαδικασία για πιθανούς συνδυασμούς των αισθητήρων που έχουν επιλεχθεί ώστε να γίνει η κατάλληλη εγγραφή. Συνολικά για τους 8 αισθητήρες (4 ροής και 4 θερμοκρασίας) υπάρχουν 30 διαφορετικοί συνδυασμοί - 15 για θερμοκασία και 15 για ροή.
Οι συνδυασμοί χωρίζονται σε δύο κατηγορίες: θερμοκρασίας και ροής. Αυτό συμβαίνει διότι δεν γίνεται να συσχετιστούν στο ίδιο xml διαφορετικής φύσης αισθητήρες. Παρακάτω βλέπουμε τους συνδυασμούς:

Αν είναι επιλεγμένοι για θερμοκρασία οι:
1+2+3+4 ή 1+2+3 ή 1+2+4 ή 1+3+4 ή 2+3+4 ή 1+2 ή 1+3 ή 1+4 ή 2+3 ή 2+4 ή 3+4 ή 1 ή 2 ή 3 ή 4

Αν είναι επιλεγμένοι για ροή οι:
5+6+7+8 ή 5+6+7 ή 5+6+8 ή 5+7+8 ή 6+7+8 ή 5+6 ή 5+7 ή 5+8 ή 6+7 ή 6+8 ή 7+8 ή 5 ή 6 ή 7 ή 8

Για παράδειγμα ο κώδικας για κάποιον συνδυασμό είναι όπως παρακάτω:

```
//(αν είναι επιλεγμένοι οι 5 + 6 + 7)

else if (sensor5CB.Checked && sensor6CB.Checked && sensor7CB.Checked &&
!sensor8CB.Checked)
        {
           XElement value1 = new XElement("Sample" + i,
              new XElement("Sensor5",
                 new XAttribute("value", b[4]),
                 new XAttribute("Time", "" + time_format())),//+ ":" + DateTime.Now.Second)),
              new XElement("Sensor6",
                 new XAttribute("value", b[5]),
                 new XAttribute("Time", "" + time_format())),//+ ":" + DateTime.Now.Second)),
              new XElement("Sensor7",
                 new XAttribute("value", b[6]),
                 new XAttribute("Time", "" + time_format())));// + ":" + DateTime.Now.Second)));
           doc2.Root.Add(value1);
        }
```

Παρατηρούμε πως αφού μπεί η τιμή του αισθητήρα που είναι αποθηκευμένη στον πίνακα b[], ακολουθεί και η ώρα και τέλος προστίθεται όλη η καταχώριση στο αρχείο .xml που έχουμε δημιουργήσει (.doc ή .doc2). Φυσικά στο τέλος όλων των πιθανών συνδυασμών υπάρχει η απαραίτητη αποθήκευση:

```
doc.Save(@"C:\xampp\htdocs\teliko_site\_temp.xml");
doc2.Save(@"C:\xampp\htdocs\teliko_site\_flow.xml");
```

Εδώ είναι και η λήξη της βασικής διαδικασίας σε ότι αφορά την κεντρική φόρμα, μιας και ακόμα δεν παρουσιάστηκε η αρχιτεκτονική πίσω από τα γραφήματα, τις συγκρίσεις κλπ. Αυτό ήταν ένας κύκλος της σύγχρονης λειτουργίας που εκτελείται σε κάθε τικ του timer (arduino_sampler) και επαναλαμβάνεται

όσο είναι ενεργός αυτός ο χρονιστής. Να σημειωθεί πως ο κώδικας για αυτή τη διαδικασία είναι πάνω από 1.000 γραμμές. Στην εικόνα της επόμενης σελίδας βλέπουμε ένα μπλοκ διάγραμμα για την διαδικασία μέχρι στιγμής:

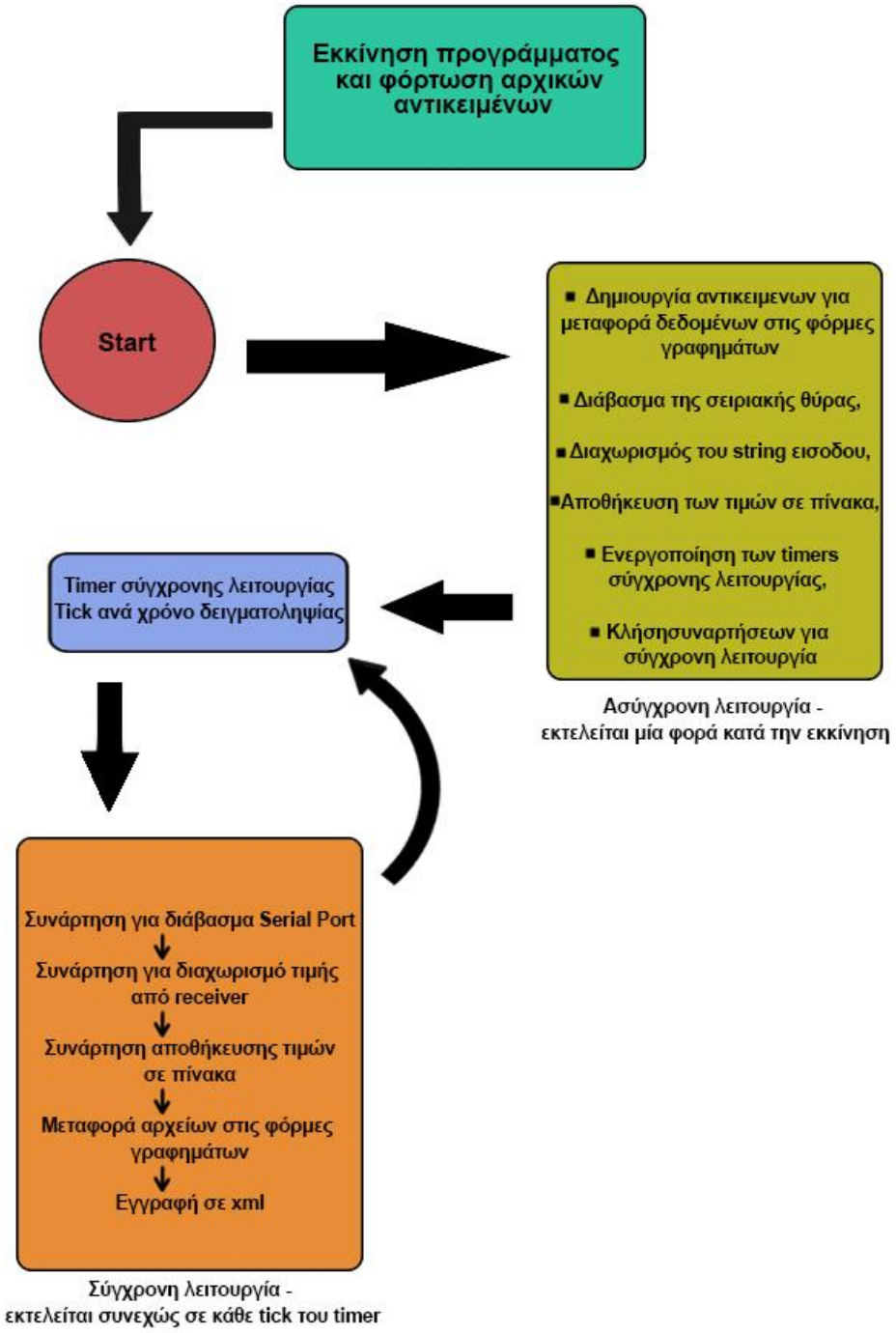


Figure 51 - Σύγχρονη και Ασύγχρονη λειτουργία

γ) Λειτουργία real time values και real time graphs

Μέχρι τώρα είδαμε οτι κατά την εκκίνηση της, η εφαρμογή διαβάζει την σειριακή θύρα (Serial port) και έπειτα με μία αλληλουχία συναρτήσεων που καλείται η μία μέσα από την άλλη, καταχωρεί τα δεδομένα σε πίνακα και τα στέλνει μέσω δύο αντικειμένων στις αντίστοιχες φόρμες των γραφημάτων. Η φόρμα που έχει να κάνει με την επιλογή Real time values είναι η φόρμα ***realtime*** και το αντικείμενο με το οποίο μεταβιβάζονται πληροφορίες από την βασική/κεντρική φόρμα σε αυτήν είναι το ***_f2.*** Τα διαγράμματα επίσης, που χρησιμοποιούνται στο πρόγραμμα είναι τα διαγράμματα ZedGraphs, που είναι γραμμένα σε C# και παρέχουν βιβλιοθήκη με δισδιάστατα γραφήματα.

Με το που επιλεχθεί από το menu η επιλογή Real time values θα ανοίξει η φόρμα που είχαμε δει στο κεφάλαιο με την περιγραφή λειτουργίας. Όσον αφορά τον κώδικα, εδώ δημιουργείται το πάνελ του γραφήματος, το grid, τα όρια που θα έχει και άλλες επιλογές που αφορούν την εμφάνιση. Η εμφάνιση των τιμών και η απεικόνιση των bars γίνεται με τη χρήση δύο κυρίως συναρτήσεων, της ***initCurves*** και της ***drawNextSpot_curve*** οι οποίες λειτουργούν χρονισμένα με βάση τον timer ***realtime_timer*** ο οποίος έχει πάρει την τιμή (interval) του χρόνου στον οποίο θα τρέχει, από την αρχική φόρμα και έτσι είναι συγχρονισμένος πλήρως με τον βασικό timer της αρχικής φόρμας. Έτσι λοιπόν σε κάθε tick αυτού του timer, καλούνται αυτές οι συναρτήσεις και προστίθενται κάθε φορά οι απαραίτητες μπάρες σύμφωνα με τις τιμές που έρχονται από την αρχική φόρμα.

Όσον αφορά την επιλογή Real time graphs, συσχετίζεται με την φόρμα ***realtimecharts*** και το αντικείμενο μεταβίβασης πληροφοριών από την κεντρική φόρμα, είναι το ***charts***. Και εδώ υπάρχει ο αντίστοιχος χρονιστής με το όνομα ***logger***, το χρονικό διάστημα (interval) του οποίου όπως και στην άλλη περίπτωση ορίζεται από την track bar της αρχικής φόρμας. Η βασική λειτουργία γίνεται με την κλήση σε κάθε χρόνο του timer της συνάρτησης ***displayCharts***, με τη βοήθεια και πάλι δύο ακόμα συναρτήσεων, ***initCurves*** και ***drawNextSpot_Curve***. Ανάλογα με την τιμή του αισθητήρα που έχει διαβαστεί από την κεντρική φόρμα (μέσω άλλης συνάρτησης) και ανάλογα με το ποιοί αισθητήρες είναι επιλεγμένοι, οι συναρτήσεις τοποθετούν το στίγμα πάνω στο γράφημα, το ενώνουν με μία γραμμή με το προηγούμενο στίγμα και ανανεώνουν το γράφημα. Σε αυτή την λειτουργία υπάρχουν και αισθητήρες ροής και αισθητήρες θερμοκρασίας, συνεπώς υπάρχουν δύο διαφορετικά γραφήματα.

δ) Σύγκριση αισθητήρων

Στη λειτουργία που περιγράψαμε στο προηγούμενο κεφάλαιο για την σύγκριση αισθητήρων, τονίστηκε το γεγονός οτι αυτό συμβαίνει μέσω αρχείων .xml που έχουν αποθηκευτεί κάπου στον Η/Υ. Η φιλοσοφία είναι να διαβάζει η εφαρμογή τα υπάρχοντα αποθηκευμένα αρχεία .xml, να τα επεξεργάζεται και να τα προσαρμόζει σε πλαίσιο σύγκρισης για την καλύτερη απεικόνιση και εποπτεία σε μη πραγματικό χρόνο (στατικά). Η αντίστοιχη φόρμα (***open_logs_form***) φορτώνει τα απαραίτητα αντικείμενα και περιμένει το πάτημα κουμπιού για την εισαγωγή κάποιου αρχείου .xml. Όλες οι συναρτήσεις που θα κληθούν προέρχονται από την δεύτερη φόρμα ***open_logs_form_functions*** (ίδια φιλοσοφία έχει ακολουθηθεί σε όλη την εφαρμογή όπως είδαμε και στην αρχή του κεφαλαίου που παρουσιάζονται οι φόρμες). Μόλις πατηθεί το κουμπί εισαγωγής αρχείου, καλείται και η αντίστοιχη συνάρτηση (***openfilen , όπου n=1,2,3,4***) από την δεύτερη φόρμα η οποία:

1. Καλεί μέσα της μία συνάρτηση για το διάβασμα των σχολίων στο αρχείο .xml. Τα σχόλια παρέχουν την εξής σημαντική πληροφορία, του ποιοί αισθητήρες έχουν επιλεχθεί και άρα ποιοί έχουν συμπεριληφθεί στην μέτρηση και ποιοί όχι.

2. Καλεί δεύτερη συνάρτηση για την ανάλυση των σχολίων και άρα των πληροφοριών με το ποιοί αισθητήρες περιλαμβάνονται στο αρχείο .xml.
3. Ενημερώνει την φόρμα open_logs_form για τις πρώτες βασικές πληροφορίες (ώρα, ποιοί αισθητήρες έχουν επιλεχθεί κ.α.)

Στη συνέχεια η φόρμα περιμένει από τον χρήστη να πατήσει κάποιο κουμπί για σύγκριση. Εάν πατήσει ο χρήστης το κουμπί για την σύγκριση των αισθητήρων του ίδιου αρχείου .xml, καλείται η συνάρτηση ***showfile***, ενώ αν πατήσει το κουμπί για τη σύγκριση των ίδιων αισθητήρων διαφορετικών αρχείων xml, καλείται η συνάρτηση ***comparesensor***.

Αν λοιπόν επιλεχθεί το πρώτο κουμπί, δηλαδή η οριζόντια σύγκριση, καλείται η ***showfile***. Μέσα σε αυτήν δημιουργείται άμεσα ένα αντικείμενο με όνομα ***_form*** που σκοπό έχει την αποστολή των δεδομένων στη φόρμα σύγκρισης, την ***saved_logs_display***. Αφού τα στείλει μέσω του νέου αυτού αντικειμένου, γίνεται μετάβαση μέσω αυτού στη φόρμα της σύγκρισης (saved_logs_display). Σε αυτήν πλέον υπάρχουν ειδικές συναρτήσεις που διαβάζουν άμεσα το .xml και τοποθετούν κάθε τιμή στο γράφημα δημιουργώντας γραμμή γραφήματος ανάλογα με τα δείγματα και με το ποιοί αισθητήρες είναι επιλεγμένοι. Επιπροσθέτως υπάρχει επιπλέον συνάρτηση ώστε να υπάρχει η λειτουργία της αυτόματης επιλογής δείγματος στο listbox, όταν ο κέρσορας του ποντικιού είναι πάνω από κάποιο δείγμα. Αν επιλεχθεί το δεύτερο κουμπί, δηλαδή η κάθετη σύγκριση και γίνει κλήση της comparesensor, τότε με παρόμοιο τρόπο δημιουργείται πάλι το αντικείμενο _form συσχετιζόμενο με την φόρμα συγκρίσεων saved_logs_display και ακολουθείται μία παρόμοια διαδικασία από την τελευταία αυτή φόρμα. Για να γίνει καλύτερα κατανοητή η αρχιτεκτονική μπορούμε να δούμε την παρακάτω εικόνα. Ουσιαστικά η διαδικασία ξεκινάει από την φόρμα εισαγωγής, έπειτα με το κουμπί εισαγωγής εισάγουμε στη φόρμα αυτή τα αρχεία μετρήσεων xml και τέλος γίνεται επιλογή είτε της οριζόντιας είτε της κάθετης σύγκρισης και μεταφερόμαστε στη φόρμα σύγκρισης:

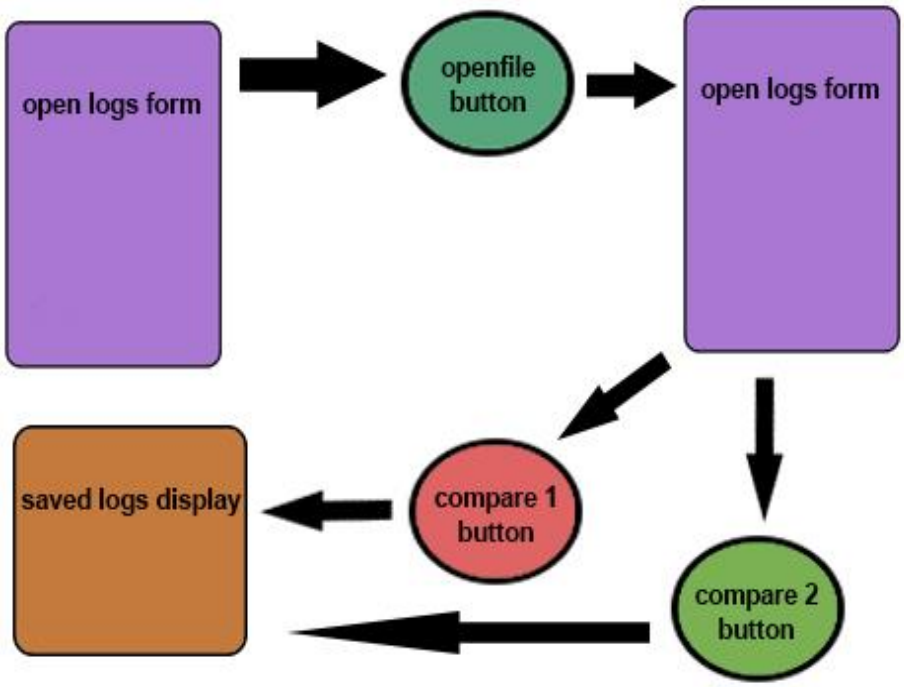


Figure 52 - Λειτουργία σύγκρισης

ε) Μετατροπή .xml σε .xlsx και .txt

Όταν γίνεται κλικ στην αντίστοιχη επιλογή μετατροπής σε .xlsx (αρχείο Excel), τότε καλείται η συνάρτηση ***ConvertToXLSX.*** Αρχικά ανοίγει παράθυρο ώστε να επιλεχθεί το επιθυμητό αρχείο .xml. Στη συνέχεια γίνεται έλεγχος για το αν είναι ή όχι εγκατεστημένο το πρόγραμμα MS Excel στον Η/Υ. Εάν είναι, τότε αρχίζει η διαδικασία μετατροπής. Αρχικά δημιουργείται ένα Excel Workbook και ένα Excel Worksheet. Γίνεται ο απαραίτητος διαχωρισμός των πληροφοριών ανάλογα με το πόσοι αισθητήρες περιέχονται στη μέτρηση και ανάλογα με τον αριθμό τους, φτιάχνονται και τόσες στήλες/γραμμές στο αρχείο xlsx. Μέχρι στιγμής επομένως το αρχείο είναι άδειο. Αφού μπούνε κάποια σχόλια όπως η ώρα και το sampling time, είναι έτοιμη η συνάρτηση να εισάγει τα δεδομένα από το xml στο xlsx. Δημιουργείται ένα αντικείμενο ***StreamReader*** το οποίο θα διαβάζει ολόκληρο το αρχείο μέχρι το τέλος του. Όταν φτάνει στο τέλος, αυτομάτως θα ξαναπηγαίνει στην αρχή.

5.2 Τεχνική ανάλυση ιστοσελίδας

α) Xampp και Apache

Το Xampp είναι μια πλατφόρμα που περιλαμβάνει συγκεκριμένα προγράμματα. Το όνομά του, είναι τα αρχικά Χ, για το γεγονός οτι εκτελείται σε διάφορες πλατφόρμες (X = cross-platform), το A για τον εξυπηρετητή Apache, το M για την βάση δεδομένων MySQL, το P για την γλώσσα κατασκευής ιστοσελίδων PHP και τέλος το P για την γλώσσα προγραμματισμού Perl (περιλαμβάνει και κάποια επιπλέον εργαλεία όπως το PhpMyAdmin, Filezilla κ.α.). Το Xampp είναι ελεύθερο πακέτο και ανοιχτού κώδικα λογισμικό και χρησιμοποιείται ευρέως για την δοκιμή και φιλοξενία ιστοσελίδων, την δημιουργία βάσεων δεδομένων καθώς και την μεταφόρτωση αρχείων και σελίδων.

Μέσω του Xampp θα γίνει χρήση του Apache ώστε να υλοποιηθεί η μεταφόρτωση της ιστοσελίδας. Ο Apache είναι ένα ανοιχτού κώδικα πρόγραμμα εξυπηρετητή (server, διακομιστής ιστού) του παγκόσμιου ιστού (www) και αυτή τη στιγμή είναι ο πιο διαδομένος web server στον κόσμο εξυπηρετώντας τις περισσότερες ιστοσελίδες παγκοσμίως μέσω του πρωτοκόλλου επικοινωνίας HTTP. Ο Apache λοιπόν θα είναι το βασικό εργαλείο το οποίο θα χρησιμοποιηθεί για την μεταφόρτωση - εξυπηρέτηση της ιστοσελίδας σε δίκτυο.

Αρχικά θα πρέπει να γίνει εγκατάσταση των παραπάνω. Το Xampp είναι δωρεάν και βρίσκεται στην ιστοσελίδα **https://www.apachefriends.org/index.html** όπου μπορεί κανείς να το κατεβάσει ελεύθερα και να το εγκαταστήσει στον Η/Υ. Θα πρέπει κατά την εγκατάσταση να επιλεχθούν όλες οι επιλογές που παρέχονται ώστε να εγκατασταθούν όλα τα απαραίτητα εργαλεία (Apache, mySQL κλπ). Η εγκατάσταση γίνεται αυτόματα στο C:\ όπου δημιουργείται ο φάκελος xampp. Ο φάκελος που μας ενδιαφέρει και που θα φιλοξενεί όλα τα αρχεία της ιστοσελίδας είναι ο φάκελος htdocs μέσα στον φάκελο xampp όπου και θα βρίσκονται τα αρχεία ιστοσελίδας PHP, το αρχείο CSS για την διαμόρφωση - παρουσίαση της ιστοσελίδας, τα αρχεία μετρήσεων κλπ.

Κατά την εκκίνηση του XAMPP πατάμε τα κουμπιά Start στο Apache και στο MySQL (κάτω εικόνα). Όπως βλέπουμε και στην εικόνα, ο Apache χρησιμοποιεί το port 80 (το οποίο θα πρέπει να είναι ελεύθερο) για να γίνονται οι λήψεις αιτήσεων για ιστοσελίδες και το port 443 για ασφαλείς συνδέσεις με

κρυπτογράφηση. Η MySQL κάνει χρήση του port 3306. Πατώντας το κουμπί Stop γίνεται και διακοπή της λειτουργίας του server. Ο server μπορεί είτε να εκκινεί χειροκίνητα πατώντας το Start - Stop, είτε να αυτόματα με την εκκίνηση των Windows, κάτι που μπορούμε να το ρυθμίσουμε από τις επιλογές του XAMPP. Αφού έχει εγκατασταθεί και ρυθμιστεί επιτυχώς το σύστημα XAMPP - Apache, μπορούν να μπουν στον φάκελο htdocs τα αρχεία της ιστοσελίδας μας ώστε να λειτουργήσει στην πράξη η ιστοσελίδα και να μεταφορτωθεί στο δίκτυο.

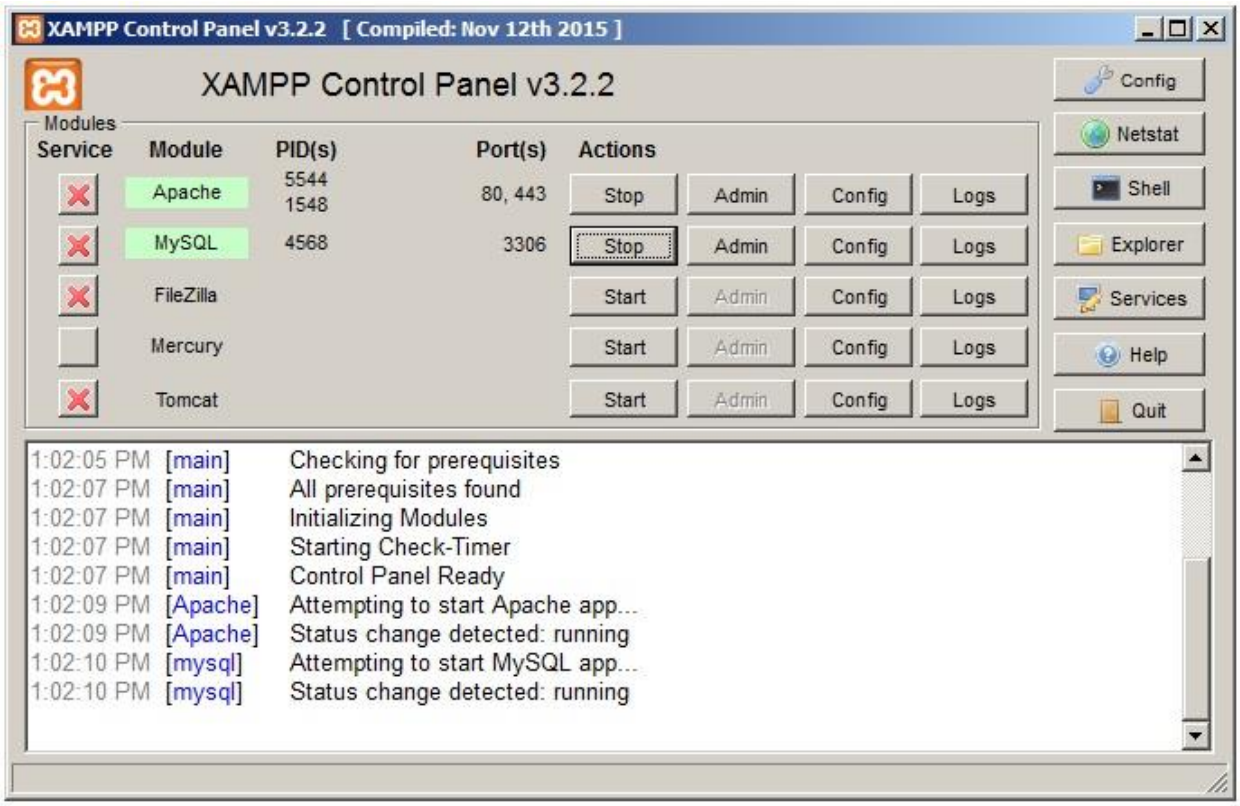


Figure 53 - Περιβάλλον XAMPP

β) Φιλοσοφία υλοποίησης

Η ιστοσελίδα είναι γραμμένη κυρίως σε γλώσσα HTML (αποθηκευμένη σε αρχείο PHP), με κάποια μικρά σημεία γλώσσας PHP και κομμάτια (scripts) από Javascript ενδιάμεσα στον κώδικα. Η βασική δόμηση της ιστοσελίδας γίνεται με τον συνδυασμό HTML - PHP και οι επιπλέον λειτουργίες του διαβάσματος των τιμών και της παρουσίασης των γραφημάτων γίνεται με την Javascript. Τέλος, η απαραίτητη μορφοποίηση ώστε η ιστοσελίδα να έχει την μορφή που βλέπει ο χρήστης γίνεται με την γλώσσα CSS (Cascading Style Sheets). Μέσα στον φάκελο htdocs θα μπούνε όλα τα αρχεία PHP της ιστοσελίδας, το αρχείο μορφοποίησης CSS και δύο φάκελοι, ένας για τις εικόνες που θα υπάρχουν και ένας για τα αρχεία παλαιών μετρήσεων.

Η διαδικασία λήψης δεδομένων από την εφαρμογή και παρουσίασής τους γίνεται με τον εξής τρόπο: Η εφαρμογή εξάγει ανά συγκεκριμένο χρονικό διάστημα αντίστοιχα αρχεία .xml μέσα στον φάκελο της

ιστοσελίδας. Αυτά τα αρχεία, τα διαβάζει η ιστοσελίδα μέσω Javascript με ειδικές τεχνικές που θα περιγραφούν στην συνέχεια και έπειτα καλούνται τα δύο ειδών γραφημάτων Google (μπάρες και μετρητές) από τα google charts. Η ιστοσελίδα μετά, ανανεώνεται κάθε 10 δευτερόλεπτα και σε κάθε αλλαγή βλέπουμε κάθε φορά τις τελευταίες (πιο πρόσφατες) τιμές των μετρήσεων. Τα αρχεία εξάγονται όσο λειτουργεί η διαδικασία εποπτείας στην εφαρμογή. Εαν κλείσει ή σταματήσει, τα αρχεία .xml διαγράφονται από τον φάκελο του website και σε περίπτωση που επιχειρήσουμε να μπούμε στην σελίδα, θα δούμε απλώς μηδενικές τιμές στα γραφήματα.

Σημαντική σημείωση: Εξαιρετικά σημαντικό είναι το γεγονός οτι για να λειτουργήσει σωστά το σύστημα θα πρέπει να οριστεί μέσα από τον κώδικα της εφαρμογής εποπτείας το path για το που θα εξάγονται αυτά τα αρχεία. Αυτό γίνεται σε αρκετά σημεία μέσα στον κώδικα της εφαρμογής, οπότε θα πρέπει κατά την εγκατάσταση του συστήματος από τον προγραμματιστή, προσεκτικά να γίνει εύρεση όλων αυτών των σημείων μέσα στον κώδικα και να αντικατασταθούν από το path του φακέλου της ιστοσελίδας ώστε να εξάγονται εκεί τα προσωρινά αρχεία xml.

γ) Διάβασμα του .xml και δημιουργία γραφημάτων

Όπως αναφέρθηκε οι λειτουργίες ανάγνωσης του .xml που εξάγει η εφαρμογή και δημιουργία γραφημάτων, γίνεται με τη χρήση της γλώσσας Javascript, μέσα σε scripts που προσθέτουμε στον στατικό - βασικό κορμό της ιστοσελίδας. Η φιλοσοφία της παραπάνω διπλής διαδικασίας είναι η εξής:

1. Αρχικά φορτώνονται τα γραφήματα άδεια, στην αρχή του κώδικα της σελίδας καθώς επίσης και οι συναρτήσεις που θα βάλουν τις τιμές στα γραφήματα και θα τα μορφοποιήσουν.
2. Έπειτα μέσα στο σώμα της σελίδας καλείται δεύτερο script στο οποίο δημιουργείται ένας πίνακας στον οποίο θα αποθηκεύονται οι τιμές των αισθητήρων.
3. Στη συνέχεια δημιουργείται XML Http Request και στέλνεται αίτημα (request) στο συγκεκριμένο αρχείο xml που θέλουμε να διαβαστεί και το οποίο έχει εξαχθεί από την εφαρμογή μέσα στον φάκελο της σελίδας.
4. Ακολουθεί ειδική συνάρτηση που διαβάζει κάθε φορά τα τελευταία δεδομένα του .xml ώστε να λαμβάνουμε την πιο πρόσφατη τιμή των αισθητήρων, ανεξάρτητα το πόσες μεσολαβούν.
5. Τα γραφήματα τοποθετούνται μέσα στο σώμα της σελίδας σε συγκεκριμένο σημείο.

Για να κατανοηθεί η τεχνική των παραπάνω και κυρίως του διαβάσματος του αρχείου .xml θα ακολουθήσει ο κώδικας της Javascript με ειδικές επεξηγήσεις - σχόλια. Κάθε σχόλιο αντιστοιχεί στον κώδικα που είναι γραμμένος ακριβώς πάνω από αυτό.

// με αυτές τις γραμμές ανοίγει το πρώτο script γλώσσας Javascript

<script type="text/javascript" src="https://www.gstatic.com/charts/loader.js"></script>
<script type="text/javascript">

//Εδώ φορτώνονται αρχικά τα δύο είδη γραφημάτων Google που θα χρησιμοποιήσουμε (corechart για τις μπάρες και gauge για τους μετρητές). Έπειτα δηλώνονται δύο επανακλήσεις των δύο συναρτήσεων γραφημάτων ώστε να εκτελεστούν όταν φορτωθεί πλήρως το Google Visual API που καλέσαμε ακριβώς απο πάνω.

```
google.charts.load('current', {'packages':['corechart','gauge']});
google.charts.setOnLoadCallback(drawPrwtoChart);
google.charts.setOnLoadCallback(drawDeuteroChart);
```

//Ακολουθεί η συνάρτηση δημιουργίας και ρύθμισης του πρώτου διαγράμματος. Όπως βλέπουμε εισάγονται 4 θέσεις ενός πίνακα στις οποίες θα αποθηκεύουμε τα δεδομένα, οι κωδικοί απο δίπλα είναι διάφορα χρώματα. Ορίζονται το ύψος και το πλάτος του γραφήματος και υπάρχει ειδική εντολή (document.getElementById) ώστε να τοποθετείται μέσα στη σελίδα το γράφημα. Τέλος με τη chart.draw(data, options) σχεδιάζεται το γράφημα.

```
function drawPrwtoChart() {
      var data = google.visualization.arrayToDataTable([
      ["Element", "Celsius", { role: "style" } ],
      ["Sensor 1", pinakas[0], "#ddd97e"],
      ["Sensor 2", pinakas[1], "#94b292"],
      ["Sensor 3", pinakas[2], "#c7988d"],
      ["Sensor 4", pinakas[3], "#8dbdc7"]
    ]);

      var options = {title:'Temperature of 4 sensors',
                     width:700,
                     height:500};

      var chart = new
google.visualization.ColumnChart(document.getElementById('1o_chart_div'));
chart.draw(data, options);

          }
```

//Εδώ είναι ο κώδικας της δεύτερης συνάρτησης, για το δεύτερο γράφημα. Στα options ορίζονται πέρα από τις διαστάσεις του και κάποιες επιλογές σχετικά με τα χρώματα στον μετρητή.

```
function drawDeuteroChart() {
      var data = google.visualization.arrayToDataTable([
        ['Label', 'Value'],
        ['Sensor 1', pinakas[0]],
        ['Sensor 2', pinakas[1]],
        ['Sensor 3', pinakas[2]],
        ['Sensor 4', pinakas[3]]
      ]);
      var options = {
        width: 600, height: 220,
        redFrom: 68, redTo: 75,
        yellowFrom:35, yellowTo: 68,
        minorTicks: 5, max: 75
```

```
        };
        var chart = new google.visualization.Gauge(document.getElementById('2o_chart_div'));
        chart.draw(data, options);

      }
```

//Μέσα στο σώμα της σελίδας υπάρχει ο παρακάτω κώδικας με τα κατάλληλα id ώστε να τοποθετούνται σε αυτό το σημείο τα γραφήματα

```
<table class="columns">
      <div id="1o_chart_div" style="border: 1px solid #ccc"></div>
       <div id="2o_chart_div" style="border: 1px solid #ccc;"></div>
 </table>
```

//Πιο κάτω ανοίγουμε πάλι script και δηλώνουμε τον πίνακα στον οποίο θα αποθηκεύουμε τις τιμές που θα διαβάζουμε και με τον οποίο γεμίζουμε τα γραφήματα. Τέσσερις θέσεις, όσοι και οι αισθητήρες.

```
var pinakas = new Array();
          pinakas[0] = 0;
          pinakas[1] = 0;
          pinakas[2] = 0;
          pinakas[3] = 0;
```

//Δημιουργούμε αντικείμενο xhttp ώστε να γίνει το ειδικό Request - αίτημα για ανάγνωση του αρχείου xml. Η myfunction είναι η συνάρτηση η οποία θα διαβάζει το αρχείο xml και θα καλείται κάθε φορά που θα αλλάζει το ready state του αιτήματος. με το .open ανοίγουμε το συγκεκριμένο αρχείο που έχει εξαχθεί μέσα στον φάκελο της ιστοσελίδας και με το .send γίνεται η αποστολή.

```
var xhttp = new XMLHttpRequest();
xhttp.onreadystatechange = function() {
   if (this.readyState == 4 && this.status == 200) {
      myFunction(this);
   }
};
xhttp.open("GET", "_temp.xml", true);
xhttp.send();
```

//Η βασική συνάρτηση η οποία εφαρμόζει την τεχνική ανάγνωσης του αρχείου .xml. Η τεχνική έχει ως φιλοσοφία να διαβάζει κάθε φορά το τελευταίο σετ τιμών, την τελευταία δηλαδή καταχώρηση του αρχείου .xml. Αρχικά προγραμματίζουμε να αντιλαμβάνεται στοιχεία μέσα στο αρχείο τα οποία έχουν το TagName 'Values'. Με τον πρώτο βρόχο επανάληψης (for), μπαίνουμε μέσα στα στοιχεία που περιλαμβάνονται στο TagName 'Values', σε όλα δηλαδή τα στοιχεία που έχει το αρχείο, εστιάζοντας όμως κάθε φορά στο τελευταίο. Έπειτα για κάθε έναν αισθητήρα, ταχτοποιούμε οτι είναι αυτός

σύμφωνα με το TagName του και με μία αντίστοιχη επανάληψη for, διαβάζουμε την τιμή του. Στη συνέχεια γίνεται καταχώρηση της τιμής που διαβάζουμε στον πίνακα.

```
function myFunction(xml)
{
   var x, i, y, j, xmlDoc, p;
   xmlDoc = xml.responseXML;
   var timi = 0;
   x=xmlDoc.getElementsByTagName('Values');
   for(i=x.length-1; i<x.length; i++)
   {
      y=x[i].getElementsByTagName('Sensor1');
      for (j = y.length-1; j < y.length; j++)
      {
            p = parseInt(y[j].getAttribute('value'))
      pinakas[0] = p;
      }

      y=x[i].getElementsByTagName('Sensor2');
      for (j = y.length-1; j < y.length; j++)
      {
       p = parseInt(y[j].getAttribute('value'))
      pinakas[1] = p;
      }

      y=x[i].getElementsByTagName('Sensor3');

      for (j = y.length-1; j < y.length; j++)
      {
      p = parseInt(y[j].getAttribute('value'))
      pinakas[2] = p;
      }

      y=x[i].getElementsByTagName('Sensor4');

      for (j = y.length-1; j < y.length; j++)
      {
      p = parseInt(y[j].getAttribute('value'))
      pinakas[3] = p;
      }

    }

  }
```

δ) Σύγκριση παλαιότερων μετρήσεων

Όπως περιγράψαμε στο κεφάλαιο λειτουργίας της ιστοσελίδας υπάρχει ειδική επιλογή - σελίδα για την σύγκριση παλαιότερων μετρήσεων σε μη πραγματικό χρόνο. Αυτή η σύγκριση γίνεται με τη μορφή πινάκων που εμφανίζονται όπως είδαμε κάθε φορά που επιλέγουμε κάποιο κουμπί ώστε να ανοίξει κάποιο αρχείο καταγραφής. Τα αρχεία αυτά είναι αρχεία .xml που έχουν τοποθετηθεί μέσα στον υποφάκελο recordings ο οποίος βρίσκεται στον φάκελο της ιστοσελίδας μέσα στο xampp. Ωστόσο δεν αρκεί μόνο η ύπαρξη των αρχείων .xml. Για να μπορούν να εμφανιστούν στην ιστοσελίδα πρέπει (πέρα από το ειδικό αρχείο μορφοποίησης - εμφάνισης CSS) κάθε αρχείο να συνοδεύεται από ένα συμπληρωματικό αρχείο PHP το οποίο είναι μοναδικό. Δηλαδή για το αρχείο recording1.xml, θα υπάρχει και το αντίστοιχο αρχείο recording1.php κ.ο.κ. Παρακάτω θα εξετάσουμε τη δομή ενός συμπληρωματικού αρχείου PHP.

//Εδώ είναι ένα δείγμα κομματιού του κώδικα μέσα στο σώμα του ειδικού αρχείου PHP. Ουσιαστικά δημιουργείται πίνακας ο οποίος φορτώνει το απαραίτητο αρχείο xml και έπειτα - με γλώσσα PHP αυτή τη φορά και όχι Javascript - για κάθε δείγμα (foreach($xml->children() as $Samples) επιστρέφει τις τιμές και τις ώρες καταγραφής τους απο το αρχείο xml.
Αν λοιπόν θέλουμε να προσθέσουμε αρχείο καταγραφής στην ιστοσελίδα, θα πρέπει ο προγραμματιστής που διαχειρίζεται το website να δημιουργεί αντίστοιχα αρχεία PHP για κάθε αρχείο xml και το μόνο που έχει να κάνει είναι να αλλάζει το αρχείο xml που θα γίνεται load, καθώς και τους τίτλους (πχ Sensors) ή και τις διαστάσεις.

```
<table class="table table-bordered table-inverse" style="text-align:center; margin-left:80px; width:5%;">
                <th colspan="3"><center>SENSOR 1</center></th>
                <tr><td>Sample</td><td>°C</td><td>Ώρα</td></tr>
                <?php
                $xml=simplexml_load_file("recording1.xml") or die("Error: Cannot create object");
                foreach($xml->children()  $Samples) {
                        echo "<tr><td></td><td>";
                        echo $Samples->Sensor1['value'];
                        echo "</td><td>";
                        echo $Samples->Sensor1['Time'];
                        echo "</td></tr>";
                }
                ?>
        </table>
```

6. Επόμενα βήματα

Η ολοκλήρωση της πτυχιακής είναι και η ολοκλήρωση μιας πρώτης πρακτικής εφαρμογής του συστήματος τηλεμετρίας - εποπτείας. Μελετήθηκε και υλοποιήθηκε στην πράξη η μέθοδος συλλογής κρίσιμων δεδομένων, η απομακρυσμένη αποστολή τους και η αναλυτική παρουσίασή τους. Ωστόσο ένα τέτοιο σύστημα είναι ένα δυναμικό πλαίσιο με πολλές δυνατότητες που μπορεί να να εμπλουτιστεί - επεκταθεί και να αποτελέσει εφαλτήριο για μελλοντικές μελέτες, έρευνες και δοκιμές που θα το αναπτύξουν ακόμα περισσότερο.

Στο πλαίσιο των παραπάνω, πρέπει αρχικά να αναφερθεί το γεγονός οτι οι αισθητήρες ροής που δοκιμάστηκαν για τις ανάγκες της πτυχιακής εργασίας, αν και λειτούργησαν με επιτυχία, είναι σχετικά φθηνοί αισθητήρες ροής νερού που πέρα από την μέτρια αξιοπιστία των υλικών τους έχουν και αρκετά μεγάλο σφάλμα. Συνεπώς ένα πρώτο μελλοντικό πεδίο για δοκιμές είναι η δοκιμή και η πιθανή προσθήκη νέων αισθητήρων ροής που θα παρέχουν αξιοπιστία ως προς τις μετρήσεις, αλλά και ως προς την ζωή τους ως εργαλεία - όργανα. Τέτοιοι αισθητήρες ροής είναι διάφοροι τύποι βιομηχανικών αισθητήρων τύπου Food Grade σχεδιασμένοι ειδικά για εφαρμογές στη βιομηχανία τροφίμων και στην παραγωγή, αλλά και στην μεταποίηση. Αυτοί οι αισθητήρες έχουν σαφώς πολύ μεγαλύτερο κόστος από τους αισθητήρες οι οποίοι δοκιμάστηκαν για τις ανάγκες της πτυχιακής εργασίας αλλά η φιλοσοφία λειτουργίας τους μπορεί να είναι η ίδια (αισθητήρες τουρμπινας). Φυσικά υπάρχουν αισθητήρες ροής με ακόμα πιο εξελιγμένες τεχνολογίες όπως ηλεκτρομαγνητικοί ή υπερήχων. Η διαδικασία με την οποία θα τους διαβάζουμε δεν είναι απαραίτητο να αλλάξει πλήρως αφού όλοι οι αισθητήρες για ανώτερες βιομηχανικές εφαρμογές εξάγουν συνήθως παλμούς, συνεπώς μπορούν εύκολα με μικρές πιθανές τροποποιήσεις να δοκιμαστούν ή και να εγκατασταθούν στο Ελαιουργείο με βάση ό,τι έχει ήδη υλοποιηθεί και για τους πιο απλούς

Ακόμα ένα άλλο πεδίο μελλοντικής επέκτασης του συστήματος είναι η επεξεργασία των τιμών ώστε να εξάγονται αποτελέσματα. Με τροποποιήσεις στον κώδικα θα μπορεί η εφαρμογή εποπτείας να εφαρμόζει επεξεργαστικές λειτουργίες σε υπάρχουσες μετρήσεις και να υπολογίζει - εξάγει συγκεκριμένα αποτελέσματα ως προς διάφορους τομείς που θα ενδιαφέρουν τον χρήστη.

Τέλος πολύ σημαντικός τομέας είναι αυτός της δυνατότητας ελέγχου της μονάδας παραγωγής απευθείας από την εφαρμογή. Πρόκειται ουσιαστικά για την επέκταση από μονάδα μη επεμβατικής εποπτείας, σε μονάδα πλήρους αυτοματισμού. Αυτό θα μπορούσε να περιλαμβάνει ειδικές πλακέτες που θα ελέγχονται από τον μικροελεγκτή ή και από την εφαρμογή στον Η/Υ και οι οποίες ανάλογα με τις τιμές των αισθητήρων που θα λαμβάνουν θα ελέγχουν με ρελέ, ή με διακόπτες κ.α., τα επιμέρους στάδια της παραγωγής. Έτσι θα μπορούσε η διαδικασία να σταματάει αυτόματα αν διαπιστωθεί μεγάλη θερμοκρασία, ή να σταματάει η παροχή νερού έαν διαπιστωθεί κάποια διαρροή, ή ακόμα να προγραμματίζεται το πότε θα ξεκινάει ή θα σταματάει ο κύκλος εργασιών του Ελαιουργείου.

7. Βιβλιογραφία

Συγγράμματα

1. "Ανάπτυξη εφαρμογών με το Arduino" - Λιώνης Σ.Π., Παπάζογλου Π. / Εκδόσεις Τζιόλα 2015
2. "Τηλεπικοινωνίες και Δίκτυα Υπολογιστών", - Αλεξόπουλος Α., Λαγογιάννης Γ. / Εκδόσεις Γιαλός 2016
3. "Αισθητήρες Μέτρησης και Ελέγχου" - Καλοβρέκτης Κωνσταντίνος, / Εκδόσεις Τζιόλα 2012

Ιστοσελίδες

1. https://www.arduino.cc - (Πληροφορίες για τον Arduino)
2. https://github.com/angeloc/simplemodbusng - (Η βιβλιοθήκη MODBUS - RTU για Arduino)
3. https://www.w3schools.com - (Μαθήματα και παραδείγματα Javascript, PHP, XML)
4. https://www.codecademy.com - (Εκμάθηση γλωσσών προγραμματισμού και κατασκευής ιστοσελίδων)