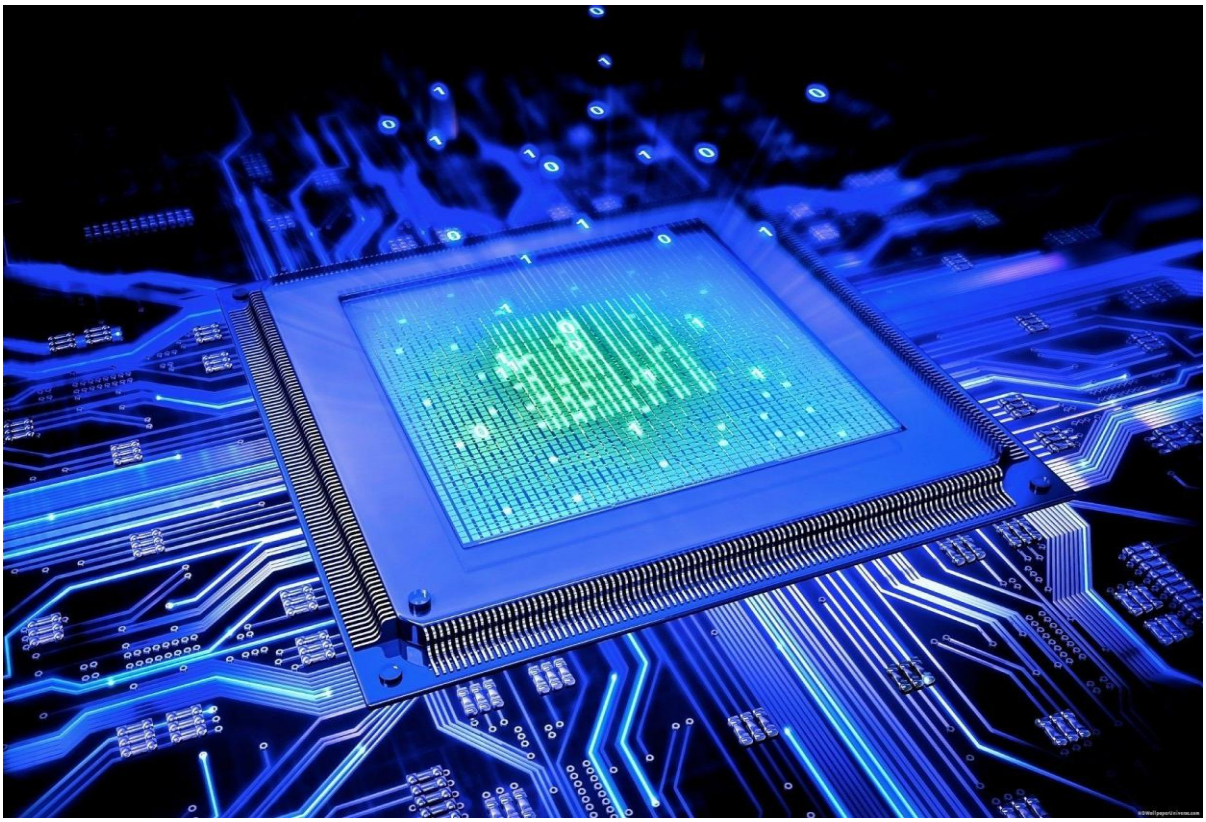
ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΔΥΤΙΚΗΣ ΕΛΛΑΔΑΣ
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ Τ.Ε.

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ 1533



Ανάπτυξη Συστήματος βασισμένο στο Arduino για την Συλλογή, την Διαχείριση και την Αποθήκευση Δεδομένων σε μία Βάση Δεδομένων μέσω Διαδικτύου

ΣΠΟΥΔΑΣΤΗΣ: ΣΤΑΥΡΟΣ ΜΠΟΛΕΤΗΣ (6573)

ΕΠΟΠΤΕΥΩΝ ΚΑΘΗΓΗΤΗΣ: ΑΘΑΝΑΣΙΟΣ ΚΑΛΑΝΤΖΟΠΟΥΛΟΣ

ΠΑΤΡΑ 2016

Πρόλογος

Η παρούσα πτυχιακή εργασία είχε ως αντικείμενο τον σχεδιασμό και την ανάπτυξη ενός συστήματος συλλογής, αποθήκευσης και επεξεργασίας περιβαλλοντικών μετρήσεων. Το συγκεκριμένο σύστημα έχει την δυνατότητα να αποθηκεύει τις ληφθείσες μετρήσεις σε μια SD (Secure Digital) κάρτα και να τις αποστέλλει σε μια MySQL βάση δεδομένων. Επίσης παρέχεται στον χρήστη η δυνατότητα να διαχειριστεί απομακρυσμένα τις μετρήσεις που αποθηκεύονται στην βάση δεδομένων είτε μέσω του GUI που δημιουργήθηκε είτε μέσω του MySQL Administrator. Η ανάπτυξη του συγκεκριμένου συστήματος στηρίχθηκε στην πλατφόρμα Arduino Mega2560 η οποία διαθέτει τον μικροελεγκτή ATmega 2560 αξιοποιώντας το Arduino IDE. Το GUI αναπτύχθηκε με το λογισμικό πακέτο LabVIEW ενώ για την διαμόρφωση της MySQL βάσης δεδομένων χρησιμοποιήθηκε το λογισμικό MySQL Administrator.

Στο πρώτο κεφάλαιο της παρούσας πτυχιακής εργασίας, γίνεται η αναφορά σε βασικές έννοιες που απαιτούνται για την ανάπτυξη του συγκεκριμένου συστήματος. Στο δεύτερο κεφάλαιο αναλύεται το υλικό που χρησιμοποιήθηκε για την ανάπτυξη του συστήματος και τα χαρακτηριστικά του μικροελεγκτή ATmega2560. Στο τρίτο κεφάλαιο γίνεται μία σύντομη περιγραφή των λογισμικών πακέτων που χρησιμοποιήθηκαν για την ανάπτυξη του προγράμματος στον μικροελεγκτή, για την ανάπτυξη του γραφικού περιβάλλοντος καθώς και για την διαμόρφωση της βάσης δεδομένων. Στο τέταρτο κεφάλαιο αναλύονται οι βιβλιοθήκες με τις συναρτήσεις που υλοποιήθηκαν για τον έλεγχο του πρωτοκόλλου επικοινωνίας I2C (Inter Integrated Circuit) και του RTC (Real Time Clock) DS3231. Το πρόγραμμα που δημιουργήθηκε για τον ATmega2560 μαζί με την εφαρμογή του γραφικού περιβάλλοντος που αναπτύχθηκε στο LabVIEW αναλύονται στο πέμπτο κεφάλαιο. Τέλος, στο έκτο κεφάλαιο παρουσιάζονται τα αποτελέσματα που προέκυψαν κατά την λειτουργία του συστήματος καθώς τα συμπεράσματα που προέκυψαν. Επίσης στο κεφάλαιο αυτό δίνονται κάποιες προτάσεις στα πλαίσια της βελτίωσης του συστήματος και της μελλοντικής επέκτασής του.

Στο σημείο αυτό θα ήθελα να ευχαριστήσω τον επιβλέπων καθηγητή μου κ. Αθανάσιο Καλαντζόπουλο για την τεράστια υποστήριξη, την καθοδήγηση και τις πολύτιμες συμβουλές που μου προσέφερε κατά την διεκπεραίωση της πτυχιακής εργασίας μου. Επίσης θα ήθελα να ευχαριστήσω τον κ. Ευάγγελο Ζυγούρη αναπλ. καθηγητή του Τμήματος Φυσικής του Πανεπιστημίου Πατρών για την καθοδήγηση και τις πολύτιμες συμβουλές που μου προσέφερε. Ακόμη, θα ήθελα να ευχαριστήσω όλους τους καθηγητές του Τμήματος Ηλεκτρολόγων Μηχανικών Τ.Ε για την καθοδήγηση και τις γνώσεις που μου μετάδωσαν όλα αυτά τα χρόνια. Τέλος, ευχαριστώ τους γονείς μου, τον αδερφό μου, τους φίλους μου και την κοπέλα μου που με στήριξαν με κάθε τρόπο σε όλη την διάρκεια των σπουδών μου και συνεχίζουν να με στηρίζουν σε κάθε βήμα της ζωής μου.

Περίληψη

Το σύστημα που αναπτύχθηκε στην παρούσα πτυχιακή εργασία αναλαμβάνει την συλλογή, την αποθήκευση, την διαχείριση, την επεξεργασία και την αποστολή δεδομένων. Τα δεδομένα αυτά προέρχονται από μια σειρά αισθητήρων για την λήψη περιβαλλοντολογικών μετρήσεων όπως θερμοκρασίας, φωτεινότητας κλπ. Για την λήψη των μετρήσεων και τον έλεγχο των αισθητήρων μέσω της πλατφόρμας Arduino Mega 2560 αξιοποιείται το πρωτόκολλο επικοινωνίας I2C (Inter Integrated Circuit). Οι παραπάνω μετρήσεις αποθηκεύονται ως αρχεία με βάση την ημερομηνία στην κάρτα SD (Secured Digital) που διαθέτει το Ethernet shield του Arduino ενώ ταυτόχρονα αποστέλλονται μέσω διαδικτύου σε μία MySQL βάση δεδομένων. Σε περίπτωση που χαθεί η σύνδεση στο διαδίκτυο δημιουργούνται προσωρινά αρχεία στην κάρτα SD που περιέχουν την πληροφορία που χρειάζεται, έτσι ώστε όταν επανέλθει η σύνδεση να αποσταλούν όλες οι μετρήσεις που δεν είχαν αποσταλεί. Για την αποθήκευση και καταγραφή των μετρήσεων ανά τακτά χρονικά διαστήματα καθώς και την καταγραφή της αντίστοιχης χρονικής στιγμής αξιοποιείται ένα RTC (Real-Time Clock) το οποίο συγχρονίζεται με έναν NTP (Network Time Protocol) Server ώστε να ελέγχεται η ορθότητα της ώρας. Επιπρόσθετα, το συγκεκριμένο σύστημα διαθέτει μια LCD (Liquid Crystal Display) οθόνη ώστε να είναι εφικτή η προβολή μηνυμάτων σχετικά με την λειτουργία του συστήματος. Τα δεδομένα που συλλέγονται από ένα ή περισσότερα τέτοια συστήματα μεταφέρονται μέσω Ethernet από την βάση δεδομένων σε ένα GUI (Graphical User Interface) που έχει αναπτυχθεί με το LabVIEW. Μέσω του GUI ο χρήστης έχει την δυνατότητα να παρατηρεί τα παραπάνω δεδομένα μέσω κατάλληλα διαμορφωμένων γραφικών παραστάσεων.

Περιεχόμενα

Κεφάλαιο 1

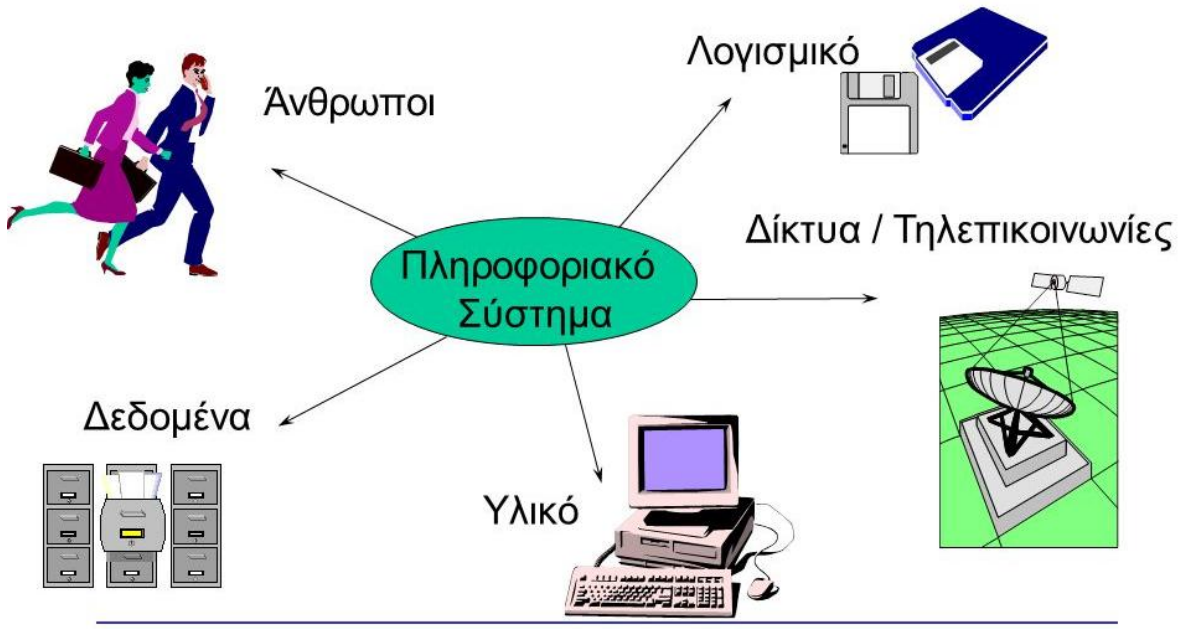
Εισαγωγή

1.1 Ιστορική αναδρομή των πληροφοριακών συστημάτων

Η ιστορία και η εξέλιξη των πληροφοριακών συστημάτων συνδέεται άμεσα με την ιστορία της επιστήμης των υπολογιστών, η οποία ξεκίνησε αρκετά πιο πριν από τον εικοστό αιώνα και την σύγχρονη επιστήμη των υπολογιστών. Με την πάροδο του χρόνου δημιουργούνται συνέχεια καινοτόμα συστήματα, που βασίζονται είτε σε ήδη υπάρχοντα είτε σε νέες ιδέες, με σκοπό την εξασφάλιση της ακεραιότητας των δεδομένων και την βελτίωση της αποτελεσματικότητας και αποδοτικότητας της όλης διαδικασίας. Τα πληροφοριακά συστήματα έχουν αναπτυχθεί στο χώρο των οργανισμών, κυρίως των επιχειρήσεων με απώτερο όφελος στην κοινωνία.

1.2 Συστήματα συλλογής, αποθήκευσης και επεξεργασίας δεδομένων

Πληροφοριακά συστήματα ονομάζονται ένα σύνολο διαδικασιών, ανθρώπινου δυναμικού και αυτοματοποιημένων υπολογιστικών συστημάτων, που προορίζονται για τη συλλογή, εγγραφή, ανάκτηση, επεξεργασία, αποθήκευση και ανάλυση πληροφοριών. Τα συστήματα αυτά περιλαμβάνουν το λογισμικό (software), το υλικό (hardware) και σε πολλές περιπτώσεις διαθέτουν κάποιο πρόσθετο υλικό για την επικοινωνία και δικτύωση. Τα πληροφοριακά συστήματα (Σχήμα 1.1) αποτελούν το μέσο για την αρμονική συνεργασία ανθρώπινου δυναμικού, δεδομένων, διαδικασιών και τεχνολογιών πληροφορίας και επικοινωνιών.



Σχήμα 1.1: Τα στοιχεία ενός πληροφοριακού συστήματος.

Ο σύγχρονος τρόπος ζωής τείνει να γίνεται όλο και πιο απαιτητικός στην μαζική συλλογή, αποθήκευση και επεξεργασία δεδομένων με σκοπό την διευκόλυνση της ζωής του ανθρώπου. Ο λόγος αυτός καθιστά τα συστήματα αυτά τόσο σημαντικά ώστε η καθημερινότητά μας να είναι εξαρτώμενη σε ένα μεγάλο βαθμό από τέτοια συστήματα.

Μερικά παραδείγματα τέτοιων συστημάτων είναι :

- συστήματα αποθήκευσης δεδομένων
- προγραμματισμού παραγωγής και υλικών
- συστήματα διαχείρισης επιχειρήσεων
- μηχανών αναζήτησης
- γεωγραφικό σύστημα πληροφοριών
- παγκόσμιο σύστημα πληροφοριών
- αυτοματισμού γραφείου

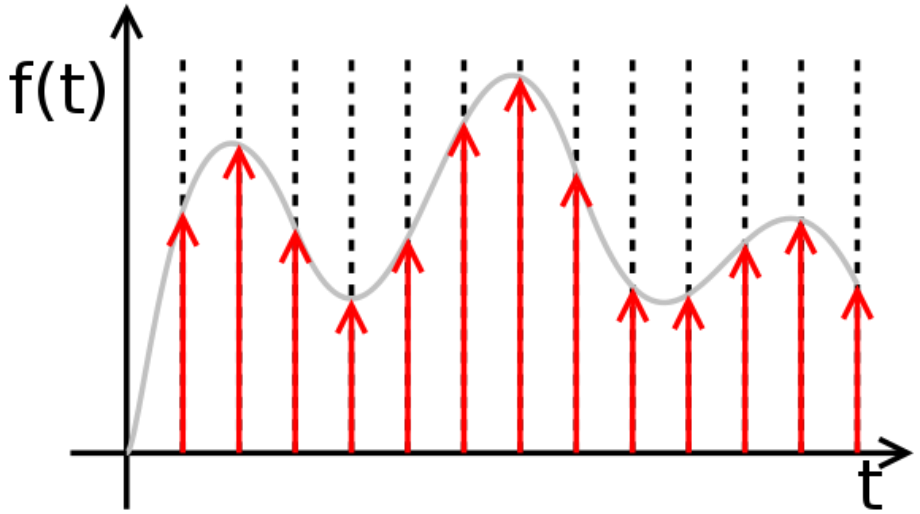
Ένα πληροφοριακό σύστημα ουσιαστικά χρησιμοποιεί την τεχνολογία των υπολογιστών για την εκτέλεση μιας σειράς προγραμματιζόμενων εργασιών με σκοπό τον πλήρες ή μερικό έλεγχο των αντίστοιχων διαδικασιών. Τα βασικά τμήματα ενός τέτοιου συστήματος είναι:

- **Υλικό** - αυτές είναι οι συσκευές όπως η οθόνη, επεξεργαστή, τον εκτυπωτή και το πληκτρολόγιο, τα οποία συνεργάζονται για να δεχθούν, επεξεργαστούν και να παρουσιάσουν τα στοιχεία και τις πληροφορίες.
- **Λογισμικό** - είναι τα προγράμματα που επιτρέπουν στο υλικό να επεξεργαστεί τα δεδομένα.
- **Βάσεις Δεδομένων** - είναι προγράμματα που επιτρέπουν την συγκέντρωση και διαχείριση των σχετιζόμενων δεδομένων.
- **Δίκτυα** - είναι τηλεπικοινωνιακά συστήματα που επιτρέπουν την διασύνδεση διάφορων υπολογιστών ή άλλων συστημάτων με σκοπό την κατανομή των πόρων.
- **Διαδικασίες** - είναι προγράμματα που αναλαμβάνουν το συνδυασμό και την αξιοποίηση των ανωτέρω τμημάτων με σκοπό την επεξεργασία των πληροφοριών και την εξαγωγή αποτελεσμάτων ή την λήψη αποφάσεων.

Τα πληροφοριακά συστήματα καλούνται να ενημερώνουν τον άνθρωπο ή και ακόμα να παίρνουν αποφάσεις στην θέση του με μεγάλη επιτυχία. Ανάλογα την σημαντικότητα του συστήματος, τις αποφάσεις που καλείται να πάρει και το περιβάλλον που θα λειτουργεί πρέπει να υπάρχει όσον δυνατόν γίνεται καλύτερη απόκριση ώστε να αποφεύγονται σφάλματα που οδηγούν σε ατυχήματα ή σε λανθασμένες ενημερώσεις.

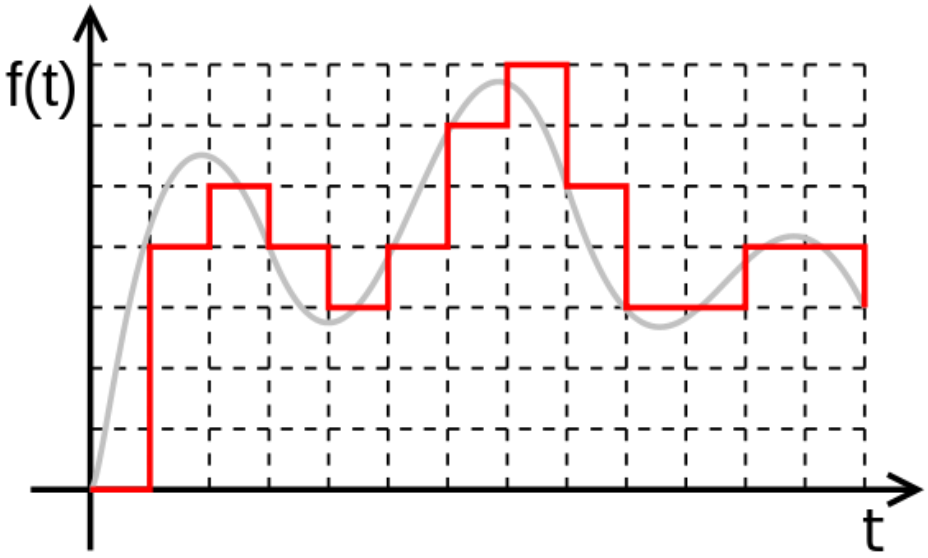
1.3 Αναλογικά και ψηφιακά σήματα

Αναλογικό σήμα είναι μια χρονικά μεταβαλλόμενη τιμή δεδομένων που μπορεί να εκφραστεί από μία μαθηματική συνάρτηση έχοντας τον χρόνο ως ανεξάρτητη μεταβλητή και την τιμή του σήματος ως εξαρτημένη. Σήμα διακριτού χρόνου (Σχήμα 1.2) είναι το αποτέλεσμα που παίρνουμε εάν από το αναλογικό σήμα παίρνουμε δεδομένα ανά τακτά διαστήματα και όχι συνεχόμενα.



Σχήμα 1.2: Διακριτό σήμα

Ως ψηφιακό σήμα μπορεί να οριστεί ένα σήμα διακριτού χρόνου το οποίο παίρνει συγκεκριμένες τιμές. Επίσης ως ψηφιακό σήμα (Σχήμα 1.3) μπορεί να ορισθεί μια κυματομορφή ενός σήματος συνεχούς χρόνου το οποίο μπορεί να αναπαρασταθεί ως μια αλληλουχία από bits.



Σχήμα 1.3: Ψηφιακό σήμα

1.4 Ψηφιακή επεξεργασία σήματος

Η ψηφιακή επεξεργασία σήματος έχει ως αντικείμενο την αναπαράσταση σημάτων διακριτού χρόνου καθώς και την επεξεργασία αυτών. Μερικές εφαρμογές της ψηφιακής επεξεργασίας είναι η ψηφιακή επεξεργασία εικόνας, ήχου, αναγνώρισης φωνής, η επεξεργασία σημάτων από αισθητήρες και ο έλεγχος συστημάτων. Βασικό στάδιο της ψηφιακής επεξεργασίας σήματος είναι η μετατροπή του αναλογικού σήματος σε ψηφιακό μέσω δειγματοληψίας και κβάντισης αξιοποιώντας ένα μετατροπέα αναλογικού σήματος σε ψηφιακό. Η αντίστροφη διαδικασία, δηλαδή η μετατροπή από ψηφιακό σε αναλογικό σήμα,

είναι εξίσου σημαντική γιατί τις περισσότερες φορές το σήμα εξόδου είναι αναλογικό παρόλο που η όλη διαδικασία είναι ψηφιακού χαρακτήρα. Οι αλγόριθμοι που υλοποιούν την επιθυμητή ψηφιακή επεξεργασία εκτελούνται κατά κύριο λόγο σε υπολογιστές καθώς και σε ισχυρούς μικροεπεξεργαστές γενικής χρήσης ή σε επεξεργαστές ψηφιακούς σήματος (Digital Signal Processors, DSPs).

Λόγο της αυξανόμενης χρήσης των ηλεκτρονικών υπολογιστών χρησιμοποιείται όλο και πιο πολύ η ψηφιακή επεξεργασία σήματος. Η μετατροπή ενός αναλογικού σήματος σε ψηφιακό γίνεται μέσω της δειγματοληψίας και της κβάντισης. Για την μετατροπή του σήματος σε διακριτές τιμές το σήμα χωρίζεται σε κλάσεις ισοδυναμίας. Ο κβαντισμός προκύπτει από την αντικατάσταση των τιμών του σήματος που αναπαριστά την κλάση ισοδυναμίας του. Βάση του θεωρήματος δειγματοληψίας Nyquist-Shannon ένα σήμα μπορεί να ανακατασκευαστεί με ακρίβεια από τα δείγματά του αν η συχνότητα δειγματοληψίας είναι μεγαλύτερη ή ίση με την υψηλότερη συχνότητα του σήματος κάτι τέτοιο όμως απαιτεί άπειρα δείγματα. Στην πράξη η συχνότητα δειγματοληψίας θα πρέπει να είναι αρκετά μεγαλύτερη από την διπλάσια συχνότητα που περιέχει το εύρος συχνοτήτων του σήματος.

1.5 Μικροελεγκτές

Μικροελεγκτής (microcontroller) είναι ένας τύπος μικροεπεξεργαστή ο οποίος λόγω των πολλαπλών ενσωματωμένων υποσυστημάτων που διαθέτει μπορεί να λειτουργήσει με ελάχιστα εξωτερικά εξαρτήματα. Έχει μεγάλη χρήση κυρίως σε ενσωματωμένα συστήματα χαμηλού και μεσαίου κόστους όπως αυτοματισμούς, ηλεκτρικές συσκευές, ηλεκτρονικά προϊόντα και κάθε είδους αυτοκινούμενα τροχοφόρα οχήματα.

Οι διαφορές που έχουν οι μικροελεγκτές από τους μικροεπεξεργαστές είναι πως στους σύγχρονους μικροεπεξεργαστές για μη ενσωματωμένα συστήματα (π.χ οι επεξεργαστές των προσωπικών υπολογιστών) δίνεται έμφαση στην υπολογιστική ισχύ. Έτσι η λειτουργικότητα του τελικού συστήματος εξαρτάται από τα εξωτερικά περιφερειακά που είναι διασυνδεμένα με την κεντρική μονάδα που δεν είναι εξειδικευμένη. Από την άλλη οι μικροελεγκτές των ενσωματωμένων συστημάτων εκτός από περιορισμένη υπολογιστική ισχύ έχουν και μικρότερες δυνατότητες διασύνδεσης με εξωτερικά περιφερειακά. Παρόλα αυτά οι μικροελεγκτές αναπτύχτηκαν με στόχο το χαμηλό κόστος, την εξειδίκευση και την χαμηλή κατανάλωση η οποία οφείλεται στην μειωμένη υπολογιστική ισχύ τους και στον μικρό αριθμό ολοκληρωμένων κυκλωμάτων που απαιτείται.

Αναλυτικά, τα πλεονεκτήματα των μικροελεγκτών είναι:

- Αυτονομία, μέσω της ενσωμάτωσης σύνθετων περιφερειακών υποσυστημάτων όπως μνήμες και θύρες επικοινωνίας. Έτσι πολλοί μικροελεγκτές δεν χρειάζονται κανένα άλλο ολοκληρωμένο κύκλωμα για να λειτουργήσουν.
- Η ενσωμάτωση περιφερειακών που σημαίνει ευκολότερη υλοποίηση εφαρμογών λόγω των απλούστερων διασυνδέσεων. Επίσης, οδηγεί σε χαμηλότερη κατανάλωση ισχύος, μεγιστοποιώντας τη φορητότητα και ελαχιστοποιεί το κόστος της συσκευής στην οποία ενσωματώνεται ο μικροελεγκτής.
- Χαμηλό κόστος.

- Μεγαλύτερη αξιοπιστία λόγω των λιγότερων διασυνδέσεων.
- Μειωμένες εκπομπές ηλεκτρομαγνητικών παρεμβολών και μειωμένη ευαισθησία σε αντίστοιχες παρεμβολές από άλλες ηλεκτρικές και ηλεκτρονικές συσκευές. Το πλεονέκτημα αυτό προκύπτει από το μικρότερο αριθμό και μήκος εξωτερικών διασυνδέσεων καθώς και τις χαμηλότερες ταχύτητες λειτουργίας.
- Περισσότεροι διαθέσιμοι ακροδέκτες για ψηφιακές εισόδους-εξόδους (για δεδομένο μέγεθος ολοκληρωμένου κυκλώματος), λόγω της μη δέσμευσής τους για τη σύνδεση εξωτερικών περιφερειακών.
- Μικρό μέγεθος συνολικού υπολογιστικού συστήματος.

Η βασική αρχιτεκτονική των μικροελεγκτών δεν διαφέρει από αυτή των κοινών μικροεπεξεργαστών, αν και στους πρώτους απαντάται συχνά η αρχιτεκτονική μνήμης τύπου Harvard, η οποία χρησιμοποιεί διαφορετικούς δίαυλους διασύνδεσης (bus) της μνήμης προγράμματος και της μνήμης δεδομένων (πχ οι σειρές AVR από την Atmel και PIC από την Microchip). Στους κοινούς μικροεπεξεργαστές συνηθίζεται η ενιαία διάταξη μνήμης τύπου von Neumann.

1.5.1 Υποσυστήματα μικροελεγκτών

Στον μικροεπεξεργαστή, το ολοκληρωμένο κύκλωμα που τον αποτελεί περιέχει μόνο την Λογική και Αριθμητική Μονάδα (Arithmetic Logic Unit, ALU), στοιχειώδεις καταχωρητές (registers), προσωρινή μνήμη RAM (Random Access Memory) πολύ υψηλής ταχύτητας (cache memory) και, κάποιες φορές, τον ελεγκτή μνήμης (memory controller). Όμως, για τη λειτουργία ενός πλήρους ενσωματωμένου υπολογιστικού συστήματος, απαιτούνται πολλά εξωτερικά υποσυστήματα και περιφερειακά:

- Κύκλωμα συνδετικής λογικής (glue logic) για τη σύνδεση των εξωτερικών μνημών και άλλων περιφερειακών παράλληλης σύνδεσης στο bus δεδομένων του επεξεργαστή.
- Μνήμη προγράμματος (τύπου ROM, ή FLASH, ή EPROM κλπ) η οποία περιέχει το λογισμικό του συστήματος. Σε κάποια μοντέλα, είναι δυνατό το κλείδωμα αυτής της μνήμης, μετά την εγγραφή της, ώστε να προστατευτεί το περιεχόμενό της από πιθανή αντιγραφή.
- Μεγάλο μέγεθος μνήμης RAM.
- Μόνιμη μνήμη αποθήκευσης παραμέτρων λειτουργίας (τύπου Electrically Erasable Programmable Read-Only Memory, EEPROM ή Non-Volatile Random-Access Memory, NVRAM). Αυτή η μνήμη έχει, έναντι της FLASH, το πλεονέκτημα της δυνατότητας διαγραφής και εγγραφής οποιουδήποτε μεμονωμένου byte.
- Κύκλωμα αρχικοποίησης (reset).
- Ελεγκτής διαχείρισης των αιτήσεων διακοπής από τα περιφερειακά (interrupt request controller).
- Κύκλωμα επιτήρησης τροφοδοσίας (brown-out detection) το οποία παρακολουθεί την τροφοδοσία και αρχικοποιεί ολόκληρο το σύστημα όταν αυτή πέσει κάτω από τα επιτρεπόμενα όρια, προλαμβάνοντας έτσι την αλλοίωση των δεδομένων.
- Κύκλωμα επιτήρησης λειτουργίας (watchdog timer) το οποίο αρχικοποιεί το σύστημα, αν αυτό εμφανίσει σημάδια δυσλειτουργίας λόγω κολλήματος (hang).

- Τοπικό ταλαντωτή για την παροχή παλμών χρονισμού (clock).
- Έναν ή περισσότερους χρονιστές-απαριθμητές υψηλής ταχύτητας (hardware timer-counter) για τη δημιουργία καθυστερήσεων, μέτρηση διάρκειας γεγονότων, απαρίθμηση γεγονότων και άλλων λειτουργιών ακριβούς χρονισμού.
- Ρολόι πραγματικού χρόνου (Real Time Clock, RTC) το οποίο τροφοδοτείται από ανεξάρτητη μπαταρία και για αυτό πρέπει να έχει πολύ χαμηλή κατανάλωση ρεύματος.
- Σειρά παράλληλων ψηφιακών εισόδων και εξόδων (Parallel Input-Output, PIO).

Γενικά, όλες οι οικογένειες μικροελεγκτών ενσωματώνουν τα περισσότερα από τα παραπάνω περιφερειακά, με διαφοροποιήσεις κυρίως στην ύπαρξη ή μη εσωτερικής μνήμης προγράμματος και στο είδος της. Έτσι, υπάρχουν:

- Μικροελεγκτές χωρίς μνήμη προγράμματος, οι οποίοι χαρακτηρίζονται ως ROM-less. Αυτοί παρέχουν πάντοτε ένα παράλληλο bus δεδομένων, πάνω στον οποίο συνδέονται οι εξωτερικές μνήμες προγράμματος και η μνήμη RAM. Τέτοιοι τύποι μικροελεγκτών προορίζονται για πιο ισχυρά υπολογιστικά συστήματα ελέγχου, με μεγαλύτερες απαιτήσεις μνήμης.
- Μικροελεγκτές με μνήμη ROM, που η μνήμη τους μπορεί να γραφεί μόνο μία φορά.
- Μικροελεγκτές με μνήμη FLASH, η οποία μπορεί να προγραμματιστεί πολλές φορές. Αυτή είναι η πιο διαδεδομένη κατηγορία. Συχνά ο προγραμματισμός της μνήμης μπορεί να γίνει ακόμη και πάνω στο κύκλωμα της ίδιας της ενσωματωμένης εφαρμογής (δυνατότητα In Circuit Programming, ISP). Αυτοί οι μικροελεγκτές έχουν ουσιαστικά αντικαταστήσει τους παλαιότερους τύπους EPROM που έσβηναν με υπεριώδη ακτινοβολία.

Ανάλογα με την εφαρμογή για την οποία προορίζεται ένας μικροελεγκτής, μπορεί να περιέχει και:

- Μία ή περισσότερες ασύγχρονες σειριακές θύρες επικοινωνίας (Universal Asynchronous Receiver Transmitter, UART).
- Σύγχρονες σειριακές θύρες επικοινωνίας (πχ Inter-Integrated Circuit-I2C, Serial Peripheral Interface-SPI, Ethernet).
- Ολόκληρα υποσυστήματα για την άμεση υποστήριξη από υλικολογισμικό (firmware) των πιο σύνθετων πρωτοκόλλων επικοινωνίας όπως Controller Area Network-CAN, High-Level Data Link Control-HDLC, Integrated Services Digital Network-ISDN, Asymmetric Digital Subscriber Line-ADSL.
- Μονάδα άμεσης εκτέλεσης πράξεων κινητής υποδιαστολής (Floating Point Processing Unit, FPU), η οποία είναι πάντοτε πιο γρήγορη από την ALU του επεξεργαστή. Τέτοιες μονάδες χαρακτηρίζουν τους μικροελεγκτές με δυνατότητες ψηφιακής επεξεργασίας σήματος. Τα τελευταία χρόνια, με την ευρύτατη διάδοση των φορητών συσκευών ήχου και εικόνας, παρατηρείται μια τάση σύγκλισης των μικροελεγκτών με τους DSPs.
- Έναν ή περισσότερους μετατροπείς αναλογικού σήματος σε ψηφιακό (Analog to Digital Converters, ADCs).
- Έναν ή περισσότερους μετατροπείς ψηφιακού σε αναλογικό σήμα (Digital to Analog Converters, DACs).

- Ελεγκτή οθόνης υγρών κρυστάλλων (Liquid Crystal Display, LCD).
- Υποσύστημα προγραμματισμού πάνω στο κύκλωμα (τύπου ISP). Χάρη σε αυτό το κύκλωμα, είναι δυνατός ο επαναπρογραμματισμός (αναβάθμιση λογισμικού) της εφαρμογής, συνδέοντας στη συσκευή μια εξωτερική συσκευή προγραμματισμού (συνήθως σε θύρα UART RS-232) ή ακόμη και από το διαδίκτυο. Αυτή η δυνατότητα απαιτεί την ύπαρξη λογισμικού υποδοχής (bootstrap) μέσα στη μνήμη προγράμματος.
- Υποσύστημα προγραμματισμού και διάγνωσης που συνήθως ακολουθεί το πρότυπο Joint Test Action Group-JTAG . Χάρη σε αυτό, είναι δυνατός ο προγραμματισμός της μνήμης προγράμματος χωρίς να απαιτείται κάποιο πρόγραμμα υποδοχής.

1.5.2 Γλώσσα προγραμματισμού

Η πιο διαδεδομένη γλώσσα προγραμματισμού των μικροελεγκτών είναι η C, η C++ και οι παραλλαγές τους. Σε τμήματα του λογισμικού όπου απαιτείται ταχύτητα ή μικρό μέγεθος χρησιμοποιούμενης μνήμης, μπορεί να χρησιμοποιείται η Assembly. Όμως η μεγαλύτερη ευκολία και ευχρηστία στην ανάπτυξη προγραμμάτων που προσφέρει η C ως γλώσσα προγραμματισμού υψηλού επιπέδου έναντι της Assembly, σε συνδυασμό με την επάρκεια μνήμης των σύγχρονων μικροελεγκτών και τις αυξημένες δυνατότητες των σύγχρονων συμβολομεταφραστών, έχουν γενικά εκτοπίσει την Assembly από τις περισσότερες εφαρμογές.

1.6 Στόχος της πτυχιακής εργασίας

Στόχος της παρούσας πτυχιακής εργασίας είναι η ανάλυση του πρωτοκόλλου I2C και η αξιοποίησή του για την ανάπτυξη ενός συστήματος που αναλαμβάνει την συλλογή, την αποθήκευση, την διαχείριση, την επεξεργασία και την αποστολή δεδομένων. Στην πτυχιακή αυτή πραγματοποιείται ο έλεγχος της μονάδας TWI(Two Wire Interface) του ATmega2560 μέσω συναρτήσεων που φτιάχτηκαν γι' αυτό τον σκοπό. Επίσης δημιουργήθηκαν και συναρτήσεις για τον έλεγχο άλλων συσκευών όπως του Real Time Clock DS3231 μέσω της πλατφόρμας Arduino Mega2560.

Τα δεδομένα που συλλέγει και διαχειρίζεται το σύστημα που υλοποιείται στην πτυχιακή, προέρχονται από μια σειρά αισθητήρων περιβαλλοντολογικών μετρήσεων. Για την αποθήκευση των δεδομένων χρησιμοποιείται μία βάση δεδομένων που δίνεται μέσω της MySQL εφαρμογής. Η επεξεργασία των δεδομένων επιτυγχάνεται από το περιβάλλον του LabVIEW και παρέχει στον χρήστη την δυνατότητα να παρακολουθεί με κυματομορφές την μεταβολή των μεγεθών σε συνάρτηση με τον χρόνο. Επιπλέον οι παραπάνω μετρήσεις αποθηκεύονται ως αρχεία με βάση την ημερομηνία στην κάρτα SD (Secured Digital) που διαθέτει το Ethernet shield του Arduino ενώ ταυτόχρονα αποστέλλονται μέσω διαδικτύου στην βάση δεδομένων. Ακόμα πραγματοποιείται σύνδεση με NTP (Network Time Protocol) Server για τον συγχρονισμό της ώρας και της ημερομηνίας.

Τα μέρη που αποτελείται το συγκεκριμένο σύστημα και θα αναλυθούν στα επόμενα κεφάλαια είναι ο μικροελεγκτής ATmega2560 της πλατφόρμας Arduino Mega2560 το Real Time Clock DS3231, το Ethernet Shield, η οθόνη LCD, ο αισθητήρας θερμοκρασίας

TCN75A ,ο αισθητήρας φωτεινότητας BH1750 και ο Logic Level Converter Bi-Directional Module 5V to 3.3V.

Κεφάλαιο 2

Το υλικό του συστήματος

2.1 Εισαγωγή

Στο κεφάλαιο αυτό γίνεται μια αναλυτική περιγραφή όσον αφορά το υλικό που χρησιμοποιήθηκε για την ανάπτυξη του συστήματος. Για κάθε επιμέρους τμήμα του υλικού που χρησιμοποιήθηκε δίνεται μία σύντομη αναφορά στα χαρακτηριστικά του καθώς και στον τρόπο λειτουργίας του. Για την ανάπτυξη του συστήματος αξιοποιήθηκε η πλατφόρμα Arduino Mega 2560 Rev3 για την οποία θα γίνει μια σύντομη περιγραφή του μικροελεγκτή ATmega 2560 (αναλύοντας κυρίως τα περιφερειακά υποσυστήματα που χρησιμοποιήθηκαν για την υλοποίηση της παρούσας πτυχιακής), το Real Time Clock DS3231, το Arduino Ethernet Shield, η οθόνη χαρακτήρων LCD, ο αισθητήρας θερμοκρασίας TCN75A, ο αισθητήρας φωτεινότητας BH1750 και ο Logic Level Converter Bi-Directional Module 5V to 3.3V.

2.2 Ιστορική αναδρομή του Arduino

Το 2005 στην πόλη της Ιβρέα, κωμόπολη της επαρχίας του Τορίνο στην περιοχή Πεδεμόντιο της βορειοδυτικής Ιταλίας παράχθηκαν οι πρώτες πλακέτες Arduino από τους Massimo Banzi και David Cueartielles. Η ανάπτυξη και υλοποίηση της πλατφόρμας Arduino πραγματοποιήθηκε στα πλαίσια ενός σχεδίου που είχε ως σκοπό την δημιουργία μίας συσκευής για τον έλεγχο προγραμμάτων και διαδραστικών σχεδίων από μαθητές, η οποία θα ήταν πιο φθηνή από άλλα πρωτότυπα συστήματα που ήταν διαθέσιμα εκείνη την περίοδο. Το όνομα του Arduino προέρχεται από τον Arduin της Ιβρέα βασιλιά της Ιταλίας.

2.3 Η πλατφόρμα Arduino Mega 2560 Rev3

Στο σχήματα 2.1 που ακολουθεί παρουσιάζεται το Arduino Mega2560 Rev3 που αξιοποιήθηκε για την εκτέλεση της παρούσας πτυχιακής εργασίας.

Σχήμα 2.1: Πάνω και κάτω όψη της πλατφόρμας Arduino Mega2560 Rev3

2.3.1 Τι είναι το Arduino

Το Arduino είναι μία open-source αναπτυξιακή πλατφόρμα που ενσωματώνει ένα μικροελεγκτή. Η πλατφόρμα Arduino έχει δυνατότητα σύνδεσης Η/Υ. Ο χρήστης μέσω του

υπολογιστή μπορεί να προγραμματίσει τον μικροελεγκτή χρησιμοποιώντας την γλώσσα Wiring και το ολοκληρωμένο αναπτυξιακό περιβάλλον που διαθέτει (Integrated Development Environment, IDE). Η γλώσσα Wiring είναι παρόμοια με την C++ και χρησιμοποιεί βιβλιοθήκες οι οποίες έχουν υλοποιηθεί με την γλώσσα C++. Η αναπτυξιακή πλατφόρμα Arduino μπορεί να χρησιμοποιηθεί για την ανάπτυξη ανεξάρτητων εφαρμογών που περιέχουν πληθώρα αισθητήρων και ελέγχουν εξωτερικές διατάξεις και συσκευές. Οι περισσότερες εκδόσεις του Arduino μπορούν να αγοραστούν προ-συναρμολογημένες.·Τα διαγράμματα και οι πληροφορίες για το υλικό είναι ελεύθερα και διαθέσιμα για αυτούς που θέλουν να συναρμολογήσουν το Arduino μόνοι τους.

2.3.2 Τα χαρακτηριστικά της πλατφόρμας Arduino Mega 2560 Rev3

Το Arduino Mega 2560 Rev3 είναι μια αναπτυξιακή πλατφόρμα βασισμένη στον μικροελεγκτή ATmega2560.της Atmel. Διαθέτει 54 ακροδέκτες ψηφιακών εισόδων και εξόδων με τους 15 από αυτούς να μπορούν να χρησιμοποιηθούν ως PWM (Pulse Width Modulation) έξοδοι, 16 ακροδέκτες αναλογικών εισόδων, 4 UARTs (hardware serial ports), έναν κεραμικό κρύσταλλο 16 MHz, μία θύρα USB (Universal Serial Port), μία υποδοχή εξωτερικής τροφοδοσίας, μία ICSP (In Circuit Serial Programming) σύνδεση, και ένα κουμπί reset

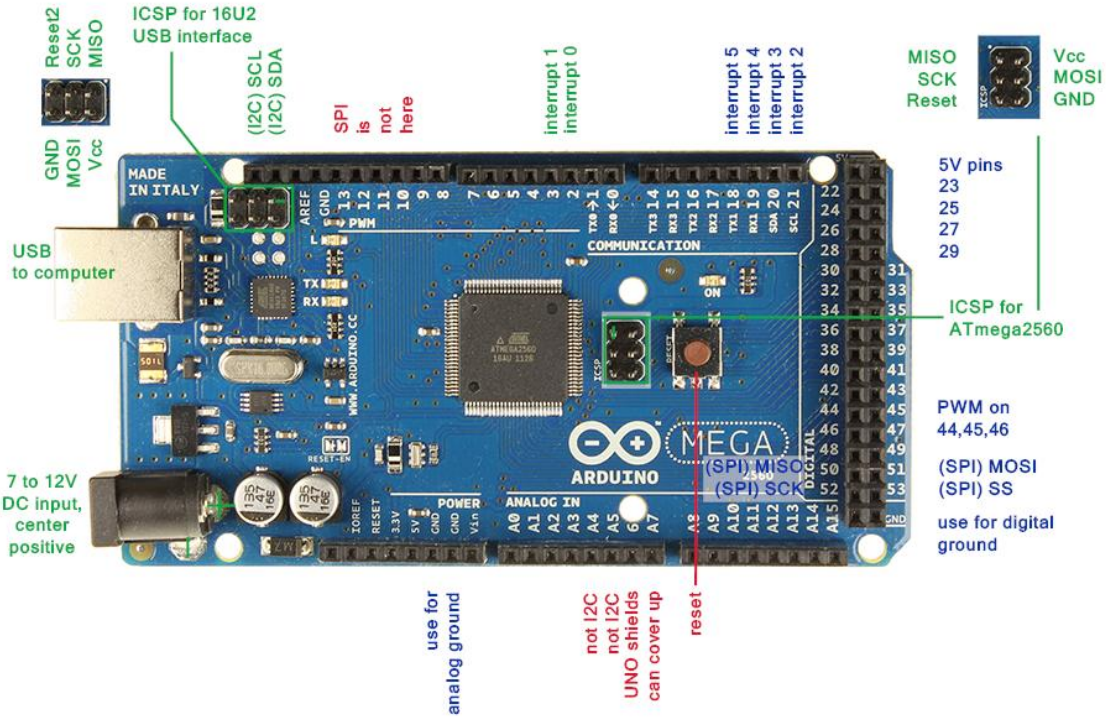
Τεχνικά Χαρακτηριστικά	
Μικροελεγκτής:	Atmega2560
Τάση λειτουργίας:	5V
Τάση εισόδου (συστήνεται):	7-12V
Τάση εισόδου (όρια):	6-20V
Ψηφιακά I/O Pins:	54 (εκ των οποίων 15 μπορούν να χρησιμοποιηθούν ως PWM)
Αναλογικά Pins εισόδου:	16
DC ρεύμα ανά I/O Pin:	20mA
DC ρεύμα για το 3.3V Pin:	50mA
Flash Memory	256 KB of which 8 KB are used by the bootloader
SRAM:	8 KB
EEPROM:	4 KB
Συχνότητα Ρολογιού:	16 MHz

Πίνακας 2.1:Τεχνικά χαρακτηριστικά της πλατφόρμας Arduino Mega2560

Εκτός από τους παραπάνω ακροδέκτες η πλατφόρμα Arduino Mega2560 (Σχήμα2.2) διαθέτει και κάποιους ακροδέκτες που υποστηρίζουν συγκεκριμένες λειτουργίες όπως παρουσιάζεται στον Πίνακα 2.2. Κάποιοι από αυτούς τους ακροδέκτες αντιπροσωπεύουν τα υποσυστήματα που διαθέτει ο μικροελεγκτής.

Λειτουργίες Ακροδεκτών της πλατφόρμας Arduino Mega2560		
Ακροδέκτες	**Λειτουργία**	**Περιγραφή**
0-53	Ψηφιακοί I/O	Κάθε ένας από αυτούς τους ακροδέκτες μπορεί να χρησιμοποιηθεί ως είσοδο/έξοδο χρησιμοποιώντας τις συναρτήσεις pinMode(), digitalWrite(), digitalRead().
A0-A16	Αναλογικοί είσοδοι	Δέχονται αναλογικά σήματα τάσης από 0-5V επιστρέφοντας μία ακέραια τιμή με 10bit ακρίβεια.
5V	Τροφοδοσία DC +5V	Προέρχεται από τον σταθεροποιητή τάσης. Επίσης μπορεί να τροφοδοτήσει απευθείας την πλακέτα παρακάμπτοντας τον σταθεροποιητή της.
3,3V	Τροφοδοσία DC +3,3V	Προέρχεται από τον σταθεροποιητή τάσης της πλατφόρμας.
GND	Γείωση	Ο ακροδέκτης γείωσης είναι σημείο αναφοράς για όλες τις διαφορές δυναμικού στην πλατφόρμα.
Vin	Εξωτερική τροφοδοσία	Ο ακροδέκτης αυτός χρησιμοποιείται για τροφοδοσία όταν η τροφοδοσία της πλατφόρμας δεν προέρχεται από την θύρα USB.(πχ Τροφοδοτικό)
AREF	Εξωτερική τάση αναφοράς	Χρησιμοποιώντας την συνάρτηση analogReference() ορίζεται ως εξωτερική τάση αναφοράς.
A4(SDA) A5(SCL) SDA SCL 20(SDA) 21(SCL)	Πρωτόκολλο 2 καλωδίων (I2C)	Οι ακροδέκτες αυτοί υποστηρίζουν το πρωτόκολλο 2 καλωδίων(I2C ή TWI)
2, 3, 18, 19, 20, 21	Εξωτερικές διακοπές (Interrupts)	Ανιχνεύουν μεταβολές στους παλμούς τάσης και ενεργοποιούν την αντίστοιχη διακοπή. Η ενεργοποίησή τους γίνεται με την συνάρτηση attachInterrupt() για κάθε ακροδέκτη αντίστοιχα.
2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13	Pulse Width Modulation(PWM)	Παρέχουν έξοδο PWM αξιοποιώντας την συνάρτηση analogWrite() η οποία δέχεται ως όρισμα έναν 8bit ακέραιο αριθμό χωρίς πρόσημο.
1→TX0 0→RX0 14→TX3 15→RX3 16→TX2 17→RX2 18→TX1 19→RX1	Σειριακή επικοινωνία	Χρησιμοποιούνται για την λήψη (RX) και αποστολή (TX) σειριακών δεδομένων. Οι ακροδέκτες αυτοί συνδέονται με το ολοκληρωμένο FTDI USB to TTL Serial στους αντίστοιχους ακροδέκτες.
50→MISO 51→MOSI 52→SCK 53→SS	Serial Peripheral Interface (SPI)	Οι ακροδέκτες αυτοί υποστηρίζουν την SPI επικοινωνία

Πίνακας 2.2: Λειτουργίες ακροδεκτών της πλατφόρμας Arduino Mega2560



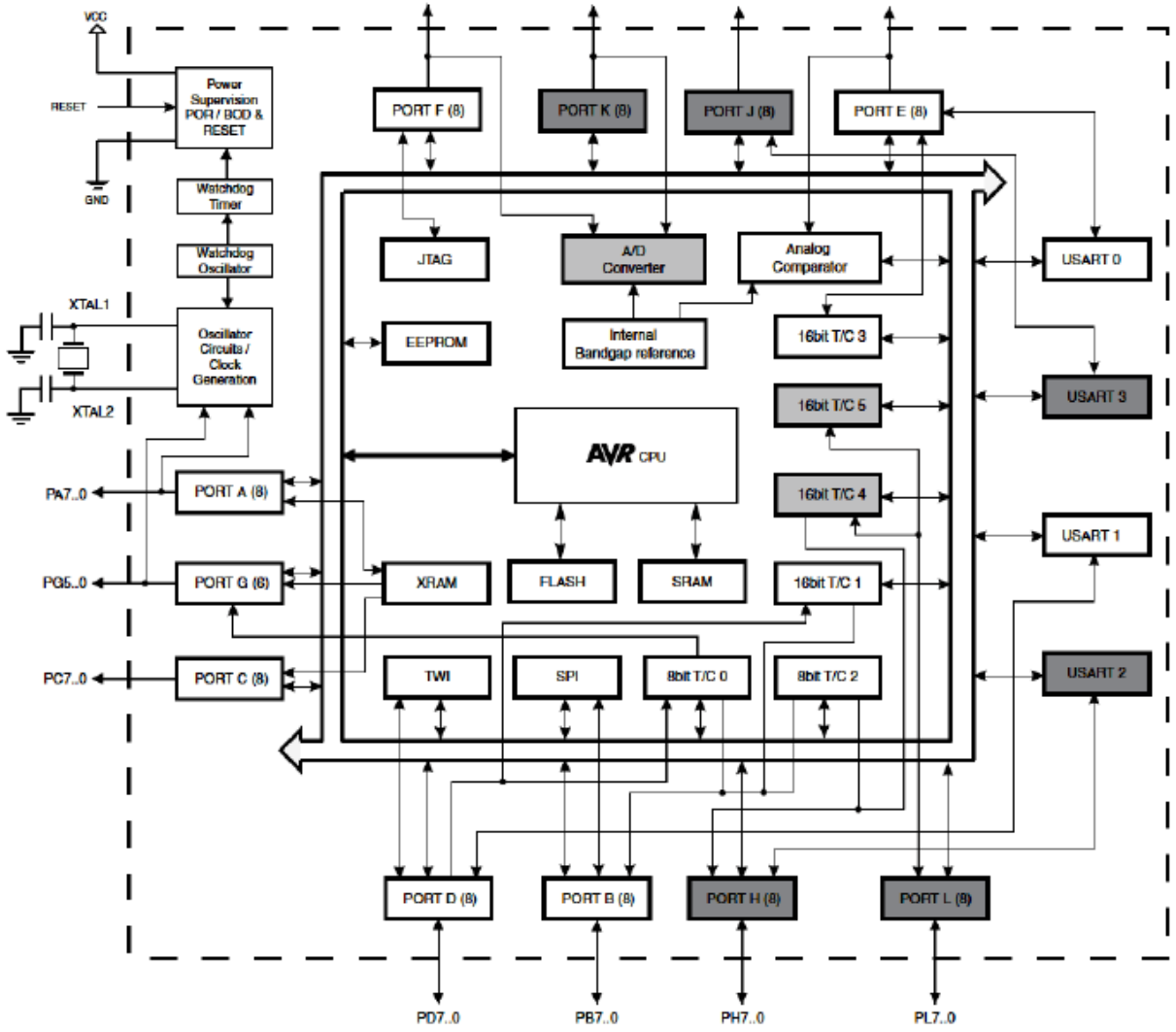
Σχήμα 2.2: Οι ακροδέκτες στο Arduino Mega2560

2.4 Ο μικροελεγκτής ATmega2560

Ο ATmega2560 είναι ένας μικροελεγκτής ο οποίος ανήκει στην κατηγορία χαμηλής ισχύος 8bit AVR της Atmel (Σχήμα 2.3). Ο μικροελεγκτής αυτός είναι βασισμένος στην RISC (Reduced Instruction Set Computing) αρχιτεκτονική. Ο ATmegs2560 συνδυάζει ISP flash memory των 256KB, 8KB SRAM , 4 KB EEPROM. Επίσης Διαθέτει 86 γενικού σκοπού I/O ακροδέκτες, 32 γενικού σκοπού καταχωρητές, έναν μετρητή πραγματικού χρόνου, 6 timer/counters, PWM, 4USARTS, 2-wire σειριακή διασύνδεση, 16 καναλιών 10bit A/D μετατροπέα και μια διεπαφή JTAG για τον on-chip εντοπισμό σφαλμάτων. Ο συγκεκριμένος μικροελεγκτής έχει την δυνατότητα να εκτελεί έως 16 MIPS(Million Instructions per Second) στα 16 MHz και η τάση λειτουργιάς του είναι 4,5-5,5 Volts. Εκτελώντας ισχυρές εντολές σε ένα μόνο κύκλο ρολογιού, ο μικροελεγκτής επιτυγχάνει μια απόδοση που πλησιάζει 1 MIPS ανά MHz, δίνοντας την δυνατότητα στον σχεδιαστή του συστήματος να βελτιστοποιεί την κατανάλωση ισχύος έναντι της ταχύτητας επεξεργασίας. Στο Σχήμα 2.3 παρουσιάζεται το διάγραμμα του ATmega2560.

2.4.1 Εξωτερικές διακοπές

Η πλατφόρμα Arduino Mega2560 υποστηρίζει μέχρι 6 εξωτερικές διακοπές (external Interrupts) που συμβολίζονται ως INT0, INT1, INT2, INT3, INT4, INT5 και αντιστοιχούν στους ψηφιακούς ακροδέκτες 21 ,20, 19, 18, 2, 3. Η εξωτερική διακοπή είναι στην πραγματικότητα ένα σήμα που δέχεται ο μικροελεγκτής οποιαδήποτε χρονική στιγμή, για ένα γεγονός το οποίο χρειάζεται άμεση προσοχή. Όταν μια διακοπή ενεργοποιηθεί, ο



Σχήμα 2.3: Μπλοκ διάγραμμα του ATmega2560

μικροελεγκτής σταματά οποιαδήποτε εργασία και εκτελεί την ρουτίνα εξυπηρέτησης διακοπής (Interrupt Service Routine,ISR). Μετά την εκτέλεση της ISR η εκτέλεση του προγράμματος συνεχίζει την εκτέλεση του προγράμματος από το σημείο στο οποίο είχε προκύψει η παραπάνω διακοπή. Για να χρησιμοποιηθεί μια εξωτερική διακοπή μέσω του Arduino IDE (Integrated Development Environment) αρκεί να χρησιμοποιηθεί η συνάρτηση attachInterrupt στην οποία πρέπει να δηλωθεί ο αριθμός της διακοπής, το όνομα της ISR και ο τρόπος λειτουργίας της διακοπής, δηλαδή attachInterrupt (interrupt, ISR_name, mode). Όπως είναι προφανές το όρισμα interrupt μπορεί να πάρει τιμές INT0, INT1, INT2 κλπ ανάλογα με την επιθυμητή διακοπή. Η ISR που καλείται κάθε φορά που ενεργοποιείται η αντίστοιχη διακοπή, δεν διαθέτει ορίσματα και δεν επιστρέφει κάποιο αποτέλεσμα. Στο όρισμα ISR_name γράφεται το όνομα της ISR, ενώ όσον αφορά τον τρόπο λειτουργίας (mode) της διακοπής υπάρχουν τέσσερις επιλογές οι οποίες είναι:

- LOW: η διακοπή ενεργοποιείται όταν στον αντίστοιχο ακροδέκτη υπάρχει χαμηλό δυναμικό (LOW).
- CHANGE: η διακοπή ενεργοποιείται στον αντίστοιχο ακροδέκτη ανιχνευθεί οποιαδήποτε αλλαγή στο δυναμικό του σήματος.

- RISING: η διακοπή ενεργοποιείται όταν ανιχνευθεί ανερχόμενο μέτωπο στο σήμα δηλαδή, όταν το δυναμικό του σήματος μεταβεί από χαμηλό σε υψηλό.
- FALLING: η διακοπή ενεργοποιείται όταν ανιχνευθεί ανερχόμενο μέτωπο στο σήμα δηλαδή, όταν το δυναμικό του σήματος μεταβεί από υψηλό σε χαμηλό.

Μεταξύ των υποστηριζόμενων διακοπών υπάρχει σειρά προτεραιότητας, έτσι στην περίπτωση που προκληθούν ταυτόχρονα περισσότερες από μια διακοπές θα εκτελεστεί πρώτα η ISR της διακοπής με την μεγαλύτερη προτεραιότητα. Η σειρά προτεραιότητας καθώς και οι ακροδέκτες που αντιστοιχούν οι διακοπές τόσο στον συγκεκριμένο μικροελεγκτή όσο και στην πλατφόρμα Arduino παρουσιάζονται στον Πίνακα 2.3. Επίσης στον πίνακα αυτό δίνονται και η αρίθμηση των διακοπών όπως υποστηρίζεται από το Arduino IDE.

Ονομασίες των interrupt	**Ακροδέκτες στον μικροελεγκτή**	**Ψηφιακοί ακροδέκτες στην πλατφόρμα**	**Αριθμός των interrupts στο Arduino IDE**
INT 0	43, PD0 (SCL/INT0)	21	2
INT 1	44, PD1 (SDA/INT1)	20	3
INT 2	45, PD2 (RXDI/INT2)	19	4
INT 3	46, PD3 (TXD1/INT3)	18	5
INT 4	6, PE4 (OC3B/INT4)	2	0
INT 5	7, PE5 (OC3C/INT5)	3	1

Πίνακας 2.3: Οι ακροδέκτες των εξωτερικών διακοπών στην πλατφόρμα Arduino Mega2560

2.4.2 Μονάδα USART

Ο ATmega2560 διαθέτει 4 USARTs (Universal Synchronous Asynchronous Receiver/ Transmitter) για την υποστήριξη TTL (Transistor – Transistor Logic) σειριακής επικοινωνίας.

Οι μονάδες UARTs είναι κυκλώματα τα οποία αναλαμβάνουν την σειριακή επικοινωνία υπολογιστών με άλλους υπολογιστές ή με άλλες συσκευές ή μικροελεγκτές. Η επικοινωνία αυτών γίνεται μέσω θυρών RS-232, RS-422 ή RS-485. Ο ρυθμός μετάδοσης (baud rate) μετριέται σε bps (bits per second). Μια μονάδα UART δέχεται ένα σύνολο bytes δεδομένων συνήθως αλφαριθμητικούς χαρακτήρες και τα αποστέλλει σειριακά ως bits δεδομένων και αντίστροφα.

Κατά αντιστοιχία με την μονάδα UART, η μονάδα USART παρέχει το απαιτούμενο υλικό για την επικοινωνία με μόντεμ και άλλες σειριακές συσκευές. Επιπρόσθετα η μονάδα USART υποστηρίζει τόσο την σύγχρονη όσο και την ασύγχρονη μεταφορά σειριακών δεδομένων.

Πρακτικές διαφορές μεταξύ σύγχρονης (η οποία είναι δυνατή μόνο με ένα USART) και ασύγχρονης λειτουργίας είναι οι εξής:

- Η σύγχρονη λειτουργία απαιτεί δεδομένα και ένα ρολόι. Η ασύγχρονη λειτουργία απαιτεί μόνο δεδομένα.
- Στη σύγχρονη λειτουργία, τα δεδομένα μεταδίδονται σε ένα σταθερό ρυθμό. Στην ασύγχρονη λειτουργία, τα δεδομένα δεν χρειάζεται να διαβιβάζονται με σταθερό ρυθμό

- Στην σύγχρονη τα δεδομένα μεταδίδονται με τη μορφή μπλοκ, ενώ στην ασύγχρονη τα δεδομένα μεταδίδονται ένα byte τη φορά.
- Η σύγχρονη λειτουργία επιτρέπει υψηλότερο DTR (Data Transfer Rate) από την ασύγχρονη λειτουργία, αν όλοι οι άλλοι παράγοντες παραμένουν σταθεροί.

Τα χαρακτηριστικά της μονάδας USART στον ATmega2560 είναι τα εξής:

- Πλήρης αμφίδρομη λειτουργία (ανεξάρτητοι σειριακοί καταχωρητές λήψης και μετάδοσης)
- Ασύγχρονη ή σύγχρονη λειτουργία
- Σύγχρονη λειτουργία αφέντη (master) ή σκλάβου (slave)
- Υψηλής ανάλυσης γεννήτριας ρυθμού μετάδοσης (baud rate generator)
- Υποστηρίζει σειριακά frames με 5, 6, 7, 8 ή 9 data bits και 1 ή 2 stop bits
- Η παραγωγή μονής ή ζυγής ισοτιμίας (parity) αλλά και ο έλεγχος ισοτιμίας υποστηρίζονται από το υλικό
- Ανίχνευση υπέρβασης ορίου μεταφοράς δεδομένων (Data OveRun Detection)
- Ανίχνευση σφαλμάτων διαμόρφωσης μηνυμάτων (Framing Error Detection)
- Φιλτράρισμα θορύβου που περιλαμβάνει ανίχνευση λανθασμένου bit εκκίνησης και ψηφιακό βαθυπερατό φίλτρο
- Υποστηρίζει τρεις διακριτές διακοπές την TX Complete, την TX Data Register Empty και την RX Complete
- Λειτουργία επικοινωνίας πολλαπλών επεξεργαστών (Multi-processor Communication Mode)
- Λειτουργία ασύγχρονης επικοινωνίας διπλής ταχύτητας (Double Speed Asynchronous Communication Mode)

Η μονάδα USART χρησιμοποιείται για την επικοινωνία με τον Η/Υ. Επίσης χρησιμοποιείται για την φόρτωση του προγράμματος στο Arduino μέσω του bootloader. Η μονάδα USART μπορεί να αξιοποιηθεί και για την επικοινωνία με συσκευές που υποστηρίζουν σειριακή επικοινωνία.

2.4.3 Μονάδα SPI

Μία άλλη μονάδα σειριακής επικοινωνίας που διαθέτει ο ATmega2560 είναι η μονάδα SPI (Serial Peripheral Interface) που επιτρέπει την μεταφορά δεδομένων με υψηλή ταχύτητα μεταξύ άλλων μικροελεγκτών και περιφερειακών μονάδων. Τα βασικά χαρακτηριστικά της μονάδας SPI είναι τα εξής:

- Διαθέτει τριών καλωδίων πλήρης αμφίδρομη σύγχρονη μεταφορά δεδομένων
- Λειτουργία αφέντη ή σκλάβου
- Μεταφορά δεδομένων με πρώτο το λιγότερο σημαντικό byte ή το περισσότερο σημαντικό byte
- Διαθέτει 7 προγραμματιζόμενους ρυθμούς μετάδοσης
- Ενεργοποιεί μια διακοπή κατά την ολοκλήρωση της μετάδοσης
- Παρέχει προστασία με flag ώστε να αποφευχθεί οποιαδήποτε διένεξη κατά το γράψιμο
- Δυνατότητα ενεργοποίησης από την κατάσταση αναμονής

- Διπλή ταχύτητα κατά την SPI λειτουργία ως αφέντης

2.4.4 Μονάδα TWI

Μία επιπλέον σειριακή μονάδα που υποστηρίζει ο ATmega2560 είναι η TWI (Two Wire Interface). Η σειριακή μονάδα επικοινωνίας TWI προορίζεται για τυπικές εφαρμογές του μικροελεγκτή και επιτρέπει την διασύνδεση έως και 128 διαφορετικών συσκευών χρησιμοποιώντας μόνο δύο καλώδια επικοινωνίας. Για την επίτευξη της παραπάνω επικοινωνίας κάθε μια από τις παραπάνω συσκευές θα πρέπει να έχει μια μοναδική διευθύνση και να διαθέτει κάποιο μηχανισμό συμβατότητας με το πρωτόκολλο TWI. Σύμφωνα με το TWI το ένα από τα δύο καλώδια επικοινωνίας χρησιμοποιείται για το ρολόι (Serial Clock Line,SCL) και το δεύτερο για τα δεδομένα (Serial Data Line,SDA). Σε κάθε ένα τα καλώδια επικοινωνίας απαιτείται μια pull-up εξωτερική αντίσταση.

Τα βασικά χαρακτηριστικά της μονάδας TWI είναι:

- Απλή, ισχυρή και ευέλικτη επικοινωνία μέσω μόνο 2 καλωδίων.
- Υποστηρίζει λειτουργία αφέντη ή σκλάβου.
- Η συσκευή έχει τη δυνατότητα λειτουργίας ως πομπός ή δέκτης.
- Διαθέτει χώρο διεύθυνσης των 7 bit επιτρέποντας έως 128 διευθύνσεις για σκλάβους.
- Υποστηρίζει διαχείριση πολλαπλών αφεντών.
- Μέγιστη ταχύτητα μεταφοράς δεδομένων 400kHz.
- Διαθέτει κυκλωματική διάταξη για την μείωση του θορύβου στους δίαυλους μετάδοσης.
- Πλήρης προγραμματιζόμενη διεύθυνση σκλάβου που υποστηρίζει την λειτουργία γενικής κλήσης από τον αφέντη.
- Διαθέτει αναγνώριση διευθύνσεων με σκοπό την επαναφορά του AVR όταν βρίσκεται σε κατάσταση αναμονής
- Συμβατό με το πρωτόκολλο I2C (Inter-Integrated Circuit)

Κάποιες βασικές έννοιες που θα χρησιμοποιηθούν στην συνέχεια και η σημασία τους είναι σημαντική για την κατανόηση της ανάλυσης που πραγματοποιείται παρακάτω εξηγούνται στον Πίνακα 2.4.

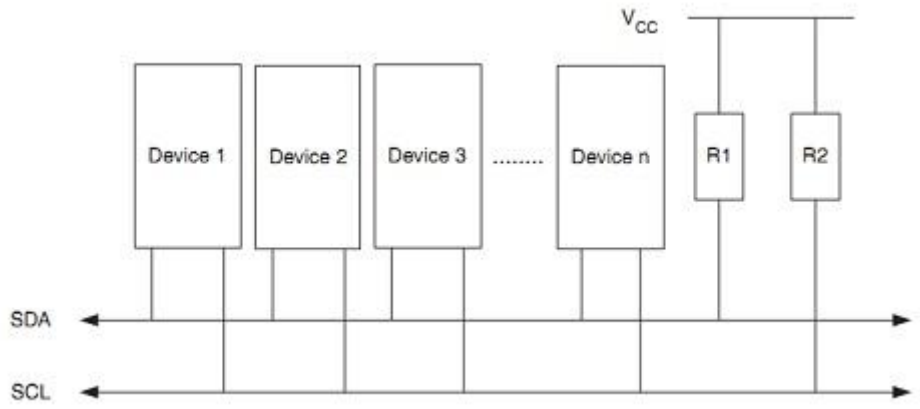
Όρος	Περιγραφή
Αφέντης	Η συσκευή η οποία εκκινεί και τερματίζει μία μετάδοση. Ο αφέντης επίσης παράγει και το ρολόι SCL.
Σκλάβος	Η συσκευή η οποία απευθύνεται ο αφέντης
Πομπός	Η συσκευή η οποία τοποθετεί τα δεδομένα στο bus
Δέκτης	Η συσκευή η οποία διαβάζει τα δεδομένα από το bus

Πίνακας 2.4: Περιγραφή των πρωταγωνιστικών όρων του TWI

Ηλεκτρική Διασύνδεση

Όπως απεικονίζεται στο Σχήμα 2.4, και οι δύο γραμμές του TWI συνδέονται με την θετική τάση τροφοδοσίας μέσω pull-up αντιστάσεων. Οι οδηγοί των παραπάνω γραμμών όλων των συμβατών με το TWI συσκευών είναι open-drain ή open-collector. Έτσι εφαρμόζεται μία wired-AND λειτουργία η οποία είναι απαραίτητη για τη λειτουργία του interface. Ένα χαμηλού επιπέδου σήμα παράγεται σε μια γραμμή του TWI όταν μία ή

περισσότερες TWI συσκευές παράγουν μηδενική έξοδο. Υψηλού επιπέδου σήμα παράγεται όταν όλες οι TWI συσκευές τακτοποιούν τις καταστάσεις των εξόδων τους, επιτρέποντας τις pull-up αντιστάσεις να ανυψώσουν το δυναμικό της γραμμής. Όλες οι συσκευές που συνδέονται στον AVR μέσω του TWI πρέπει να έχουν τροφοδοτηθεί με τάση ώστε να είναι δυνατή η λειτουργία των γραμμών του TWI. Ο αριθμός των συσκευών που μπορούν να συνδεθούν περιορίζεται μόνο από την χωρητικότητα των γραμμών του TWI μεγέθους 400 pF και τον υποστηριζόμενο 7-bit χώρο διευθύνσεων για τους σκλάβους.



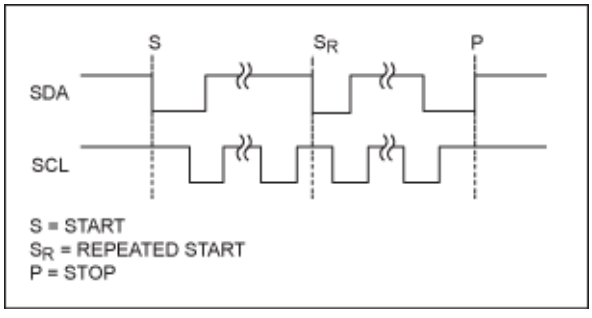
Σχήμα 2.4: Διασύνδεση του TWI BUS

Μεταφορά bits

Κάθε bit δεδομένων που μεταφέρεται στο TWI bus συνοδεύεται από έναν παλμό στη γραμμή ρολογιού. Το δυναμικό της γραμμής δεδομένων πρέπει να είναι σταθερό, όταν η γραμμή ρολογιού είναι σε υψηλό δυναμικό. Η μόνη εξαίρεση σε αυτόν τον κανόνα είναι κατά την παραγωγή των συνθηκών εκκίνησης (start) και τερματισμού (stop).

Συνθήκες εκκίνησης(Start) και τερματισμού(Stop)

Ο αφέντης εκκινεί και τερματίζει μία μετάδοση δεδομένων. Η μετάδοση ξεκινά όταν ο αφέντης αποστέλλει την κατάσταση START στο bus, και τερματίζεται όταν ο αφέντης αποστέλλει μια κατάσταση STOP. Μεταξύ μίας κατάστασης εκκίνησης και μίας κατάστασης τερματισμού, το bus θεωρείται απασχολημένο και κανένας άλλος αφέντης δεν επιτρέπεται να προσπαθήσει να καταλάβει τον έλεγχο του bus. Ιδιαίτερη περίπτωση αποτελεί όταν αποστέλλεται μία νέα κατάσταση εκκίνησης μεταξύ μίας κατάστασης εκκίνησης και μίας κατάστασης τερματισμού. Το φαινόμενο αυτό ονομάζεται επαναλαμβανόμενη κατάσταση εκκίνησης και λαμβάνει χώρα όταν ο αφέντης επιθυμεί να ξεκινήσει μία νέα μεταφορά χωρίς



Σχήμα 2.5: Καταστάσεις εκκίνησης, επαναλαμβανόμενης εκκίνησης και τερματισμού

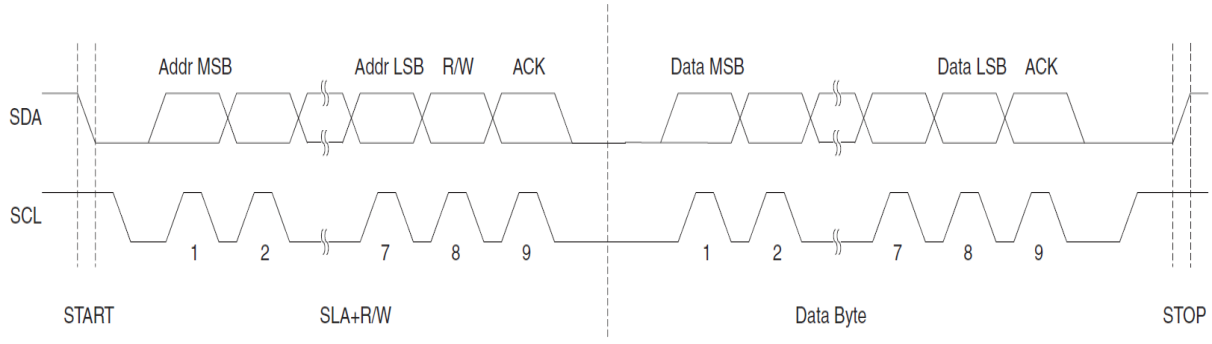
να χάσει τον έλεγχο του bus. Μετά από την επαναλαμβανόμενη κατάσταση εκκίνησης το bus θεωρείται απασχολημένο, μέχρι την επόμενη κατάσταση τερματισμού. Οι καταστάσεις εκκίνησης και τερματισμού σηματοδοτούνται από την αλλαγή του επιπέδου του δυναμικού της γραμμής δεδομένων (SDA) ενώ η γραμμή ρολογιού (SCL) βρίσκεται σε υψηλό δυναμικό όπως φαίνεται στο Σχήμα 2.5.

Address Packet Format

Όλα τα πακέτα διευθύνσεων (address packets) τα οποία μεταδίδονται στο bus του TWI έχουν μέγεθος 9 bit. Τα 7 πιο σημαντικά bits του address packet αντιστοιχούν στην διεύθυνση του σκλάβου. Το αμέσως επόμενο bit είναι το Read/Write bit ελέγχου που καθορίζει αν θα γίνει ανάγνωση ή εγγραφή δεδομένων από τον αφέντη στο σκλάβο. Το τελευταίο bit είναι το bit αναγνώρισης (acknowledge, ACK). Εάν το Read/Write bit ελέγχου έχει τιμή '1' σημαίνει πως πρόκειται να εκτελεστεί η λειτουργία ανάγνωσης και το address packet ονομάζεται SLA+R. Διαφορετικά, θα εκτελεστεί λειτουργία εγγραφής και τότε το address packet ονομάζεται SLA+W .Όταν ένας σκλάβος αναγνωρίσει την διεύθυνση του θα πρέπει να παράγει ένα ACK δηλαδή ένα σήμα αναγνώρισης θέτοντας χαμηλό δυναμικό στην γραμμή SDA στον ένατο κύκλο της γραμμής SCL. Σε περίπτωση που ο σκλάβος είναι απασχολημένος ή δεν μπορεί να εκτελέσει το αίτημα που έχει λάβει από τον αφέντη η γραμμή SCL θα πρέπει να μείνει σε υψηλό δυναμικό κατά τον κύκλο αναγνώρισης του ACK. Έπειτα ο αφέντης μπορεί να στείλει μία συνθήκη τερματισμού ή μία συνθήκη επαναλαμβανόμενης εκκίνησης για να ξεκινήσει μία νέα μετάδοση. Το πιο σημαντικό byte (MSB) της διεύθυνσης του σκλάβου μεταδίδεται πρώτο. Η διεύθυνση 0000 000 προορίζεται για μια γενική κλήση. Όταν αποστέλλεται μία γενική κλήση όλοι οι σκλάβοι πρέπει να ανταποκριθούν και να έχουν σε χαμηλό δυναμικό την γραμμή SDA κατά τον κύκλο αναγνώρισης του ACK. Μία γενική κλήση καλείται όταν ο αφέντης θέλει να μεταδώσει το ίδιο μήνυμα σε όλους του σκλάβους του συστήματος.

Data Packet Format

Όλα τα πακέτα δεδομένων (data packets) τα οποία μεταδίδονται στο TWI bus έχουν μέγεθος 9 bit, τα οποία αποτελούνται από ένα byte δεδομένων και ένα bit αναγνώρισης (Σχήμα 2.6). Κατά την διάρκεια μίας μεταφοράς δεδομένων ο αφέντης είναι υπεύθυνος για την παραγωγή του ρολογιού και των συνθηκών εκκίνησης και τερματισμού, ενώ ο δέκτης είναι υπεύθυνος για την αναγνώριση της παραλαβής. Μία αναγνώριση σηματοδοτείται από το



Σχήμα 2.6: TWI Address and Data Packet Format

δέκτη κρατώντας τη γραμμή SDA σε χαμηλό δυναμικό κατά τη διάρκεια του ένατου SCL κύκλου. Αν ο δέκτης αφήσει σε υψηλό δυναμικό την γραμμή SDA, ένα σήμα NACK σηματοδοτείται το οποίο σημαίνει πως ο δέκτης έλαβε το τελευταίο byte ή για κάποιο λόγο δεν μπορεί να λάβει άλλα bytes και έτσι ενημερώνει τον αφέντη. Το πιο σημαντικό byte του πακέτου δεδομένων μεταδίδεται πάντα πρώτο. Επίσης είναι δυνατή η ταυτόχρονη μετάδοση του πακέτου διεύθυνσης και των πακέτων δεδομένων.

Ακροδέκτες SDA και SCL

Οι ακροδέκτες SDA και SCL πραγματοποιούν την διασύνδεση μεταξύ του TWI και του υπόλοιπου συστήματος του μικροελεγκτή. Οι οδηγοί εξόδου περιέχουν έναν slew-rate limiter ώστε να συμμορφώνονται με τις προδιαγραφές του TWI. Τα στάδια εισόδου περιέχουν μία μονάδα καταστολής αιχμών (spikes) τα οποία μπορούν να αφαιρέσουν αιχμές μικρότερες των 50ns.

Μονάδα παραγωγής Bit Rate

Η μονάδα παραγωγής Bit Rate ελέγχει την περίοδο της γραμμής SCL όταν λειτουργεί ως Master. Η SCL περίοδος ελέγχεται από τις ρυθμίσεις του καταχωρητή Bit Rate TWI (TWBR) και τα prescaler bits στον καταχωρητή κατάστασης TWI (TWSR). Η λειτουργία Slave δεν εξαρτάται από τις ρυθμίσεις Bit Rate ή prescaler, αλλά από την συχνότητα χρονισμού της CPU η οποία στον σκλάβο θα πρέπει να είναι τουλάχιστον 16 φορές υψηλότερη από τη συχνότητα SCL. Οι σκλάβοι μπορούν να παρατείνουν περίοδο με χαμηλού δυναμικού στο SCL, μειώνοντας έτσι την μέση περίοδο του ρολογιού του TWI bus. Η SCL συχνότητα παράγεται σύμφωνα με την ακόλουθη εξίσωση :

*SCL frequency = CPU Clock frequency / (16 + 2(TWBR) * (PrescalerValue))*

Μονάδα Bus Interface

Η μονάδα bus Interface περιέχει τον καταχωρητή δεδομένων και διευθύνσεων (TWDR), έναν ελεγκτή εκκίνησης/τερματισμού και κυκλωμα ανίχνευσης διαιτησίας (arbitration detection). Ο καταχωρητής TWDR περιέχει τα byte δεομένων ή διευθύνσεων που θα μεταδώσει, ή θα παραλάβει. Επίσης το bus interface διαθέτει και έναν καταχωρητή που περιέχει το NACK bit που μεταδίδεται ή λαμβάνεται. Αυτός ο καταχωρητής δεν είναι προσβάσιμος απευθείας από το πρόγραμμα, αλλά μπορεί να ελεγχθεί μέσω του καταχωρητή ελέγχου TWCR. Ο ελεγκτής εκκίνησης/τερματισμού είναι η υπεύθυνος για την παραγωγή και την ανίχνευση των συνθηκών εκκίνησης, τερματισμού και επαναλαμβανόμενης εκκίνησης. Ο ελεγκτής αυτός είναι ικανός να εντοπίσει μία συνθήκη εκκίνησης ή τερματισμού ακόμα και όταν ο μικροελεγκτής βρίσκεται σε κατάσταση αναμονής. Εάν το TWI έχει ξεκινήσει μία μετάδοση ως αφέντης το arbitration detection κύκλωμα συνεχώς ελέγχει την μετάδοση ώστε να να προσδιορίσει αν η διαδικασία της διαιτησίας είναι σε εξέλιξη. Στην αντίθετη περίπτωση ενημερώνεται η μονάδα ελέγχου και ενεργοποιούνται διορθωτικές διαδικασίες ενώ ταυτόχρονα παράγονται οι αντίστοιχοι κωδικοί κατάστασης.

Μονάδα Address Match Unit

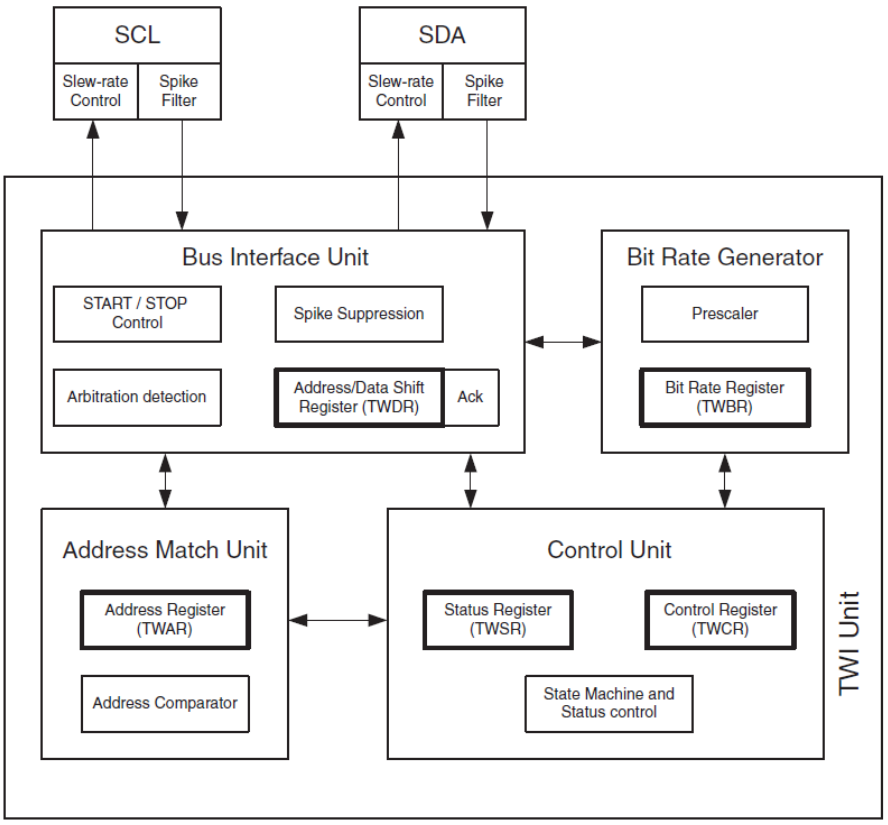
Η μονάδα ελέγχου διεύθυνσης ελέγχει εάν τα bytes διεύθυνσης που έχουν ληφθεί ταιριάζουν με την 7-bit διεύθυνση στον καταχωρητή διεύθυνσης TWAR. Κατά την σύγκριση,

η μονάδα ελέγχου ενημερώνεται επιτρέποντας να εκτελεσθούν οι κατάλληλες λειτουργίες. Το TWI θα αναγνωρίσει ή δεν θα αναγνωρίσει την διεύθυνση ανάλογα με τις ρυθμίσεις στον καταχωρητή TWCR. Η μονάδα αυτή έχει την ικανότητα να συγκρίνει διευθύνσεις ακόμα και όταν ο AVR βρίσκεται σε κατάσταση αναμονής.

Μονάδα Ελέγχου

Η μονάδα ελέγχου παρακολουθεί το bus του TWI και ανταποκρίνεται ανάλογα με τις ρυθμίσεις του καταχωρητή ελέγχου TWCR. Όταν ένα γεγονός που απαιτεί την προσοχή της εφαρμογής εμφανίζεται στο bus, το Interrupt Flag (TWINT) παίρνει την τιμή '1'. Στον επόμενο κύκλο ρολογιού, ο καταχωρητής κατάστασης TWSR ενημερώνεται με τον κωδικό κατάστασης που αντιπροσωπεύει την διαδικασία που πραγματοποιήθηκε. Ο καταχωρητής κατάστασης TWSR περιέχει τις σχετικές πληροφορίες για την κατάσταση όταν το Interrupt Flag έχει την τιμή '1'. Όλες τις άλλες φορές ο καταχωρητής κατάστασης περιέχει έναν ειδικό κωδικό κατάστασης που δείχνει ότι καμία σχετική πληροφορία κατάστασης δεν είναι διαθέσιμη. Όσο υπάρχει το TWINT, η γραμμή SCL μένει σε χαμηλό δυναμικό ώστε να δώσει χρόνο στην εφαρμογή για να ολοκληρώσει τις διεργασίες της πριν να επιτραπεί η συνέχιση της μετάδοσης του TWI. Το TWINT εμφανίζεται στις παρακάτω περιπτώσεις:

- Μετά από μία μετάδοση συνθήκης εκκίνησης ή επαναλαμβανόμενης εκκίνησης
- Μετά από μία μετάδοση SLA+R/W
- Μετά από την μετάδοση ενός byte διεύθυνσης
- Σε περίπτωση που χαθεί μία διαιτησία
- Μετά από την παραλαβή ενός byte δεδομένων



Σχήμα 2.7: Επισκόπηση του μοντέλου TWI

- Μετά από την παραλαβή μίας συνθήκης τερματισμού ή επαναλαμβανόμενης εκκίνησης ενώ λειτουργεί ως σκλάβος
- Όταν ένα σφάλμα του bus εμφανιστεί λόγο μίας μη επιτρεπτής συνθήκης εκκίνησης ή τερματισμού
- Όταν το TWI έχει διευθυνσιοδοτηθεί ως σκλάβος και έχει πραγματοποιηθεί γενική κλήση

Παρακάτω δίνονται οι συμβολισμοί και οι ορισμοί για κάποιες βασικές λειτουργίες και έννοιες του TWI που αναφέρθηκαν παραπάνω.

- **S:** START condition (συνθήκη εκκίνησης)
- **Rs:** REPEATED START condition (συνθήκη επαναλαμβανόμενης εκκίνησης)
- **R:** Read bit (bit ανάγνωσης, υψηλό δυναμικό στην γραμμή SDA)
- **W:** Write bit (bit εγγραφής, χαμηλό δυναμικό στην γραμμή SDA)
- **A:** Acknowledge bit (bit αναγνώρισης, χαμηλό δυναμικό στην γραμμή SDA)
- **NACK:** Not acknowledge bit (bit μη αναγνώρισης υψηλό δυναμικό στην γραμμή SDA)
- **Data:** 8-bit data (δεδομένα των 8 bit)
- **P:** STOP condition (συνθήκη τερματισμού)
- **SLA:** Slave Address (διεύθυνση σκλάβου)

Ο έλεγχος του TWI γίνεται μέσω καταχωρητών οι οποίοι δομούν τις λειτουργίες του και επιτρέπουν στον χρήστη την διαχείρισή του μέσω αυτών. Οι καταχωρητές αυτοί αναλύονται στην συνέχεια.

TWBR-TWI Bit Rate Register

Bit	7	6	5	4	3	2	1	0
TWBR	**TWBR7**	**TWBR6**	**TWBR5**	**TWBR4**	**TWBR3**	**TWBR2**	**TWBR1**	**TWBR0**
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

Σχήμα 2.8: Τα bit του καταχωρητή TWBR

Ο καταχωρητής TWBR επιλέγει τον συντελεστή διαίρεσης της bit rate γεννήτριας. Η γεννήτρια bit rate είναι ουσιαστικά ένας διαιρέτης συχνότητας που παράγει την συχνότητα του SCL ρολογιού, κατά την λειτουργία ως αφέντης.

TWCR-TWI Control Register

Bit	7	6	5	4	3	2	1	0
TWCR	**TWINT**	**TWEA**	**TWSTA**	**TWSTO**	**TWWC**	**TWEN**	**-**	**TWIE**
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

Σχήμα 2.9: Τα bit του καταχωρητή TWCR

Ο TWCR καταχωρητής χρησιμοποιείται για να ελέγχει την λειτουργία του TWI. Χρησιμοποιείται για να ενεργοποιεί το TWI, για να ξεκινήσει μια λειτουργία ως αφέντης εφαρμόζοντας μια συνθήκη εκκίνησης στο bus, για να παράγει μια συνθήκη αναγνώρισης

λήψης, για να παράγει μια συνθήκη τερματισμού και για να ελέγχει το bus κατά την διάρκεια εγγραφής δεδομένων στον κατάχωρητή TWDR.

Bit 7-TWINT (TWI Interrupt Flag): Το bit αυτό ελέγχεται από το υλικό και ενεργοποιείται όταν το TWI έχει τελειώσει την αντίστοιχη διαδικασία, του και περιμένει ανταπόκριση από το πρόγραμμα της εφαρμογής.

Bit 6-TWEA (TWI Enable Acknowledge Bit): Το bit TWEA ελέγχει την παραγωγή παλμών αναγνώρισης. Εάν το TWEA bit έχει τιμή '1', ο παλμός ACK παράγεται στο TWI bus εάν ισχύουν οι παρακάτω συνθήκες.

1. Η διεύθυνση του σκλάβου έχει ληφθεί
2. Έχει ληφθεί γενική κλήση, όσο το bit TWGCE στον καταχωρητή TWAR έχει τιμή '1'
3. Ένα byte δεδομένων έχει ληφθεί κατά την λειτουργία ως Master Receiver ή Slave Receiver

Γράφοντας την τιμή '0' στο TWEA bit, η συσκευή αποσυνδέεται προσωρινά από το TWI. Η αναγνώριση της διεύθυνσης μπορεί να συνεχιστεί όταν το TWEA bit πάρει την τιμή '1'.

Bit 5–TWSTA (TWI START Condition Bit): Η εφαρμογή γράφει το bit TWSTA ως '1' όταν επιθυμεί να λειτουργεί ως αφέντης. Το υλικό του TWI ελέγχει εάν το bus είναι διαθέσιμο και παράγει μία συνθήκη εκκίνησης εάν το bus είναι ελεύθερο. Ωστόσο, εάν το bus δεν είναι ελεύθερο, το TWI περιμένει μέχρι να εντοπιστεί μία συνθήκη τερματισμού και τότε παράγεται μία νέα συνθήκη εκκίνησης για να διεκδικήσει την λειτουργία ως αφέντη στο bus. Το bit αυτό πρέπει να γίνεται '0'από το πρόγραμμα όταν η συνθήκη εκκίνησης μεταδοθεί.

Bit 4-TWSTO (TWI STOP Condition Bit): Το bit αυτό όταν παίρνει την τιμή '1' κατά την λειτουργία ως αφέντης παράγεται μία συνθήκη τερματισμού. Όταν εκτελεστεί η συνθήκη τερματισμού το TWSTO bit παίρνει την τιμή '0' αυτόματα. Στην λειτουργία ως σκλάβος, το bit αυτό μπορεί να πάρει την τιμή '1' για να επανέλθει η εφαρμογή από κάποιο σφάλμα. Έτσι δεν θα παραχθεί μία συνθήκη τερματισμού, αλλά το TWI θα ελευθερώσει τις γραμμές SDA και SCL θέτοντάς τα σε υψηλό δυναμικό.

Bit 3-TWWC (TWI Write Collision Flag): Το bit αυτό παίρνει την τιμή '1' όταν υπάρξει προσπάθεια εγγραφής στον καταχωρητή δεδομένων TWDR και το bit TWINT έχει την τιμή '0'. Το bit αυτό επιστρέφει στην αρχική του κατάσταση μόλις υπάρξει προσπάθεια εγγραφής στον καταχωρητή δεδομένων TWDR και το bit TWINT είναι '1'.

Bit 2-TWEN (TWI Enable Bit): Το bit αυτό ενεργοποιεί την διαδικασία TWI και κατά επέκταση το TWI interface. Όταν το bit TWEN παίρνει την τιμή '1', το TWI καταλαμβάνει τους ακροδέκτες εισόδου/εξόδου που είναι συνδεδεμένοι με τους ακροδέκτες SCL και SDA, ενεργοποιώντας έτσι τους slew-rate limiters και τα spike filters. Αν το bit αυτό έχει την τιμή '1', το TWI απενεργοποιείται και όλες οι μεταδόσεις τερματίζονται, ανεξάρτητα εάν έχουν ολοκληρωθεί όλες οι ενεργές διαδικασίες.

Bit 1-Res (Reserved Bit): Είναι ένα δεσμευμένο bit χωρίς κάποια λειτουργία και διαβάζεται πάντα ως '0'.

Bit 0-TWIE (TWI Interrupt Enable): Όταν το bit αυτό παίρνει την τιμή '1'και το I-bit στο καταχωρητή SREG που ενεργοποιεί τις διακοπές του μικροελεγκτή, έχει την τιμή '1' το αίτημα διακοπής ενεργοποιείται για όσο το TWINT flag έχει την τιμή '1'.

TWSR - TWI Status Register

Bit	7	6	5	4	3	2	1	0
TWSR	**TWS7**	**TWS6**	**TWS5**	**TWS4**	**TWS3**	**-**	**TWPS1**	**TWPS0**
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	1	1	1	1	1	0	0	0

Σχήμα 2.10: Τα bit του καταχωρητή TWSR

Bits 7:3-TWS (TWI Status): Τα Bit από 7 έως και 3 περιγράφουν την κατάσταση που βρίσκεται το TWI. Ο καταχωρητής αυτός επίσης περιέχει και τα prescaler bits τα οποία θα πρέπει να μην ληφθούν υπόψη ώστε να βρεθεί ο σωστός κωδικός κατάστασης του TWI.

Bit 2-Re (Reserved Bit): Το bit αυτό δεν χρησιμοποιείται για κάποια διεργασία και διαβάζεται πάντα ως '0'.

Bits 1:0-TWPS (TWI Prescaler Bits): Τα bits αυτά ελέγχουν τον διαιρέτη bit rate.

TWPS1	TWPS0	Prescaler Value
0	0	1
0	1	4
1	0	16
1	1	64

Πίνακας 2.5: Οι τιμές που μπορεί να πάρει ο διαιρέτης(prescaler)

TWDR - TWI Data Register

Bit	7	6	5	4	3	2	1	0
TWDR	**TWD7**	**TWD6**	**TWD5**	**TWD4**	**TWD3**	**TWD2**	**TWD1**	**TWD0**
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	1	1	1	1	1	1	1	1

Σχήμα 2.12: Τα bit του καταχωρητή TWDR

Ο καταχωρητής TWDR περιέχει το επόμενο byte που είναι να μεταδοθεί ή το τελευταίο byte το οποίο παρέλαβε, ανάλογα με την λειτουργία του κάθε φορά. Επιτρέπεται η εγγραφή σε αυτόν τον καταχωρητή μόνο όταν δεν βρίσκεται σε διαδικασία μεταφοράς byte. Αυτό συμβαίνει όταν το bit διακοπής TWINT έχει την τιμή '1'. Τα δεδομένα στον καταχωρητή TWDR παραμένουν αμετάβλητα για όσο το bit αυτό έχει την τιμή '1'.

Καθώς τα δεδομένα μεταφέρονται έξω από τον καταχωρητή TWDR τα δεδομένα από το bus μεταφέρονται μέσα σε αυτόν. Ο καταχωρητής TWDR περιέχει πάντα το τελευταίο byte που εμφανίστηκε στο bus εκτός από την περίπτωση κατά την οποία το bus έχει επανέλθει από κατάσταση αναμονής. Σε αυτή την περίπτωση το περιεχόμενο του TWDR δεν είναι προσδιορισμένο. Σε περίπτωση που χαθεί η διαιτησία δεν χάνονται δεδομένα κατά την μετάδοση από τον αφέντη στον σκλάβο.

Bits 7:0-TWD (TWI Data Register): Αυτά τα 8 bit περιέχουν το byte δεδομένων που είναι να μεταδοθεί, ή το πιο πρόσφατο byte δεδομένων που παραλήφθηκε στο TWI.

TWAR-TWI (Slave) Address Register

Bit	7	6	5	4	3	2	1	0
TWAR	**TWA6**	**TWA5**	**TWA4**	**TWA3**	**TWA2**	**TWA1**	**TWA0**	**TWCGE**
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	1	1	1	1	1	1	1	0

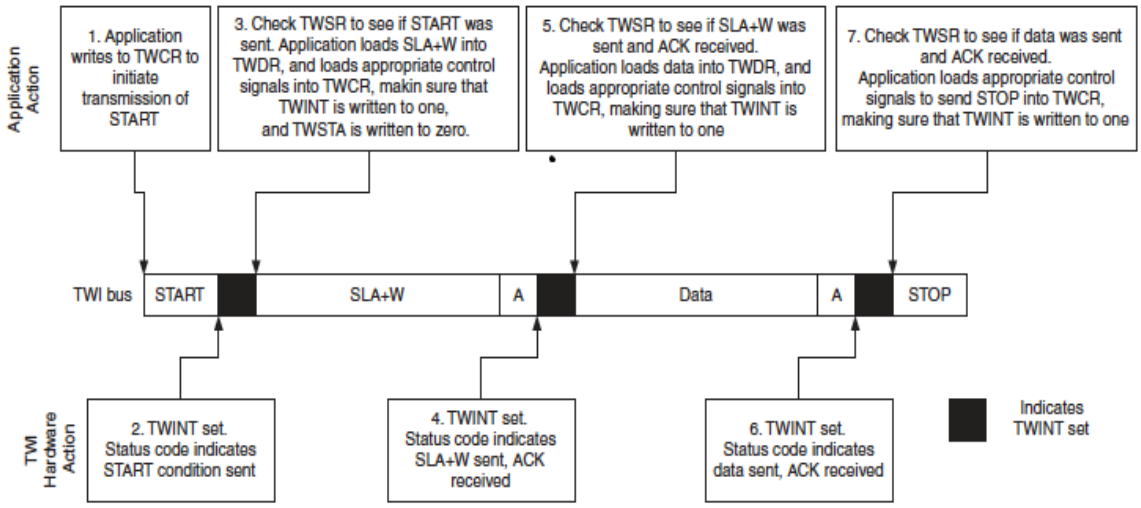
Σχήμα 2.13: Τα bit του καταχωρητή TWAR

Στον καταχωρητή TWAR δίνεται το byte που περιέχει την διεύθυνση του σκλάβου με τον οποίο πρόκειται να επικοινωνήσει.

Bits 7:1-TWA (TWI Slave Address Register): Τα 7 αυτά bit περιέχουν την διεύθυνση του σκλάβου της μονάδας TWI.

Bit 0–TWGCE (TWI General Call Recognition Enable Bit): Το bit αυτό όταν έχει την τιμή '1' ενεργοποιείται ο εντοπισμός γενικής κλήσης που δίνεται από το TWI.

Στην συνέχεια ακολουθεί ένα παράδειγμα (Σχήμα 2.14) ολοκληρωμένης επικοινωνίας μέσω του TWI επιδεικνύοντας την χρήση των παραπάνω καταχωρητών που αναφέρθηκαν για κάθε περίπτωση.



Σχήμα 2.14: Διασύνδεση μίας εφαρμογής με το TWI σε μία τυπική μετάδοση

Γενικά το TWI μπορεί να λειτουργήσει με 4 τρόπους, ως Master Trasnmitter (MT), ως Master Receiver (MR), ως Slave Receiver (SR) ή ως Slave Trasnmitter (ST). Καθένας από αυτούς τους τρόπους μπορεί να χρησιμοποιηθεί σε μία εφαρμογή. Ανάλογα με το πώς λειτουργεί το TWI υπάρχουν και αντίστοιχες τιμές στον καταχωρητή κατάστασης ώστε να μπορεί να γίνει έλεγχος αναγνώρισης σφάλματος σε κάθε διαδικασία που πραγματοποιεί το TWI. Παρακάτω δίνονται οι πίνακες που αναλύουν την σημασία για όλους τους κωδικούς καταστάσεων που προκύπτουν για όλες τις λειτουργίες του TWI.

Status Code στον καταχωρητή TWSR (χωρίς τα prescaler bits)	Η Κατάσταση που βρίσκεται το TWI
0x08	Μία συνθήκη εκκίνησης έχει μεταδοθεί
0x10	Μια συνθήκη επαναλαμβανόμενης εκκίνησης έχει μεταδοθεί
0x18	Ένα σήμα SLA+W έχει μεταδοθεί και έχει ληφθεί το σήμα αναγνώρισης ACK
0x20	Ένα σήμα SLA+W έχει μεταδοθεί και δεν έχει ληφθεί σήμα αναγνώρισης (NOT ACK)
0x28	Έχει μεταδοθεί ένα byte δεδομένων και έχει ληφθεί σήμα αναγνώρισης ACK
0x30	Έχει μεταδοθεί ένα byte δεδομένων και δεν έχει ληφθεί το σήμα αναγνώρισης (NOT ACK)
0x38	Έχει χαθεί η διαιτησία του bus κατά την μετάδοση ενός σήματος SLA+W ή ενός byte δεδομένων

Πίνακας 2.6: Κωδικοί κατάστασης σε λειτουργία Master Transmitter

Status Code στον καταχωρητή TWSR (χωρίς τα prescaler bits)	Η Κατάσταση που βρίσκεται το TWI
0x08	Μία συνθήκη εκκίνησης έχει μεταδοθεί
0x10	Μια συνθήκη επαναλαμβανόμενης εκκίνησης έχει μεταδοθεί
0x38	Έχει χαθεί η διαιτησία κατά την διάρκεια ενός σήματος SLA+R ή δεν έχει ληφθεί το bit μη αναγνώρισης NOT ACK
0x40	Έχει μεταδοθεί ένα σήμα SLA+R και έχει ληφθεί σήμα αναγνώρισης ACK
0x48	Έχει μεταδοθεί ένα σήμα SLA+R και έχει ληφθεί σήμα NOT ACK
0x50	Έχει ληφθεί ένα byte δεδομένων και έχει επιστραφεί ένα σήμα αναγνώρισης ACK
0x58	Έχει μεταδοθεί ένα byte δεδομένων και έχει επιστραφεί σήμα NOT ACK

Πίνακας 2.7: Κωδικοί κατάστασης σε λειτουργία Master Receiver

Status Code στον καταχωρητή TWSR (χωρίς τα prescaler bits)	Η Κατάσταση που βρίσκεται το 2-wire Serial Bus
0xF8	Δεν είναι διαθέσιμη κάποια πληροφορία για την κατάσταση του TWI
0x00	Σφάλμα στο bus κατά την εκκίνηση κάποιας παράνομης συνθήκης εκκίνησης ή τερματισμού.

Πίνακας 2.8: Κωδικοί κατάστασης που ισχύουν σε όλες τις λειτουργίες

Status Code στον καταχωρητή TWSR (χωρίς τα prescaler bits)	Η Κατάσταση που βρίσκεται το TWI
0x60	Το σήμα SLA+W έχει μεταδοθεί και το σήμα αναγνώρισης ACK έχει επιστραφεί
0x68	Η διαιτησία έχει χαθεί κατά την μετάδοση του σήματος SLA+R/W στον αφέντη. Το SLA+W σήμα έχει ληφθεί και το σήμα αναγνώρισης ACK έχει επιστραφεί.
0x70	Η διεύθυνση γενικής κλήσης έχει μεταδοθεί και έχει επιστραφεί σήμα αναγνώρισης ACK
0x78	Η διαιτησία έχει χαθεί κατά την μετάδοση του σήματος SLA+R/W στον αφέντη. Η διεύθυνση γενικής κλήσης έχει ληφθεί. Το σήμα αναγνώρισης ACK έχει επιστραφεί.
0x80	Το προηγούμενο σήμα SLA+W έχει ληφθεί και το σήμα αναγνώρισης ACK έχει επιστραφεί.
0x88	Το προηγούμενο σήμα SLA+W έχει ληφθεί και το σήμα μη αναγνώρισης NOT ACK έχει επιστραφεί.
0x90	Έχει ληφθεί γενική κλήση. Έχουν ληφθεί δεδομένα και σήμα αναγνώρισης ACK έχει ληφθεί.
0x98	Έχει ληφθεί γενική κλήση. Έχουν ληφθεί δεδομένα, σήμα μη αναγνώρισης NOT ACK έχει ληφθεί.
0xA0	Μία συνθήκη τερματισμού ή επαναλαμβανόμενης εκκίνησης έχει μεταδοθεί όσο λειτουργεί ως σκλάβος.

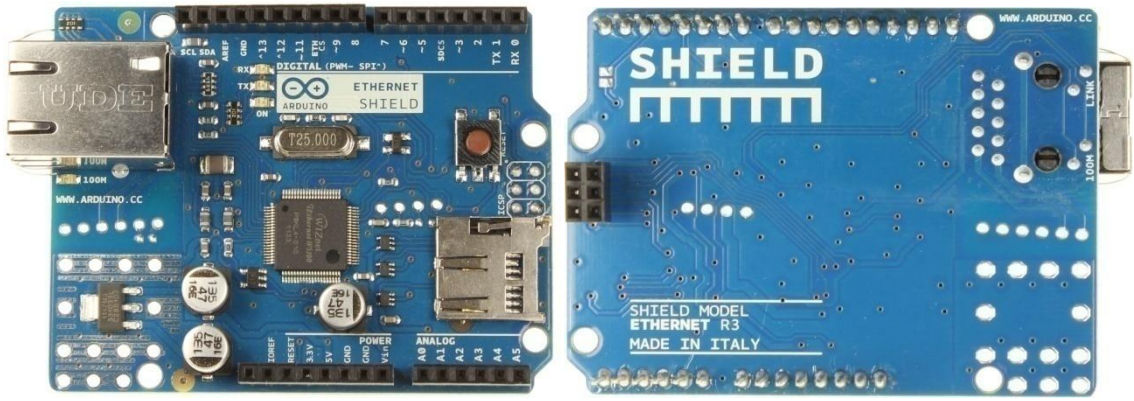
Πίνακας 2.9: Κωδικοί κατάστασης σε λειτουργία Slave Receiver

Status Code στον καταχωρητή TWSR (χωρίς τα prescaler bits)	Η Κατάσταση που βρίσκεται το TWI
0xA8	Το σήμα SLA+R έχει ληφθεί και έχει επιστραφεί το σήμα αναγνώρισης ACK.
0xB0	Έχει χαθεί η διαιτησία κατά την διάρκεια ενός σήματος SLA+R/W στην λειτουργία ως αφέντης. Το σήμα αναγνώρισης ACK έχει ληφθεί.
0xB8	Έχει μεταδοθεί ένα byte δεδομένων και το σήμα αναγνώρισης ACK έχει ληφθεί.
0xC0	Έχει μεταδοθεί ένα byte δεδομένων και το σήμα μη αναγνώρισης NOT ACK έχει ληφθεί.
0xC8	Το τελευταίο byte δεδομένων έχει μεταδοθεί στον καταχωρητή TWDR (TWEA = 0) και το σήμα αναγνώρισης ACK έχει ληφθεί.

Πίνακας 2.10: Κωδικοί κατάστασης σε λειτουργία Slave Transmitter

2.5 Arduino Ethernet Shield

Το Arduino Ethernet Shield είναι μία ηλεκτρονική διάταξη που δίνει την δυνατότητα στο Arduino να συνδεθεί στο διαδίκτυο. Η επικοινωνία της πλακέτας αυτής με το Arduino γίνεται με την χρήση της θύρας SPI. Το Ethernet Shield που χρησιμοποιήθηκε λειτουργεί με τον ελεγκτή Ethernet Wiznet W5100. Η τάση λειτουργίας του είναι στα 5V και παρέχεται από την πλατφόρμα του Arduino. Ο ελεγκτής W5100 διαθέτει την στοίβα δικτύου (IP stack) υποστηρίζοντας έτσι τα πρωτόκολλα TCP (Transmission Control Protocol) και UDP (User Datagram Protocol) με ταχύτητα επικοινωνίας 10/100 Mbps. Το Ethernet Shield μπορεί να υποστηρίξει μέχρι και 5 ταυτόχρονες διαφορετικές συνδέσεις τύπου socket.



Σχήμα 2.15: Πάνω και κάτω όψη του Ethernet Shield

Το Ethernet Shield διαθέτει μία θύρα επικοινωνίας RJ-45 στην οποία συνδέεται το καλώδιο Ethernet, μία ενσωματωμένη υποδοχή για κάρτα μνήμης μεγέθους micro SD και ένα κουμπί επανεκκίνησης. Το Arduino επικοινωνεί και με τον ελεγκτή W5100 και με την SD κάρτα μέσω της SPI επικοινωνίας. Οι ψηφιακοί ακροδέκτες 50,51,52 εξυπηρετούν αυτόν τον σκοπό, καθώς επίσης ο ψηφιακός ακροδέκτης 10, ο οποίος χρησιμοποιείται για να επιλέγει τον ελεγκτή W5100 και ο ψηφιακός ακροδέκτης 4 για την επιλογή της SD κάρτας. Για να επιτευχθεί η επιλογή της κάρτας SD θα πρέπει ο ψηφιακός ακροδέκτης 4 να είναι δηλωμένος ως έξοδος και να βρίσκεται σε υψηλό δυναμικό. Για να επιτευχθεί η επιλογή του ελεγκτή W5100 θα πρέπει ο ψηφιακός ακροδέκτης 10 να είναι δηλωμένος ως έξοδος και να βρίσκεται σε υψηλό δυναμικό.

Το Ethernet Shield διαθέτει επιπλέον και μια σειρά από LED όπου ο ρόλος τους είναι τελείως βοηθητικός με σκοπό την γνωστοποίηση της κατάστασής του όταν βρίσκεται σε λειτουργία.

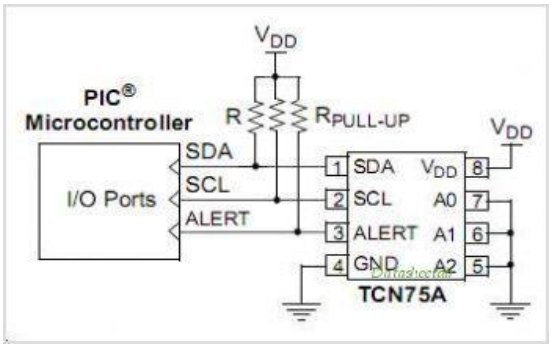
- **PWR:** Δείχνει ότι το Ethernet Shield έχει τροφοδοσία.
- **LINK:** Δείχνει αν υπάρχει συνδεδεμένο καλώδιο δικτύου και επίσης αναβοσβήνει όταν το Ethernet Shield δέχεται ή στέλνει δεδομένα.
- **FULLD:** Δείχνει αν η σύνδεση είναι πλήρως αμφίδρομη.
- **100Mbps:** Δείχνει την ταχύτητα της σύνδεσης στο δίκτυο, αν είναι αναμμένο τότε υποστηρίζεται ταχύτητα 100Mbps.
- **RX:** Αναβοσβήνει όταν το Ethernet Shield λαμβάνει δεδομένα.
- **TX:** Αναβοσβήνει όταν το Ethernet Shield στέλνει δεδομένα.

- **COLL:** Αναβοσβήνει όταν εντοπιστεί κάποια διένεξη στο δίκτυο

2.6 Αισθητήρας θερμοκρασίας TCN75A

Ο ψηφιακός αισθητήρας θερμοκρασίας TCN75A μπορεί να μετατρέψει θερμοκρασίες από -40°C έως και +125°C σε μία ψηφιακή λέξη με τυπικό σφάλμα ±1°C. Ο αισθητήρας αυτός διαθέτει καταχωρητές οι οποίοι είναι διαθέσιμοι για να χρησιμοποιήσει ο χρήστης ανάλογα με τις ανάγκες της εφαρμογής του. Επίσης διαθέτει ανάλυση μέτρησης από 0,5°C μέχρι και 0,0625°C και έναν ακροδέκτη προειδοποίησης alert ο οποίος παράγει ένα σήμα (υψηλού ή χαμηλού δυναμικού το οποίο ελέγχεται από τον χρήστη), όταν η θερμοκρασία ξεπερνά τα όρια που μπορεί να μετρήσει ο αισθητήρας.

Σχήμα 2.16: Ο Αισθητήρας θερμοκρασίας TCN75A



Σχήμα 2.17: Εφαρμογή του αισθητήρα θερμοκρασίας TCN75A

MSOP,SOIC	Ακροδέκτης	Λειτουργία
1	SDA	Σειριακά δεδομένα (είσοδος/έξοδος)
2	SCL	Είσοδος σειριακού ρολογιού
3	ALERT	Προειδοποίηση για τα όρια θερμοκρασίας
4	GND	Γείωση
5	A2	Bit επιλογής διεύθυνσης 2
6	A1	Bit επιλογής διεύθυνσης 1
7	A0	Bit επιλογής διεύθυνσης 0
8	VDD	Είσοδο της τάσης τροφοδοσίας

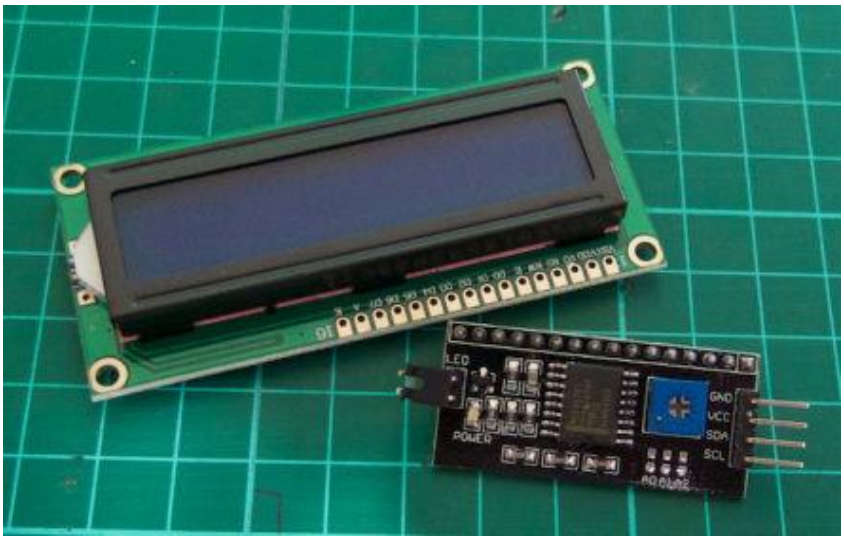
Πίνακας 2.11: Πίνακάς λειτουργιών των ακροδεκτών του TCN75A

Τα γενικά χαρακτηριστικά του είναι:

- Ικανότητα μετατροπής της θερμοκρασίας σε ψηφιακή έξοδο
- Ελάχιστο τυπικό σφάλμα ±1°C και μέγιστο τυπικό σφάλμα ±2°C σε θερμοκρασίες από -40°C έως και +125°C.
- Επιλογή ακρίβειας από τον χρήστη : από 0,5°C έως 0,0625°C
- Τάση λειτουργίας : από 2,7 έως και 5,5 Volts
- Συμβατό με το I2C Interface
- Τυπικό ρεύμα λειτουργίας 200μA
- Ελάχιστο ρεύμα λειτουργίας 2μA
- Ικανότητα μέτρησης σε πάρα πολύ γρήγορο χρόνο
- Διατίθεται σε πακέτα τύπου MSOP-8 και τύπου SOIC-8

2.7 Οθόνη LCD

Στην συγκεκριμένη πτυχιακή χρησιμοποιήθηκε μία οθόνη υγρών κρυστάλλων (Σχήμα 2.18) μεγέθους 16 χαρακτήρων ανά γραμμή και 2 γραμμών (16x2). Για την σύνδεση της οθόνης με το Arduino χρησιμοποιήθηκε το PCF8574 backpack (Σχήμα 2.18), το οποίο χρησιμοποιείται ώστε να είναι δυνατήτ η διασύνδεση της συγκεκριμένης LCD οθόνης μέσω του I2C πρωτοκόλλου. Το ποτενσιόμετρο που διαθέτει το backpack αναλαμβάνει την ρύθμιση της φωτεινότητας της οθόνης. Οι ακρόδεκτες του backpack είναι οι SDA (γραμμή δεδομένων), SCL (γραμμή ρολογιού), VCC (τάση τροφοδοσίας) και GND (γείωση). Η χρήση της οθόνης στο σύστημα που αναπτύχθηκε έγινε με σκοπό την ενημέρωση του χρήστη για τα στάδια που βρίσκεται το σύστημα και για ενημερώσεις διάφορων σφαλμάτων εφόσον αυτά προκύψουν.



Σχήμα 2.18: Η 16x2 LCD οθόνη και το backpack PCF8574

2.8 Αισθητήρας φωτεινότητας BH1750

Ο αισθητήρας φωτεινότητας BH1750 (Σχήμα 2.19) έχει ενσωματωμένο έναν 16 bit AD μετατροπέα. Ο αισθητήρας αυτός λειτουργεί με βάσει το πρωτόκολλο I2C. Επίσης έχει

διαθέσιμους καταχωρητές που μπορούν να προγραμματιστούν από τον χρήστη με σκοπό την παραμετροποίηση της ακρίβειας και τον τρόπο λειτουργίας του. Οι ακροδέκτες του αισθητήρα είναι οι SDA,SCL,ADDR,GND,VCC. Ο ακροδέκτης ADDR δίνει διαφορετική διεύθυνση όταν είναι γειωμένος και όταν είναι υπό τάση.

Σχήμα 2.19: Ο αισθητήρας φωτεινότητας BH1750

Τα γενικά χαρακτηριστικά του αισθητήρα είναι:

- Συμβατός με το I2C bus (3.3V τάση λειτουργίας)
- Η φασματική του απόκριση είναι πολύ κοντά με την αντίδραση των ανθρώπινων ματιών
- Διαθέτει ψηφιακό μετατροπέα σε lumen
- Μεγάλο εύρος και υψηλή ακρίβεια (1-65535 lx)
- Μικρό ρεύμα λειτουργίας
- 50/60 Hz λειτουργία
- 1.8V ελάχιστη τάση εισόδου Interface
- Δεν χρειάζεται εξωτερικά μέρη για να λειτουργήσει
- Η εξάρτιση από την φωτεινή πηγή είναι πολύ μικρή
- Διαθέτει 2 διευθύνσεις σκλάβου για επιλογή από τον χρήστη
- Μικρή μεταβολή μέτρησης (+/- 20%)
- Η επίδραση των υπέρυθρων είναι πολύ μικρή

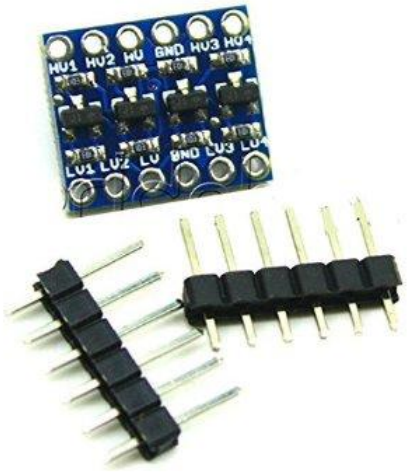
2.9 Logic Level Converter Bi-Directional Module

Το module Logic Level Converter Bi-Directional περιέχει δύο κυκλώματα τα οποία εξυπηρετούν την ανύψωση ή την υποβίβαση της τάσης ενός σήματος. Το κύκλωμα υποβίβασης της τάσης δεν είναι άλλο από την διάταξη ενός διαιρέτη τάσης ο οποίος υποβιβάζει την τάση ενός σήματος κατά 66%. Το κύκλωμα της ανύψωσης της τάσης περιέχει ένα MOSFET. Μέσω του ημιαγωγού αυτού πραγματοποιείται η ανύψωση της τάσης ενός σήματος.

Όπως φαίνεται και στο Σχήμα 2.20 το module αυτό διαθέτει 6 ακροδέκτες στην πάνω πλευρά και 6 στην κάτω πλευρά. Ο ακροδέκτης που συμβολίζεται με HV είναι ο ακροδέκτης της υψηλότερης τάσης τροφοδοσίας ενώ ο ακροδέκτης που συμβολίζεται με LV είναι ο ακροδέκτης της χαμηλότερης τάσης τροφοδοσίας. Οι ακροδέκτες HV1-HV4 και οι ακροδέκτες LV1-LV4 δέχονται τα σήματα που θα υποβιβαστούν και θα ανυψωθούν

αντίστοιχα. Επίσης υπάρχει ένας ακροδέκτης γείωσης (GND) σε κάθε πλευρά. Η γείωση πρέπει να είναι κοινή.

Η συγκεκριμένη πλατφόρμα μπορεί να μετατρέπει σήματα 5V σε 3,3V και το ανάποδο και σήματα 2,8V σε 1,7V και το ανάποδο. Στην συγκεκριμένη πτυχιακή χρησιμοποιήθηκε για να μπορέσει να συνδεθεί ο αισθητήρας φωτεινότητας με τους ακροδέκτες SDA και SCL μιας και η τάση λειτουργίας του δεν ήταν συμβατή με αυτήν που μπορεί να παράγει η πλατφόρμα Arduino.



Σχήμα 2.20: Logic Level Converter Bi-Directional Module

Η πλατφόρμα αυτή υποστηρίζει μέχρι και 4 ταυτόχρονες μετατροπές, μίας και διαθέτει 4 ακροδέκτες HV1-HV4 και 4 LV1-LV4.

Στα σήματα που εισέρχονται στους ακροδέκτες HV1-HV4 πραγματοποιείται υποβιβασμός της τάσης τους κατά 66% και το σήμα με την νέα τάση λαμβάνεται από τον αντίστοιχο ακροδέκτη LV1-LV4.

Στα σήματα που εισέρχονται στους ακροδέκτες LV1-LV4 πραγματοποιείται ανύψωση της τάσης τους και το σήμα με την νέα τάση λαμβάνεται από τον αντίστοιχο ακροδέκτη HV1-HV4.

2.10 Real Time Clock (RTC) DS3231

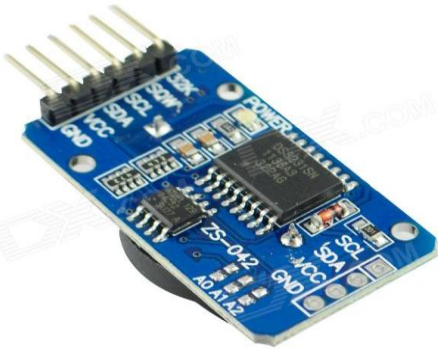
Το DS3231 είναι μία πλακέτα που διαθέτει ένα RTC (DS3231 SN1136A3) μεγάλης ακρίβειας και έναν ταλαντωτή κρύσταλλο αντιστάθμισης θερμοκρασίας (Temperature Compensated Crystal Oscillator, TXCO). Η πλακέτα διαθέτει μία είσοδο μπαταρίας και διατηρεί την λειτουργίας της όταν η κύρια τροφοδοσία της πλακέτας διακόπτεται. Ο κρύσταλλος που διαθέτει εξυπηρετεί στην μακροπρόθεσμη ακρίβεια της συσκευής. Επίσης η πλακέτα DS3231 περιέχει την μνήμη EEPROM 24C32 η οποία είναι μία μνήμη μεγέθους 4KBytes και επικοινωνεί μέσω του πρωτοκόλλου I2C με δική της ανεξάρτητη διεύθυνση σκλάβου.

Το RTC που περιέχει η πλακέτα διατηρεί τα δευτερόλεπτα, τα λεπτά, τις ώρες, την μέρα, την ημερομηνία, τους μήνες καθώς και πληροφορίες χρόνου. Η ημερομηνία στο τέλος του μήνα ρυθμίζεται αυτόματα για τους μήνες που έχουν λιγότερες από 31 ημέρες

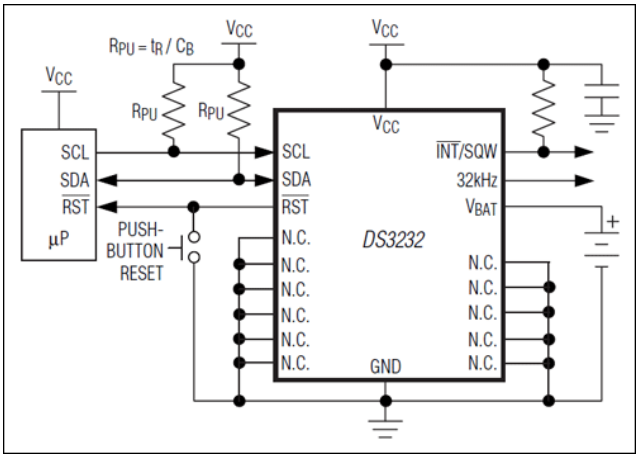
συμπεριλαμβανομένων των διορθώσεων για το δίσεκτο έτος. Το ρολόι μπορεί να λειτουργήσει σε 24-ωρη ή 12-ωρη μορφή με ένα δείκτη AM/PM. Επίσης διαθέτει 2 alarms τα οποία μπορούν να ρυθμιστούν για να ενεργοποιηθούν, με βάση την ώρα, ή ημερομηνία, ή και διάφορους συνδυασμούς αυτών κατά επιθυμία του χρήστη. Επιπλέον περιλαμβάνει μία έξοδο τετραγωνικών παλμών που είναι προγραμματιζόμενη για γενική χρήση. Η διεύθυνση και τα δεδομένα μεταφέρονται σειριακά μέσω του πρωτοκόλλου I2C.

Η πλακέτα DS3231 έχει την δυνατότητα να αναγνωρίσει αυτόματα την απώλεια τροφοδοσίας, με το κύκλωμα σύγκρισης που διαθέτει, και να μεταβεί στην εφεδρική τροφοδοσία όταν είναι απαραίτητο. Τα τεχνικά χαρακτηριστικά της πλακέτας είναι:

- Σχετικό σφάλμα ±2ppm σε λειτουργία από 0°C έως +40°C
- Σχετικό σφάλμα ±3,5ppm σε λειτουργία από -40C έως +85°C
- Σχετικό σφάλμα ±3°C στην μέτρηση θερμοκρασίας
- Διαθέτει 2 alarms
- Μπορεί να παράγει τετραγωνικό κύμα
- Διαθέτει εφεδρική μπαταρία
- Λειτουργεί με ταχύτητα 400kHz με το I2C Interface
- Θερμοκρασία λειτουργίας από -40C έως +85°C
- RST (Reset) έξοδο για την επανεκκίνηση της συσκευής



Σχήμα 2.21: Η πλακέτα του DS3231



Σχήμα 2.22: Παράδειγμα σύνδεσης του RTC DS3231

Το DS3231 διαθέτει καταχωρητές (Πίνακας 2.12) στους οποίους έχει πρόσβαση ο χρήστης και μπορεί να ελέγχει την συσκευή για να πραγματοποιεί διάφορες λειτουργίες όπως να θέτει την ώρα ή την ημερομηνία, να χειριστεί το τετραγωνικό παλμό ή τα alarms καθώς και τον αισθητήρα θερμοκρασίας που διαθέτει.

ADDRESS	BIT 7 MSB	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0 LSB	FUNCTION	RANGE
00h	0	10 Seconds			Seconds				Seconds	00–59
01h	0	10 Minutes			Minutes				Minutes	00–59
02h	0	12/$\overline{24}$	$\overline{AM}$/PM 20 Hour	10 Hour	Hour				Hours	1–12 + $\overline{AM}$/PM 00–23
03h	0	0	0	0	0	Day			Day	1–7
04h	0	0	10 Date		Date				Date	01–31
05h	Century	0	0	10 Month	Month				Month/ Century	01–12 + Century
06h	10 Year				Year				Year	00–99
07h	A1M1	10 Seconds			Seconds				Alarm 1 Seconds	00–59
08h	A1M2	10 Minutes			Minutes				Alarm 1 Minutes	00–59
09h	A1M3	12/$\overline{24}$	$\overline{AM}$/PM 20 Hour	10 Hour	Hour				Alarm 1 Hours	1–12 + $\overline{AM}$/PM 00–23
0Ah	A1M4	DY/$\overline{DT}$	10 Date		Day				Alarm 1 Day	1–7
					Date				Alarm 1 Date	1–31
0Bh	A2M2	10 Minutes			Minutes				Alarm 2 Minutes	00–59
0Ch	A2M3	12/$\overline{24}$	$\overline{AM}$/PM 20 Hour	10 Hour	Hour				Alarm 2 Hours	1–12 + $\overline{AM}$/PM 00–23
0Dh	A2M4	DY/$\overline{DT}$	10 Date		Day				Alarm 2 Day	1–7
					Date				Alarm 2 Date	1–31
0Eh	$\overline{EOSC}$	BBSQW	CONV	RS2	RS1	INTCN	A2IE	A1IE	Control	—
0Fh	OSF	0	0	0	EN32kHz	BSY	A2F	A1F	Control/Status	—
10h	SIGN	DATA	DATA	DATA	DATA	DATA	DATA	DATA	Aging Offset	—
11h	SIGN	DATA	DATA	DATA	DATA	DATA	DATA	DATA	MSB of Temp	—
12h	DATA	DATA	0	0	0	0	0	0	LSB of Temp	—

Πίνακας 2.12: Οι καταχωρητές του DS3231 και η δομή λειτουργίας τους

Κεφάλαιο 3

Λογισμικό που χρησιμοποιήθηκε

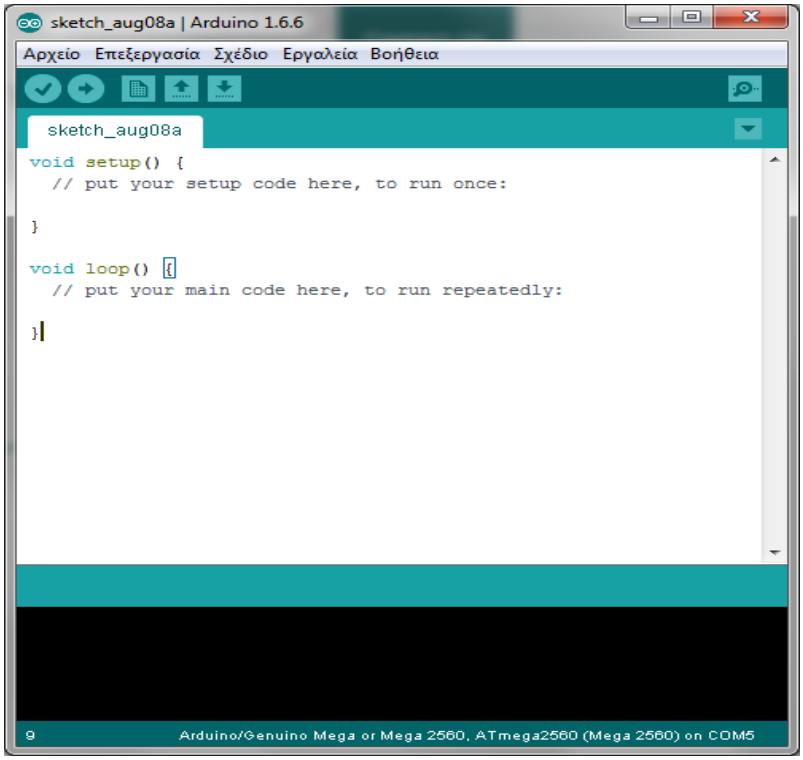
3.1 Εισαγωγή

Στο κεφάλαιο αυτό γίνεται αναφορά στα πακέτα λογισμικού που χρησιμοποιήθηκαν για να την ανάπτυξη της παρούσας πτυχιακής εργασίας, τα οποία είναι το Arduino IDE (Integrated Development Environment), το LabVIEW της National Instruments καθώς και το MySQL Administrator. Στο περιβάλλον του Arduino IDE αναπτύχθηκε το τμήμα του προγράμματος που αφορά την επικοινωνία TWI, την συλλογή μετρήσεων, καθώς και την αποστολή – αποθήκευση των δεδομένων στην βάση μέσω του Ethernet Shield. Το λογισμικό πακέτο LabVIEW χρησιμοποιήθηκε για την δημιουργία του GUI (Graphical User Interface) στο οποίο ο χρήστης μπορεί να χειριστεί τα δεδομένα που βρίσκονται στην βάση για να συγκρίνει διάφορες τιμές με την χρήση κυματομορφών. Το λογισμικό MySQL Administrator αξιοποιήθηκε για την δημιουργία, την διαχείριση και τον έλεγχο της βάσης δεδομένων.

3.2 Περιβάλλον προγραμματισμού του Arduino

Το Arduino IDE είναι μία εφαρμογή γραμμένη σε γλώσσα JAVA η οποία είναι διαθέσιμη για τα λειτουργικά Mac, Linux και Windows. Η γλώσσα προγραμματισμού η οποία χρησιμοποιείται για την συγγραφή προγραμμάτων στο περιβάλλον αυτό είναι η Wiring-Processing (C, C++). Η έκδοση της εφαρμογής Arduino IDE που χρησιμοποιήθηκε για την υλοποίηση αυτής της πτυχιακής είναι η 1.6.6. Τα εργαλεία GNU toolchain και AVR Libc χρησιμοποιούνται από την εφαρμογή αυτή για την μεταγλώττιση των προγραμμάτων από την Wiring-Processing στις κατάλληλες εντολές της γλώσσας μηχανής. Επίσης για την φόρτωση του εκτελέσιμου κώδικα στην μνήμη του μικροελεγκτή χρησιμοποιείται το εργαλείο avrdude.

Στο περιβάλλον αυτό ο χρήστης μπορεί να δημιουργήσει προγράμματα που ονομάζονται sketches και αποθηκεύονται ως αρχεία με επέκταση .ino. Το ίδιο το Arduino IDE (Σχήμα 3.1) παρέχει στην χρήστη ποικίλα παραδείγματα που είναι πολύ χρήσιμα για την κατανόηση του περιβάλλοντος. Επίσης το περιβάλλον ανάπτυξης διαθέτει μια περιοχή επεξεργασίας κειμένου για τη συγγραφή του κώδικα, μια γραμμή εργαλείων με κουμπιά συντομεύσεων καθώς και ένα μενού επιλογών. Στο κάτω μέρος βρίσκεται μια περιοχή στην οποία εμφανίζονται μηνύματα τα οποία σχετίζονται με τυχόν σφάλματα που προέκυψαν κατά την σύνταξη του κώδικα ή σφάλματα που προέκυψαν κατά την μεταγλώττιση του κώδικα καθώς και μηνύματα σχετικά με τα χαρακτηριστικά του κώδικα και του υλικού που χρησιμοποιείται, όπως το μέγεθος του κώδικα και πόσο χώρο καταλαμβάνει. Επίσης κάτω δεξιά στο περιβάλλον ανάπτυξης εμφανίζεται η σειριακή θύρα που είναι συνδεμένη η πλατφόρμα που επικοινωνεί καθώς και το όνομα της πλατφόρμας αυτή (π.χ Arduino Mega 2560 COM 9). Η γραμμή εργαλείων έχει επιλογές για την δημιουργία νέου sketch, την αποθήκευσή του και το άνοιγμα νέου sketch. Επίσης υπάρχουν επιλογές για τον έλεγχο και φόρτωση sketch στην πλατφόρμα που έχει επιλεχθεί, καθώς και επιλογή για την σειριακή θύρα που θα επικοινωνήσει.



Σχήμα 3.1: Το περιβάλλον Arduino IDE

Το περιβάλλον Arduino IDE είναι αρκετά φιλικό και οικείο προς τον χρήστη γιατί το αναπτυξιακό του περιβάλλον θυμίζει σημειωματάριο και χρησιμοποιεί την λογική του τετραδίου (Sketchbook). Την πρώτη φορά που εκτελείται το Arduino IDE δημιουργείται αυτόματα ένας κατάλογος sketchbook στον οποίο ο χρήστης μπορεί να αποθηκεύσει τα προγράμματα (sketches) που ανέπτυξε. Πριν την εκτέλεση ενός προγράμματος θα πρέπει να γίνεται η επιλογή των βασικών παραμέτρων όπως, της σειριακής θύρας και της σωστής πλατφόρμας. Η επιλογή αυτή γίνεται από το μενού Εργαλεία→Πλακέτα και Εργαλεία→ Σειριακή Θύρα αντίστοιχα. Εφόσον έχουν ολοκληρωθεί οι παραπάνω ρυθμίσεις ο χρήστης έχει την δυνατότητα επιλέγοντας upload να φορτώσει στην πλατφόρμα το πρόγραμμα που ανέπτυξε. Κατά την φόρτωση του προγράμματος η πλατφόρμα κάνει επανεκκίνηση και φορτώνει τον νέο εκτελέσιμο κώδικα. Στην περιοχή μηνυμάτων θα εμφανιστεί εάν η διαδικασία ολοκληρώθηκε επιτυχώς ή αν υπήρξε σφάλμα, καθώς και τι σφάλμα είναι αυτό.

Οι συντομεύσεις που υπάρχουν στο αναπτυξιακό περιβάλλον του Arduino IDE είναι:

- **Επαλήθευση (Verify)→** Ελέγχει το πρόγραμμα για τυχόν συντακτικά λάθη(Compile)
- **Φόρτωση (Upload)→**Μεταγλωττίζει το πρόγραμμα και φορτώνει τον εκτελέσιμο κώδικα στην πλατφόρμα που έχει επιλεχθεί.
- **Νέο (New)→** Δημιουργεί ένα νέο πρόγραμμα
- **Άνοιγμα (Open)→** Παρουσιάζει τα υπάρχων προγράμματα στον φάκελο Sketchbook για να επιλέξει ο χρήστης.
- **Αποθήκευση (Save)→** Αποθηκεύει το τρέχον πρόγραμμα
- **Σειριακή οθόνη (Serial Monitor)→** Ενεργοποιεί το Serial Monitor

Παρόλα αυτά υπάρχουν και πρόσθετες επιλογές που είναι διαθέσιμες στο χρήστη που αντιστοιχούν στα μενού εργαλείων και τις αντίστοιχες επιλογές. Κάποιες από τις βασικές επιλογές είναι:

Επεξεργασία(Edit)

- Copy for Forum: Αντιγράφει τον κώδικα από το πρόγραμμα στο πρόχειρο σε κατάλληλη μορφή προς δημοσίευση στο forum του Arduino.
- Copy as HTML: Αντιγράφει το πρόγραμμα στο πρόχειρο σε μορφή HTML που είναι κατάλληλη για την ενσωμάτωση του σε ιστοσελίδες.
- Επιπλέον διαθέτει και πρόσθετες επιλογές όπως αντιγραφή, αποκοπή, αναζήτηση, επιλογή όλων, αναίρεση κλπ.

Σκίτσο (Sketch)

- Compile – Verify: Ελέγχει το πρόγραμμα για τυχόν σφάλματα. Επίσης υπολογίζει το χώρο μνήμης που δεσμεύει το πρόγραμμα και το εμφανίζει στο παράθυρο μηνυμάτων.
- Upload: Μεταγλωττίζει το πρόγραμμα και φορτώνει το μεταγλωττισμένο κώδικα στην επιλεγμένη πλατφόρμα μέσω της σειριακής θύρας.
- Show Sketch Folder: Ανοίγει τον τρέχοντα φάκελο που είναι αποθηκευμένο το πρόγραμμα.
- Add File: Προσθέτει ένα νέο αρχείο στο τρέχον πρόγραμμα το οποίο εμφανίζεται σε μια νέα καρτέλα.
- Include Library: Συμπεριλαμβάνει μία βιβλιοθήκη στο πρόγραμμα χρησιμοποιώντας την δήλωση "#include"

Εργαλεία (Tools)

- Auto Format: Μορφοποιεί αυτόματα το κώδικα. Για παράδειγμα αυτόματα στοιχίζει τον κώδικα που βρίσκεται μέσα στα άγκιστρα που δηλώνουν την αρχή και το τέλος του κώδικα.
- Archive Sketch: Αρχειοθετεί ένα αντίγραφο του τρέχοντος προγράμματος σε μορφή (.zip). Το αρχείο αυτό αποθηκεύεται στον ίδιο φάκελο με το πρόγραμμα.
- Board: Επιλογή της αναπτυξιακής πλατφόρμα που θα χρησιμοποιηθεί.
- Serial Port: Επιλογή της σειριακής θύρας μέσω της οποίας θα γίνει η σύνδεση με την διαθέσιμη αναπτυξιακή πλατφόρμα.
- Programmer: Επιλογή του Hardware Programmer που θα χρησιμοποιηθεί για την φόρτωση του κώδικα. Συνήθως χρησιμοποιείται ο onboard programmer της επιλεγμένης αναπτυξιακής πλατφόρμας. Η επιλογή του κατάλληλου programmer είναι απαραίτητη όταν πρόκειται να φορτωθεί ο bootloader σε έναν νέο μικροελεγκτή.
- Burn Bootloader: Η επιλογή αυτή επιτρέπει την εγγραφή ενός bootloader στο μικροελεγκτή. Αυτό δεν είναι απαραίτητο, αλλά είναι χρήσιμο στη περίπτωση που έχει αγοραστεί ένας νέος μικροελεγκτής ATmega, ο οποίος συνήθως δεν έχει προεγκατεστημένο τον συγκεκριμένο bootloader.

Help (βοήθεια)

- Στο παράθυρο τις βοήθειας παρέχεται βοήθεια για το συγκεκριμένο περιβάλλον προγραμματισμού.

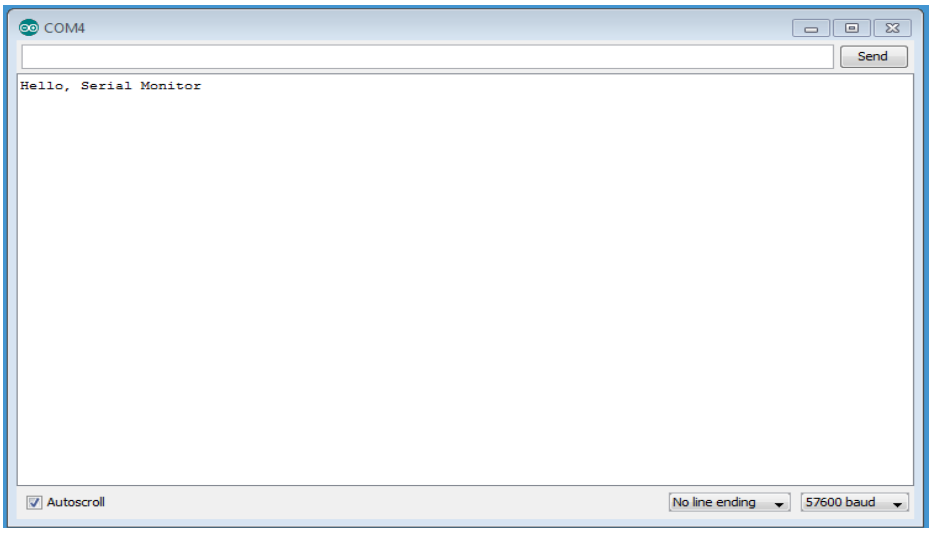
3.2.1 Οι βιβλιοθήκες για το Arduino IDE

Οι βιβλιοθήκες είναι αρχεία γραμμένα σε C ή C++ (.c, .cpp) τα οποία παρέχουν στα προγράμματα επιπλέον λειτουργικότητα μιας και περιέχουν έτοιμες συναρτήσεις για συγκεκριμένες λειτουργίες (π.χ την επικοινωνία και τον έλεγχο ενός αισθητήρα θερμοκρασίας). Για να χρησιμοποιηθεί μία υπάρχουσα βιβλιοθήκη σε κάποιο πρόγραμμα θα πρέπει πρώτα να εισαχθεί στο Arduino IDE. Η εισαγωγή της βιβλιοθήκης γίνεται από το μενού εργαλείων Σκίτσο→Προσθήκη βιβλιοθήκης και επιλογή της επιθυμητής βιβλιοθήκης. Μόλις γίνει η προσθήκη θα εισαχθεί μία δήλωση #include στο πάνω μέρος του πρόγραμματος για κάθε αρχείο κεφαλίδας (.h) στο φάκελο της βιβλιοθήκης. Οι δηλώσεις αυτές κάνουν τις δημόσιες συναρτήσεις και τις σταθερές της βιβλιοθήκης διαθέσιμες στο συγκεκριμένο πρόγραμμα. Επίσης σηματοδοτούν το περιβάλλον Arduino IDE να συνδέσει τον κώδικα της βιβλιοθήκης με το συγκεκριμένο πρόγραμμα κατά την μεταγλώττιση και την παραγωγή του εκτελέσιμου κώδικα.

Για την προσθήκη μίας βιβλιοθήκης που έχει δημιουργηθεί από τον χρήστη θα πρέπει αρχικά στον φάκελο που βρίσκονται οι υπάρχουσες βιβλιοθήκες να δημιουργηθεί ένας νέος φάκελος με το όνομα της βιβλιοθήκης του χρήστη. Στον φάκελο αυτό θα πρέπει να προστεθούν ένα ή περισσότερα αρχεία γραμμένα σε C ή C++ που θα περιέχουν τον αντίστοιχο κώδικα και τουλάχιστον ένα αρχείο κεφαλίδας με τις δηλώσεις των γενικών μεταβλητών και των συναρτήσεων. Επιπλέον επειδή οι βιβλιοθήκες θα μεταγλωττιστούν μαζί με το πρόγραμμα του χρήστη θα αυξηθεί ο όγκος που καταλαμβάνει το πρόγραμμα στην μνήμη της πλατφόρμας Arduino..

3.2.2 Serial Monitor

Το Arduino IDE παρέχει μία σειριακή οθόνη για την απεικόνιση των δεδομένων που αποστέλλονται μέσω της σειριακής θύρας της πλατφόρμας Arduino. Επίσης μπορεί να χρησιμοποιηθεί για να σταλθούν δεδομένα σειριακά στο Arduino κάνοντας Send το επιθυμητό μήνυμα. Κάθε φορά που ανοίγει το Serial Monitor γίνεται επανεκκίνηση στην συνδεδεμένη πλατφόρμα Arduino. Όταν ανοίγει το Serial Monitor θα πρέπει να ελέγχεται ο ρυθμός μεταφοράς δεδομένων (baud rate) ώστε να είναι ίδιος με αυτόν που έχει δηλωθεί.



Σχήμα 3.2: Το Serial Monitor του Arduino IDE

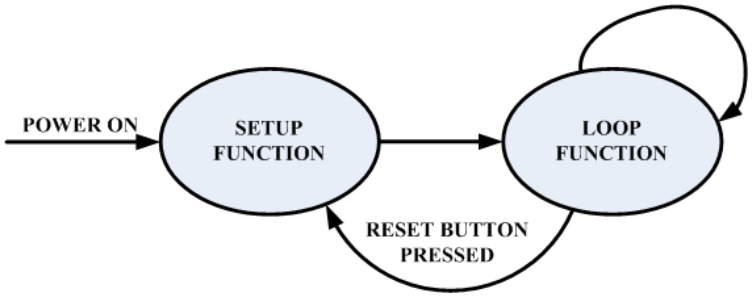
3.2.3 Δομή κώδικα στο Arduino IDE

Τα προγράμματα στο περιβάλλον Arduino IDE χωρίζονται σε 3 βασικά μέρη (Πρόγραμμα 3.1). Το πρώτο μέρος είναι προσθήκη των βιβλιοθηκών και η δήλωση των γενικών μεταβλητών (Global variables). Στο δεύτερο μέρος γίνονται οι απαραίτητες αρχικοποιήσεις και βρίσκονται μέσα στα όρια της συνάρτησης setup. Στο τρίτο μέρος γράφεται το κυρίως πρόγραμμα και βρίσκεται μέσα στα όρια της συνάρτησης loop. Για να μπορεί να εκτελεστεί ένα πρόγραμμα θα πρέπει να έχει και τα 3 μέρη ακόμα και αν είναι κενά. Επίσης το τμήμα του κώδικα ή το κείμενο που βρίσκεται μεταξύ των συμβόλων "/*" και "*/" ή μετά από το σύμβολο "//" αποτελεί σχόλιο του προγράμματος και δεν λαμβάνεται υπόψη κατά την μεταγλώττιση και την παραγωγή του εκτελέσιμου κώδικα που θα φορτωθεί στον μικροελεγκτή.

```
//δήλωση γενικών μεταβλητών
Και προσθήκη βιβλιοθηκών
Void setup()
{
//αρχικοποιήσεις
}
Void loop()
{
//Κώδικας
}
```

Πρόγραμμα 3.1:Η δομή του προγράμματος στο Arduino IDE

Κατά την εκτέλεση του προγράμματος συμφώνα με το Σχήμα 3.3 όταν ο μικροελεγκτής τεθεί υπό τάση θα εκτελέσει πρώτα την συνάρτηση setup και στην συνέχεια θα εκτελεί κυκλικά την συνάρτηση loop. Στην περίπτωση που πατηθεί το μπουτόν reset ο μικροελεγκτής θα σταματήσει την εκτέλεση της συνάρτησης loop και θα εκτελέσει την συνάρτηση setup. Στην συνέχεια θα εκτελεί κυκλικά την συνάρτηση loop.



Σχήμα 3.3: Το διάγραμμα ροής στο Arduino IDE

Συνάρτηση void setup()

Η συνάρτηση setup τρέχει μόνο μία φορά στην αρχή του κώδικα και δεν επιστρέφει κάποιο αποτέλεσμα. Ο κώδικας που βρίσκεται ανάμεσα στα όρια της συνάρτησης αυτής θα τρέξει μόνο μία φορά. Η συνάρτηση αυτή δεν μπορεί να κληθεί από άλλη συνάρτηση. Στον κώδικα της setup συνήθως γίνονται οι αρχικοποιήσεις συσκευών που είναι συνδεδεμένες στην πλατφόρμα Arduino καθώς και άλλες αρχικοποιήσεις.

Συνάρτηση void loop()

Η συνάρτηση loop εκτελείται αμέσως μετά το τέλος της setup και σε αυτή περιγράφεται η κύρια λειτουργία του προγράμματος. Ο κώδικας που βρίσκεται μέσα στην loop εκτελείται συνεχώς από τον μικροελεγκτή.

Γενικές Συναρτήσεις

Υπάρχουν συναρτήσεις οι οποίες είναι έτοιμες και παρέχονται από την ομάδα σχεδιασμού του Arduino. Τα ορίσματα της συνάρτησης αναγράφονται μέσα στις παρενθέσεις που ακολουθούν το όνομα της συνάρτησης. Ο κώδικας της συνάρτησης βρίσκεται μέσα σε αγκύλες. Δίνεται επίσης. η δυνατότητα στον χρήστη να δημιουργήσει τις δικές του συναρτήσεις ανάλογα με τις ανάγκες του.

Μεταβλητές

Οι μεταβλητές χωρίζονται σε 2 βασικές κατηγορίες τις τοπικές και τις γενικές μεταβλητές. Οι τοπικές μεταβλητές ορίζονται μέσα σε μία συνάρτηση και δεν μπορούν να αξιοποιηθούν σε οποιαδήποτε συνάρτηση που περιλαμβάνεται στο πρόγραμμα γιατί η χρήση τους περιορίζεται μέσα στην συνάρτηση αυτή. Οι γενικές μεταβλητές χρησιμοποιούνται για την αποθήκευση των δεδομένων και είναι διαθέσιμες σε όλες τις συναρτήσεις του προγράμματος. Οι τύποι δεδομένων που μπορούν να αποθηκευτούν σε μια μεταβλητή είναι ακέραιοι αριθμοί με ή χωρίς πρόσημο, αριθμοί κινητής υποδιαστολής, σειρές αλφαριθμητικών χαρακτήρων κ.α.

3.3 Εισαγωγή στο LabVIEW

Το LabVIEW είναι ένα περιβάλλον γραφικού προγραμματισμού το οποίο αναπτύχθηκε από την National Instruments. Η γλώσσα προγραμματισμού που χρησιμοποιεί, ονομάζεται γλώσσα G (Graphical Language) και είναι ένα είδος γλώσσας VPL (Visual Programming Language). Αυτό σημαίνει πως παρέχει στον χρήστη την δυνατότητα να προγραμματίζει με τον χειρισμό γραφικών εικονιδίων των οποίων η διασύνδεση επιτυγχάνεται μέσω καλωδίων (wires) τα οποία επιτρέπουν την μεταφορά των δεδομένων .Το LabVIEW μπορεί να εκτελέσει πολλές εντολές παράλληλα με τη βοήθεια του multithreading.

Το LabVIEW ξεκίνησε ως περιβάλλον για την υλοποίηση εφαρμογών που είχαν να κάνουν με την συλλογή και επεξεργασία δεδομένων καθώς και με εφαρμογές επικοινωνίας και ελέγχου οργάνων χρησιμοποιώντας διάφορα πρωτόκολλα επικοινωνίας. Το LabVIEW περιλαμβάνει εκτεταμένη υποστήριξη για διασύνδεση με συσκευές, όργανα, κάμερες και άλλες συσκευές. Ακόμα, διαθέτει πολλά γραφικά αντικείμενα που παρέχονται μέσω διαφορών πακέτων επέκτασης του LabVIEW. Επίσης το LabVIEW διαθέτει πολλές βιβλιοθήκες που περιέχουν ένα μεγάλο αριθμό συναρτήσεων που αναλαμβάνουν την συλλογή δεδομένων, την δημιουργία σημάτων, την εκτέλεση μαθηματικών πράξεων, την επεξεργασία σημάτων, την ανάλυση σημάτων κλπ.

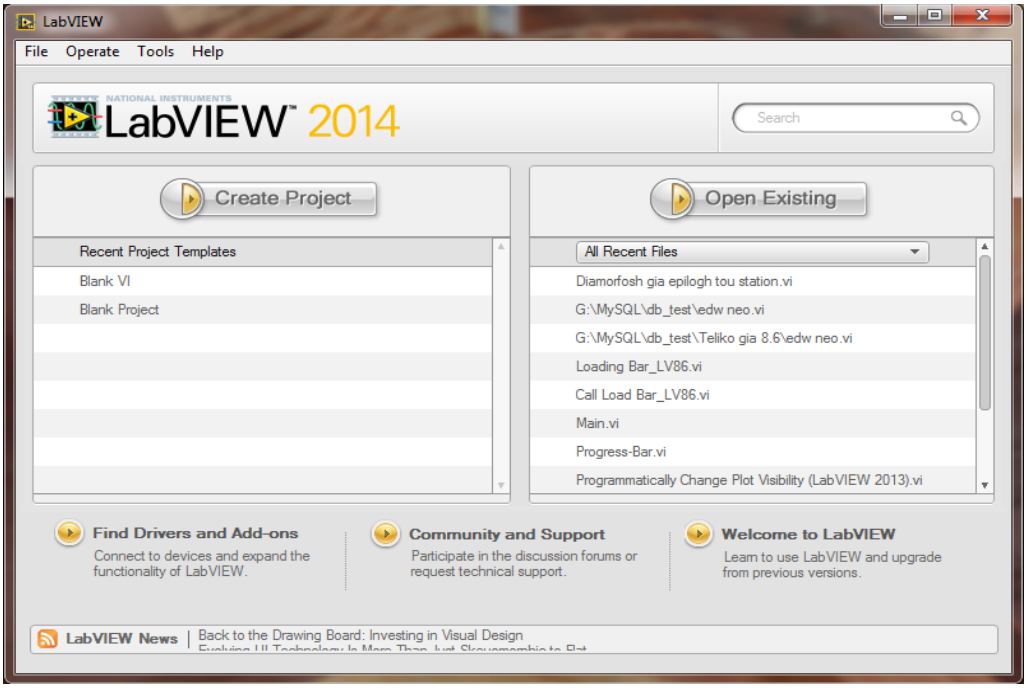
Μία εφαρμογή που πραγματοποιείται στο LabVIEW ονομάζεται VI (Virtual Instrument) και αποτελείται από δύο τμήματα. Το πρώτο τμήμα είναι το Front Panel που ο χρήστης δημιουργεί το GUI (Graphical User Interface) του VI. Το δεύτερο τμήμα είναι το Block Diagram που αποτελεί τον κώδικα του VI και μπορεί να περιέχει βρόχους επανάληψης, συναρτήσεις, subVIs (τα οποία είναι VIs που καλούνται από το συγκεκριμένο VI), τερματικά εικονίδια κα. Ο γραφικός προγραμματισμός επιτρέπει σε χρήστες οι οποίοι δεν είναι ιδιαίτερα

εξοικειωμένοι με κάποια κλασσική γλώσσα προγραμματισμού να υλοποιούν τα προγράμματα που επιθυμούν με την μεταφορά και διασύνδεση εικονιδίων.

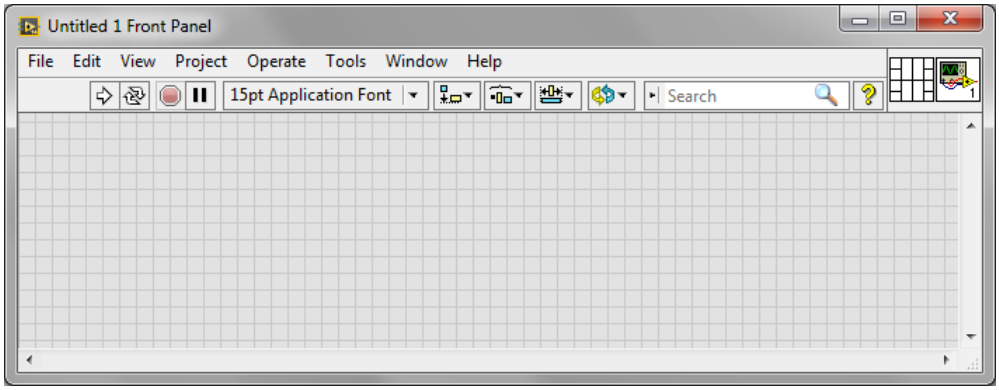
Το περιβάλλον προγραμματισμού LabVIEW θεωρείται απλό στην χρήση για την δημιουργία μικρών εφαρμογών από νέους χρήστες. Στις παραγράφους που ακολουθούν θα γίνει πιο αναλυτική παρουσίαση του LabVIEW 2014 που χρησιμοποιήθηκε στην παρούσα πτυχιακή.

3.3.1 Το πρόγραμμα LabVIEW 2014

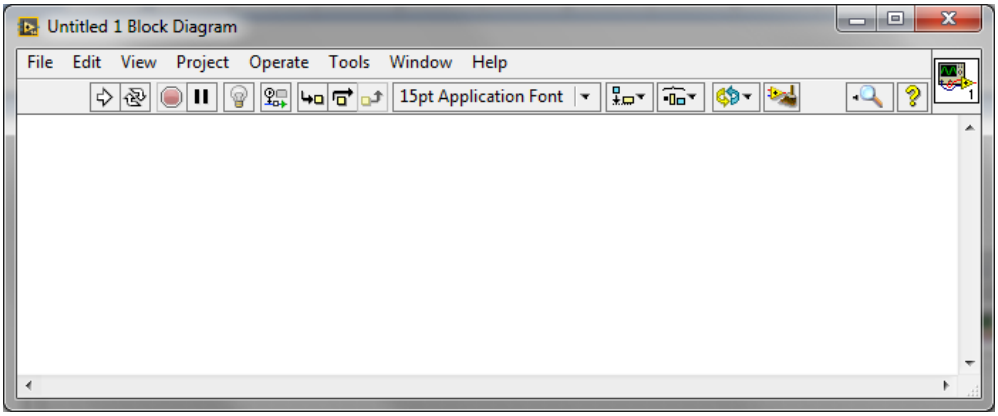
Κατά την εκκίνηση του LabVIEW ανοίγει το παράθυρο Getting Started (Σχήμα 3.4) που πρόκειται για το αρχικό μενού. Στο αρχικό μενού αριστερά υπάρχουν οι επιλογές για το VI που θα χρησιμοποιηθεί ή θα δημιουργηθεί. Δεξιά υπάρχουν επιλογές για παραδείγματα και τα πρόσφατα αρχεία που έχουν δημιουργηθεί. Στο κάτω μέρος υπάρχουν επιλογές για τα διαθέσιμα updates, τα διαθέσιμα forums για εκπαίδευση καθώς και άλλες πληροφορίες σχετικά με το LabVIEW. Για την δημιουργία ενός καινούριου VI υπάρχει η επιλογή Blank VI. Μόλις επιλεχθεί το Blank VI θα ανοίξουν δύο νέα παράθυρα. Το πρώτο παράθυρο είναι το Front Panel (Σχήμα 3.5) και το δεύτερο παράθυρο είναι το Block Diagram (Σχήμα 3.6).



Σχήμα 3.4: Το μενού έναρξης του LabVIEW 2014



Σχήμα 3.5: Το παράθυρο του Front Panel του LabVIEW 2014



Σχήμα 3.6: Το παράθυρο του Block Diagram του LabVIEW 2014

Κάθε ένα από αυτά τα παράθυρα διαθέτει μία γραμμή εργαλείων. Οι πλέον βασικές επιλογές των γραμμών εργαλείων του Front Panel και του Block Diagram είναι:

→ Θέτει σε λειτουργία το VI
→Σταματά την εκτέλεση του VI
→Θέτει σε συνεχή λειτουργία το VI
→ Διακόπτει την εκτέλεση του VI
13pt Application Font → Μέγεθος και είδος γραμματοσειράς
→ Στοίχιση των αντικειμένων
→ Κατανομή των αντικειμένων
→ Αλλαγή των διαστάσεων των αντικειμένων
→ Καθορισμό της σειράς επικάλυψης των αντικειμένων
→ Ενεργοποίηση/Απενεργοποίηση του Context Help
→ Εμφάνιση της ροής δεδομένων στο Block Diagram
→ Εκτέλεση της επόμενης συνάρτησης ή SubVI ακόμα και μέσα σε μία δομή
→ Εκτέλεση της επόμενης συνάρτησης, δομής ή SubVI
→ Επιστροφή στην κανονική εκτέλεση του VI

Στο παράθυρο Context Help εμφανίζεται μια σύντομη περιγραφή της επιλεγμένης από τον χρήστη συνάρτησης αναλύοντας τις διαθέσιμες εισόδους και εξόδους. Επίσης δίνει μια συντόμευση για την πλήρη περιγραφή της συγκεκριμένης συνάρτησης.

Front Panel

Στο Front Panel ο χρήστης έχει την δυνατότητα να δημιουργήσει το γραφικό περιβάλλον του VI ανάλογα με τις ανάγκες της εφαρμογής του. Με δεξί κλικ στο Front Panel εμφανίζεται η Controls Palette που προσφέρει στον χρήστη τα απαραίτητα αντικείμενα για την ανάπτυξη του γραφικού περιβάλλοντος. Όταν εισάγεται ένα αντικείμενο στο Front Panel, τότε στο Block Diagram δημιουργείται το αντίστοιχο τερματικό εικονίδιο.

Block Diagram

Στο Block Diagram ο χρήστης αναπτύσσει τον κώδικα του VI. Εκεί βρίσκονται όλα τα τερματικά εικονίδια των εισόδων και εξόδων τα οποία μεταφέρουν τα δεδομένα από το Block Diagram στο Front Panel και αντίστροφα. Μέσω της Function Palette είναι διαθέσιμα στον χρήστη όλες οι απαραίτητες συναρτήσεις για την ανάπτυξη της εφαρμογής.

Tools Pallete

Η παλέτα εργαλείων (tools pallete) που παρουσιάζεται στο Σχήμα 3.7 είναι κοινή στο Front Panel και στο Block Diagram και επιτρέπει στον χρήστη να καθορίσει την λειτουργία του κέρσορα του ποντικιού όπως περιγράφεται παρακάτω.

Σχήμα 3.7: Παλέτα εργαλείων του LabVIEW

 →Automatic Tool Selection

 → Operate Value

 → Position/Size/Select

 → Edit Text

 → Connect Wire

 → Probe Data

 → Set Color

 → Get Color

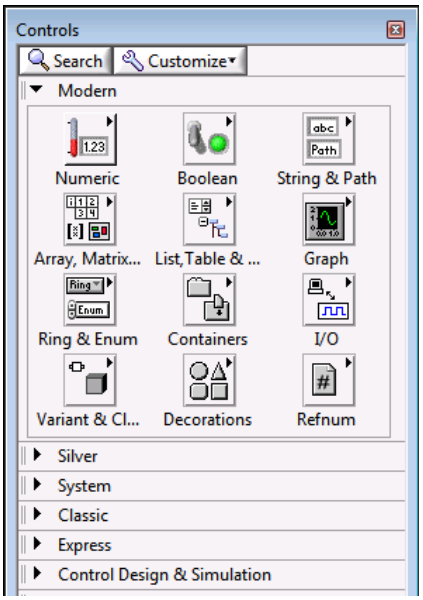
 → Set/Clear Breakpoint

 → Scroll Window

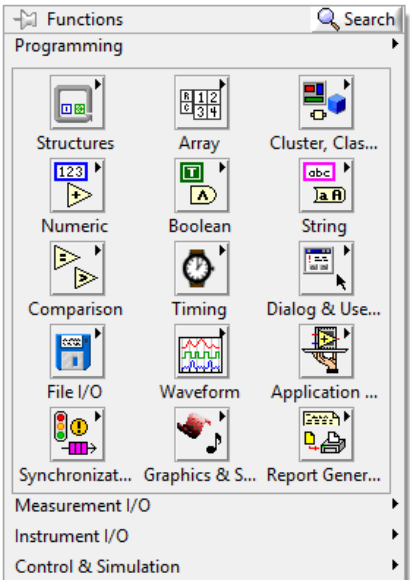
 → Object Shortcut Menu

Controls Palette του Front Panel

Στην Controls Palette (Σχήμα 3.8) υπάρχουν όλα τα διαθέσιμα αντικείμενα Controls και Indicators τα οποία είναι ομαδοποιημένα σύμφωνα με το είδος τους, για την δημιουργία του Front Panel. Η εισαγωγή αυτών των στοιχείων γίνεται με την Drag and Drop διαδικασία.



Σχήμα 3.8: Controls Palette του LabVIEW

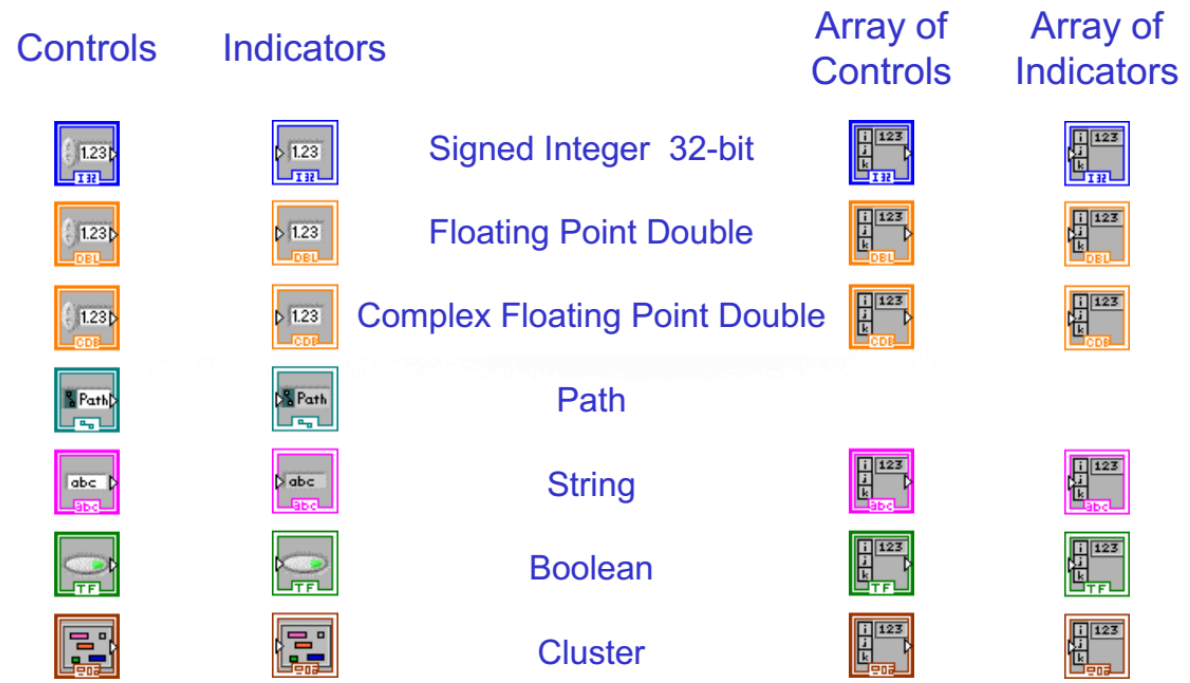


Σχήμα 3.9: Function Palette του LabVIEW

Function Palette του Block Diagram

Όπως και στο Front Panel έτσι και στο Block Diagram οι συναρτήσεις που είναι διαθέσιμες καθώς και τα διαθέσιμα SubVIs για την δημιουργία του Block Diagram περιέχονται στην Function Palette (Σχήμα 3.9) και εισάγονται στο Block Diagram με την Drag and Drop διαδικασία. Στην συγκεκριμένη παλέτα όλες οι συναρτήσεις είναι ομαδοποιημένες ανάλογα με το είδος τους.

Οι τύποι δεδομένων που υποστηρίζονται από το LabVIEW και οι συμβολισμοί τους παρουσιάζονται στο Σχήμα 3.10.



Σχήμα 3.10: Τύποι δεδομένων του LabVIEW

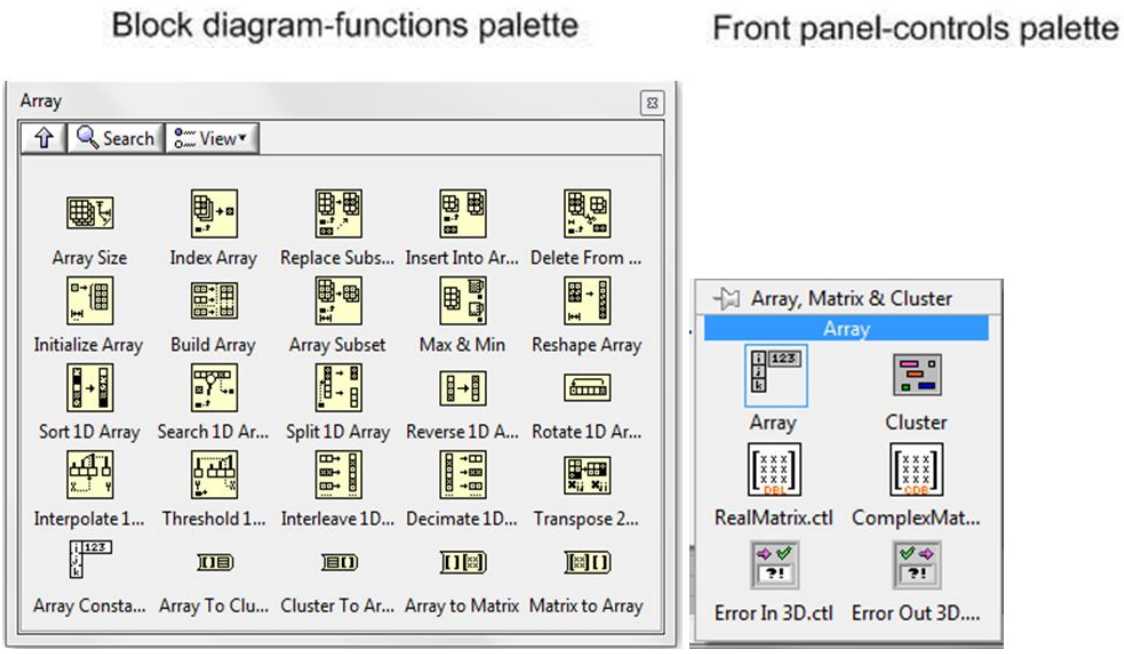
3.3.2 Οι συναρτήσεις του LabVIEW

Το σύνολο των συναρτήσεων του LabVIEW είναι διαθέσιμο μέσα από την Function Palette. Η ομαδοποίηση αυτών των συναρτήσεων γίνεται σε κατηγορίες με βάση το είδος των δεδομένων που διαχειρίζονται. Κάποιες από τις πιο βασικές κατηγορίες και σχετικές με την παρούσα πτυχιακή εργασία είναι:

1. Strings
2. Arrays
3. Loop Structures
4. Case Structures
5. Flat Sequence Structures
6. Event Structures
7. Clusters
8. Graphs
9. SubVIs

Arrays

Οι πίνακες (Arrays) ομαδοποιούν δεδομένα ίδιου τύπου (π.χ αριθμητικά, αλφαριθμητικά, κ.α) και μπορούν να έχουν παραπάνω από μία διαστάσεις. Το LabVIEW 2014 διαθέτει μία μεγάλη γκάμα συναρτήσεων και SubVIs σχετικά με τα Arrays. Στο σχήμα 3.11 παρουσιάζονται οι διαθέσιμες συναρτήσεις του Block Diagram και τα αντικείμενα του Front Panel που ανήκουν στην κατηγορία Array.



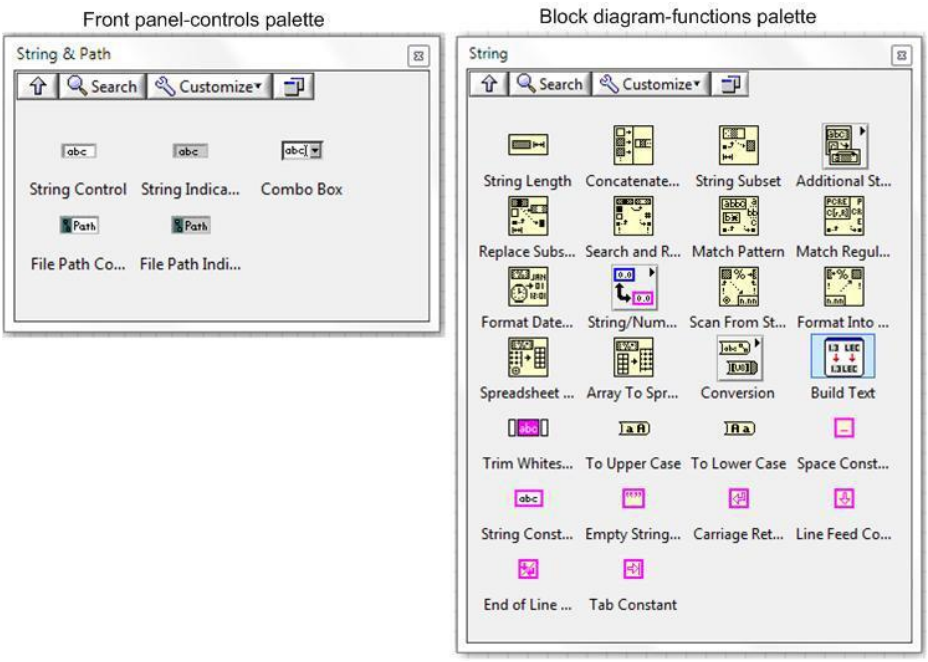
Σχήμα 3.11: Array συναρτήσεις του LabVIEW

Strings

Τα Strings είναι ακολουθίες ASCII χαρακτήρων και συνήθως χρησιμοποιούνται για την προβολή μηνυμάτων ή την αποστολή εντολών σε κάποιο όργανο. Η γκάμα των συναρτήσεων String του LabVIEW είναι αρκετά μεγάλη και παρέχει μεγάλες δυνατότητες στον χρήστη όσον αφόρα την διαχείριση των Strings. Στο σχήμα 3.12 παρουσιάζονται οι διαθέσιμες συναρτήσεις του Block Diagram και τα αντικείμενα του Front Panel που ανήκουν στην κατηγορία String.

Loop Structures

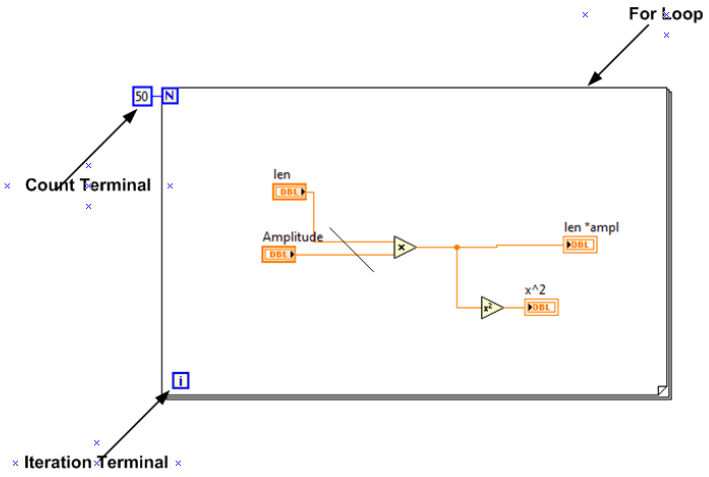
Οι Loop Structures είναι βρόχοι επανάληψης οι οποίοι χρησιμοποιούνται για την επανάληψη κάποιων διαδικασιών στο VI. Υπάρχουν διαφορετικοί βρόχοι επανάληψης και καθένας από αυτούς εκτελεί διαφορετικές λειτουργίες. Τα είδη των υποστηριζόμενων βρόχων επανάληψης είναι η While Loop, For Loop και Timed Loop. Παρακάτω δεν αναφέρεται πιο αναλυτικά η Timed Loop μιας και χρησιμοποιήθηκαν μόνο οι δύο πρώτοι βρόχοι επανάληψης στην ανάπτυξη αυτής της πτυχιακής εργασίας.



Σχήμα 3.12: String συναρτήσεις του LabVIEW

Βρόχος επανάληψης For Loop

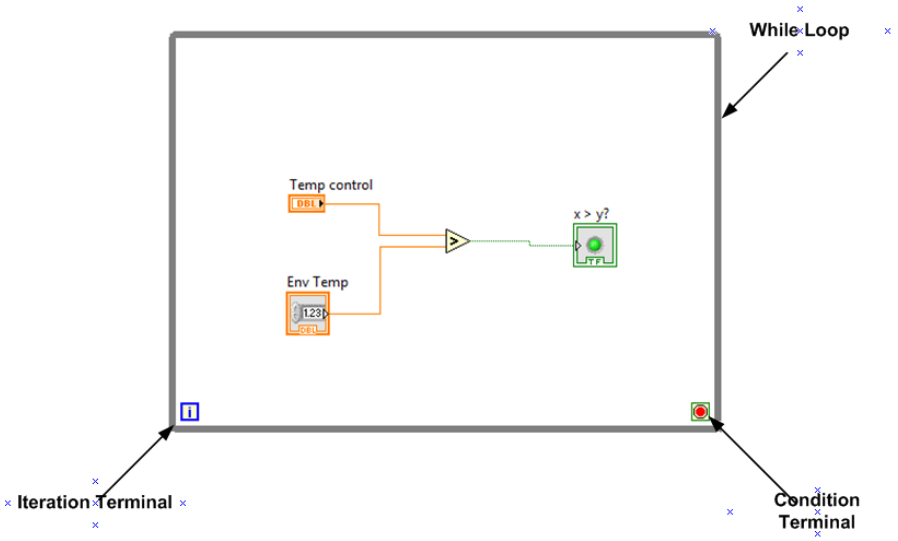
Ο βρόχος For Loop εκτελείται όσες φορές επιθυμεί ο χρήστης ανάλογα με τον αριθμό που έχει δοθεί στην μεταβλητή N του Σχήματος 3.13. Η μεταβλητή i περιέχει τον αριθμό που δείχνει πόσες φορές εκτελέστηκε το περιεχόμενο του βρόχου. Όταν η μεταβλητή i γίνει ίση με την μεταβλητή N τότε σταματά η εκτέλεση του βρόχου επανάληψης.



Σχήμα 3.13: Δομή της For Loop

Βρόχος επανάληψης While Loop

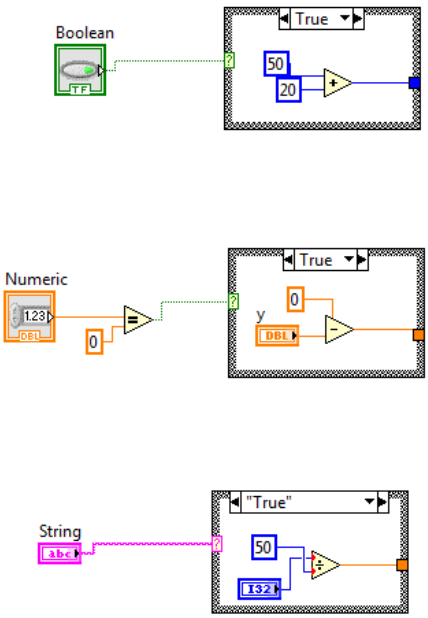
Ο βρόχος While Loop εκτελείται μέχρι να ενεργοποιηθεί η συνθήκη τερματισμού όπως φαίνεται στο Σχήμα 3.14. Το περιεχόμενο του βρόχου θα εκτελεστεί τουλάχιστον μία φορά. Η μεταβλητή i περιέχει τον αριθμό που δείχνει πόσες φορές εκτελέστηκε το περιεχόμενο του βρόχου.



Σχήμα 3.14: Δομή της While Loop

Case Structues

Οι δομές case αποτελούνται από τουλάχιστον δύο περιπτώσεις. Ανάλογα με την επιλογή που έχει γίνει στον Case selector, θα εκτελεστεί κώδικας της αντίστοιχης περίπτωσης. Οι τύποι δεδομένων που συνδέονται με τον Case selector (Σχήμα 3.15), μπορεί να είναι τύπου Boolean,Integer, String, και Enumerated type.

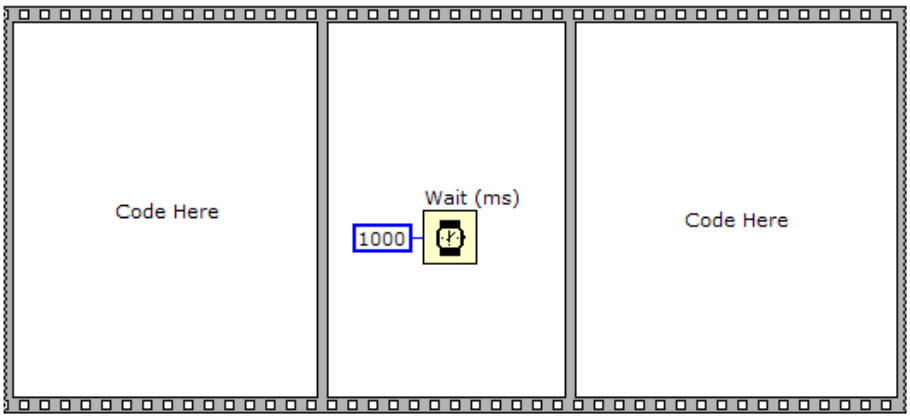


Σχήμα 3.15: Case Structure

Flat Sequence Structures

Οι δομές Flat Sequence (Σχήμα 3.16) αποτελούνται από ένα ή περισσότερα sub diagrams ή αλλιώς πλαίσια τα οποία περιέχουν κώδικα. Η εκτέλεση του κώδικα γίνεται το αριστερό προς το δεξιό πλαίσιο. Σε κάθε περίπτωση για να εκτελεστεί το επόμενο πλαίσιο θα

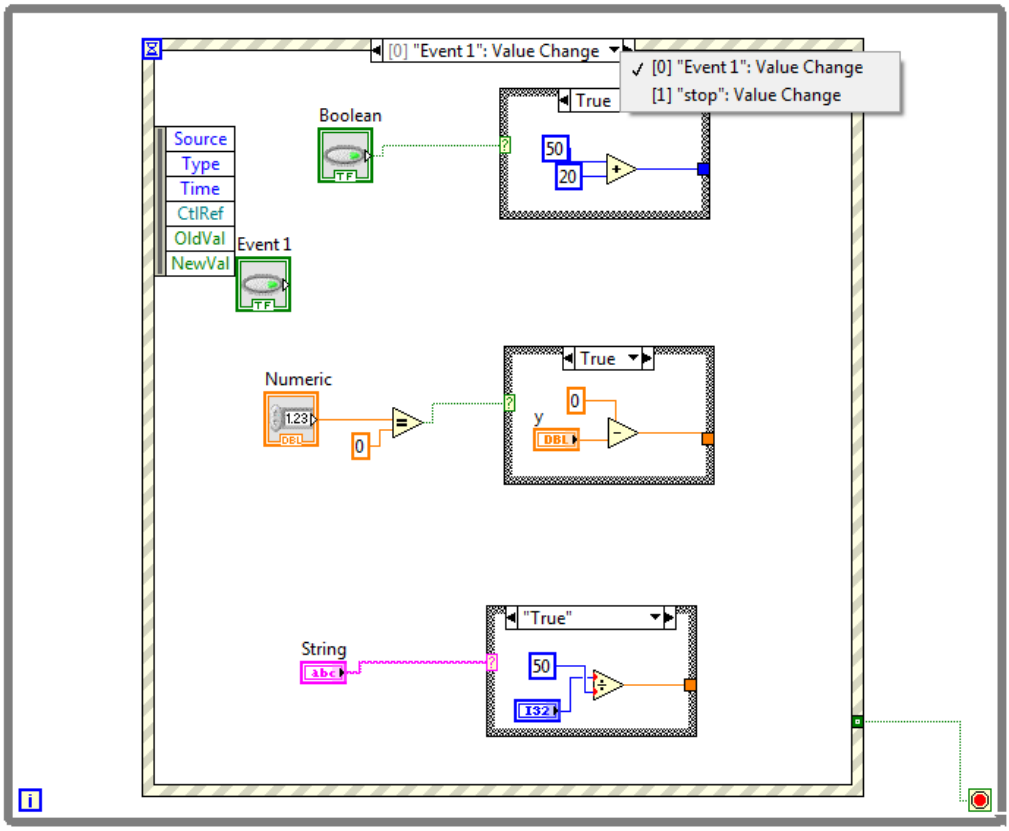
πρέπει αφενός να έχει ολοκληρωθεί η εκτέλεση του προηγούμενου πλαισίου και αφετέρου να είναι διαθέσιμα όλα τα δεδομένα που αποτελούν τις εισόδους του συγκεκριμένου πλαισίου.



Σχήμα 3.16: Flat Sequence Structure

Event Structures

Οι δομές Event (Σχήμα 3.17) αποτελούνται από περιπτώσεις οι οποίες εκτελούνται όταν συμβεί το αντίστοιχο γεγονός στο Front Panel (πχ το πάτημα ενός κουμπιού, μεταβολή της τιμής κάποιου αντικειμένου). Οι δομές αυτές αποτελούνται από μία ή περισσότερες περιπτώσεις.



Σχήμα 3.17: Παράδειγμα μίας δομής Event

Σύμφωνα με το Σχήμα 3.17 επιλέγοντας τον event selector εμφανίζονται όλες οι περιπτώσεις που είναι διαθέσιμες στην συγκεκριμένη δομή Event. Στον event selector αναγράφεται το όνομα του αντικειμένου και η συνθήκη λειτουργίας του από τα οποία εξαρτάται η κάθε περίπτωση (πχ μόλις αλλάξει η τιμή στο κουμπί με όνομα Event 1).

Graphs

Το LabVIEW παρέχει στον χρήστη την δυνατότητα προβολής δεδομένων μέσω γραφικών παραστάσεων. Οι βασικοί τύποι γραφικών παραστάσεων που διαθέτει είναι:

- Waveform Chart
- Waveform Graph
- XY Graph

Οι γραφικές παραστάσεις τοποθετούνται στο Front Panel μέσω της ομάδας Graph της Controls Palette.

Waveform Chart

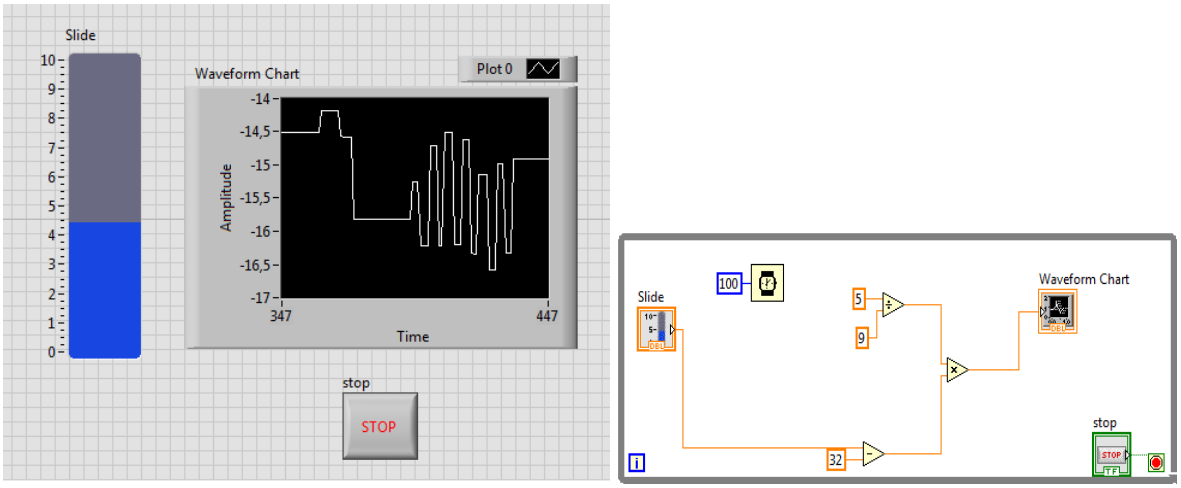
Τα Waveform Charts (Σχήμα 3.18) είναι αριθμητικοί indicators οι οποίοι σχεδιάζουν μία ή περισσότερες γραφικές παραστάσεις. Τα Waveform Charts διατηρούν ιστορικό δεδομένων το οποίο το εμφανίζουν προσθέτοντας κάθε φορά στο ιστορικό και ένα στοιχείο. Το μέγεθος του ιστορικού μπορεί να επιλεχθεί από τον χρήστη από την επιλογή Chart History Length.

Waveform Graph

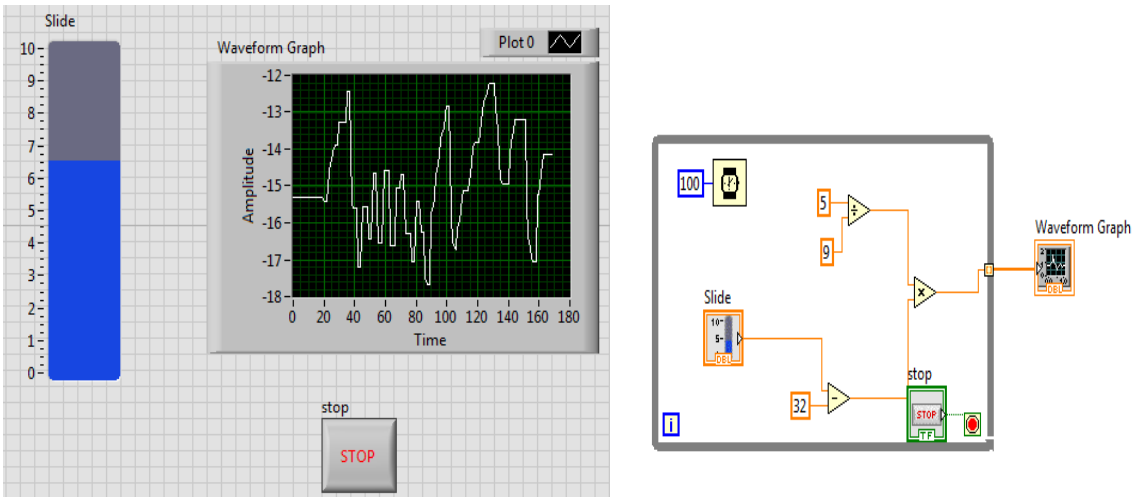
Τα Waveform Graphs (Σχήμα 3.19) είναι αριθμητικοί indicators, οι οποίο σχεδιάζουν μία ή περισσότερες γραφικές παραστάσεις, οι οποίες έχουν ομοιόμορφη δειγματοληψία μετρήσεων. Τα δεδομένα αυτά προέρχονται από έναν ολοκληρωμένο πίνακα δεδομένων. Οι γραφικές αυτές ανανεώνονται μόνο όταν όλα τα δεδομένα είναι διαθέσιμα.

XY Graph

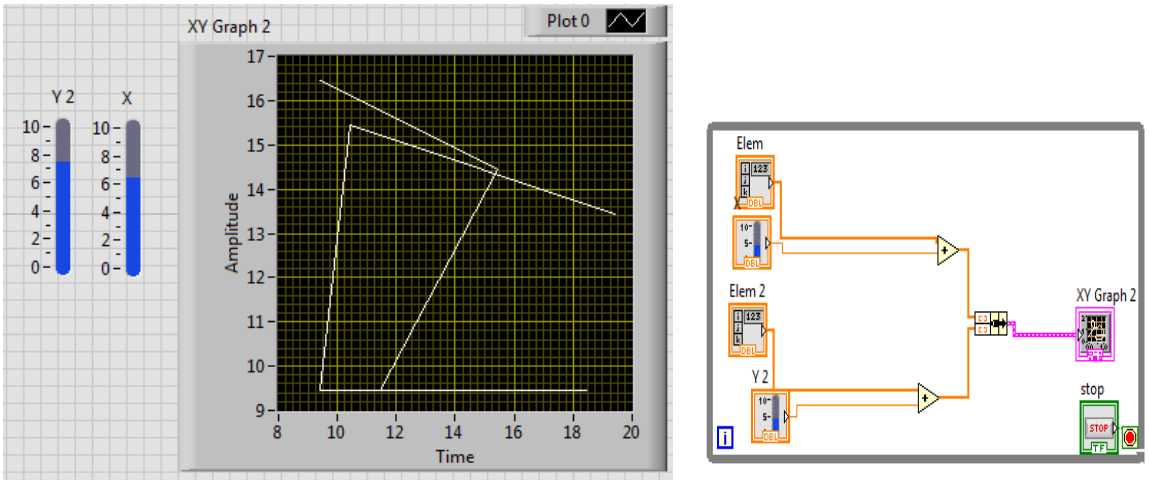
Τα XY Graphs (Σχήμα 3.20) είναι αριθμητικοί indicators οι οποίοι σχεδιάζουν μία ή περισσότερες γραφικές παραστάσεις ανεξαρτήτως εάν είναι ομοιόμορφης δειγματοληψίας ή όχι. Τα XY Graphs αντλούν τα δεδομένα τους από ένα Cluster ο οποίος περιέχει ένα πίνακα με τα στοιχεία του άξονα X και ένα πίνακα με τα στοιχεία του άξονα Y. Οι γραφικές ανανεώνονται μόνο όταν όλα τα δεδομένα είναι διαθέσιμα.



Σχήμα 3.18: Παράδειγμα ενός Waveform Chart



Σχήμα 3.19: Παράδειγμα ενός Waveform Graph



Σχήμα 3.20: Παράδειγμα ενός XY Graph

3.3.3 LabVIEW to MySQL

Το LabVIEW to MySQL είναι ένα toolkit το οποίο επιτρέπει την πρόσβαση σε MySQL βάσεις δεδομένων από εφαρμογές που έχουν υλοποιηθεί με το LabVIEW.

Εικ.	Ονομασία	Εικ.	Ονομασία
	Open_Connection.vi		Update_Blob_Data.vi
	Close_Connection.vi		Read_Blob_Data_&_ Save_In_File.vi
	Change_Database.vi		Update_Blob_Data_ From_File.vi
	Read_Column_Field.vi		Execute_Query.vi
	Update_Column_Field.vi		Execute_Non_Query.vi
	Read_Blob_Data.vi		

Πίνακας 3.1: Οι συναρτήσεις του LabVIEW to MySQL.

Επειδή οι συναρτήσεις του LabVIEW to MySQL toolkit που παρουσιάζονται στον Πίνακα 3.1 έχουν υλοποιηθεί αξιοποιώντας τις δυνατότητας του .NET framework, θα πρέπει να είναι εγκατεστημένο εκτός από το LabVIEW και ο MySQL Connector/Net που διατίθεται δωρεάν από την ιστοσελίδα της MySQL.

Το πλεονέκτημα του LabVIEW to MySQL toolkit είναι ότι μέσω του συνόλου των συναρτήσεων που διαθέτει δίνεται η δυνατότητα σε αρχάριους χρήστες να υλοποιούν τις πλέον συνηθισμένες διαδικασίες όπως η ανάκτηση ή η εγγραφή των δεδομένων μιας στήλης ενός πίνακα ή ολόκληρου του πίνακα μιας βάσης δεδομένων χωρίς να είναι απαραίτητη η σύνταξη SQL (Structured Query Language) εντολών. Επιπρόσθετα το MySQL to LabVIEW διαθέτει συναρτήσεις για την διαχείριση δεδομένων τύπου BLOB (Binary Large OBject) τα οποία μπορεί να ανακτώνται από ή να αποθηκεύονται σε ένα αρχείο που επιλέγει ο χρήστης. Για την επίτευξη πιο σύνθετων διαδικασιών ο χρήστης έχει την δυνατότητα μέσω των συναρτήσεων Execute_Query.vi και Execute_Non_Query.vi να συντάξει και να εκτελέσει τα SQL ερωτήματα (queries) προς την βάση δεδομένων που επιθυμεί.

Από τις παραπάνω συναρτήσεις για την υλοποίηση της παρούσας πτυχιακής εργασίας χρησιμοποιήθηκαν το Open_Connection.vi για την σύνδεση με την βάση, το Execute_Query.vi για την εκτέλεση των SQL Queries και το Close_Connection.vi για τον τερματισμό της σύνδεσης με την βάση δεδομένων.

3.4 Εισαγωγή στη MySQL

Μια βάση δεδομένων μπορεί να θεωρηθεί ως μία ξεχωριστή εφαρμογή η οποία αναλαμβάνει την αποθήκευση ενός συνόλου δεδομένων. Κάθε βάση δεδομένων διαθέτει ένα ή περισσότερα διακριτά APIs (Application Program Interfaces) για την εγγραφή, την ανάκτηση, την διαχείριση, την αναζήτηση και την αντιγραφή των δεδομένων.

Για την αποθήκευση και την διαχείριση μεγάλου όγκου δεδομένων χρησιμοποιούνται τα συστήματα RDBMS (Relation Database Management Systems). Σε αυτά τα συστήματα όλα τα δεδομένα αποθηκεύονται σε διαφορετικούς πίνακες (tables) έχοντας όμως αλληλεξάρτηση μεταξύ τους χρησιμοποιώντας primary ή foreign keys.

Το RDBMS είναι ένα λογισμικό το οποίο μπορεί να δημιουργήσει μία βάση δεδομένων με πίνακες, στήλες και ευρετήρια, μπορεί να εγγυηθεί την ακεραιότητα αναφορών μεταξύ των σειρών των διαφόρων πινάκων, μπορεί να ενημερώνει τα ευρετήρια αυτόματα καθώς και να ερμηνεύει ένα SQL query συνδυάζοντας πληροφορίες από διάφορους πίνακες. Κάποιοι θεμελιώδεις ορισμοί που σχετίζονται με τις βάσεις δεδομένων είναι:

- **Database:** Μία βάση δεδομένων είναι μία συλλογή από πίνακες με τα αντίστοιχα δεδομένα τους.
- **Table:** Είναι ένας πίνακας δεδομένων ο οποίος μοιάζει με ένα υπολογιστικό φύλλο.
- **Column:** Μία στήλη που περιέχει στοιχεία από δεδομένα ίδιου τύπου (πχ μία στήλη που περιέχει ταχυδρομικούς κώδικες).
- **Row:** Μία σειρά είναι μία ομάδα σχετιζόμενων δεδομένων (πχ τα δεδομένα μιας πλήρης εγγραφής).
- **Redundancy:** Διπλή αποθήκευση δεδομένων. Ο μηχανισμός αυξάνει τον όγκο δεδομένων αλλά βοηθά στην συνολική αύξηση της ταχύτητας του συστήματος.
- **Primary Key:** Ένα πρωτεύον κλειδί είναι μοναδικό και σχετίζεται μόνο με μια γραμμή ενός πίνακα.
- **Foreign Key:** Ένα ξένο κλειδί χρησιμοποιείται για την διασύνδεση δύο πινάκων.
- **Compound Key:** Ένα σύνθετο κλειδί αποτελείται από πολλαπλές στήλες, επειδή μία στήλη δεν είναι απαραίτητα μοναδική.
- **Index:** Ένας δείκτης σε μία βάση δεδομένων λειτουργεί όπως το ευρετήριο στο πίσω μέρος ενός βιβλίου.
- **Referential Integrity:** Η ακεραιότητα των αναφορών διασφαλίζει ότι ένα ξένο κλειδί δείχνει πάντα σε μία υπάρχουσα γραμμή ενός πίνακα.

Η MySQL είναι ένα ανοικτού κώδικα (open-source) σύστημα διαχείρισης βάσεων δεδομένων που χρησιμοποιείται πολύ συχνά σε εφαρμογές διαδικτύου. Συστήματα που απαιτούν ένα πλήρως εξοπλισμένο σύστημα διαχείρισης βάσεων δεδομένων χρησιμοποιούν επίσης την MySQL. Η MySQL υποστηρίζεται από πολλά λειτουργικά συστήματα και πολλές γλώσσες προγραμματισμού συμπεριλαμβανομένων PHP, Perl, C, C++, Java, κλπ. Επίσης, υποστηρίζει μεγάλου όγκου βάσεις δεδομένων που μπορεί να περιλαμβάνουν πινάκες οι οποίοι να ξεπερνούν τις 50 εκατομμύρια γραμμές. Το προεπιλεγμένο μέγεθος αρχείου για ένα πίνακα είναι τα 4GB. το όριο αυτό μπορεί να αυξηθεί (αν το λειτουργικό σύστημα μπορεί να το χειριστεί) έως το θεωρητικό όριο των 8 εκατομμυρίων TB. Ακόμα, η MySQL είναι

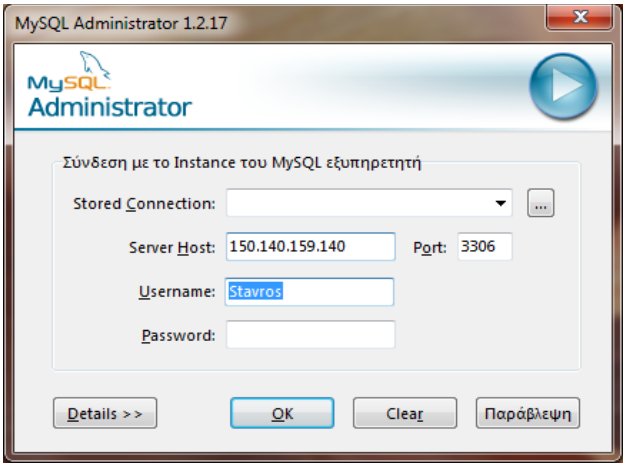
προσαρμόσιμη αφού μέσω της GPL άδειας ανοικτού κώδικα επιτρέπει στους προγραμματιστές να τροποποιήσουν το λογισμικό της ανάλογα με τις ανάγκες τους.

Στην συνέχεια αυτής της παραγράφου παρουσιάζεται η λειτουργία του λογισμικού MySQL Administrator που χρησιμοποιήθηκε για την διαμόρφωση και την διαχείριση της βάσης δεδομένων. Επίσης γίνεται μια σύντομη αναφορά στις βασικές SQL εντολές που αξιοποιήθηκαν για την ανάπτυξη του συστήματος που παρουσιάζεται στην συγκεκριμένη πτυχιακή εργασία.

3.4.1 Η εφαρμογή MySQL Administrator

Η εφαρμογή MySQL Administrator (Σχήμα 3.21) αναλαμβάνει την εκτέλεση λειτουργιών, όπως την διαμόρφωση, την παρακολούθηση, την εκκίνηση και τον τερματισμό ενός MySQL server. Επίσης, αναλαμβάνει την διαχείριση των χρηστών, τις συνδέσεις και πολλά άλλα καθήκοντα διαχείρησης. Τα βασικά πλεονεκτήματα της εφαρμογής MySQL Administrator έναντι αντίστοιχων εφαρμογών είναι:

- Παρέχει στον χρήστη ένα εύχρηστο και φιλικό γραφικό περιβάλλον
- Παρέχει επισκόπηση των βασικών ρυθμίσεων της MySQL με σκοπό την απόδοση, την αξιοπιστία και την ασφάλεια των MySQL servers
- Εμφανίζει τους δείκτες απόδοσης με την χρήση γραφημάτων

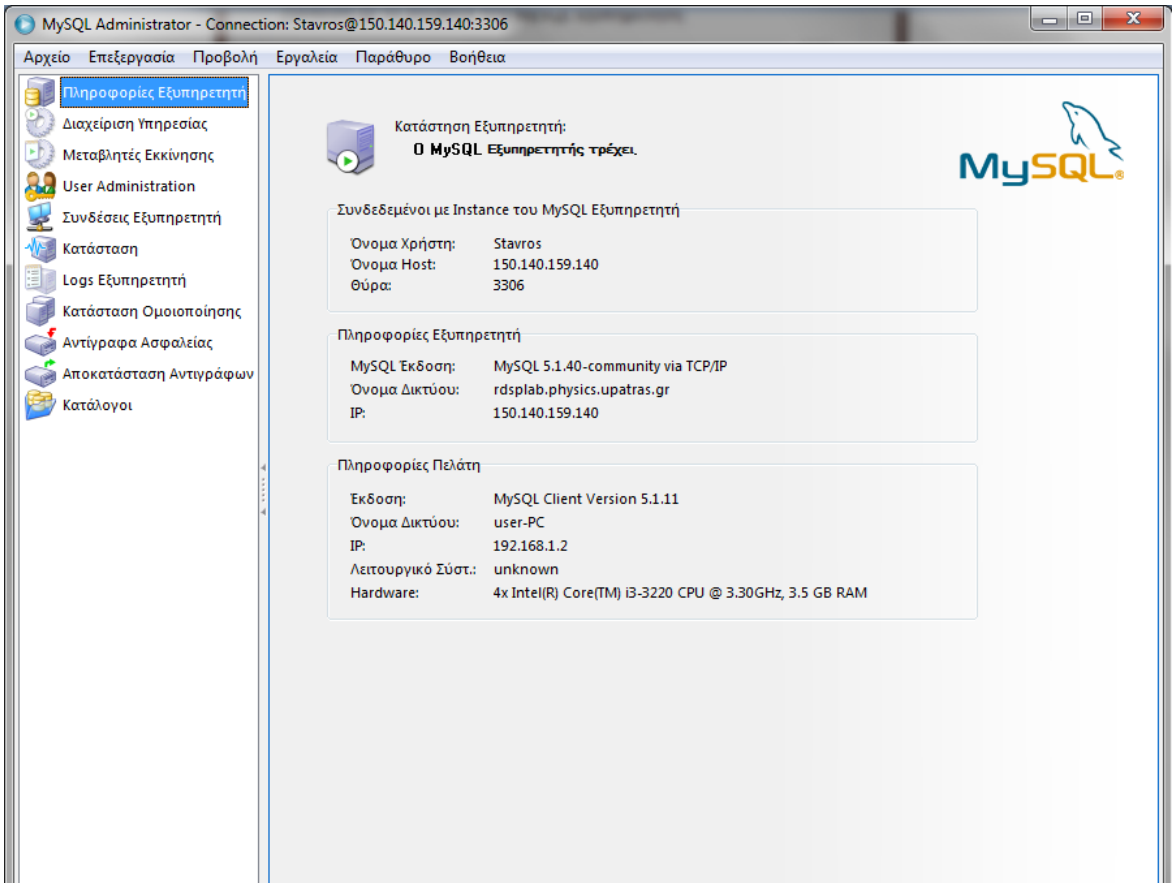


Σχήμα 3.21: Έναρξη του MySQL Administrator

Όταν πραγματοποιείται η σύνδεση στον MySQL server εμφανίζεται το αρχικό μενού (Σχήμα 3.22) που διαθέτει διάφορες επιλογές. Μία σύντομη επεξήγηση κάθε διαθέσιμης επιλογής δίνεται παρακάτω:

- **Πληροφορίες Εξυπηρετητή:** Δίνονται πληροφορίες σχετικά με τον MySQL server που έχει πραγματοποιηθεί σύνδεση, το μηχάνημα που λειτουργεί ο server καθώς και πληροφορίες που σχετίζονται με την σύνδεση.
- **Διαχείριση Υπηρεσίας:** Ελέγχει την εκκίνηση και την διακοπή του MySQL server.
- **Μεταβλητές Εκκίνησης:** Πραγματοποιείται η διαμόρφωση των μεταβλητών εκκίνησης για τον MySQL server.

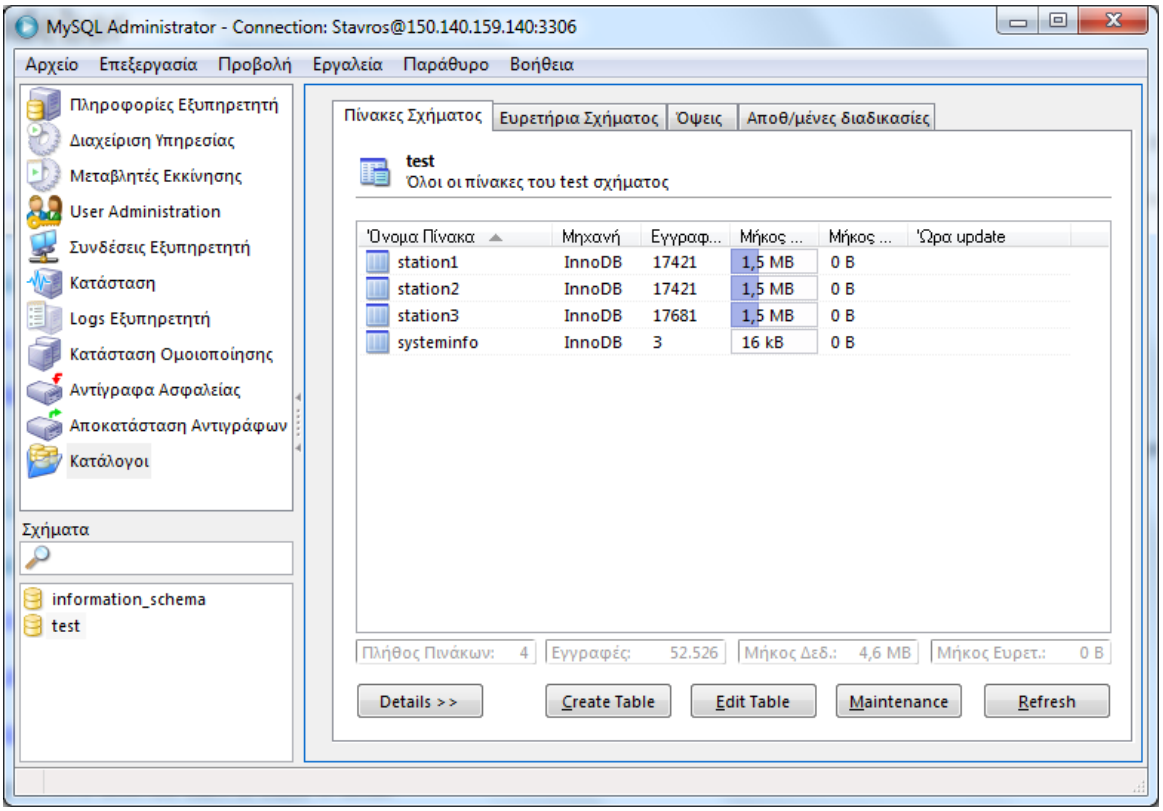
- **Συνδέσεις Εξυπηρετητή:** Πραγματοποιείται η προβολή ή ο τερματισμός των τρέχων συνδέσεων στον MySQL server.
- **User Administration:** Πραγματοποιείται η δημιουργία νέων χρηστών ή διαγραφή χρηστών. Επίσης πραγματοποιείται η εκχώρηση δικαιωμάτων σε ένα χρήστη.
- **Κατάσταση:** Παρουσιάζεται η απόδοση του server με την χρήση γραφημάτων



Σχήμα 3.22: Αρχικό Μενού του MySQL Administrator

- **Logs Εξυπηρετητή:** Εμφανίζονται τα αρχεία καταγραφής του server
- **Κατάσταση Ομοιοποίησης:** Σχεδιασμός και διαχείριση των έργων που σχετίζονται με την δημιουργία των αντιγράφων ασφαλείας. Επιτρέπει την επιλογή των βάσεων δεδομένων για τις οποίες θα δημιουργηθεί αντίγραφο ασφαλείας και την έναρξη της δημιουργίας του αντίγραφου ασφαλείας.
- **Αντίγραφα Ασφαλείας:** Πραγματοποιείται η επαναφορά των βάσεων δεδομένων από το επιλεγμένο αντίγραφο ασφαλείας.
- **Αποκατάσταση Αντιγράφων:** Δίνονται πληροφορίες σχετικά με τον αφέντη server αλλά και με τους συνδεδεμένους σκλάβους servers αντιγράφων.
- **Κατάλογοι:** Πραγματοποιείται η προβολή των πληροφοριών σχετικά με τις βάσεις δεδομένων, τους πίνακες, τις στήλες, τα ευρετήρια και τις σειρές. Επίσης με την επιλογή αυτή δίνεται στον χρήστη η δυνατότητα διαχείρισης των πινάκων.

Στην συνέχεια αναλύονται κάποιες από τις παραπάνω επιλογές που αξιοποιήθηκαν στα πλαίσια του συστήματος που δημιουργήθηκε στην παρούσα πτυχιακή εργασία.

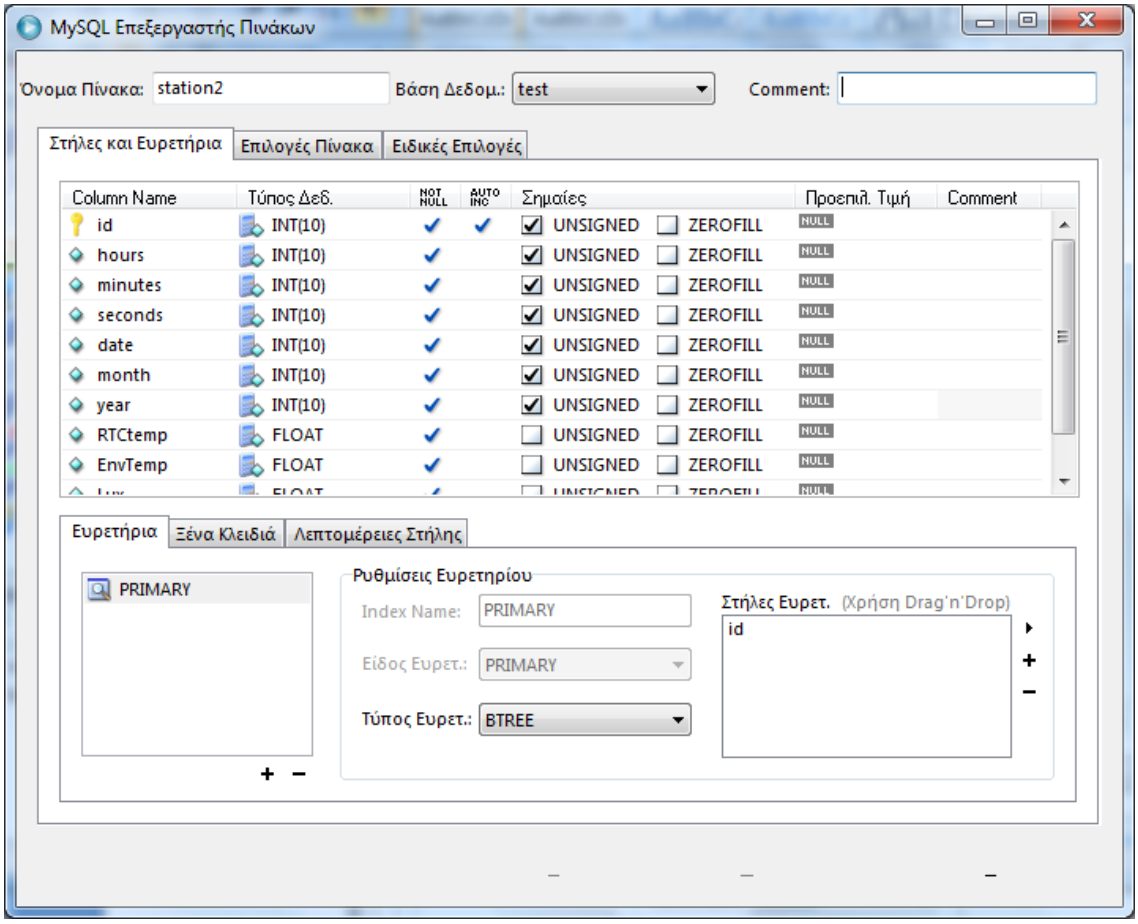


Σχήμα 3.23: Κατάλογοι και tables στην βάση δεδομένων

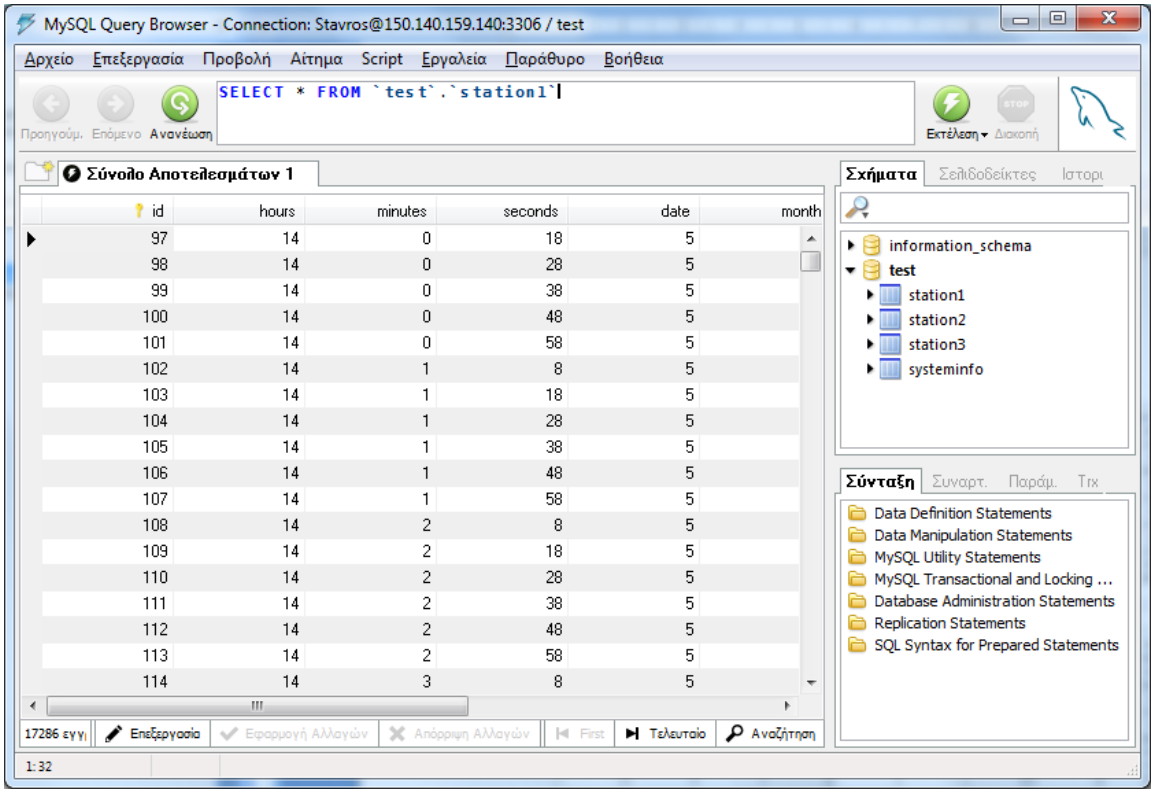
Επιλέγοντας την συντόμευση Κατάλογοι εμφανίζονται τα σχήματα με τους αντίστοιχους πίνακές τους. Στο σχήμα test όπως φαίνεται στο Σχήμα 3.23 υπάρχουν διάφοροι πίνακες όπως ο station2, systeminfo, station1 κα. Επίσης δίνονται και κάποια στοιχεία για κάθε πίνακα, όπως το μέγεθος του περιεχομένου, τον αριθμό των εγγραφών που έχουν γίνει, την ώρα ενημέρωσης κα. Στο κάτω μέρος εμφανίζονται πληροφορίες που αφορούν το επιλεγμένο σχήμα όπως, το πλήθος των πινάκων, τον αριθμό των συνολικών εγγραφών, το μέγεθος των συνολικών δεδομένων.

Για την δημιουργία ενός σχήματος ή ενός πίνακα χρειάζεται δεξί κλικ στο αντίστοιχο πλαίσιο και δημιουργία σχήματος ή πίνακα. Κατά την δημιουργία του πίνακα ο χρήστης καλείται να δημιουργήσει στήλες ανάλογα με τα δεδομένα που θέλει να περιέχει (Σχήμα 3.24). Η βάση δεδομένων καλύπτει μία μεγάλη γκάμα τύπων δεδομένων που δέχεται και μπορεί να επεξεργαστεί.

Κάνοντας δεξί κλικ σε ένα πίνακα και επιλέγοντας την επεξεργασία δεδομένων, εμφανίζονται τα δεδομένα που είναι αποθηκευμένα σε αυτόν τον πίνακα. Το παράθυρο που ανοίγει (Σχήμα 3.25) δίνει στον χρήστη την δυνατότητα να ανανεώσει τα δεδομένα ή και να τα διαχειριστεί εκτελώντας διάφορα SQL queries στο πλαίσιο που υπάρχει στο πάνω μέρος για την εκτέλεση εντολών.



Σχήμα 3.24: Παράθυρο επεξεργασίας πίνακα



Σχήμα 3.25: Παράθυρο δεδομένων ενός πίνακα

3.4.2 SQL (Structured Query Language)

Η SQL είναι μια ειδικού σκοπού γλώσσα προγραμματισμού η οποία έχει σχεδιαστεί για την διαχείριση δεδομένων σε συστήματα RDBMS. Η SQL αποτελείται από μία γλώσσα προγραμματισμού για τον ορισμό δεδομένων και μία γλώσσα προγραμματισμού για τον χειρισμό δεδομένων. Το πεδίο εφαρμογής της SQL περιλαμβάνει τις διαδικασίες εισαγωγής δεδομένων, εκτέλεσης queries, πραγματοποίησης ενημερώσεων, την διαγραφή δεδομένων, την δημιουργία σχημάτων, την τροποποίηση σχημάτων και τον έλεγχο πρόσβασης δεδομένων. Η SQL γλώσσα υποδιαιρείται σε πολλά στοιχεία μεταξύ των οποίων είναι:

- **Clauses:** Είναι τα συντακτικά στοιχεία των statements και των queries. Σε ορισμένες περιπτώσεις είναι προαιρετικά.
- **Expressions:** Μπορούν να επιστρέφουν είτε τιμές είτε πίνακες που αποτελούνται από τις στήλες και τις σειρές των δεδομένων
- **Predicates:** Καθορίζουν τις συνθήκες που μπορούν να ληφθούν υπόψη ως SQL λογική τριών τιμών (three-valued logic, 3VL) ή ως Boolean τιμές και χρησιμοποιούνται είτε για να περιορίσουν τα αποτελέσματα των statements και των queries είτε για τον έλεγχο της ροής του προγράμματος.
- **Queries:** Ανακτούν δεδομένα βάση συγκεκριμένων κριτηρίων. Αποτελούν σημαντικό στοιχείο της SQL.
- **Statements:** Μπορεί να έχουν επίδραση στα σχήματα και στα δεδομένα. Επίσης ελέγχουν τις συνδιαλλαγές (transactions), τις συνδέσεις, την ροή του προγράμματος και εκτελούν διαγνωστικές διεργασίες.

Στα statements επίσης ανήκει και ο χαρακτήρας ";" που αποτελεί τον τερματικό χαρακτήρα των statements. Επίσης τα επιπλέον κενά μεταξύ των statements αγνοούνται από την SQL. Στην συνέχεια, αναφέρονται οι βασικές εντολές της SQL που χρησιμοποιήθηκαν κατά την υλοποίηση της συγκεκριμένης πτυχιακής.

SELECT

Η εντολή SELECT χρησιμοποιείται για την επιλογή δεδομένων από την βάση δεδομένων. Το αποτέλεσμα επιστρέφεται ως ένας πίνακας αποτελεσμάτων. Για την επιλογή μίας στήλης δεδομένων από ένα πίνακα η εντολή αυτή συντάσσεται ως εξής:

SELECT column_name, column_name FROM table_name;

Για την επιλογή όλων των δεδομένων ενός πίνακα η εντολή SELECT συντάσσεται ως εξής:

*SELECT * FROM table_name;*

WHERE

Η εντολή WHERE χρησιμοποιείται για την εξαγωγή των δεδομένων που εκπληρώνουν μία συγκεκριμένη συνθήκη. Η εντολή αυτή συνδυάζεται με άλλες εντολές όπως η εντολή SELECT, και συντάσσεται ως εξής:

SELECT column_name,column_name FROM table_name WHERE column_name operator value;

BETWEEN

Η εντολή BETWEEN επιλέγει τιμές ανάμεσα σε όρια. Οι τιμές μπορεί να είναι αριθμοί, κείμενα ή ημερομηνίες. Η εντολή αυτή συνδυάζεται με άλλες εντολές όπως η εντολή SELECT, και συντάσσεται ως εξής:

SELECT column_name(s) FROM table_name WHERE column_name BETWEEN value1 AND value2;

INSERT INTO

Η εντολή INSERT INTO χρησιμοποιείται για την εισαγωγή δεδομένων σε ένα πίνακα. Όταν πρόκειται να εισαχθούν νέα δεδομένα σε μία γραμμή ενός πίνακα η εντολή αυτή συντάσσεται ως εξής:

INSERT INTO table_name VALUES (value1,value2,value3,...);

Αν η εισαγωγή των δεδομένων πρόκειται να πραγματοποιηθεί σε μια νέα γραμμή του πίνακα αλλά μόνο σε συγκεκριμένες στήλες η εντολή INSERT INTO ως εξής:

INSERT INTO table_name (column1,column2,column3,...) VALUES (value1,value2,value3,...);

MAX

Η εντολή MAX σε συνδυασμό με την εντολή SELECT επιστρέφει την μεγαλύτερη τιμή της επιλεγμένης στήλης. Σε αυτή την περίπτωση η σύνταξη είναι:

SELECT MAX(column_name) FROM table_name;

MIN

Η εντολή MIN σε συνδυασμό με την εντολή SELECT επιστρέφει την μικρότερη τιμή της επιλεγμένης στήλης. Σε αυτή την περίπτωση η σύνταξη είναι:

SELECT MIN(column_name) FROM table_name;

Κεφάλαιο 4

Ανάπτυξη βιβλιοθηκών για το TWI και το RTC

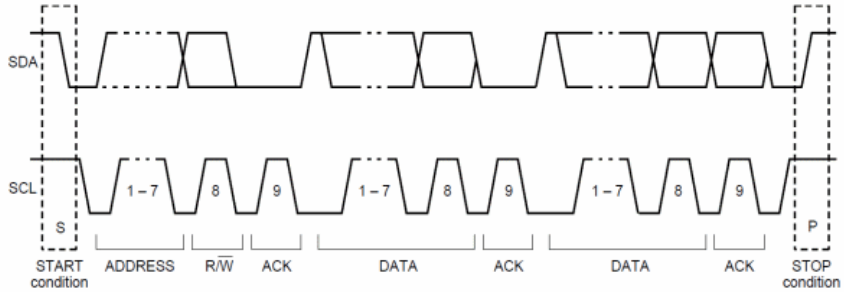
4.1 Εισαγωγή

Ένας από τους στόχους της παρούσας πτυχιακής είναι η μελέτη και ανάλυση του πρωτοκόλλου I2C. Στο πρωτόκολλο αυτό στηρίχθηκε η επικοινωνία και η λειτουργία όλων των συσκευών του συστήματος. Αρχικά, υλοποιήθηκε η δημιουργία συναρτήσεων για την παραμετροποίηση του I2C όπως ο ορισμός της ταχύτητα επικοινωνίας και η ενεργοποίηση των αντίστοιχων ακροδεκτών. Έπειτα δημιουργήθηκαν συναρτήσεις που επιτρέπουν την ανάγνωση και την εγγραφή τιμών σε συσκευές που υποστηρίζουν το πρωτόκολλο I2C. Η γκάμα των συναρτήσεων που δημιουργήθηκε καλύπτει ένα μεγάλο πλήθος συσκευών. Εκτός από τις συναρτήσεις που αφορούν το πρωτόκολλο I2C, δημιουργήθηκαν και συναρτήσεις για τον έλεγχο του RTC DS3231. Οι συναρτήσεις αυτές σε συνδυασμό με τις συναρτήσεις του I2C μπορούν και αξιοποιούν κάθε δυνατότητα του ρολογιού όπως τα Alarms, την EEPROM που περιέχει και φυσικά την ανάγνωση και το γράψιμο της ώρας και ημερομηνίας.

Στο κεφάλαιο αυτό αναλύεται κάθε συνάρτηση που πραγματοποιήθηκε για το πρωτόκολλο I2C και για το RTC και παρουσιάζεται αναλυτικά ο κώδικάς τους, όπως αναπτύχθηκε στο περιβάλλον Arduino IDE για τον μικροελεγκτή ATmega2560. Για να μπορεί το περιβάλλον Arduino IDE να αναγνωρίσει τις συναρτήσεις αυτές έχουν συμπεριληφθεί στον κώδικα τα αντίστοιχα αρχεία που δημιουργήθηκαν για κάθε περίπτωση.

4.2 Συναρτήσεις για την υλοποίηση της επικοινωνίας I2C

Για να την επικοινωνία του hardware με τον μικροελεγκτή ATmega2560 χρησιμοποιήθηκε το πρωτόκολλο I2C. Παρόλα αυτά δεν χρησιμοποιήθηκε η έτοιμη βιβλιοθήκη που παρέχει το Arduino IDE αλλά δημιουργήθηκαν νέες συναρτήσεις για την ανάπτυξη της επικοινωνίας αυτής. Οι συναρτήσεις που έχουν δημιουργηθεί για την επικοινωνία μέσω I2C έχουν βασιστεί στην λογική που παρουσιάζεται στο Σχήμα 4.1.



Σχήμα 4.1 : Επικοινωνία I2C

Το πρώτο βήμα σε μία μετάδοση στο TWI είναι η μετάδοση της συνθήκης εκκίνησης. Αυτό επιτυγχάνεται γράφοντας μία συγκεκριμένη τιμή στον καταχωρητή TWCR (Πίνακας 4.1), καθοδηγώντας το hardware του TWI να μεταδώσει μία συνθήκη εκκίνησης. Είναι σημαντικό να ελέγχεται το TWINT bit ώστε να έχει την κατάλληλη τιμή (‘1’) για να

επιτρέψει την μετάδοση της συνθήκης εκκίνησης. Όταν μεταδοθεί η συνθήκη εκκίνησης το bit TWINT έχει τιμή ‘0’ και ο καταχωρητής TWSR ενημερώνεται με τον κώδικα κατάστασης που αντιστοιχεί στην επιτυχής αποστολή της συνθήκης εκκίνησης. Μόλις το πρόγραμμα αναγνωρίσει την σωστή τιμή στον καταχωρητή TWSR ακολουθεί η εγγραφή της διεύθυνσης και του bit read/wite στον καταχωρητή TWDR. Για την ολοκλήρωση της μετάδοσης γράφεται η αντίστοιχη τιμή σύμφωνα με τον Πίνακα 4.1 στον καταχωρητή TWCR ώστε να πραγματοποιηθεί ο εντοπισμός του ACK. Έπειτα ο καταχωρητής TWSR θα ενημερωθεί με τον αντίστοιχο κωδικό κατάστασης για την επιτυχή αποστολή της διεύθυνσης. Μόλις αναγνωρισθεί η επιτυχή αποστολή της διεύθυνσης ακολουθεί η αποστολή δεδομένων τα οποία γράφονται στον καταχωρητή TWDR και ακολουθείται η ίδια διαδικασία όπως και στην αποστολή της διεύθυνσης. Όταν πια όλα τα δεδομένα έχουν σταλεί, η συνθήκη τερματισμού αποστέλλεται γράφοντας την κατάλληλη τιμή στον καταχωρητή TWCR. Μετά από κάθε μετάδοση είτε δεδομένων είτε άλλων συνθηκών θα πρέπει το bit TWINT στον καταχωρητή TWCR να γίνεται λογικό ‘0’ για την επιτυχή ολοκλήρωση κάθε διαδικασίας.

Παρακάτω παρουσιάζονται οι τιμές που πρέπει να λαμβάνει ο καταχωρητής TWCR για την μετάδοση των αντίστοιχων συνθηκών και την επιτυχή επικοινωνία TWI. Επίσης, στο Πρόγραμμα 4.1 δίνεται ένα παράδειγμα μιας ολοκληρωμένης επικοινωνίας του TWI.

Λειτουργία	**Τιμή στον καταχωρητή ελέγχουTWCR**
Συνθήκη εκκίνησης	TWCR=(1<<TWINT)|(1<<TWSTA)|(1<<TWEN)|(1<<TWEA);
Συνθήκη τερματισμού	TWCR=(1<<TWINT)|(1<<TWSTO)|(1<<TWEN);
Καθαρισμός του TWI και εντοπισμός του ACK	TWCR = (1<<TWINT)|(1<<TWEN)|(1<<TWEA);
Καθαρισμός του TWI και εντοπισμός του NACK	TWCR=(1<<TWINT)|(1<<TWEN);

Πίνακας 4.1 : Τιμές του καταχωρητή TWCR

```
writeAddr = devAddr<<1;
point = data;
TWCR=(1<<TWINT)|(1<<TWSTA)|(1<<TWEN)|(1<<TWEA);  //Send START Condition
while ((TWCR & (1<<TWINT)) == 0){};                 // Wait for TWI to be ready
stat = TWIcheckStatus();                            //Check for the right status code
if (stat != 0x08)
        goto fun_end;
TWDR=writeAddr;                                     //Send the SLA+W
TWCR =(1<<TWINT)|(1<<TWEN)|(1<<TWEA);        // Wait ACK and clear TWINT to proceed
while ((TWCR & (1<<TWINT)) == 0){};                 // Wait for TWI to be ready
stat = TWIcheckStatus();                            //Check for the right status code
if (stat != 0x18)
        goto fun_end;
TWDR=regAddr;                                       //Send Register Address
```

Πρόγραμμα 4.1: Παράδειγμα μίας ολοκληρωμένης μετάδοσης TWI

```
TWCR =(1<<TWINT)|(1<<TWEN)|(1<<TWEA);        // Wait ACK and clear TWINT to proceed

while ((TWCR & (1<<TWINT)) == 0){};                    // Wait for TWI to be ready
stat = TWIcheckStatus();                               //Check for the right status code
if (stat != 0x28)
        goto fun_end;
for (int i=0; i<lngth; i++)
{
        TWDR= *point;                                  //Send Data
        TWCR =(1<<TWINT)|(1<<TWEN)|(1<<TWEA); // Wait ACK and clear TWINT to proceed
        while ((TWCR & (1<<TWINT)) == 0){};            // Wait for TWI to be ready
        point++;
}
stat = TWIcheckStatus();                               //Check for the right status code
if (stat != 0x28)
        goto fun_end;
TWCR=(1<<TWINT)|(1<<TWSTO)|(1<<TWEN);                  //Send STOP Condition
stat = TWIcheckStatus();                               //Check for the right status code
if (stat != 0xF8)
        goto fun_end;
fun_end: return stat;                                  //Return Status Code
```

Πρόγραμμα 4.1: Παράδειγμα μίας ολοκληρωμένης μετάδοσης TWI (Συνέχεια)

Οι συναρτήσεις που έχουν δημιουργηθεί μπορούν να διαβάζουν και να γράφουν καταχωρητές για 8bit και 16bit διευθύνσεις. Επιπλέον έχουν προσαρμοστεί στις ανάγκες του συστήματος και έχουν διάφορες δυνατότητες που αναλύονται στην συνέχεια.

TWI_Init

Αρχικά, για την ενεργοποίηση των ακροδεκτών SDA και SCL και την αρχικοποίηση της ταχύτητας επικοινωνίας δημιουργήθηκε η συνάρτηση TWI_Init. Η συνάρτηση αυτή δέχεται ως όρισμα μία boolean τιμή (true/false). Όταν είναι true ενεργοποιεί τις εσωτερικές pull-up αντιστάσεις του μικροελεγκτή για τους ακροδέκτες SDA και SCL ενώ όταν είναι θα πρέπει να χρησιμοποιηθούν εξωτερικές pull-up αντιστάσεις. Σε κάθε περίπτωση η ταχύτητα επικοινωνίας που επιλέγεται μέσω του διαιρέτη συχνότητας είναι 400KHz.

```
uint8_t  TWI_Init(boolean val )
{
if (val==true)
{
        PORTC |=1<<4;          // activate internal pullups for TWI .enable SDA
        PORTC |=1<<5;         //enable SCL
        PORTC|=0x30;
}
else
{
        PORTC&=0xCF;
```

Πρόγραμμα 4.2: Κώδικας της συνάρτησης TWI_Init

```
}                                   //set SCL to 400khz
TWSR &= ~(1 << 0);
TWSR &= ~(1 << 1);
TWBR=0x0C;
TWCR=(1<<TWEN)+1;           //enable TWI
}
```

Πρόγραμμα 4.2: Κώδικας της συνάρτησης TWI_Init(Συνέχεια).

TWIcheckStatus(void)

Η συνάρτηση TWIcheckStatus ελέγχει την κατάσταση που βρίσκεται το TWI μέσω του καταχωρητή κατάστασης TWSR και επιστρέφει τους κωδικούς κατάστασης που αντιστοιχούν στην κατάσταση που βρίσκεται το bus. Η συνάρτηση αυτή δεν δέχεται κάποιο όρισμα.

```
uint8_t TWIcheckStatus(void)
{
uint8_t stat;
stat=TWSR & 0xF8;           //Status Code
return stat;
}
```

Πρόγραμμα 4.3: Κώδικας της συνάρτησης TWIcheckStatus

BurstReadReg

Η συνάρτηση BurstReadReg δέχεται τέσσερα ορίσματα. Το devAddr το οποίο είναι η 8-bit διεύθυνση της συσκευής που θα επικοινωνήσει. Το regAddr το οποίο είναι η διεύθυνση του καταχωρητή της συσκευής που θα διαβαστεί πρώτος. Τον πίνακα data ο οποίος θα περιέχει τα δεδομένα που θα αναγνωστούν. Το lngth που είναι ένας ακέραιος αριθμός και δηλώνει τον αριθμό των bytes που θα αναγνωστούν. Η συνάρτηση αυτή αναλαμβάνει την επικοινωνία με μία συσκευή 8-bit διεύθυνσης και διαβάζει όσα δεδομένα ζητήσει ο χρήστης, ξεκινώντας από τον καταχωρητή που δηλώνει ο χρήστης.

```
uint8_t BurstReadReg (uint8_t devAddr, uint8_t regAddr, uint8_t* data,int lngth)
{
uint8_t writeAddr, readAddr, statR;
uint8_t* point;
writeAddr = devAddr<<1;
readAddr = writeAddr + 1;
TWCR=(1<<TWINT)|(1<<TWSTA)|(1<<TWEN)|(1<<TWEA);          //Send Start Condition
while ((TWCR & (1<<TWINT))==0) {};                        // Wait for TWI to be ready
statR = TWIcheckStatus();                                 //Check for the right status code
if (statR != 0x08)
        goto fun_end;
TWDR=writeAddr;                                           //Send the SLA+W
TWCR = (1<<TWINT)|(1<<TWEN)|(1<<TWEA);                    //Clear TWI to proceed wait (ACK)
while ((TWCR & (1<<TWINT))==0) {};                        // Wait for TWI to be ready
statR = TWIcheckStatus();                                 //Check for the right status code
```

Πρόγραμμα 4.4: Κώδικας της συνάρτησης BurstReadReg

```
if (statR != 0x18)
        goto fun_end;
TWDR=regAddr;
TWCR = (1<<TWINT)|(1<<TWEN)|(1<<TWEA);          //Clear TWI to proceed send (ACK)
while ((TWCR & (1<<TWINT))==0) {};                      // Wait for TWI to be ready
statR = TWIcheckStatus();                               //Check for the right status code
if (statR != 0x28)
        goto fun_end;
TWCR=(1<<TWINT)|(1<<TWSTA)|(1<<TWEN)|(1<<TWEA);         //Send rep.Start
while ((TWCR & (1<<TWINT))==0) {};                      // Wait for TWI to be ready
statR = TWIcheckStatus();                               //Check for the right status code
if (statR != 0x10)
        goto fun_end;
TWDR=readAddr;                                          //Send SLA+R
TWCR = (1<<TWINT)|(1<<TWEN)|(1<<TWEA);          //Clear TWI to proceed send (ACK)
while ((TWCR & (1<<TWINT))==0) {};                      // Wait for TWI to be ready
statR = TWIcheckStatus();                               //Check for the right status code
if (statR != 0x40)
        goto fun_end;
for (int i=0; i<lngth; i++)
{
        point = data++;
        TWCR = (1<<TWINT)|(1<<TWEN)|(1<<TWEA);          //Clear TWI to proceed send (ACK)
        while ((TWCR & (1<<TWINT))==0) {};                      // Wait for TWI to be ready
        *point = TWDR;
}
TWCR=(1<<TWINT)|(1<<TWEN);                      // Send NACK and clear TWINT to proceed
while ((TWCR & (1<<TWINT))==0) {};                      // Wait for TWI to be ready
statR = TWIcheckStatus();                               //Check for the right status code
if (statR != 0x058)
        goto fun_end;
TWCR=(1<<TWINT)|(1<<TWSTO)|(1<<TWEN);                   //send STOP Condition
statR = TWIcheckStatus();                               //Check for the right status code
if (statR != 0xF8)
        goto fun_end;
fun_end: return statR;                                  //Return status code
}
```

Πρόγραμμα 4.4: Κώδικας της συνάρτησης BurstReadReg(Συνέχεια)

BurstReadOpecode_8B

Η συνάρτηση BurstReadOpecode_8B δέχεται τρία ορίσματα. Το devAddr, το οποίο είναι η 8-bit διεύθυνση της συσκευής που θα επικοινωνήσει. Τον πίνακα data στον οποίο θα αποθηκευτούν τα δεδομένα που θα αναγνωστούν. Το lngth που είναι ένας ακέραιος αριθμός και δηλώνει τον αριθμό των bytes που θα αναγνωστούν. Η συνάρτηση αυτή επικοινωνεί με

μία συσκευή 8-bit διεύθυνσης η οποία επιστρέφει τα δεδομένα που επιθυμεί ο χρήστης, χωρίς να τα διαβάσει από κάποιο καταχωρητή.

```
uint8_t BurstReadOpecode_8B (uint8_t devAddr, uint8_t* data, int lngth)
{
uint8_t writeAddr, readAddr, statR;
uint8_t* point;
writeAddr = devAddr<<1;
readAddr = writeAddr + 1;
TWCR=(1<<TWINT)|(1<<TWSTA)|(1<<TWEN)|(1<<TWEA);          //send Start Condition
 while ((TWCR & (1<<TWINT))==0) {};                      // Wait for TWI to be ready
statR = TWIcheckStatus();                               //Check for the right status code
if (statR != 0x08)
        goto fun_end;
TWDR=writeAddr;                                         //Send the SLA+W
TWCR = (1<<TWINT)|(1<<TWEN)|(1<<TWEA);                  //Clear TWI to proceed wait (ACK)
while ((TWCR & (1<<TWINT))==0) {};                      // Wait for TWI to be ready
statR = TWIcheckStatus();
if (statR != 0x18)
        goto fun_end;
TWCR=(1<<TWINT)|(1<<TWSTA)|(1<<TWEN)|(1<<TWEA);          //Send rep.Start
while ((TWCR & (1<<TWINT))==0) {};                      // Wait for TWI to be ready
TWDR=readAddr;                                          //Send SLA+R
TWCR = (1<<TWINT)|(1<<TWEN)|(1<<TWEA);                  //Clear TWI to proceed send (ACK)
while ((TWCR & (1<<TWINT))==0) {};                      // Wait for TWI to be ready
statR = TWIcheckStatus();                               //Check for the right status code
if (statR != 0x40)
        goto fun_end;
for (int i=0; i<lngth; i++)
{
        point = data++;
        TWCR = (1<<TWINT)|(1<<TWEN)|(1<<TWEA);          //Clear TWI to proceed send (ACK)
        while ((TWCR & (1<<TWINT))==0) {};              // Wait for TWI to be ready
        *point = TWDR;
}
TWCR=(1<<TWINT)|(1<<TWEN);                      // Send NACK and clear TWINT to proceed
while ((TWCR & (1<<TWINT))==0) {};                      // Wait for TWI to be ready
statR = TWIcheckStatus();
if (statR != 0x058)
        goto fun_end;
TWCR=(1<<TWINT)|(1<<TWSTO)|(1<<TWEN);                   //send STOP Condition
statR = TWIcheckStatus();                               //Check for the right status code
if (statR != 0xF8)
        goto fun_end;
fun_end: return statR;                                  //Return Status Code
}
```

Πρόγραμμα 4.5: Κώδικας της συνάρτησης BurstReadOpcode_8B

BurstWriteReg

Η συνάρτηση BurstWriteReg δέχεται τέσσερα ορίσματα. Το devAddr το οποίο είναι η 8-bit διεύθυνση της συσκευής που θα επικοινωνήσει. Το regAddr είναι η διεύθυνση του καταχωρητή της συσκευής από τον οποίο θα ξεκινήσει η εγγραφή των δεδομένων. Τον πίνακα data ο οποίος περιέχει τα δεδομένα που θα γραφτούν. Το lngth που είναι ένας ακέραιος αριθμός και δηλώνει τον αριθμό των bytes που θα εγγραφούν.

```
uint8_t BurstWriteReg (uint8_t devAddr, uint8_t regAddr, uint8_t* data, int lngth)
{
uint8_t writeAddr, stat;
uint8_t* point;
writeAddr = devAddr<<1;
point = data;
TWCR=(1<<TWINT)|(1<<TWSTA)|(1<<TWEN)|(1<<TWEA);          //Send START Condition
while ((TWCR & (1<<TWINT)) == 0){};                      // Wait for TWI to be ready
stat = TWIcheckStatus();                                 //Check for the right status code
if (stat != 0x08)
        goto fun_end;
TWDR=writeAddr;                                          //Send the SLA+W
TWCR = (1<<TWINT)|(1<<TWEN)|(1<<TWEA);       // Wait ACK and clear TWINT to proceed
while ((TWCR & (1<<TWINT)) == 0){};                      // Wait for TWI to be ready
stat = TWIcheckStatus();                                 //Check for the right status code
if (stat != 0x18)
        goto fun_end;
TWDR=regAddr;

while ((TWCR & (1<<TWINT)) == 0){};                      // Wait for TWI to be ready
stat = TWIcheckStatus();                                 //Check for the right status code
 if (stat != 0x28)
        goto fun_end;
for (int i=0; i<lngth; i++)
{
        TWDR= *point;
        TWCR = (1<<TWINT)|(1<<TWEN)|(1<<TWEA);       // Wait ACK and clear TWINT
        while ((TWCR & (1<<TWINT)) == 0){};                  // Wait for TWI to be ready
        point++;
}
stat = TWIcheckStatus();                                 //Check for the right status code
if (stat != 0x28)
        goto fun_end;
TWCR=(1<<TWINT)|(1<<TWSTO)|(1<<TWEN);                    //Send STOP Condition
stat = TWIcheckStatus();                                 //Check for the right status code
if (stat != 0xF8)
        goto fun_end;
fun_end: return stat;                                    //Return Status Code
}
```

Πρόγραμμα 4.6: Κώδικας της συνάρτησης BurstWriteReg

ReadReg

Η συνάρτηση ReadReg δέχεται δύο ορίσματα. Το devAddr το οποίο είναι η 8-bit διεύθυνση της συσκευής που θα επικοινωνήσει. Το regAddr είναι η διεύθυνση του καταχωρητή της συσκευής από τον οποίο θα αναγνωστούν τα δεδομένα. Η συνάρτηση αυτή επικοινωνεί με μία συσκευή 8-bit διεύθυνσης και διαβάζει ένα καταχωρητή επιστρέφοντας την τιμή του.

```
uint8_t ReadReg (uint8_t devAddr, uint8_t regAddr)
{
uint8_t writeAddr, readAddr, statR;
uint8_t readValue;
writeAddr = devAddr<<1;
readAddr = writeAddr + 1;
TWCR=(1<<TWINT)|(1<<TWSTA)|(1<<TWEN)|(1<<TWEA);          //Send Start Condition
while ((TWCR & (1<<TWINT))==0) {};                        // Wait for TWI to be ready
statR = TWIcheckStatus();                              //Check for the right status code
if (statR != 0x08)
        goto fun_end;
TWDR=writeAddr;                                           //Send the SLA+W

TWCR = (1<<TWINT)|(1<<TWEN)|(1<<TWEA);                 //Clear TWI to proceed wait (ACK)
while ((TWCR & (1<<TWINT))==0) {};                        // Wait for TWI to be ready
statR = TWIcheckStatus();                              //Check for the right status code
if (statR != 0x18)
        goto fun_end;
TWDR=regAddr;
TWCR = (1<<TWINT)|(1<<TWEN)|(1<<TWEA);                 //Clear TWI to proceed send (ACK)
while ((TWCR & (1<<TWINT))==0) {}                         ;// Wait for TWI to be ready
statR = TWIcheckStatus();                              //Check for the right status code
if (statR != 0x28)
        goto fun_end;
TWCR=(1<<TWINT)|(1<<TWSTA)|(1<<TWEN)|(1<<TWEA);          //Send rep.Start
while ((TWCR & (1<<TWINT))==0) {};                        // Wait for TWI to be ready
statR = TWIcheckStatus();                              //Check for the right status code
if (statR != 0x10)
        goto fun_end;
TWDR=readAddr;                                            //Send SLA+R
TWCR = (1<<TWINT)|(1<<TWEN)|(1<<TWEA);                 //Clear TWI to proceed send (ACK)
while ((TWCR & (1<<TWINT))==0) {};                        // Wait for TWI to be ready
statR = TWIcheckStatus();                              //Check for the right status code
if (statR != 0x40)
        goto fun_end;
TWCR = (1<<TWINT)|(1<<TWEN)|(1<<TWEA);                 //Clear TWI to proceed send (ACK)
while ((TWCR & (1<<TWINT))==0) {};                        // Wait for TWI to be ready
readValue = TWDR;
TWCR=(1<<TWINT)|(1<<TWEN);                    // Send NACK and clear TWINT to proceed
```

Πρόγραμμα 4.7: Κώδικας της συνάρτησης ReadReg

```
while ((TWCR & (1<<TWINT))==0) {};                    // Wait for TWI to be ready
statR = TWIcheckStatus();                            //Check for the right status code
if (statR != 0x058)
        goto fun_end;
TWCR=(1<<TWINT)|(1<<TWSTO)|(1<<TWEN);                //send STOP Condition
 return readValue;
statR = TWIcheckStatus();                            //Check for the right status code
if (statR != 0xF8)
        goto fun_end;
fun_end: return statR;                               //Return Status Code
}
```

Πρόγραμμα 4.7: Κώδικας της συνάρτησης ReadReg(Συνέχεια)

WriteReg

Η συνάρτηση WriteReg δέχεται τρία ορίσματα. Το devAddr το οποίο είναι η 8-bit διεύθυνση της συσκευής που θα επικοινωνήσει. Το regAddr είναι η διεύθυνση του καταχωρητή της συσκευής στον οποίο θα εγγραφούν τα δεδομένα. Την μεταβλητή data που περιέχει τα 8-bit δεδομένα. Η συνάρτηση αυτή επικοινωνεί με μία συσκευή 8-bit διεύθυνσης και γράφει σε ένα καταχωρητή την τιμή που βρίσκεται στην μεταβλητή data.

```
uint8_t WriteReg (uint8_t devAddr, uint8_t regAddr, uint8_t data)
{
uint8_t writeAddr, stat;
writeAddr = devAddr<<1;
TWCR=(1<<TWINT)|(1<<TWSTA)|(1<<TWEN)|(1<<TWEA);      //Send START Condition
while ((TWCR & (1<<TWINT)) == 0){};                  // Wait for TWI to be ready
stat = TWIcheckStatus();                             //Check status code
if (stat != 0x08)
        goto fun_end;
TWDR=writeAddr;                                      //Send the SLA+W
TWCR = (1<<TWINT)|(1<<TWEN)|(1<<TWEA);       // Wait ACK and clear TWINT to proceed
while ((TWCR & (1<<TWINT)) == 0){};                  // Wait for TWI to be ready
stat = TWIcheckStatus();                             //Check status code
if (stat != 0x18)
        goto fun_end;
TWDR=regAddr;
TWCR = (1<<TWINT)|(1<<TWEN)|(1<<TWEA);       // Wait ACK and clear TWINT to proceed
 while ((TWCR & (1<<TWINT)) == 0){};                 // Wait for TWI to be ready
stat = TWIcheckStatus();                             //Check status code
if (stat != 0x28)
        goto fun_end;
TWDR= data;
TWCR = (1<<TWINT)|(1<<TWEN)|(1<<TWEA);       // Wait ACK and clear TWINT to proceed
 while ((TWCR & (1<<TWINT)) == 0){};                 // Wait for TWI to be ready
```

Πρόγραμμα 4.8: Κώδικας της συνάρτησης WriteReg

```
stat = TWIcheckStatus();                                    //Check status code
if (stat != 0x28)
        goto fun_end;
TWCR=(1<<TWINT)|(1<<TWSTO)|(1<<TWEN);                       //Send STOP Condition
stat = TWIcheckStatus();                                    //Check status code
if (stat != 0xF8)
        goto fun_end;
fun_end: return stat;                                       //Return Status Code
}
```

Πρόγραμμα 4.8: Κώδικας της συνάρτησης WriteReg (Συνέχεια)

WriteOpecode_8B

Η συνάρτηση WriteOpecode_8B δέχεται δύο ορίσματα. Το devAddr το οποίο είναι η 8-bit διεύθυνση της συσκευής που θα επικοινωνήσει. Την μεταβλητή Opecode η οποία περιέχει την 8-bit εντολή που θα αποσταλεί στην συσκευή . Η συνάρτηση αυτή στέλνει μια εντολή σε μία συσκευή 8-bit διεύθυνσης.

```
uint8_t WriteOpecode_8B (uint8_t devAddr, uint8_t Opecode)
{
uint8_t writeAddr, stat;
writeAddr = devAddr<<1;
TWCR=(1<<TWINT)|(1<<TWSTA)|(1<<TWEN)|(1<<TWEA); //Send START Condition
while ((TWCR & (1<<TWINT)) == 0){};                         // Wait for TWI to be ready
stat = TWIcheckStatus();                                    //Check for the right status code
if (stat != 0x08)
        goto fun_end;
TWDR=writeAddr;                                             //Send the SLA+W
TWCR = (1<<TWINT)|(1<<TWEN)|(1<<TWEA);       // Wait ACK and clear TWINT to proceed
while ((TWCR & (1<<TWINT)) == 0){};                         // Wait for TWI to be ready
stat = TWIcheckStatus();                                    //Check for the right status code
if (stat != 0x18)
        goto fun_end;
TWDR=Opecode;
TWCR = (1<<TWINT)|(1<<TWEN)|(1<<TWEA);       // Wait ACK and clear TWINT to proceed
while ((TWCR & (1<<TWINT)) == 0){};                         // Wait for TWI to be ready
stat = TWIcheckStatus();                                    //Check for the right status code
if (stat != 0x28)
        goto fun_end;
TWCR=(1<<TWINT)|(1<<TWSTO)|(1<<TWEN);                        //Send STOP Condition
stat = TWIcheckStatus();                                    //Check for the right status code
if (stat != 0xF8)
        goto fun_end;
fun_end: return stat;                                       //Return Status Code
}
```

Πρόγραμμα 4.9: Κώδικας της συνάρτησης WriteOpecode_8B

WriteReg_16B

Η συνάρτηση WriteReg_16B δέχεται τρία ορίσματα. Το devAddr το οποίο είναι η 8-bit διεύθυνση της συσκευής που θα επικοινωνήσει. Το WAddr το οποίο είναι η 16-bit διεύθυνση του καταχωρητή της συσκευής. Την μεταβλητή data που περιέχει τα 8-bit δεδομένα. Η συνάρτηση αυτή επικοινωνεί με μία συσκευή 8-bit διεύθυνσης και γράφει σε ένα καταχωρητή 16-bit διεύθυνσης την τιμή που βρίσκεται στην μεταβλητή data.

```
uint8_t WriteReg_16B(uint8_t devAddr,uint16_t WAddr ,uint8_t data)
{
uint8_t writeAddr,stat;
writeAddr = devAddr<<1;
uint8_t FWA=WAddr >> 8;
uint8_t SWA=WAddr -(FWA <<8);
TWCR=(1<<TWINT)|(1<<TWSTA)|(1<<TWEN)|(1<<TWEA);          //Send START Condition
while ((TWCR & (1<<TWINT)) == 0){};                      // Wait for TWI to be ready
stat = TWIcheckStatus();                                 //Check for the right status code
if (stat != 0x08)
        goto fun_end;
TWDR=writeAddr;                                          //Send the SLA+W
TWCR = (1<<TWINT)|(1<<TWEN)|(1<<TWEA);       // Wait ACK and clear TWINT to proceed
while ((TWCR & (1<<TWINT)) == 0){};                      // Wait for TWI to be ready
stat = TWIcheckStatus();                                 //Check for the right status code
if (stat != 0x18)
        goto fun_end;
TWDR=FWA;
TWCR = (1<<TWINT)|(1<<TWEN)|(1<<TWEA);
while ((TWCR & (1<<TWINT)) == 0){};
TWDR=SWA;
TWCR = (1<<TWINT)|(1<<TWEN)|(1<<TWEA);       // Wait ACK and clear TWINT to proceed
while ((TWCR & (1<<TWINT)) == 0){};                      // Wait for TWI to be ready
stat = TWIcheckStatus();                                 //Check for the right status code
if (stat != 0x28)
        goto fun_end;
TWDR= data;
TWCR = (1<<TWINT)|(1<<TWEN)|(1<<TWEA);       // Wait ACK and clear TWINT to proceed
while ((TWCR & (1<<TWINT)) == 0){};                      // Wait for TWI to be ready
stat = TWIcheckStatus();                                 //Check for the right status code
if (stat != 0x28)
        goto fun_end;
TWCR=(1<<TWINT)|(1<<TWSTO)|(1<<TWEN);                    //Send STOP Condition
stat = TWIcheckStatus();                                 //Check for the right status code
if (stat != 0xF8)
        goto fun_end;
fun_end: return stat;                                    //Return Status Code
}
```

Πρόγραμμα 4.10: Κώδικας της συνάρτησης WriteOpecode_16B

ReadReg_16B

Η συνάρτηση ReadReg δέχεται δύο ορίσματα. Το devAddr το οποίο είναι η 8-bit διεύθυνση της συσκευής που θα επικοινωνήσει. Το WAddr το οποίο είναι η 16-bit διεύθυνση του καταχωρητη. Η συνάρτηση αυτή επικοινωνεί με μία συσκευή 8-bit διεύθυνσης και επιστρέφει το περιεχόμενο του επιθυμητού καταχωρητή.

```
uint8_t ReadReg_16B(uint8_t devAddr,uint16_t WAddr)
{
uint8_t writeAddr, readAddr,statR;
uint8_t readValue;
uint8_t FWA=WAddr >> 8;
uint8_t SWA=WAddr -(FWA <<8);
writeAddr = devAddr<<1;
readAddr = writeAddr + 1;
TWCR=(1<<TWINT)|(1<<TWSTA)|(1<<TWEN)|(1<<TWEA);          //Send START Condition
while ((TWCR & (1<<TWINT))==0) {};                        //Wait for TWI to be ready
statR = TWIcheckStatus();                                 //Check for the right status code
if (statR != 0x08)
        goto fun_end;
TWDR=writeAddr;                                           //Send the SLA+W
TWCR = (1<<TWINT)|(1<<TWEN)|(1<<TWEA);       //Wait for ACK and clear TWI to proceed
while ((TWCR & (1<<TWINT))==0) {}; /                       //Wait for TWI to be ready
statR = TWIcheckStatus();                                 //Check for the right status code
if (statR != 0x18)
        goto fun_end;
TWDR=FWA;
TWCR = (1<<TWINT)|(1<<TWEN)|(1<<TWEA);       //Wait for ACK and clear TWI to proceed
while ((TWCR & (1<<TWINT))==0) {};                        //Wait for TWI to be ready
TWDR=SWA;
TWCR = (1<<TWINT)|(1<<TWEN)|(1<<TWEA);       //Wait for ACK and clear TWI to proceed
while ((TWCR & (1<<TWINT))==0) {};                        //Wait for TWI to be ready
statR = TWIcheckStatus();                                 //Check for the right status code
if (statR != 0x28)
        goto fun_end;
TWCR=(1<<TWINT)|(1<<TWSTA)|(1<<TWEN)|(1<<TWEA);          //Send rep.Start
while ((TWCR & (1<<TWINT))==0) {};
statR = TWIcheckStatus();                                 //Check for the right status code
if (statR != 0x10)
        goto fun_end;
TWDR=readAddr;                                            //Send SLA+R
TWCR = (1<<TWINT)|(1<<TWEN)|(1<<TWEA);       //Wait for ACK and clear TWI to proceed
while ((TWCR & (1<<TWINT))==0) {};                        //Wait for TWI to be ready
statR = TWIcheckStatus();                                 //Check for the right status code
if (statR != 0x40)
        goto fun_end;
TWCR = (1<<TWINT)|(1<<TWEN)|(1<<TWEA);       //Wait for ACK and clear TWI to proceed
```

Πρόγραμμα 4.11: Κώδικας της συνάρτησης ReadReg_16B

```
while ((TWCR & (1<<TWINT))==0) {};                    //Wait for TWI to be ready
readValue = TWDR;
TWCR=(1<<TWINT)|(1<<TWEN);                    //Send NACK and clear TWI to proceed
while ((TWCR & (1<<TWINT))==0) {};                    //Wait for TWI to be ready
statR = TWIcheckStatus();                    //Check for the right status code
if (statR != 0x058)
        goto fun_end;
TWCR=(1<<TWINT)|(1<<TWSTO)|(1<<TWEN);                    //Send STOP Condition
return readValue;
statR = TWIcheckStatus();                    //Check for the right status code
if (statR != 0xF8)
        goto fun_end;
fun_end: return statR;                    //Return Status Code
}
```

Πρόγραμμα 4.11: Κώδικας της συνάρτησης ReadReg_16B(Συνέχεια)

PageWriteReg_16B

Η συνάρτηση PageWriteReg_16B δέχεται τέσσερα ορίσματα. Το devAddr το οποίο είναι η 8-bit διεύθυνση της συσκευής που θα επικοινωνήσει. Το WAddr το οποίο είναι η 16-bit διεύθυνση του καταχωρητή από τον οποίο θα ξεκινήσει η εγγραφή. Τον πίνακα data που περιέχει τα δεδομένα που θα εγγραφούν. Το μήκος lngth που δηλώνει τον αριθμό των bytes που θα εγγραφούν. Η συνάρτηση αυτή δημιουργήθηκε για την εγγραφή δεδομένων ξεκινώντας από μια αρχική 16-bit διεύθυνση καταχωρητή ή μνήμης. Στην περίπτωση της EEPROM η συγκεκριμένη συνάρτηση μπορεί να αξιοποιηθεί για την εγγραφή δεδομένων σε διευθύνσεις που ανήκουν στην ίδια σελίδα.

```
uint8_t PageWriteReg_16B (uint8_t devAddr, uint16_t WAddr, uint8_t* data, int lngth)
{
uint8_t writeAddr, stat;
uint8_t* point;
uint8_t FWA=WAddr >> 8;
uint8_t SWA=WAddr -(FWA <<8);
writeAddr = devAddr<<1;
point = data;
TWCR=(1<<TWINT)|(1<<TWSTA)|(1<<TWEN)|(1<<TWEA);          //Send START Condition
while ((TWCR & (1<<TWINT)) == 0){};                    // Wait for TWI to be ready
stat = TWIcheckStatus();                    //Check for the right status code
if (stat != 0x08)
        goto fun_end;
TWDR=writeAddr;                    //Send the SLA+W
TWCR = (1<<TWINT)|(1<<TWEN)|(1<<TWEA);        // Wait ACK and clear TWINT to proceed
while ((TWCR & (1<<TWINT)) == 0){};                    // Wait for TWI to be ready
stat = TWIcheckStatus();                    //Check for the right status code
if (stat != 0x18)
        goto fun_end;
TWDR=FWA;
```

Πρόγραμμα 4.12: Κώδικας της συνάρτησης PageWriteReg_16B

```
TWCR = (1<<TWINT)|(1<<TWEN)|(1<<TWEA);        // Wait ACK and clear TWINT to proceed
while ((TWCR & (1<<TWINT)) == 0){};                    // Wait for TWI to be ready
TWDR=SWA;
TWCR = (1<<TWINT)|(1<<TWEN)|(1<<TWEA);        // Wait ACK and clear TWINT to proceed
while ((TWCR & (1<<TWINT)) == 0){};                    // Wait for TWI to be ready
stat = TWIcheckStatus();                                 //Check for the right status code
if (stat != 0x28)
        goto fun_end;
for (int i=0; i<lngth; i++)
{
        TWDR= *point;
        TWCR = (1<<TWINT)|(1<<TWEN)|(1<<TWEA);        // Wait ACK and clear TWINT
        while ((TWCR & (1<<TWINT)) == 0){};                    // Wait for TWI to be ready
        point++;
}
stat = TWIcheckStatus();                                 //Check for the right status code
if (stat != 0x28)
        goto fun_end;
TWCR=(1<<TWINT)|(1<<TWSTO)|(1<<TWEN);                  //Send STOP Condition
stat = TWIcheckStatus();                                 //Check for the right status code
if (stat != 0xF8)
        goto fun_end;
fun_end: return stat;                                    //Return Status Code
}
```

Πρόγραμμα 4.12: Κώδικας της συνάρτησης PageWriteReg_16B (Συνέχεια)

BurstReadReg_16B

Η συνάρτηση BurstReadReg_16B δέχεται τέσσερα ορίσματα. Το devAddr το οποίο είναι η 8-bit διεύθυνση συσκευής που θα επικοινωνήσει. Το WAddr το οποίο είναι η 16-bit διεύθυνση καταχωρητή. Τον πίνακα data που θα περιέχει τα δεδομένα που θα διαβαστούν. Το μήκος lngth που δηλώνει τον αριθμό των bytes που θα διαβαστούν. Η συνάρτηση αυτή έχει σκοπό την ανάγνωση δεδομένων μήκους lngth ξεκινώντας από την δοθείσα 16-bit διεύθυνση.

```
uint8_t BurstReadReg_16B (uint8_t devAddr, uint8_t WAddr,uint8_t* data,int lngth)
{
uint8_t writeAddr, readAddr,statR;
uint8_t readValue;
uint8_t FWA=WAddr >> 8;
uint8_t SWA=WAddr -(FWA <<8);
writeAddr = devAddr<<1;
readAddr = writeAddr + 1;
TWCR=(1<<TWINT)|(1<<TWSTA)|(1<<TWEN)|(1<<TWEA);        //Send START Condition
while ((TWCR & (1<<TWINT))==0) {};                      //Wait for TWI to be ready
statR = TWIcheckStatus();                                //Check for the right status
if (statR != 0x08)
```

Πρόγραμμα 4.13: Κώδικας της συνάρτησης BurstReadReg_16B

```
        goto fun_end;
TWDR=writeAddr;                                         //Send the SLA+W
TWCR = (1<<TWINT)|(1<<TWEN)|(1<<TWEA);      //Wait ACK and clear TWI to be ready
while ((TWCR & (1<<TWINT))==0) {};                      //Wait for TWI to be ready
statR = TWIcheckStatus();                               //Check for the right status code
if (statR != 0x18)
        goto fun_end;
TWDR=FWA;
TWCR = (1<<TWINT)|(1<<TWEN)|(1<<TWEA);      //Wait ACK and clear TWI to be ready
while ((TWCR & (1<<TWINT))==0) {};                      //Wait for TWI to be ready
TWDR=SWA;
TWCR = (1<<TWINT)|(1<<TWEN)|(1<<TWEA);      //Wait ACK and clear TWI to be ready
while ((TWCR & (1<<TWINT))==0) {};                      //Wait for TWI to be ready
statR = TWIcheckStatus();                               //Check for the right status code
if (statR != 0x28)
        goto fun_end;
TWCR=(1<<TWINT)|(1<<TWSTA)|(1<<TWEN)|(1<<TWEA);         //Send rep.Start
while ((TWCR & (1<<TWINT))==0) {};
statR = TWIcheckStatus();                               //Check for the right status code
if (statR != 0x10)
        goto fun_end;
TWDR=readAddr;                                          //Send SLA+R
TWCR = (1<<TWINT)|(1<<TWEN)|(1<<TWEA);      //Wait ACK and clear TWI to be ready
while ((TWCR & (1<<TWINT))==0) {};                      //Wait for TWI to be ready
statR = TWIcheckStatus();                               //Check for the right status code
if (statR != 0x40)
        goto fun_end;
TWCR = (1<<TWINT)|(1<<TWEN)|(1<<TWEA);      //Wait ACK and clear TWI to be ready
while ((TWCR & (1<<TWINT))==0) {};                      //Wait for TWI to be ready
for (int i=0; i<lngth-1; i++)
{
        data[i] = TWDR;
        TWCR = (1<<TWINT)|(1<<TWEN)|(1<<TWEA);      //Clear TWI to proceed send (ACK)
        while ((TWCR & (1<<TWINT))==0) {};                  // Wait for TWI to be ready
}
data[lngth-1] = TWDR;
TWCR=(1<<TWINT)|(1<<TWEN);                  // Send NACK and clear TWINT to proceed
while ((TWCR & (1<<TWINT))==0) {};                      // Wait for TWI to be ready
TWCR=(1<<TWINT)|(1<<TWSTO)|(1<<TWEN);                   //Send STOP Condition
statR = TWIcheckStatus();                               //Check for the right status code
if (statR != 0xF8)
        goto fun_end;
fun_end: return statR;                                  //Return Status Code
}
```

Πρόγραμμα 4.13: Κώδικας της συνάρτησης BurstReadReg_16B (Συνέχεια)

BurstWriteReg_16B

Η συνάρτηση BurstWriteReg_16B δέχεται τέσσερα ορίσματα. Το devAddr το οποίο είναι η 8-bit διεύθυνση της συσκευής που θα επικοινωνήσει. Το WAddr το οποίο είναι η 16-bit διεύθυνση καταχωρητή ή μνήμης. Τον πίνακα data που περιέχει τα δεδομένα που θα εγγραφούν. Το μήκος lngth που δηλώνει το μήκος των δεδομένων που θα εγγραφούν. Επειδή η συνάρτηση αυτή καλείται μέσα στην συνάρτηση των interrupt δηλώνεται ως volatile για να μπορεί να ανταλλάσει τα δεδομένα της με το υπόλοιπο πρόγραμμα. Η συνάρτηση δημιουργήθηκε με σκοπό την εγγραφή δεδομένων στην μνήμη EEPROM αλλάζοντας αυτόματα σελίδα στην περίπτωση τα εναπομείναντα δεδομένα δεν χωρούν στην τρέχουσα σελίδα .

```
volatile uint8_t BurstWriteReg_16B (uint8_t devAddr, uint16_t WAddr, uint8_t* data, int lngth)
{
uint16_t n_addr,pg_len=32,pg,pages;
int byte_num,data_len;
uint8_t *p,stat;
data_len=lngth;
p = data;
pg = WAddr/pg_len;                                  //Starting page
n_addr = (pg+1)*pg_len;                             //The Address of the first byte of the next page
byte_num = n_addr - WAddr;                          //Remaining bytes of the starting  page
if(byte_num>=data_len)                              //Check the remaining space of the starting  page
 {                                                  //The data is stored only in remaining space of the starting page
        stat = PageWriteReg_16B(devAddr,WAddr,p,data_len); //Write data in the starting  page
        delay(6);
        if (stat != 0xF8)                           //Check write status
        goto fun_end;
}
else
 {                                                  //The data is stored  in more than one pages
        stat = PageWriteReg_16B(devAddr,WAddr,p,byte_num);      //Write data in the start page
        delay(6);
        if (stat != 0xF8)                           //Check write status
        goto fun_end;
        data_len  = data_len - byte_num;            //Calculation of the remaining bytes
        p = p + byte_num;                           //Set the data pointer to following data
        pages = data_len/pg_len;                    //Calculation of the number of the filled pages
for(int i=0; i<pages; i++)                          //Write data to the following pages
        {
                stat = PageWriteReg_16B(devAddr,n_addr,p,pg_len);   //Write data to the next page
        delay(6);
        if (stat != 0xF8)                                           //Check write status
        goto fun_end;
        n_addr = n_addr + pg_len;       //Calculation of the starting address of the next page
        p = p + pg_len;                 //Set the data pointer to following data
        data_len  = data_len - pg_len;                  //Calculation of remaining data
```

Πρόγραμμα 4.14: Κώδικας της συνάρτησης BurstWriteReg_16B

```
        }
if (data_len > 0)                                   //Check if there is remaining data
{
stat = PageWriteReg_16B(devAddr,n_addr,p,data_len); //Write remaining data to the last page
delay(6);
if (stat != 0xF8)                                   //Check write status
        goto fun_end;
}
}
fun_end: return stat;                               //Return the status
}
```

Πρόγραμμα 4.14: Κώδικας της συνάρτησης BurstWriteReg_16B(Συνέχεια)

4.3 Συναρτήσεις για την λειτουργία του RTC DS3231

Για τον έλεγχο του RTC DS3231 δεν χρησιμοποιήθηκε έτοιμη βιβλιοθήκη, αλλά οι συναρτήσεις που δημιουργήθηκαν για το πρωτόκολλο I2C καθώς και κάποιες συναρτήσεις που δημιουργήθηκαν για τον έλεγχο της ώρας, ημερομηνίας καθώς και τον υπόλοιπων δυνατοτήτων του. Ο έλεγχος του RTC έγινε με βάση την εγγραφή και την ανάγνωση των διαθέσιμων καταχωρητών που παρέχει στον χρήστη για την αξιοποίηση των λειτουργιών του. Οι καταχωρητές αυτοί παρουσιάζονται στο Σχήμα 4.2 και αναλύονται παρακάτω.

Οι πρώτοι 7 καταχωρητές χρησιμοποιούνται για την καταγραφή της ώρας και της ημερομηνίας και χρησιμοποιούν BCD (Binary Coded Decimal) μορφοποίηση για τις τιμές. Δηλαδή, κάθε δεκαδικό ψηφίο αντιπροσωπεύεται από ένα σταθερό αριθμό bits, που στην συγκεκριμένη περίπτωση είναι 4.

Καταχωρητής δευτερολέπτων

Ο 8-bit καταχωρητής δευτερολέπτων με διεύθυνση 0x00 αναλαμβάνει την καταγραφή των δευτερολέπτων και παίρνει τιμές από 0-59. Ο καταχωρητής αυτός χρησιμοποιεί τα bits 0-3 για την καταγραφή των μονάδων των δευτερολέπτων και τα bits 4-6 για την καταγραφή των δεκάδων των δευτερολέπτων. Το bit 7 δεν χρησιμοποιείται επομένως έχει πάντα την τιμή ‘0’.

Καταχωρητής λεπτών

Ο 8-bit καταχωρητής λεπτών με διεύθυνση 0x01 αναλαμβάνει την καταγραφή των λεπτών και παίρνει τιμές από 0-59. Ο καταχωρητής αυτός χρησιμοποιεί τα bits 0-3 για την καταγραφή των μονάδων των λεπτών και τα bits 4-6 για την καταγραφή των δεκάδων των λεπτών. Το bit 7 δεν χρησιμοποιείται επομένως έχει πάντα την τιμή ‘0’.

ADDRESS	BIT 7 MSB	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0 LSB	FUNCTION	RANGE
00h	0	10 Seconds			Seconds				Seconds	00–59
01h	0	10 Minutes			Minutes				Minutes	00–59
02h	0	12/$\overline{24}$	$\overline{AM}$/PM 20 Hour	10 Hour	Hour				Hours	1–12 + $\overline{AM}$/PM 00–23
03h	0	0	0	0	0	Day			Day	1–7
04h	0	0	10 Date		Date				Date	01–31
05h	Century	0	0	10 Month	Month				Month/ Century	01–12 + Century
06h	10 Year				Year				Year	00–99
07h	A1M1	10 Seconds			Seconds				Alarm 1 Seconds	00–59
08h	A1M2	10 Minutes			Minutes				Alarm 1 Minutes	00–59
09h	A1M3	12/$\overline{24}$	$\overline{AM}$/PM 20 Hour	10 Hour	Hour				Alarm 1 Hours	1–12 + $\overline{AM}$/PM 00–23
0Ah	A1M4	DY/$\overline{DT}$	10 Date		Day				Alarm 1 Day	1–7
					Date				Alarm 1 Date	1–31
0Bh	A2M2	10 Minutes			Minutes				Alarm 2 Minutes	00–59
0Ch	A2M3	12/$\overline{24}$	$\overline{AM}$/PM 20 Hour	10 Hour	Hour				Alarm 2 Hours	1–12 + $\overline{AM}$/PM 00–23
0Dh	A2M4	DY/$\overline{DT}$	10 Date		Day				Alarm 2 Day	1–7
					Date				Alarm 2 Date	1–31
0Eh	$\overline{EOSC}$	BBSQW	CONV	RS2	RS1	INTCN	A2IE	A1IE	Control	—
0Fh	OSF	0	0	0	EN32kHz	BSY	A2F	A1F	Control/Status	—
10h	SIGN	DATA	DATA	DATA	DATA	DATA	DATA	DATA	Aging Offset	—
11h	SIGN	DATA	DATA	DATA	DATA	DATA	DATA	DATA	MSB of Temp	—
12h	DATA	DATA	0	0	0	0	0	0	LSB of Temp	—

Σχήμα 4.2: Οι καταχωρητές του RTC DS3231

Καταχωρητής ωρών

Ο 8-bit καταχωρητής ωρών με διεύθυνση 0x02 αναλαμβάνει την καταγραφή των ωρών και παίρνει τιμές από 0-23 ή 0-12 με δείκτη μμ-πμ. Ο καταχωρητής αυτός χρησιμοποιεί τα bits 0-3 για την καταγραφή των μονάδων των ωρών. Το bit 6 δίνει την δυνατότητα στον χρήστη να επιλέξει τον τρόπο μέτρησης της ώρας ανάμεσα σε 24ωρο ή σε 12ωρο με την λειτουργία δείκτη μμ-πμ. Το bit 5 εξαρτάται από το bit 6, εάν δηλαδή το bit 6 μετράει σε 24ωρη μορφή τότε το bit 5 μετράει τις εικοσάδες των ωρών, αλλιώς περιέχει την πληροφορία AM/PM. Το bit 4 περιέχει τις δεκάδες των ωρών ενώ το bit 7 δεν χρησιμοποιείται επομένως έχει πάντα την τιμή ‘0’.

Καταχωρητής ημέρας της εβδομάδας

Ο 8-bit καταχωρητής ημέρας της εβδομάδας με διεύθυνση 0x03 αναλαμβάνει την καταγραφή των ημερών της εβδομάδας και παίρνει τιμές από 1-7. Ο καταχωρητής αυτός χρησιμοποιεί τα bits 0-2 για την καταγραφή της ημέρας. Τα υπόλοιπα bits έχουν την τιμή ‘0’.

Καταχωρητής ημερομηνίας

Ο 8-bit καταχωρητής ημερομηνίας με διεύθυνση 0x04 αναλαμβάνει την καταγραφή της ημερομηνίας και παίρνει τιμές από 1-31. Ο καταχωρητής αυτός χρησιμοποιεί τα bits 0-3 για την καταγραφή των μονάδων της ημερομηνίας και τα bits 4-5 για τις δεκάδες. Τα υπόλοιπα bits έχουν τιμή ‘0’.

Καταχωρητής μήνα

Ο 8-bit καταχωρητής μήνα με διεύθυνση 0x05 αναλαμβάνει την καταγραφή των μηνών και παίρνει τιμές από 1-12. Ο καταχωρητής αυτός χρησιμοποιεί τα bits 0-3 για την καταγραφή των μονάδων των μηνών και το bit 4 για την καταγραφή των δεκάδων των μηνών. Τα bits 5-6 είναι πάντα ‘0’ ενώ το bit 7 χρησιμοποιείται για να ενημερώνει πως πέρασε ένας αιώνας όταν η τιμή του γίνεται ‘1’.

Καταχωρητής χρονιάς

Ο 8-bit καταχωρητής χρονιάς με διεύθυνση 0x06 αναλαμβάνει την καταγραφή των χρόνων και παίρνει τιμές από 0-99. Ο καταχωρητής αυτός χρησιμοποιεί τα bit 0-3 για την καταγραφή των μονάδων των χρόνων και τα bits 4-7 για την καταγραφή των δεκάδων των χρόνων.

Καταχωρητής δευτερολέπτων Alarm1

Ο 8-bit καταχωρητής δευτερολέπτων Alarm1 με διεύθυνση 0x07 περιέχει την τιμή των δευτερολέπτων που θα συγκριθεί με τον καταχωρητή των δευτερολέπτων του RTC ώστε να είναι εφικτή η ενεργοποίηση του Alarm1. Τα bits του καταχωρητή αυτού είναι όπως και του καταχωρητή δευτερολέπτων εκτός από το bit 7 (A1M1) που μπορεί να πάρει τιμές ‘0’ ή ‘1’ ανάλογα με την λειτουργία του Alarm που αναλύεται παρακάτω.

Η λειτουργία των Alarms επιλέγεται βάση του συνδυασμού των καταχωρητών που αντιστοιχούν σε κάθε Alarm. Πιο συγκεκριμένα ο συνδυασμός των τιμών του bit 7, που μπορεί να πάρει τιμές ‘1’ ‘ή ‘0,’ των καταχωρητών με διεύθυνση 0x07, 0x08, 0x09 και 0x0A ευθύνεται για την επιλογή της λειτουργίας του Alarm1 και ο συνδυασμός των τιμών του bit 7, που μπορεί να πάρει τιμές ‘1’ ή ‘0’ ,των καταχωρητών με διεύθυνση 0x0B, 0x0C και 0x0D ευθύνεται για

DY/$\overline{DT}$	ALARM 1 REGISTER MASK BITS (BIT 7)				ALARM RATE
	A1M4	A1M3	A1M2	A1M1	
X	1	1	1	1	Alarm once per second
X	1	1	1	0	Alarm when seconds match
X	1	1	0	0	Alarm when minutes and seconds match
X	1	0	0	0	Alarm when hours, minutes, and seconds match
0	0	0	0	0	Alarm when date, hours, minutes, and seconds match
1	0	0	0	0	Alarm when day, hours, minutes, and seconds match

DY/$\overline{DT}$	ALARM 2 REGISTER MASK BITS (BIT 7)			ALARM RATE
	A2M4	A2M3	A2M2	
X	1	1	1	Alarm once per minute (00 seconds of every minute)
X	1	1	0	Alarm when minutes match
X	1	0	0	Alarm when hours and minutes match
0	0	0	0	Alarm when date, hours, and minutes match
1	0	0	0	Alarm when day, hours, and minutes match

Πίνακας 4.2: Λειτουργία των Alarms του RTC DS3231

την επιλογή της λειτουργίας του Alarm2. Επίσης σε κάποιες περιπτώσεις χρειάζεται και το bit 6 (DY/$\overline{DT}$), που μπορεί να πάρει τιμές ‘0’ ή ‘1’, των καταχωρητών με διεύθυνση 0x0A

(Για το Alarm1) και 0x0D (Για το Alarm2) για να επιλεχθούν οι αντίστοιχες λειτουργίες όπως φαίνεται στον Πίνακα 4.2.

Ανάλογα με την επιλεγμένη λειτουργία κάθε Alarm, το RTC συγκρίνει τους καταχωρητές του αντίστοιχου Alarm με τους καταχωρητές του RTC. Αν εντοπιστεί ισότητα στο σύνολο των καταχωρητών που απαιτεί η επιλεγμένη λειτουργία και εφόσον είναι ενεργοποιημένη η λειτουργία τόσο των γενικών διακοπών όσο και του συγκεκριμένου Alarm, παράγεται ένα σήμα διακοπής, το οποίο εμφανίζεται στην έξοδο INT/SQW.

Καταχωρητής λεπτών Alarm1

Ο 8-bit καταχωρητής λεπτών Alarm1 με διεύθυνση 0x08 περιέχει την τιμή των λεπτών που θα συγκριθεί με τον καταχωρητή των λεπτών του RTC ώστε να είναι εφικτή η ενεργοποίηση του Alarm1. Τα bits του καταχωρητή αυτού είναι όπως και του καταχωρητή λεπτών εκτός από το bit 7 (A1M2) που μπορεί να πάρει τιμές ‘0’' ή ‘1’ ανάλογα με την λειτουργία του Alarm.

Καταχωρητής ωρών Alarm1

Ο 8-bit καταχωρητής ωρών Alarm1 με διεύθυνση 0x09 περιέχει την τιμή των ωρών που θα συγκριθεί με τον καταχωρητή των ωρών του RTC ώστε να είναι εφικτή η ενεργοποίηση του Alarm1. Τα bits του καταχωρητή αυτού είναι όπως και του καταχωρητή λεπτών εκτός από το bit 7 (A1M3) που μπορεί να πάρει τιμές ‘0’ ή ‘1’ ανάλογα με την λειτουργία του Alarm1.

Καταχωρητής ημέρας/ημερομηνίας Alarm1

Ο 8-bit καταχωρητής ημέρας/ημερομηνίας Alarm1 με διεύθυνση 0x0A περιέχει την τιμή των ημερών της εβδομάδας ή ημερομηνίας, ανάλογα με την τιμή που έχει το bit 6 (DY/$\overline{\text{DT}}$) του συγκεκριμένου καταχωρητή, που θα συγκρίνει με τον καταχωρητή των ημερών της εβδομάδας ή της ημερομηνίας του RTC ώστε να είναι εφικτή η ενεργοποίηση του Alarm1. Αν η τιμή του bit 6 είναι ‘1’ τότε τα bits 0-2 περιέχουν την ημέρα της εβδομάδας και τα bits 3-5 δεν λαμβάνονται υπόψη. Αν η τιμή του bit 6 είναι ‘0’ τότε τα bits 0-5 περιέχουν την ημερομηνία κατά αντιστοιχία με τον καταχωρητή ημερομηνίας του RTC. Τέλος, το bit 7 (A1M4) μπορεί να πάρει τιμές ‘0’ ή ‘1’ ανάλογα με την λειτουργία του Alarm.

Καταχωρητής λεπτών Alarm2

Ο 8-bit καταχωρητής λεπτών Alarm2 με διεύθυνση 0x0B έχει την ίδια δομή και λειτουργία με τον καταχωρητή λεπτών Alarm1 αλλά χρησιμοποιείται για την ενεργοποίηση του Alarm 2 συμφώνα με την επιλεγμένη λειτουργία των Alarms.

Καταχωρητής ωρών Alarm2

Ο 8-bit καταχωρητής ωρών Alarm2 με διεύθυνση 0x0C έχει την ίδια δομή και λειτουργία με τον καταχωρητή ωρών Alarm1 αλλά χρησιμοποιείται για την ενεργοποίηση του Alarm 2 συμφώνα με την επιλεγμένη λειτουργία των Alarms.

Καταχωρητής ημέρας/ημερομηνίας Alarm2

Ο 8-bit καταχωρητής ημέρας/ημερομηνίας Alarm2 με διεύθυνση 0x0D έχει την ίδια δομή και λειτουργία με τον καταχωρητή ημέρας/ημερομηνίας Alarm1 αλλά χρησιμοποιείται για την ενεργοποίηση του Alarm 2 συμφώνα με την επιλεγμένη λειτουργία των Alarms.

Καταχωρητής ελέγχου

Ο 8-bit καταχωρητής ελέγχου με διεύθυνση 0x0E είναι ένας από τους πιο βασικούς καταχωρητές γιατί αναλαμβάνει τον έλεγχο του RTC. Τα bits 0-1 του καταχωρητή ελέγχου ενεργοποιούν τα Alarm1 και Alarm2 αντίστοιχα όταν έχουν την τιμή ‘1’. Το bit 2 όταν έχει την τιμή ‘0’ στην έξοδο INT/SQW παράγεται ένας τετραγωνικός παλμός, ενώ όταν έχει την τιμή ‘1’ στην έξοδο INT/SQW δημιουργείται ένα σήμα διακοπής που προέρχεται από την λειτουργία των Alarms. Τα bits 3-4 χρησιμοποιούνται για την επιλογή της συχνότητας του τετραγωνικού παλμού, σύμφωνα με Πίνακα 4.3, υπό την προϋπόθεση ότι το bit 2 έχει την τιμή ‘0’. Το bit 5 εξαναγκάζει την μέτρηση της θερμοκρασίας από τον αισθητήρα του RTC όταν το bit 2 (BSY) στον καταχωρητή κατάστασης είναι ‘0’. Όταν το bit 6 του καταχωρητή κατάστασης έχει την τιμή ‘1’ ενεργοποιείται η παράγωγη τετραγωνικών παλμών στην έξοδο INT/SQW υπό την προϋπόθεση ότι η τιμή του bit 2 είναι ‘0’ και ταυτόχρονα η τάση της εφεδρικής μπαταρίας είναι μεγαλύτερη από την τάση τροφοδοσίας. Στην περίπτωση που ισχύουν οι προηγούμενες προϋποθέσεις αλλά η τιμή του bit 6 είναι‘0’, η έξοδος INT/SQW μπαίνει σε κατάσταση υψηλής εμπέδησης. Το bit 7 ενεργοποιεί τον ενσωματωμένο ταλαντωτή όταν έχει την τιμή ‘0’. Εφόσον το συγκεκριμένο bit έχει την τιμή ‘1’, ο ταλαντωτής σταματά την στιγμή που η τροφοδοσία του RTC προέρχεται από την εφεδρική μπαταρία. Όταν η τάση τροφοδοσίας δεν προέρχεται από την εφεδρική μπαταρία η λειτουργία του ταλαντωτή εξαρτάται από την τιμή του bit 7. Στις περιπτώσεις που ο ταλαντωτής απενεργοποιείται το περιεχόμενο των καταχωρητών παραμένει αμετάβλητο.

Bit4	**Bit3**	**Συχνότητα τετραγωνικού παλμού**
0	0	1 Hz
0	1	1024 Hz
1	0	4096 Hz
1	1	8192 Hz

Πίνακας 4.3: Επιλογές της συχνότητας τετραγωνικού παλμού του RTC

Καταχωρητής κατάστασης

Τα bits 0-1 του καταχωρητή κατάστασης με διεύθυνση 0x0F όταν προκληθεί ένα σήμα διακοπής που προέρχεται από την λειτουργία των Alarm 1 και Alarm 2 αντίστοιχα. Προϋπόθεση για την παραπάνω λειτουργία είναι να έχει ενεργοποιηθεί η λειτουργία του αντίστοιχου Alarm μέσω του καταχωρητή ελέγχου (bits 0-1) και ταυτόχρονα το bit 2 (INTCN) του καταχωρητή ελέγχου να έχει την τιμή '1'. Μέσω του προγράμματος ο χρήστης έχει την δυνατότητα να γράψει στα bits 0-1 του καταχωρητή κατάστασης μόνο την τιμή '0'. Το bit 2 ελέγχει ένα η συσκευή είναι απασχολημένη (‘1’) εκτελώντας κάποια TCXO (Temperature Compensated Crystal Oscillator) λειτουργία. Το bit 3 ενεργοποιεί την έξοδο των 32kHz όταν έχει τιμή ‘1’. Τα bits 4-6 δεν χρησιμοποιούνται επομένως έχουν πάντα την

τιμή ‘0’. Το bit 7 δείχνει αν ο ταλαντωτής είναι σταματημένος ή είχε σταματήσει για κάποια χρονική περίοδο. Η λειτουργία αυτή μπορεί να αξιοποιηθεί ώστε επιβεβαιωθεί η αξιοπιστία του μετρούμενου χρόνου από το RTC. Το συγκεκριμένο bit διατηρεί την τιμή ‘0’ μέχρι να εγγραφεί σε αυτό η τιμή ‘0’.

Καταχωρητής αντιστάθμισης γήρανσης

Ο καταχωρητής αντιστάθμισης γήρανσης με διεύθυνση 0x10 περιέχει μια κωδικοποιημένη τιμή η οποία δίνεται από τον χρήστη και προστίθεται ή αφαιρείται από τις διαθέσιμες κωδικοποιημένες τιμές του πίνακα χωρητικοτήτων. Το bit 7 περιέχει το πρόσημο ενώ τα υπόλοιπα bits την κωδικοποιημένη τιμή. Μια θετική τιμή σε αυτό τον καταχωρητή έχει ως αποτέλεσμα την αύξηση της χωρητικότητας που προκαλεί μείωση της συχνότητας του ταλαντωτή. Ενώ μια αρνητική τιμή προκαλεί την μείωση της χωρητικότητας και κατά επέκταση την αύξηση της συχνότητας του ταλαντωτή.

Καταχωρητής θερμοκρασίας

Επειδή η θερμοκρασία δίνεται σαν μια 10-bit τιμή με ακρίβεια 0.25°C δεν μπορεί να αποθηκευτεί σε ένα 8-bit καταχωρητή. Για τον λόγο αυτό ο καταχωρητής θερμοκρασίας στην πραγματικότητα αποτελείται από δύο 8-bit καταχωρητές, τον καταχωρητή θερμοκρασίας MSB με διεύθυνση 0x11 και τον καταχωρητή θερμοκρασίας LSB με διεύθυνση 0x12. Το bit 7 του καταχωρητή θερμοκρασίας MSB περιέχει το πρόσημο ενώ τα υπόλοιπα bits περιέχουν το ακέραιο μέρος της μετρούμενης θερμοκρασίας. Τα δύο πιο σημαντικά bits του καταχωρητή θερμοκρασίας LSB αν πολλαπλασιαστούν με το συντελεστή 0,25 δίνουν το δεκαδικό μέρος της μετρούμενης θερμοκρασίας. Τα υπόλοιπα bits του καταχωρητή θερμοκρασίας LSB έχουν πάντα την τιμή ‘0’.

Με βάση τις λειτουργίες των παραπάνω καταχωρητών δημιουργήθηκαν και οι κατάλληλες συναρτήσεις για την λήψη ώρας, ημερομηνίας, θερμοκρασίας και τον έλεγχο των Alarms του RTC DS3231. Επίσης δημιουργήθηκαν οι συναρτήσεις για την μετατροπή των τιμών που διαβάζονται από το RTC από BCD σε αριθμούς κινητής υποδιαστολής και αντίστροφα.

Num2Bcd

Η συνάρτηση Num2Bcd επιστρέφει τον ακέραιο αριθμό που δέχεται ως όρισμα σε BCD αριθμό.

```
uint8_t Num2Bcd(uint8_t value)          //Function turn num to bcd
{
uint8_t bcdnum=(( (value/10)<<4) + (value%10) );
return bcdnum;
}
```

Πρόγραμμα 4.15: Κώδικας της συνάρτησης Num2Bcd

Bcd2Num

Η συνάρτηση Bcd2Num επιστρέφει τον BCD αριθμό που δέχεται ως όρισμα σε ακέραιο αριθμό.

```
uint8_t Bcd2Num(uint8_t value)          //Function turn bcd to num
{
uint8_t  num=(value & 15) + 10 * ((value & (15<<4)) >>4);
return num;
}
```

Πρόγραμμα 4.16: Κώδικας της συνάρτησης Bcd2Num

encodeSM

Η συνάρτηση encodeSM επιστρέφει τον ακέραιο αριθμό που δέχεται ως όρισμα ο οποίος αντιπροσωπεύει τα λεπτά ή τα δευτερόλεπτα, σε BCD μορφή ώστε να την αναγνωρίσει το RTC.

```
uint8_t encodeSM(uint8_t value)          //Function to encode Sec,Min for the RTC
{
if ((value>=0) & (value<60))
        {
        uint8_t bcdnum=Num2Bcd(value);
        return bcdnum;
        }
}
```

Πρόγραμμα 4.17: Κώδικας της συνάρτησης encodeSM

encodeDate

Η συνάρτηση encodeDate επιστρέφει τον ακέραιο τον αριθμό που δέχεται ως όρισμα και αντιπροσωπεύει την ημερομηνία, σε BCD μορφή ώστε να την αναγνωρίσει ο καταχωρητής ημερομηνίας του RTC.

```
uint8_t encodeDate(uint8_t value)          //Function to encode Date for the RTC
{
if ((value>=1) & (value<32))
        {
        uint8_t bcdnum=Num2Bcd(value);
        return bcdnum;
        }
}
```

Πρόγραμμα 4.18: Κώδικας της συνάρτησης encodeDate

encodeHour

Η συνάρτηση encode Hour επιστρέφει τον ακέραιο τον αριθμό που δέχεται ως όρισμα και αντιπροσωπεύει την ώρα, σε BCD μορφή ώστε να την αναγνωρίσει ο καταχωρητής ώρας του RTC.

```
uint8_t encodeHour(uint8_t value)          //Function to encode Hour for the RTC
{
if ((value>=0) & (value<24))
        {
        uint8_t bcdnum=Num2Bcd(value);
```

Πρόγραμμα 4.19: Κώδικας της συνάρτησης encodeHour

```
        return bcdnum;
        }
}
```

Πρόγραμμα 4.19: Κώδικας της συνάρτησης encodeHour (Συνέχεια)

encodeYear

Η συνάρτηση encodeYear επιστρέφει τον ακέραιο αριθμό που δέχεται ως όρισμα και αντιπροσωπεύει τα χρόνια, σε BCD μορφή ώστε να την αναγνωρίσει ο καταχωρητής των χρόνων του RTC.

```
uint8_t encodeYear(uint8_t value)         //Function to encode Year for the RTC
{
if ((value>=0) & (value<100))
        {
        uint8_t bcdnum=Num2Bcd(value);
        return bcdnum;
        }
}
```

Πρόγραμμα 4.20: Κώδικας της συνάρτησης encodeYear

encodeDow

Η συνάρτηση encodeDow επιστρέφει τον ακέραιο αριθμό που δέχεται ως όρισμα και αντιπροσωπεύει την μέρα της εβδομάδας, σε BCD μορφή ώστε να την αναγνωρίσει ο καταχωρητής των ημερών της εβδομάδας του RTC.

```
uint8_t encodeDow(uint8_t value)         //Function to encode Dow for the RTC
{
if ((value>0) & (value<8))
        {
        uint8_t bcdnum=Num2Bcd(value);
        return bcdnum;
        }
}
```

Πρόγραμμα 4.21: Κώδικας της συνάρτησης encodeDow

encodeMonth

Η συνάρτηση encodeMonth επιστρέφει τον ακέραιο αριθμό που δέχεται ως όρισμα και αντιπροσωπεύει τους μήνες, σε BCD μορφή ώστε να την αναγνωρίσει ο καταχωρητής μηνών του RTC.

```
uint8_t encodeMonth(uint8_t value)       //Function to encode Month for the RTC
{
if ((value >0) & (value<13))
        {
```

Πρόγραμμα 4.22: Κώδικας της συνάρτησης encodeMonth

```
        uint8_t decvalue=Num2Bcd(value);
        return decvalue;
        }
}
```

Πρόγραμμα 4.22: Κώδικας της συνάρτησης encodeMonth (Συνέχεια)

decodMonth

Η συνάρτηση decodMonth μετατρέπει τον BCD αριθμό που δέχεται ως όρισμα και αντιπροσωπεύει τους μήνες από το RTC, σε ακέραιο.

```
uint8_t decodMonth(uint8_t value)
uint8_t decvalue=(value & 15) +10 *((value &(1<<4))>>4);
return decvalue;
 }
```

Πρόγραμμα 4.23: Κώδικας της συνάρτησης decodMonth

decodSMD

Η συνάρτηση decodSMD μετατρέπει τον BCD αριθμό που δέχεται ως όρισμα και αντιπροσωπεύει τα λεπτά ή τα δευτερόλεπτα από το RTC, σε ακέραιο.

```
uint8_t decodSMD(uint8_t value)         //  read the correct values for sec,min,dow
{
uint8_t decvalue=value & 127;
decvalue=Bcd2Num(decvalue);
return decvalue;
}
```

Πρόγραμμα 4.24: Κώδικας της συνάρτησης decodSMD

decodYear

Η συνάρτηση decodYear μετατρέπει τον BCD αριθμό που δέχεται ως όρισμα και αντιπροσωπεύει τα χρόνια από το RTC, σε ακέραιο. Επίσης, η συνάρτηση αυτή ελέγχει το bit του αιώνα σε περίπτωση που υπάρχει αλλαγή. Τα χρόνια καταγράφονται από το 2000 και μετά.

```
uint16_t decodYear(uint8_t value)                //  read the correct value for year from RTC
{
uint16_t decvalue;
uint8_t b=ReadReg(0x68,0x05);
int c=b>>7;                                      //check for century bit
if (c==1)
        {
        cent+=100;
        WriteReg(0x68,0x06,0);
        decvalue=Bcd2Num(value) + cent;          //if bit for cent 1 add 100
        WriteReg(0x68,0x05,(b & 127));
```

Πρόγραμμα 4.25: Κώδικας της συνάρτησης decodYear

```
        }

if (c==0)
        {
        decvalue=Bcd2Num(value) + cent;
        }
        return decvalue;}
```

Πρόγραμμα 4.25: Κώδικας της συνάρτησης decodYear (Συνέχεια)

decodHour

Η συνάρτηση decodHour μετατρέπει τον BCD αριθμό που δέχεται ως όρισμα και αντιπροσωπεύει την ώρα από το RTC, σε ακέραιο.

```
uint8_t decodHour(uint8_t value)          // read the correct value for Hour from RTC
{
uint8_t decvalue=(value & 127);
decvalue=(decvalue & 15) + 10 * ((value & (3<<4)) >>4);
return decvalue;
}
```

Πρόγραμμα 4.26: Κώδικας της συνάρτησης decodHour

RtcSetTime

Η συνάρτηση RtcSetTime θέτει την ώρα στο RTC εγγράφοντας τους αντίστοιχους καταχωρητές. Τα τρία ορίσματα που δέχεται είναι η ώρα, τα λεπτά και τα δευτερόλεπτα.

```
void RtcSetTime(uint8_t h,uint8_t m,uint8_t s)          // Set time at DS3231
{
  uint8_t dd[3];
  dd[2]=encodeHour(h);
  dd[1]=encodeSM(m);
  dd[0]=encodeSM(s);
  BurstWriteReg(0x68,0x00,dd,3);                        //Write hour, minutes and seconds to RTC
}
```

Πρόγραμμα 4.27: Κώδικας της συνάρτησης RtcSetTime

RtcSetDate

Η συνάρτηση RtcSetDate θέτει την ημερομηνία στο RTC εγγράφοντας τους αντίστοιχους καταχωρητές. Τα τρία ορίσματα που δέχεται είναι η ημερομηνία, ο μήνας και η χρονιά.

```
void RtcSetDate(uint8_t d,uint8_t m, uint16_t y)          // Set Date at DS3231
{
 uint8_t dd[3];
  dd[2]=encodeYear(y);
  dd[1]=encodeMonth(m);
  dd[0]=encodeDate(d);
  BurstWriteReg(0x68,0x04,dd,3);                          //Write date, month and year to RTC
}
```

Πρόγραμμα 4.28: Κώδικας της συνάρτησης RtcSetDate

RtcSetDow

Η συνάρτηση RtcSetDow θέτει την ημέρα της εβδομάδας στο RTC εγγράφοντας τον αντίστοιχο καταχωρητή. Το όρισμα που δέχεται είναι ο αριθμός που αντιπροσωπεύει την αντίστοιχη μέρα (π.χ 1=Δευτέρα κτλ).

```
void RtcSetDow(uint8_t d)                // Set Dow at DS3231
{
 WriteReg(0x68,0x03,encodeDow(d));
}
```

Πρόγραμμα 4.29: Κώδικας της συνάρτησης RtcSetDow

RtcGetHour

Η συνάρτηση RtcGetHour αναλαμβάνει την ανάγνωση της ώρας από το RTC.

```
uint8_t RtcGetHour()                     // Read Hour from DS3231
{
 BurstReadReg(0x68,0x00,aa,7);
 uint8_t h=decodHour(aa[2]);
 return h;
}
```

Πρόγραμμα 4.30: Κώδικας της συνάρτησης RtcGetHour

RtcGetMin

Η συνάρτηση RtcGetMin αναλαμβάνει την ανάγνωση των λεπτών από το RTC.

```
uint8_t RtcGetMin()                      //Read Min from DS3231
{
 BurstReadReg(0x68,0x00,aa,7);
 uint8_t m=decodSMD(aa[1]);
```

Πρόγραμμα 4.31: Κώδικας της συνάρτησης RtcGetMin

```
 return m;
}
```

Πρόγραμμα 4.31: Κώδικας της συνάρτησης RtcGetMin (Συνέχεια)

RtcGetSec

Η συνάρτηση RtcGetSec αναλαμβάνει την ανάγνωση των δευτερολέπτων από το RTC.

```
uint8_t RtcGetSec()                      // Read Sec from DS3231
{
 BurstReadReg(0x68,0x00,aa,7);
 uint8_t s=decodSMD(aa[0]);
 return s;
}
```

Πρόγραμμα 4.32: Κώδικας της συνάρτησης RtcGetSec

RtcGetDow

Η συνάρτηση RtcGetDow αναλαμβάνει την ανάγνωση της ημέρας της εβδομάδας από το RTC.

```
uint8_t RtcGetDow()                    // Read Dow from DS3231
{
  BurstReadReg(0x68,0x00,aa,7);
  uint8_t d=decodSMD(aa[3]);
  return d;
}
```

Πρόγραμμα 4.33: Κώδικας της συνάρτησης RtcGetDow

RtcGetDate

Η συνάρτηση RtcGetDate αναλαμβάνει την ανάγνωση της ημερομηνίας από το RTC.

```
uint8_t RtcGetDate()                   // Read Date from DS3231
{
  BurstReadReg(0x68,0x00,aa,7);
  uint8_t dt=decodSMD(aa[4]);
  return dt;
}
```

Πρόγραμμα 4.34: Κώδικας της συνάρτησης RtcGetDate

RtcGetMonth

Η συνάρτηση RtcGetMonth αναλαμβάνει την ανάγνωση του μήνα από το RTC.

```
uint8_t RtcGetMonth()                  // Read Month from DS3231
{
  BurstReadReg(0x68,0x00,aa,7);
```

Πρόγραμμα 4.35: Κώδικας της συνάρτησης RtcGetMonth

```
  uint8_t mnth=decodMonth(aa[5]);
  return mnth;
}
```

Πρόγραμμα 4.35: Κώδικας της συνάρτησης RtcGetMonth (Συνέχεια)

RtcGetYear

Η συνάρτηση RtcGetYear αναλαμβάνει την ανάγνωση της χρονιάς από το RTC.

```
uint16_t RtcGetYear()                  // Read Year from DS3231
{
 BurstReadReg(0x68,0x00,aa,7);
 uint16_t y=decodYear(aa[6]);
 return y;
}
```

Πρόγραμμα 4.36: Κώδικας της συνάρτησης RtcGetYear

RtcGetTemp

Η συνάρτηση RtcGetTemp αναλαμβάνει την ανάγνωση της θερμοκρασίας από το RTC.

```
float RtcGetTemp()                    // Read Temp from DS3231
{
```

Πρόγραμμα 4.37: Κώδικας της συνάρτησης RtcGetTemp

```
 uint8_t msb=ReadReg(0x68,0x11);      //Read MSB
 uint8_t lsb=ReadReg(0x68,0x12);      //Read LSB
 float temp=msb + ((lsb >>6) * 0.25);
 return temp;
}
```

Πρόγραμμα 4.37: Κώδικας της συνάρτησης RtcGetTemp(Συνέχεια)

InitAlarms

Η συνάρτηση InitAlarms αναλαμβάνει την ενεργοποίηση των Alarms του RTC.

```
void InitAlarms()                     //Init all the Alarms
{
 uint8_t g=ReadReg(0x68,0x0F);
 WriteReg(0x68,0x0F,(g & B11111100));
 delay(100);
 uint8_t m=ReadReg(0x68,0x0E);
 WriteReg(0x68,0x0E,(m | B00000111));        //Set Alarm bits Enable
}
```

Πρόγραμμα 4.38: Κώδικας της συνάρτησης InitAlarms

DisarmAlarms

Η συνάρτηση DisarmAlarms αναλαμβάνει την απενεργοποίηση των Alarms του RTC.

```
void DisarmAlarms()                   //Disable all Alarms
{
 uint8_t v=ReadReg(0x68,0x0E);
 uint8_t b=(v | B00000100);
 WriteReg(0x68,0x0E,(b & B11111100));        //Set zero the Alarm enable bits
 WriteReg(0x68,0x07,0);                      //Set all Alarm registers to zero
 WriteReg(0x68,0x08,0);
 WriteReg(0x68,0x09,0);
 WriteReg(0x68,0x0A,0);
 WriteReg(0x68,0x0B,0);
 WriteReg(0x68,0x0C,0);
 WriteReg(0x68,0x0D,0);
}
```

Πρόγραμμα 4.39: Κώδικας της συνάρτησης DisarmAlarms

Alarm1persec

Η συνάρτηση Alarm1persec αναλαμβάνει την παραμετροποίηση του Alarm1 ώστε να λειτουργεί κάθε δευτερόλεπτο γράφοντας τις κατάλληλες τιμές στους αντίστοιχους καταχωρητές του RTC.

```
void Alarm1persec()     //Setting the register values needed for Alarm1 working per sec
{
 WriteReg(0x68,0x07,0x80);
 WriteReg(0x68,0x08,0x80);
 WriteReg(0x68,0x09,0x80);
 WriteReg(0x68,0x0A,0x80);
}
```

Πρόγραμμα 4.40: Κώδικας της συνάρτησης Alarm1persec

Alarm2permin

Η συνάρτηση Alarm2permin αναλαμβάνει την παραμετροποίηση του Alarm2 ώστε να λειτουργεί κάθε λεπτό γράφοντας τις κατάλληλες τιμές στους αντίστοιχους καταχωρητές του RTC.

```
void Alarm2permin()     //Setting the register values needed for Alarm 2 working per min
{
 WriteReg(0x68,0x0B,0x80);
 WriteReg(0x68,0x0C,0x80);
 WriteReg(0x68,0x0D,0x80);
}
```

Πρόγραμμα 4.41: Κώδικας της συνάρτησης Alarm2permin

Set_Alarm

Η συνάρτηση Set_Alarm δέχεται τρία ορίσματα. Η μεταβλητή alarm παίρνει τιμές 1 και 2 και αντιπροσωπεύει το αντίστοιχο Alarm του RTC. Η μεταβλητή type παίρνει τιμές από 1 έως 5 και αντιπροσωπεύει την λειτουργία του κάθε Alarm (πχ όταν το alarm είναι 1 και το type είναι 1 τότε το alarm 1 ενεργοποιείται όταν ταιριάζουν τα δευτερόλεπτα κτλ). Οι τιμές του type αντιπροσωπεύουν τις λειτουργίες του κάθε alarm σύμφωνα με τον αριθμό της γραμμής στον Πίνακα 4.2. Ο πίνακας data περιέχει τις τιμές που θα γραφθούν στους καταχωρητές των Alarms και αντιπροσωπεύουν τα λεπτά, δευτερόλεπτα, την ώρα κλπ που θα συγκριθούν με τις τιμές των αντίστοιχων καταχωρητών του RTC.

```
uint8_t Set_Alarm(int alarm, int type , uint8_t* data)      //This function sets the alarm 1 or 2
{
 uint8_t a=127;
 uint8_t b=128;
 if (alarm==1)
 {
  if (type==1)                                              //When second match
  {
```

Πρόγραμμα 4.42: Κώδικας της συνάρτησης Set_Alarm

```
  WriteReg(0x68,0x07,encodeSM(a & data[0]));

  WriteReg(0x68,0x08,0x80);
  WriteReg(0x68,0x09,0x80);
  WriteReg(0x68,0x0A,0x80);
 }
 if (type==2)                                    //When min and sec match
 {
  WriteReg(0x68,0x07,encodeSM(a & data[0]));
  WriteReg(0x68,0x08,encodeSM(a & data[1]));
  WriteReg(0x68,0x09,0x80);
  WriteReg(0x68,0x0A,0x80);
 }
 if (type==3)                                    //When hours , min and sec match
 {
   WriteReg(0x68,0x07,encodeSM(a & data[0]));
   WriteReg(0x68,0x08,encodeSM(a & data[1]));
   WriteReg(0x68,0x09,encodeHour(a & data[2]));
   WriteReg(0x68,0x0A,0x80);
 }
 if (type==4)                                    //When date hours , min and seconds match
 {
   WriteReg(0x68,0x07,encodeSM(a & data[0]));
   WriteReg(0x68,0x08,encodeSM(a & data[1]));
   WriteReg(0x68,0x09,encodeHour(a & data[2]));
   WriteReg(0x68,0x0A,encodeDate(B00111111 & data[4]));
  }
  if (type==5)                                   //When day hours  , min and seconds match
  {
   WriteReg(0x68,0x07,encodeSM(a & data[0]));
   WriteReg(0x68,0x08,encodeSM(a & data[1]));
   WriteReg(0x68,0x09,encodeHour(a & data[2]));
   WriteReg(0x68,0x0A,encodeDow(a & data[3]));
   WriteReg(0x68,0x0A,encodeDow(B01000000 | data[3]));
   }
}
if (alarm==2)
{
 if (type==1)                                    // When minutes match
 {
  WriteReg(0x68,0x0B,encodeSM(a & data[1]));
  WriteReg(0x68,0x0C,0x80);
  WriteReg(0x68,0x0D,0x80);
 }
 if (type==2)                                    //When hours and minutes match
 {
```

Πρόγραμμα 4.42: Κώδικας της συνάρτησης Set_Alarm (Συνέχεια)

```
    WriteReg(0x68,0x0B,encodeSM(a & data[1]));

    WriteReg(0x68,0x0C,encodeHour(a & data[2]));
    WriteReg(0x68,0x0D,0x80);
   }
   if (type==3)                                          // When date , hours and minutes match
   {
    WriteReg(0x68,0x0B,encodeSM(a & data[1]));
    WriteReg(0x68,0x0C,encodeHour(a & data[2]));
    WriteReg(0x68,0x0D,encodeDate(B00111111 & data[4]));
    }
    if (type==4)                                         //When day, hours and minutes match
    {
    WriteReg(0x68,0x0B,encodeSM(a & data[1]));
    WriteReg(0x68,0x0C,encodeHour(a & data[2]));
    WriteReg(0x68,0x0D,encodeDow(a & data[3]));
    WriteReg(0x68,0x0D,encodeDow(B01000000 & data[3]));
   }
  }
}
```

Πρόγραμμα 4.42: Κώδικας της συνάρτησης Set_Alarm (Συνέχεια)

Κεφάλαιο 5

Η ανάπτυξη του συστήματος

5.1 Εισαγωγή

Το σύστημα που αναπτύχθηκε στην παρούσα πτυχιακή εργασία χωρίζεται σε τρία στάδια που είναι η συλλογή, η αποθήκευση και η επεξεργασία δεδομένων. Τα στάδια της συλλογής και αποθήκευσης πραγματοποιήθηκαν στο περιβάλλον του Arduino IDE με την χρήση της πλατφόρμας Arduino Mega2560, ενώ το στάδιο της επεξεργασίας στο GUI που αναπτύχτηκε με το LabVIEW 2014.

Στο στάδιο της συλλογής, το σύστημα λαμβάνει μετρήσεις από τρεις αισθητήρες που είναι θερμοκρασίας χώρου, θερμοκρασίας RTC και φωτεινότητας χώρου. Η επικοινωνία μεταξύ των συσκευών και του μικροελεγκτή έγινε μέσω του πρωτοκόλλου I2C. Η δειγματοληψία γίνεται με βάση το Alarm1 του RTC που χρησιμοποιήθηκε. Κατά την αποθήκευση των λαμβανόμενων μετρήσεων εκτός από τα δεδομένα που προέρχονται από τις παραπάνω συσκευές καταγράφεται επίσης και η ώρα και η ημερομηνία λήψης. Επίσης το σύστημα συγχρονίζεται με έναν NTP Server χρησιμοποιώντας το δεύτερο Alarm του RTC ώστε να είναι εφικτή η διόρθωση της ώρας του RTC. Επιπλέον υπάρχουν ενδεικτικές Led λυχνίες που δείχνουν πότε γίνεται η δειγματοληψία μετρήσεων, ο συγχρονισμός της ώρας και η αποθήκευση των δεδομένων.

Στο στάδιο της αποθήκευσης, το σύστημα αποθηκεύει τα δεδομένα κάθε ένα λεπτό στην SD κάρτα δημιουργώντας αυτόματα αρχεία .txt που περιέχουν τις μετρήσεις μίας ημέρας. Το όνομα των παραπάνω αρχείων βασίζεται στην ημερομηνία λήψης των μετρήσεων. Επιπρόσθετα, το συγκεκριμένο σύστημα αποθηκεύει τις τρέχουσες μετρήσεις σε μια MySQL βάση δεδομένων υπό την προϋπόθεση πως υπάρχει σύνδεση στο διαδίκτυο. Σε περίπτωση που δεν υπάρχει σύνδεση στο διαδίκτυο ή στην συγκεκριμένη βάση δεδομένων, το σύστημα έχει την δυνατότητα να το αναγνωρίσει και να αποθηκεύσει τα δεδομένα σε προσωρινά αρχεία στην SD κάρτα. Όταν η σύνδεση με την βάση δεδομένων αποκατασταθεί αρχικά αποστέλλονται στην βάση δεδομένων τα δεδομένα των προσωρινών αρχείων. Μόλις ολοκληρωθεί η αποστολή των δεδομένων ενός προσωρινού αρχείου, το αρχείο αυτό διαγράφεται αυτόματα από την SD κάρτα.

Το σύστημα αυτό επιπλέον διαθέτει μία LCD οθόνη που αναγράφει τα στάδια που βρίσκεται το σύστημα, τους συγχρονισμούς με τον NTP Server αλλά και τυχόν σφάλματα όπως η απώλεια σύνδεσης στο δίκτυο κα. Με σκοπό την ανάπτυξη πολλαπλών σταθμών εργασίας, η απαιτουμένη παραμετροποίηση κάθε σταθμού πραγματοποιείται μέσω της EEPROM που διαθέτει το RTC. Για τον λόγο αυτό έχει δημιουργηθεί ένα επιπλέον πρόγραμμα στο περιβάλλον του Arduino IDE το οποίο αποθηκεύει στην EEPROM βασικές πληροφορίες για το σύστημα όπως, το όνομά του, το αν λειτουργεί με static IP (Internet Protocol) διεύθυνση ή με δυναμική IP διεύθυνση μέσω του DHCP (Dynamic Host Configuration Protocol), την IP διεύθυνση του NTP Server, τον κωδικό πρόσβασης καθώς και όνομα χρήστη για την βάση δεδομένων, την IP διεύθυνση του MySQL server, την mac

διεύθυνση της συσκευής κ.α. Αυτές οι πληροφορίες ανακτώνται αυτόματα από το σύστημα κατά την έναρξή του.

Στο στάδιο της επεξεργασίας, μέσω του GUI που έχει πραγματοποιηθεί στο LabVIEW ο χρήστης έχει την δυνατότητα να αναλύσει τα δεδομένα που επιθυμεί σε κυματομορφές επιλέγοντας την ημερομηνία και τον σταθμό από τον οποίο προέρχονται. Στην συνέχεια μπορεί να επιλέξει τις κυματομορφές που επιθυμεί να εμφανιστούν (π.χ μόνο θερμοκρασίας ή όλα κ.α).

5.2 Το κυρίως πρόγραμμα του ATmega2560

Οι συναρτήσεις που αναλύθηκαν στο προηγούμενο κεφάλαιο, δημιουργήθηκαν για την επικοινωνία των συσκευών με τον ATmega2560. Παρόλα αυτά έχουν χρησιμοποιηθεί και άλλες συναρτήσεις που προέρχονται από έτοιμες βιβλιοθήκες για τον πλήρη έλεγχο των συσκευών και του συστήματος. Οι βιβλιοθήκες που αξιοποιήθηκαν είναι η SPI, Ethernet, EthernetUdp, Time, Wire, LCD, LiquidCrystal_I2C, sha1, mysql, SD, TCN75A, BH1750. Η χρήση τους έχει να κάνει με την λειτουργία των πρωτοκόλλων επικοινωνίας SPI και Ethernet, την σύνδεση με την βάση δεδομένων MySQL, την λειτουργία της οθόνης LCD, την λειτουργία της SD κάρτας καθώς και την λειτουργία των αισθητήρων θερμοκρασίας και φωτεινότητας.

Κατά την ανάπτυξη και υλοποίηση του προγράμματος που θα εκτελεί ο ATmega2560 αρχικά πραγματοποιούνται οι δηλώσεις των γενικών μεταβλητών καθώς και η προσθήκη των απαιτούμενων αρχείων κεφαλίδας για την ενσωμάτωση των απαραίτητων βιβλιοθηκών (Πρόγραμμα 5.1).

```
#include "Arduino.h"
#include "dnecode.h"
#include "Rtc.h"
#include <SPI.h>
#include <Ethernet.h>
#include <EthernetUdp.h>
#include <Time.h>
#include <Wire.h>
#include <LCD.h>
#include <LiquidCrystal_I2C.h>
#include <sha1.h>
#include <mysql.h>
#include <SD.h>
#include <TCN75A.h>
#include <BH1750.h>
#define I2C_ADDR    0x3F          // <<----- Add your address here.  Find it from I2C Scanner
#define BACKLIGHT_PIN     3             //Definition of lcd pins
#define En_pin  2
#define Rw_pin  1
#define Rs_pin  0
```

Πρόγραμμα 5.1: Οι δηλώσεις των μεταβλητών και η προσθήκη των αρχείων κεφαλίδας

```
#define D4_pin  4
#define D5_pin  5
#define D6_pin  6
#define D7_pin  7
#define row 6
#define col 9
#define resolution 0.0625                    //Resolution for Temp Sensor
TCN75A sensor;
BH1750 lightMeter;
LiquidCrystal_I2C lcd(I2C_ADDR,En_pin,Rw_pin,Rs_pin,D4_pin,D5_pin,D6_pin,D7_pin);
byte server_addr[4];                         //Db IP address buffer
byte timeServer[4];                          //NTP Server address buffer
byte mac[6];
byte backupNtpServer[4];
byte DnServer[4];
byte Gateway[4];
byte Subnet[4];
byte StaticIP[4];
Connector my_conn;                           //mySQl connector define
const long timeZoneOffset = 10800L;          //gmt+3 Athens Summer light
unsigned int localPort = 8888;               //My local port for connection
const int NTP_PACKET_SIZE= 48;
byte packetBuffer[NTP_PACKET_SIZE];
EthernetUDP Udp;
unsigned long ntpLastUpdate = 0;
time_t prevDisplay = 0;
char user[20];
char password[20];
char SystemName[20];
String db="test.station2 ";
String colums="(hours, minutes, seconds, date, month, year, RTCtemp,EnvTemp,Lux)";
String envt[6];
String rtct[6];
volatile int AlarmONE=0;
volatile int AlarmTWO=0;
volatile int line=0;
uint16_t tempBuff[row][col];
String dataBuff[row][col];
volatile int SdWriteFlag=0;
volatile int DbWriteFlag=0;
String date="";
String mnth="";
String yr="";
String fileName="";
String SdFormat="";
String TempName="";
```

Πρόγραμμα 5.1: Οι δηλώσεις των μεταβλητών και η προσθήκη των αρχείων κεφαλίδας (Συνέχεια)

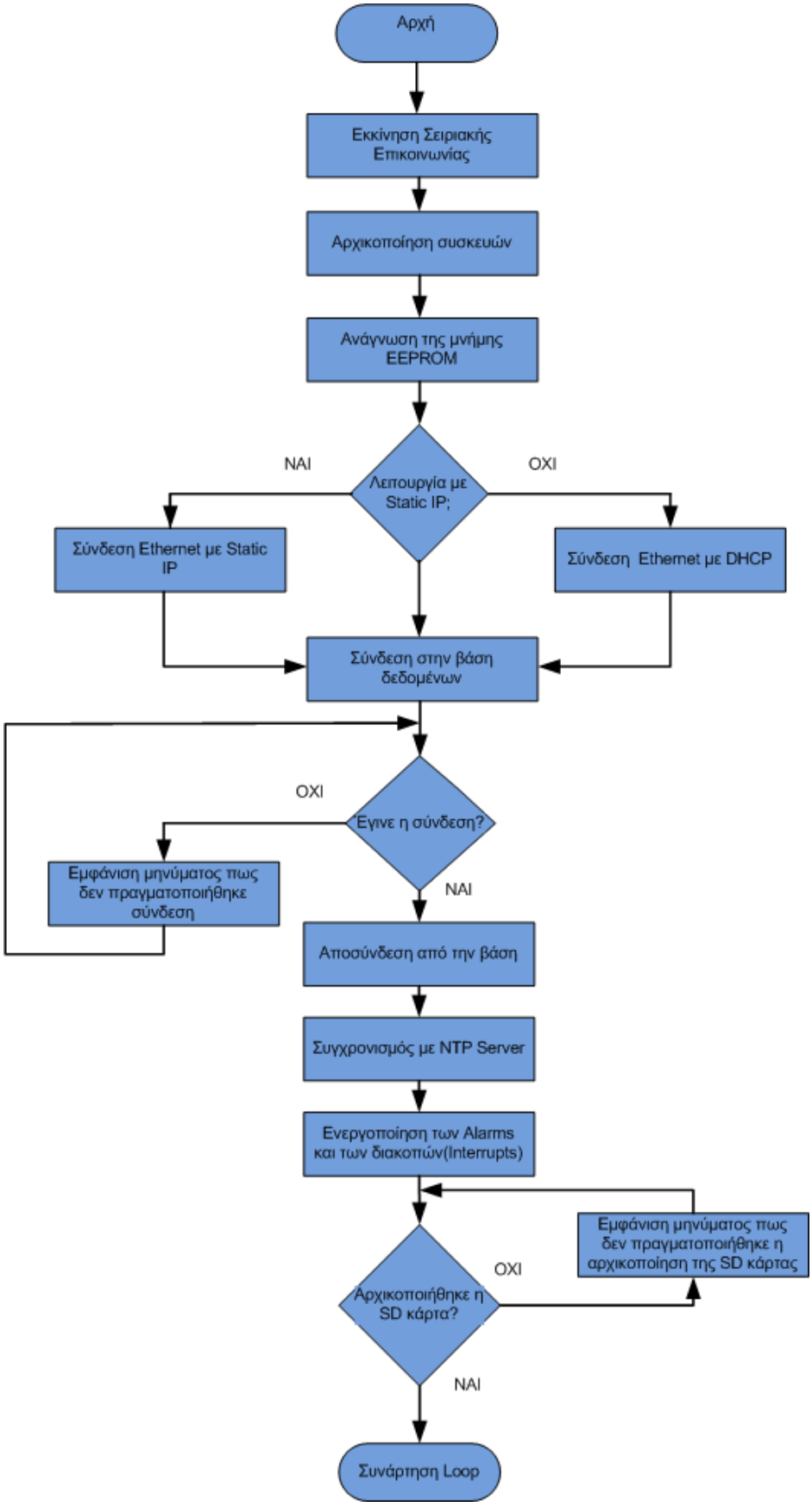
```
String dt="";
String mn="";
String db_command="";
int TempCounter=0;
int ConLostFlag=0;
int TempExistFlag=0;
String TempArray[100]="";
unsigned long byt=0;
int pos=0;
int valcounter=0;
int f_bytes=0;
unsigned long  p2=0;
int temptemp=0;
String buf[10]="";
File myFile;                                    //define sd file for SD
```

Πρόγραμμα 5.1: Οι δηλώσεις των μεταβλητών και η προσθήκη των αρχείων κεφαλίδας (Συνέχεια)

5.2.1 Η συνάρτηση setup

Στο Σχήμα 5.2 παρουσιάζεται το διάγραμμα ροής της συνάρτησης setup. Αρχικά η συνάρτηση setup ενεργοποιεί την σειριακή επικοινωνία με ρυθμό μετάδοσης 115200 bps. Στην συνέχεια, γίνεται η ενεργοποίηση και η παραμετροποίηση του αισθητήρα φωτεινότητας καθορίζοντας την ακρίβεια των μετρήσεων. Αφού πραγματοποιηθεί η αρχικοποίηση της LCD οθόνης, γίνεται η αρχικοποίηση του TWI και ξεκινά η ανάγνωση των δεδομένων της EEPROM ώστε να ληφθούν οι απαραίτητες πληροφορίες του συστήματος. Οι πληροφορίες αυτές περιλαμβάνουν το όνομά του συστήματος, την IP διεύθυνση του NTP Server, την IP διεύθυνση του MySQL server, την διεύθυνση του DNS (Domain Name System) server, το Subnet Mask, την Static IP διεύθυνση, το username για την βάση δεδομένων, το password για την βάση δεδομένων καθώς και την πληροφορία για το αν το σύστημα θα λειτουργεί με Static IP διεύθυνση ή με δυναμική IP διεύθυνση μέσω του DHCP.

Στην συνέχεια η συνάρτηση setup αναλαμβάνει την αρχικοποίηση του Ethernet Shield ανάλογα με την επιλογή λειτουργίας του με Static IP ή με DHCP ανάλογα την πληροφορία της EEPROM και γίνεται μία προσπάθεια σύνδεσης στην βάση δεδομένων, ώστε να ελεγχθεί η σύνδεση στο διαδίκτυο και η κατάσταση του MySQL server. Παράλληλα εμφανίζονται και τα αντίστοιχα μηνύματα για την επιτυχία ή την αποτυχία ολοκλήρωσης αυτών των διαδικασιών. Σε περίπτωση που δεν υπάρξει σύνδεση με τον MySQL server εμφανίζεται το αντίστοιχο μήνυμα και η διαδικασία σταματά εκεί. Εάν πραγματοποιηθεί η σύνδεση με τον MySQL server, τότε το σύστημα αποσυνδέεται από τον MySQL server και πραγματοποιείται σύνδεση με τον NTP Server για τον συγχρονισμό της ώρας και ημερομηνίας. Ακολουθεί η ενεργοποίηση των διακοπών του ATmega2560 και των Alarms του RTC ώστε το Alarm1 να λειτουργεί κάθε δευτερόλεπτο και το Alarm2 κάθε λεπτό. Τέλος ελέγχεται αν πραγματοποιήθηκε η αρχικοποίηση της SD κάρτας. Αν η αρχικοποίηση της SD κάρτας δεν ολοκληρώθηκε με επιτυχία τότε η διαδικασία σταματά εκεί. Παρακάτω δίνεται ο κώδικας της συνάρτησης setup (Πρόγραμμα 5.2) όπως αναπτύχθηκε στο περιβάλλον Arduino IDE



Σχήμα 5.1: Διάγραμμα ροής της συνάρτησης setup

```
void setup()
{
  Serial.begin(115200);                          //Serial communication begin with 115200baud rate
  lightMeter.begin();                            //Light sensor Initialize
  int w_reg;
  sensor.begin();                                //Temperature sensor Initialize
  sensor.set_address(0);
  sensor.set_resolution(3);                      //Resolution for Temperature sensor
  pinMode(8,OUTPUT);
  pinMode(9,OUTPUT);
  pinMode(5,OUTPUT);
  lcd.begin (16,2);                              //ATTENTION LCD MUST INIT FIRST
lcd.setBacklightPin(BACKLIGHT_PIN,POSITIVE);
  lcd.setBacklight(HIGH);                        //Enable Lcd BackLight
  TWI_Init(true);                                //Enable TWI with eternal pullups
                                                 //reading info from eeprom
  int Control = ReadReg_16B(0x57,0);
  for (int mm=0; mm<6; mm++)
  {
    mac[mm]=ReadReg_16B(0x57,mm+1);
  }
for (int ss=0; ss<4; ss++)
{
  timeServer[ss]=ReadReg_16B(0x57,ss+7);  //NTP SERVER
  backupNtpServer[ss]=ReadReg_16B(0x57,ss+11); //BACKUP NTP
  server_addr[ss]=ReadReg_16B(0x57,ss+15);  //DB SERVER
  DnServer[ss]=ReadReg_16B(0x57,ss+19);
  Gateway[ss]=ReadReg_16B(0x57,ss+23);
  Subnet[ss]=ReadReg_16B(0x57,ss+27);
  StaticIP[ss]=ReadReg_16B(0x57,ss+31);
 }
int SystemNl=ReadReg_16B(0x57,95);
int UserNl=ReadReg_16B(0x57,96);
int Passl=ReadReg_16B(0x57,97);
for (int pp=0; pp<SystemNl; pp++)
{
  char abc = ReadReg_16B(0x57,pp+35);
  SystemName[pp] = abc;
}
for (int rr=0; rr<UserNl; rr++)
{
  char abd = ReadReg_16B(0x57,rr+55);
  user[rr] = abd;
}

for (int uu=0; uu<Passl; uu++)
```

Πρόγραμμα 5.2: Κώδικας της συνάρτησης setup

```
{

  char abe = ReadReg_16B(0x57,uu+55);
  password[uu] = abe;
 }
  DisarmAlarms();                                  //Disable all Alarms for disable all flags
  InitAlarms();                                    //Enable Alarms
  Alarm1persec();                                   //Set Alarm 1 working per sec
  Alarm2permin();                                  //Set Alarm 2 working per min
if (Control==1)                                                  //Work with static IP if its true
{
  Ethernet.begin(mac,StaticIP,DnServer,Gateway,Subnet);          //Ethernet connection
  lcd.clear();
  lcd.setCursor (0,0);
  lcd.print("Working on Static IP");
}
else                                                             //Work with DHCP
{
  int i = 0;
  int DHCP = 0;
  DHCP = Ethernet.begin(mac);
    while( DHCP == 0 && i < 30)
    {
      delay(1000);
      DHCP = Ethernet.begin(mac);
      i++;
    }
  if(!DHCP)                                                      //IF DCHP failed
  {
    lcd.clear();
    lcd.setCursor (0,0);
    lcd.print("DHCP FAILED");
    Serial.println("DHCP FAILED");
  }
  else
  {                                                              //IF DHCP succeed
    lcd.clear();
    lcd.setCursor (0,0);
    lcd.print("DHCP Success");
    Serial.println("DHCP Success");
  }
}
    pinMode(10,OUTPUT);                                          //CS for Ethernet
    if (!(my_conn.mysql_connect(server_addr, 3306, user, password)))          //Try to connect to DB
    {                                                            //If it failed print failure at lcd
      lcd.clear();
```

Πρόγραμμα 5.2: Κώδικας της συνάρτησης setup (Συνέχεια)

```
      lcd.setCursor (0,0);

      lcd.print("CONNECTION");
      lcd.setCursor(0,1);
      lcd.print("FAILED");
      Serial.println("NO CONNECTION TO INTRNT");
      for(;;);  //IF THERE IS NO CONNECTION TO INTERNET DON'T GO FURTHER
    }
   my_conn.disconnect();                                    //Disconnect from DB
getTimeAndDate();                              //Take time and date from NTP Server
   uint8_t aa[5];
   int16_t bb;
   aa[0]=second();
   aa[1]=minute();
   aa[2]=hour();
   aa[3]=day();
   aa[4]=month();
   bb=year()-2000;
   if (year()>=2100)
    {
      bb=year()-2100;
    }                                              //Write time and date from NTP Server to RTC
   WriteReg(0x68,0x00,encodeSM(aa[0]));
   WriteReg(0x68,0x01,encodeSM(aa[1]));
   WriteReg(0x68,0x02,encodeHour(aa[2]));
   WriteReg(0x68,0x04,encodeDate(aa[3]));
   WriteReg(0x68,0x05,encodeMonth(aa[4]));
   WriteReg(0x68,0x06,encodeYear(bb));
  delay(1);
//Print Hour and Date to LCD
  lcd.setCursor (0,0);
  lcd.print(RtcGetHour());
  lcd.print(":");
  lcd.print(RtcGetMin());
  lcd.print(":");
  lcd.print(RtcGetSec());
  lcd.print(" ");
  lcd.print("RTC Synch");
  lcd.setCursor (0,1);
  lcd.print(RtcGetDate());
  lcd.print("/");
  lcd.print(RtcGetMonth());
```

Πρόγραμμα 5.2: Κώδικας της συνάρτησης setup (Συνέχεια)

```
  lcd.print("/");
  lcd.print(RtcGetYear());
  delay(1000);
  Serial.print("Initializing SD card...");
  if (!SD.begin(4))                                    //Check if SD initialized
  {
    delay(100);
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("SD Init Failed");
    Serial.println("initialization failed!");
    return;
  }
  delay(100);
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("SD Init Done");
  Serial.println("initialization done.");
  delay(1000);
  attachInterrupt(digitalPinToInterrupt(19),AlarmInterrupt,FALLING);//Set the External
interrupt at DP19  Arduino Mega 2560
  uint8_t g=ReadReg(0x68,0x0F);     //Disable Interrupt flags to make sure nothing is ON
  WriteReg(0x68,0x0F,(g & B11111100));
  Serial.println("Setup Completed");
  delay(10);
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("Setup Cmpltd");
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("I AM ");
  lcd.setCursor(5,0);
  lcd.print(SystemName);                               //Print Station's Name
}
```

Πρόγραμμα 5.2: Κώδικας της συνάρτησης setup (Συνέχεια)

Στο τελευταίο κομμάτι του κώδικα της setup καλούνται κάποιες συναρτήσεις που είναι πολύ σημαντικές για την λειτουργία του συστήματος για τις οποίες θα γίνει η ανάλυση της λειτουργίας τους και η παρουσίαση του κώδικά τους παρακάτω.

sendNTPpacket

Η συνάρτηση sendNTPpakcet επικοινωνεί με τον NTP Server στέλνοντας ένα αίτημα για την ώρα-ημερομηνία στον NTP Server που δείχνει το όρισμα address. Απαραίτητη

προϋπόθεση για την αξιοποίηση αυτής της συνάρτησης είναι η ενσωμάτωση της βιβλιοθήκης Ethernet.

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

Πρόγραμμα 5.3: Κώδικας της συνάρτησης sendNTPpacket

getTimeAndDate

Η συνάρτηση getTimeAndDate λαμβάνει το πακέτο δεδομένων που περιέχει την ώρα και ημερομηνία από τον NTP Server και αξιοποιώντας την παράμετρο αντιστάθμισης για την ώρα της ζώνης (time offset) ενημερώνει τον ATmega2560 με την τρέχουσα ώρα και ημερομηνία. Απαραίτητη προϋπόθεση για την αξιοποίηση αυτής της συνάρτησης είναι η ενσωμάτωση της βιβλιοθήκης Ethernet.

```
int getTimeAndDate()
{
   int flag=0;
   Udp.begin(localPort);
   sendNTPpacket(timeServer);
   delay(1000);
   if (Udp.parsePacket())
   {
      Udp.read(packetBuffer,NTP_PACKET_SIZE);         // read the packet into the buffer
      unsigned long highWord, lowWord, epoch;
      highWord = word(packetBuffer[40], packetBuffer[41]);
      lowWord = word(packetBuffer[42], packetBuffer[43]);
      epoch = highWord << 16 | lowWord;
      epoch = epoch - 2208988800 + timeZoneOffset;       //Add offset for the given zone
      flag=1;
      setTime(epoch);
      ntpLastUpdate = now();
    }
      Udp.stop();
      return flag;
 }
```

Πρόγραμμα 5.4: Κώδικας της συνάρτησης getTimeAndDate

AlarmInterrupt

Η συνάρτηση AlarmInterrupt αναγνωρίζει ποιο Alarm του RTC είναι υπεύθυνο για το σήμα εξωτερικής διακοπής. Στην περίπτωση που είναι υπεύθυνο το Alarm1 το οποίο διεγείρεται κάθε δευτερόλεπτο, αυξάνει κατά 1 την μεταβλητή AlarmONE, ενεργοποιεί και απενεργοποιεί το ενδεικτικό LED και απενεργοποιεί το αντίστοιχο flag στον καταχωρητή κατάσταση του RTC. Όταν η μεταβλητή AlarmONE γίνει 10 (δηλαδή έχουν περάσει 10 δευτερόλεπτα), διαβάζει την ώρα, την ημερομηνία και την θερμοκρασία από το RTC, διαβάζει την θερμοκρασία από τον αισθητήρα θερμοκρασίας και την φωτεινότητα από τον αισθητήρα φωτεινότητας. Τέλος οι μετρήσεις αυτές αποθηκεύονται προσωρινά σε ένα πίνακα.

Όταν η παραπάνω διαδικασία επαναληφθεί 6 φορές (δηλαδή έχει περάσει 1 λεπτό) οι μετρήσεις αντιγράφονται στον πίνακα dataBuff και μέσω των μεταβλητών SDWriteFlag και DBWriteFlag ενημερώνεται η συνάρτηση loop ώστε να ενεργοποιηθούν οι διαδικασίες εγγραφής για την βάση δεδομένων και την SD κάρτα.

Στην περίπτωση που διεγερθεί το Alarm2 (που λειτουργεί ανά 1 λεπτό) αυξάνει κατά 1 η μεταβλητή AlarmTWO που χρησιμοποιείται στην συνάρτηση loop για την ενεργοποίηση της διαδικασίας συγχρονισμού. Επίσης απενεργοποιεί το αντίστοιχο flag στον καταχωρητή κατάστασης του RTC.

```
void AlarmInterrupt()
{
 uint8_t g=ReadReg(0x68,0x0F);              //Read Status Register of RTC
 uint8_t h=g<<7;                            // flag of Alarm1
 uint8_t c=g<<6;                            //flag of Alarm2
 if (h==128)                                //if Alarm 1 Flag on
 {
  WriteReg(0x68,0x0F,(g & B11111110));      //Write Alarm1 Flag off
  digitalWrite(9,HIGH);                     //LED ON
  AlarmONE+=1;
  digitalWrite(9,LOW);                      //LED OFF
  if (AlarmONE==10)                         //IF alarm 1 flag enabled 10 times
  {
   AlarmONE=0;
   digitalWrite(5,HIGH);                    //LED ON
//Read all the values from RTC and sensors
    if (RtcGetDate()<10)
    {
     tempBuff[line][0] = "0" + (String(RtcGetDate()));
    }
    else
    {
     tempBuff[line][0] = (RtcGetDate());
    }
    if (RtcGetMonth()<10)
```

Πρόγραμμα 5.5: Κώδικας της συνάρτησης AlarmInterrupt

```
{
  tempBuff[line][1] = "0" +  (String(RtcGetMonth()));
}
else
{
  tempBuff[line][1] = (RtcGetMonth());
}
if (RtcGetHour()<10)
{
  tempBuff[line][3] = "0"+ (String(RtcGetHour()));
}
else
{
  tempBuff[line][3] = (RtcGetHour());
}
if (RtcGetMin()<10)
{
  tempBuff[line][4] = "0" + (String(RtcGetMin()));
}
else
{
  tempBuff[line][4] = (RtcGetMin());
if (RtcGetSec()<10)
{
  tempBuff[line][5] ="0" + (String(RtcGetSec()));
else
  tempBuff[line][5] = (RtcGetSec());
tempBuff[line][2] = (RtcGetYear());
float rtc = RtcGetTemp();
rtct[line] = String(rtc);
tempBuff[line][6] = (RtcGetTemp());
uint8_t ff[2];
BurstReadReg(0x48,0x00,ff,2);
float Envtemp = ff[0] + (ff[1] >> 4) * resolution;
envt[line] = String(Envtemp,4);
tempBuff[line][7] = (Envtemp);
uint8_t lx[2];
BurstReadOpecode_8B(0x23,lx,2);
uint16_t lux = (lx[0]<<8) | lx[1];
tempBuff[line][8] = (lux);
digitalWrite(5,LOW);                    //LED OFF
line+=1;
if (line==6)
{
  for (int ii=0; ii<6; ii++)
  {
```

Πρόγραμμα 5.5: Κώδικας της συνάρτησης AlarmInterrupt (Συνέχεια)

```
        for (int jj=0; jj<9; jj++)

        {
          dataBuff[ii][jj]=String(tempBuff[ii][jj]);
        }
        dataBuff[ii][7] = envt[ii]; //dior8wnw tis times gia thn 8ermokrasia
        dataBuff[ii][6] = rtct[ii];
       }
       line=0;
       SdWriteFlag=1;                          //Enable SD write flag
       DbWriteFlag=1;                          //Enable Database write flag
     }
    }
  }
 if ((c==128) | (c==192))                      // if Alarm 2 enabled
 {
  AlarmTWO+=1;
  WriteReg(0x68,0x0F,(g & B11111101));         // Write Alarm2 Flag off
 }
}
```

Πρόγραμμα 5.5: Κώδικας της συνάρτησης AlarmInterrupt (Συνέχεια)

ReadLine

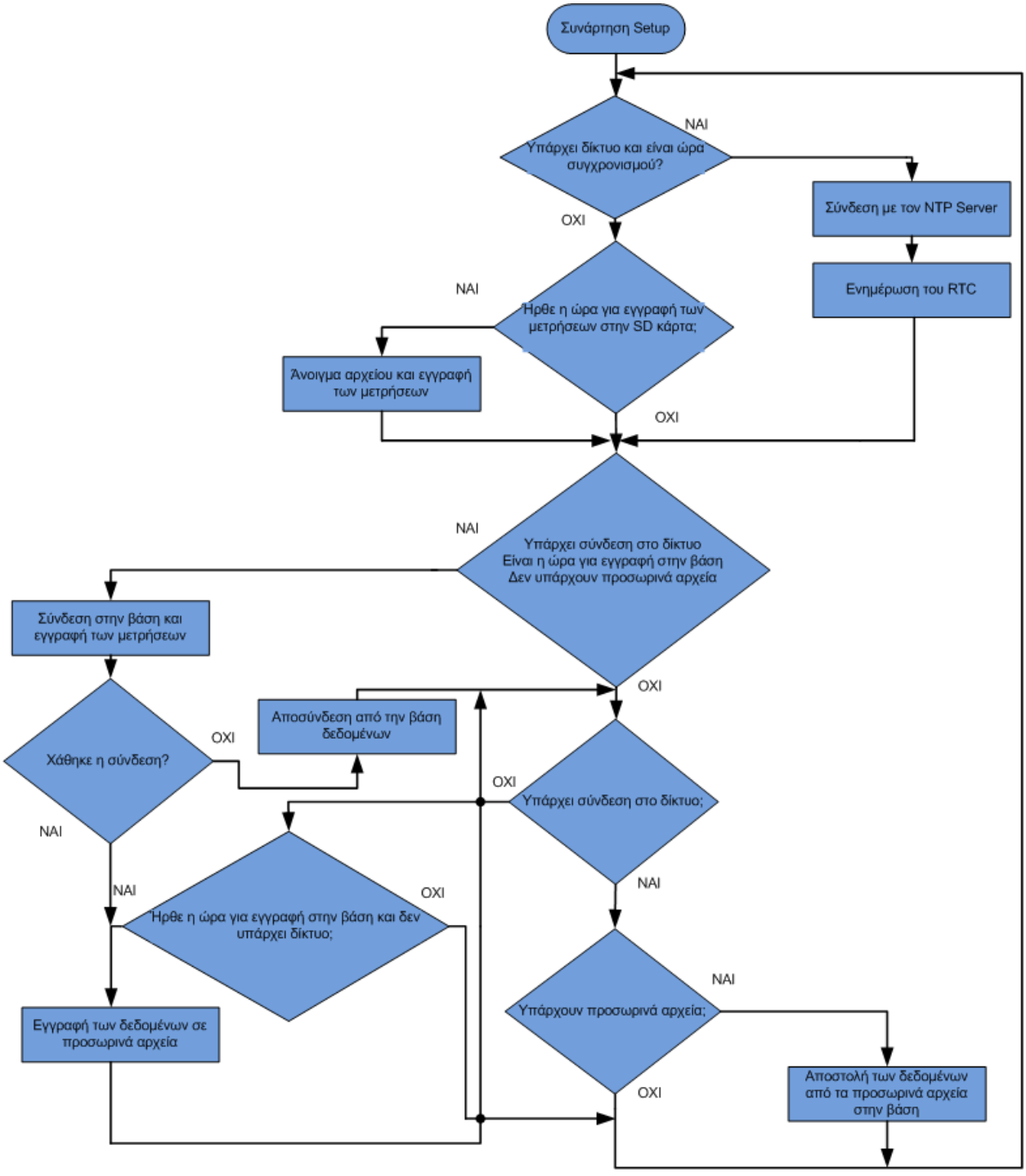
Η συνάρτηση ReadLine δημιουργήθηκε για να μετράει το μέγεθος ενός αρχείου στην SD κάρτα μέχρι να εντοπίσει τον τερματικό χαρακτήρα \n και επιστρέφει τον καταμετρημένο αριθμό των bytes.

```
String ReadLine()
{
 String rec = "";
 char ch;
 while (myFile.available())
 {
  ch = myFile.read();
  if (ch=='\n')
  {
   byt+=1;
   return String(rec);
  }
  else
  {
   rec+=ch;
   byt+=1;
  }
 }
 return "";
}
```

Πρόγραμμα 5.6: Κώδικας της συνάρτησης ReadLine

5.2.2 Η συνάρτηση loop

Στο Σχήμα 5.3 παρουσιάζεται το διάγραμμα ροής της συνάρτησης loop. Αρχικά η συνάρτηση loop ελέγχει αν έχει περάσει ο χρόνος που έχει οριστεί ώστε να πραγματοποιηθεί η σύνδεση με τον NTP Server για τον συγχρονισμό της ώρας και της ημερομηνίας. Για να γίνει ο συγχρονισμός ώρας, θα πρέπει να υπάρχει σύνδεση στο διαδίκτυο, το σύστημα να μην είναι συνδεδεμένο στην βάση δεδομένων και να μην έχει ενεργοποιηθεί το Alarm 1. Αν οι



Σχήμα 5.2: Διάγραμμα ροής της συνάρτησης loop

παραπάνω προϋποθέσεις ισχύουν τότε πραγματοποιείται η σύνδεση με τον NTP Server και ενημερώνεται το RTC με την τρέχουσα ώρα και ημερομηνία.

Έπειτα, η συνάρτηση loop ελέγχει αν έχει παρέλθει το χρονικό διάστημα του 1 λεπτού για την εκτέλεση της διαδικασίας εγγραφής των μετρήσεων στην SD κάρτα. Κατά την εκτέλεση της διαδικασίας εγγραφής των μετρήσεων στην SD κάρτα δημιουργούνται αρχεία δεδομένων για κάθε μέρα με όνομα που βασίζεται στην ημερομηνία που λήφθηκαν οι μετρήσεις πχ 06082016.txt (δηλαδή 6 Αυγούστου του 2016). Η διαδικασία εγγραφής στην SD κάρτα, διαβάζει την ημερομηνία που λήφθηκε κάθε μέτρηση και ελέγχει αν υπάρχει αλλαγή. Εάν δεν έχει αλλάξει η ημερομηνία συνεχίζει την εγγραφή των μετρήσεων στο αντίστοιχο αρχείο δεδομένων. Όταν εντοπίσει αλλαγή ημερομηνίας δημιουργεί αυτόματα νέο αρχείο και αποθηκεύει τα δεδομένα στο νέο αρχείο. Σε κάθε αρχείο που δημιουργείται πραγματοποιείται μία διαμόρφωση που περιέχει την ώρα, την ημερομηνία καθώς και τις ονομασίες των αισθητήρων από τους οποίους λήφθηκαν οι μετρήσεις. Κάθε γραμμή μετρήσεων που γράφεται στην SD κάρτα έχει ως τερματικό χαρακτήρα το \n ώστε να μπορεί να εντοπιστεί από την συνάρτηση ReadLine.

Αμέσως μετά τον έλεγχο για την εγγραφή στην SD κάρτα, ακολουθεί ο έλεγχος για την εγγραφή των δεδομένων στην βάση δεδομένων Για να πραγματοποιηθεί η σύνδεση στον MySQL server, θα πρέπει να υπάρχει σύνδεση στο διαδίκτυο, και να μην υπάρχουν προσωρινά αρχεία. Αν ισχύουν οι παραπάνω προϋποθέσεις ξεκινά η διαδικασία σύνδεσης στον MySQL server και η εγγραφή των δεδομένων στην βάση δεδομένων με την δημιουργία και την εκτέλεση των κατάλληλων queries. Εφόσον η εγγραφή έχει πραγματοποιηθεί το σύστημα αποσυνδέεται από την βάση δεδομένων. Σε περίπτωση που δεν πραγματοποιηθεί σύνδεση με την βάση δεδομένων δημιουργείται ένα προσωρινό αρχείο με όνομα Tempddmm.txt (όπου dd = date και mm= month) στην SD κάρτα. Σε αυτό το αρχείο θα γραφθούν τα queries που δεν στάλθηκαν στην βάση δεδομένων.

Εφόσον ολοκληρωθεί ο έλεγχος για την εγγραφή των δεδομένων στην βάση δεδομένων και η σύνδεση στον MySQL server δεν είναι εφικτή ακολουθεί ο έλεγχος σύνδεσης στον MySQL server ο οποίος πραγματοποιείται σε κάθε κύκλο εκτέλεσης της συνάρτησης loop Όσο η σύνδεση στον MySQL server δεν είναι εφικτή, η εγγραφή των δεδομένων δεν γίνεται στην βάση δεδομένων αλλά συνεχίζεται η αποθήκευση των queries στο αντίστοιχο προσωρινό αρχείο. Αν πραγματοποιηθεί σύνδεση στην βάση τότε το σύστημα ελέγχει αν υπάρχουν προσωρινά αρχεία. Σε αυτή την περίπτωση ξεκινάει την αποστολή των δεδομένων που περιέχουν τα προσωρινά αρχεία. Όσο υπάρχει σύνδεση και προσωρινά αρχεία οι τρέχουσες μετρήσεις καταγράφονται στο πιο πρόσφατο προσωρινό αρχείο. Προφανώς η διαδικασία εγγραφής δεδομένων των προσωρινών αρχείων στην βάση δεδομένων είναι πιο γρήγορη από μία κανονική εγγραφή που πραγματοποιείται ανά λεπτό αφού πραγματοποιείται σε κάθε εκτέλεση της συνάρτησης loop. Κάθε φορά που ολοκληρώνεται η εγγραφή των δεδομένων ενός προσωρινού αρχείου στην βάση δεδομένων, το συγκεκριμένο αρχείο διαγράφεται αυτόματα.

Παρακάτω δίνεται ο κώδικας της συνάρτησης loop (Πρόγραμμα 5.7) όπως αναπτύχθηκε στο περιβάλλον Arduino IDE.

```
if (AlarmTWO==5 && DbWriteFlag==0 && ConLostFlag==0 && AlarmONE==0)
  {
    getTimeAndDate();              //Take time and date from NTP Server
    digitalWrite(8,HIGH);
    uint8_t aa[5];
    int16_t bb;
    aa[0]=second();//Time and date from ntp server are available from ATmega time functions
    aa[1]=minute();
    aa[2]=hour();
    aa[3]=day();
    aa[4]=month();
    bb=year()-2000;
    if (year()>=2100)
      {
        bb=year()-2100;
      }                                                    //Write time and Date to RTC
    WriteReg(0x68,0x00,encodeSM(aa[0]));
    delay(1);
    WriteReg(0x68,0x01,encodeSM(aa[1]));
    delay(1);
    WriteReg(0x68,0x02,encodeHour(aa[2]));
    delay(1);
    WriteReg(0x68,0x04,encodeDate(aa[3]));
    delay(1);
    WriteReg(0x68,0x05,encodeMonth(aa[4]));
    delay(1);
    WriteReg(0x68,0x06,encodeYear(bb));
    delay(1);                                            //Print time and date at LCD
    lcd.setCursor (0,0);
    lcd.print(RtcGetHour());
    lcd.print(":");
    lcd.print(RtcGetMin());
    lcd.print(":");
    lcd.print(RtcGetSec());
    lcd.print(" ");
    lcd.print("RTC Synch");
    lcd.setCursor (0,1);
    lcd.print(RtcGetDate());
    lcd.print("/");
    lcd.print(RtcGetMonth());
    lcd.print("/");
    lcd.print(RtcGetYear());
    AlarmTWO=0;
    digitalWrite(8,LOW);
```

Πρόγραμμα 5.7: Κώδικας της συνάρτησης loop

```
  }

if (SdWriteFlag==1)                                    //Check flag for SD write
  {
   for (int k=0; k<row; k++)
   {
    date=dataBuff[k][0];                               //Read Date from all measurements
    mnth=dataBuff[k][1];
    yr=dataBuff[k][2];
    if (tempBuff[k][0]<10)
    {
     date = "0" + dataBuff[k][0];
    }
    if (tempBuff[k][1]<10)
    {
     mnth ="0" + dataBuff[k][1];
    }
    fileName = date + mnth+ yr + ".txt";               //Create the file name
    if (SD.exists(fileName)==false)                    //Check if day has passed
    {
     myFile=SD.open(fileName,FILE_WRITE);
     if (myFile)
     {
      myFile.println("**********************************");
      SdFormat="Date: \t" + date + "/" + mnth + "/" + yr;          //Format SD file
      myFile.println(SdFormat);
      myFile.println("**********************************\n");
      myFile.println("Time \t     RtcTemp \t    EnvTemp \t    Lux");
      myFile.close();
     }
    }
    myFile= SD.open(fileName,FILE_WRITE);              //Open File and start writing
    if (myFile)
    {
     for (int j=0; j<(col-3);j++)
     {
      if (j<2)
      {
       if (tempBuff[k][j+3]<10)
       {
        tSTR = tSTR + "0" + dataBuff[k][j+3] + ":";
       }
       else
       {
        tSTR = tSTR + dataBuff[k][j+3] + ":";
       }
```

Πρόγραμμα 5.7: Κώδικας της συνάρτησης loop (Συνέχεια)

```
        }
        else if(j<3)
        {
          if (tempBuff[k][j+3] < 10)
          {
            tSTR = tSTR + "0" + dataBuff[k][j+3] +"\t";
          myFile.print(tSTR);
          tSTR = "";
          }
          else
          {
            tSTR = tSTR + dataBuff[k][j+3] +"\t";
          myFile.print(tSTR);
          tSTR = "";
          }
        }
        else
        {
          tSTR = dataBuff[k][j+3] +"    \t ";
          myFile.print(tSTR);
          tSTR = "";
        }
      }
      myFile.print("\n");
      myFile.close();
    }

   }
   SdWriteFlag=0;                                    //When writing is finished set flag to zero
  }
if (DbWriteFlag==1 && ConLostFlag==0 && TempExistFlag==0)
  {
    Serial.println("************************************************");
   String command="";
   if (my_conn.mysql_connect(server_addr, 3306, user, password))          //Connect to DB
   {
     for (int d=0; d<row; d++)
     {
       command = "INSERT INTO " + db + colums + "VALUES (" + dataBuff[d][3] + "," +
dataBuff[d][4] + "," + dataBuff[d][5] + "," + dataBuff[d][0] + "," + dataBuff[d][1] + "," +
dataBuff[d][2]
+ "," + dataBuff[d][6] + ","  + dataBuff[d][7] + "," + dataBuff[d][8] + ")";          //Make Query
       char a[command.length() +1];
       command.toCharArray(a,sizeof(a));                          //Turn in to char array
       my_conn.cmd_query(a);                                      //Send Query
       command="";
```

Πρόγραμμα 5.7: Κώδικας της συνάρτησης loop (Συνέχεια)

```
    }
    my_conn.disconnect();                                   //Disconnect from DB
    DbWriteFlag=0;
   }
    else                                                    //If connection to DB failed
   {
    for (int t=0; t<row; t++)
    {
     dt=dataBuff[t][0];
     mn=dataBuff[t][1];
     TempName="Temp" + dt + mn + ".txt";                    //Create the name of Temp file
     myFile=SD.open(TempName,FILE_WRITE);                   //Make the Temp file
     if (myFile)
      {
       if (myFile.size()==0)
       {
        TempArray[TempCounter] = TempName;                  //Save Temp file's name
        TempCounter+=1;
       }
       db_command = "INSERT INTO " + db + colums + "VALUES (" + dataBuff[t][3] + "," +
dataBuff[t][4] + "," + dataBuff[t][5] + "," + dataBuff[t][0] + "," + dataBuff[t][1] + "," + dataBuff[t][2]
+ "," + dataBuff[t][6] +  ","  + dataBuff[t][7] + "," + dataBuff[t][8] + ")";          //Make Query
       myFile.println(db_command);                          //Write Query to file
       db_command="";
       myFile.close();
      }
    }
    DbWriteFlag=0;                                          //Set to 0 DB write flag
    ConLostFlag=1;                                          //Set to 1 Connection flag
    TempExistFlag=1;                                        //Set to 1 Temp flag
   }
 }
        if (DbWriteFlag==1 && TempExistFlag==1)
         {
          for (int p=0; p<row; p++)
           {
            dt=dataBuff[p][0];
            mn=dataBuff[p][1];
            TempName="Temp" + dt + mn + ".txt";             //Create Temp file's name
            myFile=SD.open(TempName,FILE_WRITE);
```

Πρόγραμμα 5.7: Κώδικας της συνάρτησης loop (Συνέχεια)

```
            if (myFile)
              {
               if (myFile.size()==0)
               {
```

Πρόγραμμα 5.7: Κώδικας της συνάρτησης loop (Συνέχεια)

```
                    TempArray[TempCounter] = TempName;          //Save Temp file's name

                    TempCounter+=1;
                   }
                  db_command = "INSERT INTO " + db + colums + "VALUES (" + dataBuff[p][3] +
"," + dataBuff[p][4] + "," + dataBuff[p][5] + "," + dataBuff[p][0] + "," + dataBuff[p][1] + "," +
dataBuff[p][2] + "," + dataBuff[p][6] + ","  + dataBuff[p][7] +  "," + dataBuff[p][8] + ")";
          //Make Query
                   myFile.println(db_command);                                  //Write Query to file
                   db_command="";
                   myFile.close();
                  }
                 }
                 myFile.close();
                 DbWriteFlag=0;                                          //Set DB write flag to 0
               }
               if (ConLostFlag==1)                                       //Check if connection is back
               {
                if (my_conn.mysql_connect(server_addr, 3306, user, password))        //Connect to DB
                {
                 ConLostFlag=0;
                 my_conn.disconnect();                                   //Disconnect from DB
                }
                else
                {
                 delay(1000);
                }
               }
if (ConLostFlag==0 && TempExistFlag==1)
  {
   String Tname="";
   byt=p2;
   Tname=TempArray[temptemp];                    //Read file's name
   myFile = SD.open(Tname);                      //Open Temp file
   myFile.seek(byt);                             //Go read from the byte it had stopped
   f_bytes = myFile.available();
   while (f_bytes && pos <10)
   {
    myFile.seek(byt);
    buf[pos] = ReadLine();
    pos+=1;

    valcounter+=1;
    f_bytes = myFile.available();
   }
   myFile.close();
```

Πρόγραμμα 5.7: Κώδικας της συνάρτησης loop (Συνέχεια)

```
if (pos==10 | f_bytes==0)       //If 10 lines(Query) done or no more bytes available on the file
{
  String cc="";
   if (my_conn.mysql_connect(server_addr, 3306, user, password))       //Connect to DB
   {
    for (int i=0; i<valcounter ;i++)
    {
      cc= buf[i];
      char aa[cc.length()+1];
      cc.toCharArray(aa,sizeof(aa));
      my_conn.cmd_query(aa);                                          //Send Query
      buf[i]="";
    }
    my_conn.disconnect();                                             //Disconnect from DB
    pos=0;
    valcounter=0;
    p2=byt;
   }
   else                                                          //If it doesn't connected to DB
   {
    ConLostFlag=1;                                                    //No connection flag set to 1
   }
   if (f_bytes==0)                                               //If no more available bytes in file
   {
    SD.remove(TempArray[temptemp]);                                   //Delete temp file
    temptemp+=1;                                                      //Move to next file
    byt=0;
    p2=0;
   }
    if (TempCounter==temptemp)                                        //If Temp files over
  {
    TempCounter=0;                                                //Counter of temp file's set to 0
    temptemp=0;
    TempExistFlag=0;                                        //Flag for the existence of temp files set to 0
  }
 }
}
```

Πρόγραμμα 5.7: Κώδικας της συνάρτησης loop (Συνέχεια)

5.2.3 Το βοηθητικό πρόγραμμα για την αρχικοποίηση της EEPROM

Κατά την εκκίνηση του συστήματος, εκτελείται η διαδικασία ανάγνωσης της EEPROM που έχει ενσωματωμένη το RTC DS3231. Η EEPROM 24C32 έχει μέγεθος 4KB και επικοινωνεί μέσω του TWI. Η χρησιμότητα της μνήμης αυτής στο σύστημα που αναπτύχθηκε, είναι σημαντική καθώς περιέχει όλες τις πληροφορίες που περιλαμβάνουν το όνομά του συστήματος, την IP διεύθυνση του NTP Server, την IP διεύθυνση του MySQL

server, την διεύθυνση του DNS (Domain Name System) server, το Subnet Mask, την Static IP διεύθυνση, το username για την βάση δεδομένων, το password για την βάση δεδομένων καθώς και την πληροφορία για το αν το σύστημα θα λειτουργεί με Static IP διεύθυνση ή με δυναμική IP διεύθυνση μέσω του DHCP.

Στο πρόγραμμα (Πρόγραμμα 5.8) αυτό που αναπτύχθηκε στο περιβάλλον Arduino IDE, με την βοήθεια των συναρτήσεων του TWI γίνονται οι εγγραφές των παραπάνω παραμέτρων στην EEPROM. Κάθε παράμετρος γράφεται σε συγκεκριμένη θέση στην EEPROM και καταλαμβάνει ένα συγκεκριμένο μέγεθος.

```
#include "Arduino.h"                    //Include libraries
#include "TWI.h"
#include <SPI.h>
byte control_byte;                      //Initialize variables
byte mac[6];
byte mainNtpIP[4];
byte backupNtpIP[4];
byte DbServerIP[4];
byte DnServer[4];
byte Gateway[4];
byte Subnet[4];
byte StaticIP[4];
String SystemName="";
String DbUserName="";
String DbPassword="";
byte bb[100];
String Stra[20]="";
char SysName[20];
char Stra1[20];
String abc;
void setup()                            //Setup Function
{
  TWI_Init(true);                       //Enbale TWI
  control_byte=0;                       //Control byte
  mac[0]=0x90;                          //Mac address of Ethernet Shield
  mac[1]=0xD2;
  mac[2]=0xDA;
  mac[3]=0x0B;
  mac[4]=0xD2;
  mac[5]=0xBD;

  mainNtpIP[0]=195;                     //NTP Server IP
  mainNtpIP[1]=167;
  mainNtpIP[2]=30;
```

Πρόγραμμα 5.8: Κώδικας του προγράμματος της EEPROM

```
  mainNtpIP[3]=249;

  backupNtpIP[0]=195;                    //Backup NTP Server IP
  backupNtpIP[1]=167;
  backupNtpIP[2]=30;
  backupNtpIP[3]=249;

  DbServerIP[0]=150;                     //Database IP
  DbServerIP[1]=140;
  DbServerIP[2]=159;
  DbServerIP[3]=140;

  DnServer[0]=195;                       //Dn Server IP
  DnServer[1]=251;
  DnServer[2]=8;
  DnServer[3]=37;

  Gateway[0]=195;                        //Gateway IP
  Gateway[1]=251;
  Gateway[2]=12;
  Gateway[3]=193;

  Subnet[0]=255;                         //Subnet IP
  Subnet[1]=255;
  Subnet[2]=255;
  Subnet[3]=192;

  StaticIP[0]=195;                       //Static IP
  StaticIP[1]=251;
  StaticIP[2]=12;
  StaticIP[3]=204;

  SystemName="Station2";                 //system name
  DbUserName="Stavros";                  //User name for database
  DbPassword="Stavros";                  //Password for database
//Write all these data to eeprom

  WriteReg_16B(0x57,0,control_byte);
  delay(10);
  BurstWriteReg_16B(0x57,1,mac,6);
  delay(10);
  BurstWriteReg_16B(0x57,7,mainNtpIP,4);
```

Πρόγραμμα 5.8: Κώδικας του προγράμματος της EEPROM (Συνέχεια)

```
  delay(10);

  BurstWriteReg_16B(0x57,11,backupNtpIP,4);
  delay(10);
  BurstWriteReg_16B(0x57,15,DbServerIP,4);
  delay(10);
  BurstWriteReg_16B(0x57,19,DnServer,4);
  delay(10);
  BurstWriteReg_16B(0x57,23,Gateway,4);
  delay(10);
  BurstWriteReg_16B(0x57,27,Subnet,4);
  delay(10);
  BurstWriteReg_16B(0x57,31,StaticIP,4);
  delay(10);
//TurnStrings in to bytes and write them at eeprom
  byte bytes[SystemName.length() + 1];
  SystemName.getBytes(bytes, SystemName.length() +1);

  WriteReg_16B(0x57,95,(SystemName.length()));
  delay(10);
  BurstWriteReg_16B(0x57,35,bytes,20);
  delay(10);

  byte a[DbUserName.length() +1];
  DbUserName.getBytes(a,DbUserName.length()+1);

  WriteReg_16B(0x57,96,(DbUserName.length()));
  delay(10);
  BurstWriteReg_16B(0x57,55,a,20);
  delay(10);

  byte b[DbPassword.length() +1];

  DbPassword.getBytes(b,DbPassword.length()+1);
  WriteReg_16B(0x57,97,(DbPassword.length()));
  delay(10);

  BurstWriteReg_16B(0x57,75,a,20);
  delay(10);
```

Πρόγραμμα 5.8: Κώδικας του προγράμματος της EEPROM (Συνέχεια)

5.3 Το LabVIEW Interface

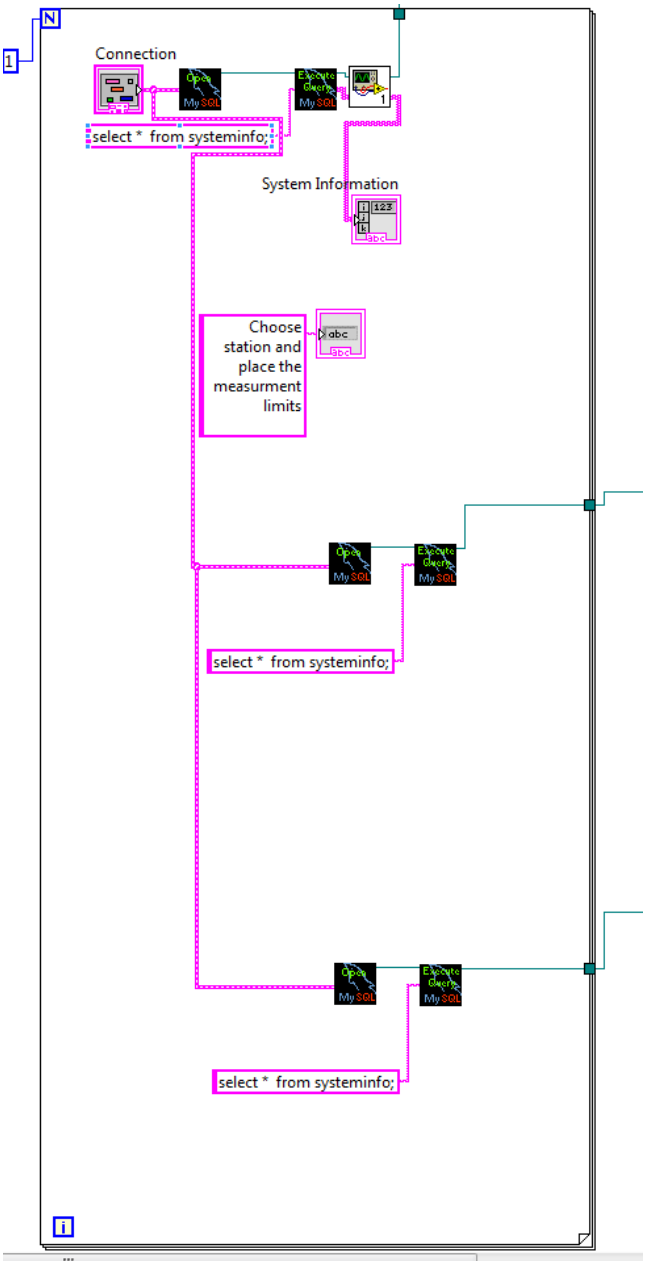
Το GUI (Σχήμα 5.3) που δημιουργήθηκε με το λογισμικό πακέτο του LabVIEW έχει σκοπό την λήψη των δεδομένων από την βάση δεδομένων και την δημιουργία γραφημάτων για περαιτέρω ανάλυση. Ο χρήστης έχει την δυνατότητα να επιλέξει τα δεδομένα που θέλει να επεξεργαστεί μέσω του Front Panel. Η επικοινωνία του LabVIEW με την βάση δεδομένων έχει πραγματοποιηθεί αξιοποιώντας το LabVIEW to MySQL toolkit.

Σχήμα 5.3: Το Front Panel της εφαρμογής

5.3.1 Το Block Diagram της εφαρμογής

Στο Block Diagram βρίσκεται ο κώδικας λειτουργίας της εφαρμογής που πραγματοποιήθηκε. Μέσω του κώδικα αυτού πραγματοποιείται η σύνδεση με την βάση δεδομένων, η ανάκτηση των δεδομένων καθώς και η δημιουργία των αντίστοιχων γραφημάτων.

Στο παρακάτω κομμάτι του κώδικα του Blog Diagram (Σχήμα 5.4) πραγματοποιούνται τρεις ανεξάρτητες συνδέσεις στην βάση δεδομένων, μία για κάθε σταθμό. Τα στοιχεία για την σύνδεση όπως η IP διεύθυνση, το Port, το όνομα της βάσης και οι κωδικοί πρόσβασης παρέχονται από τον cluster με όνομα Connection.

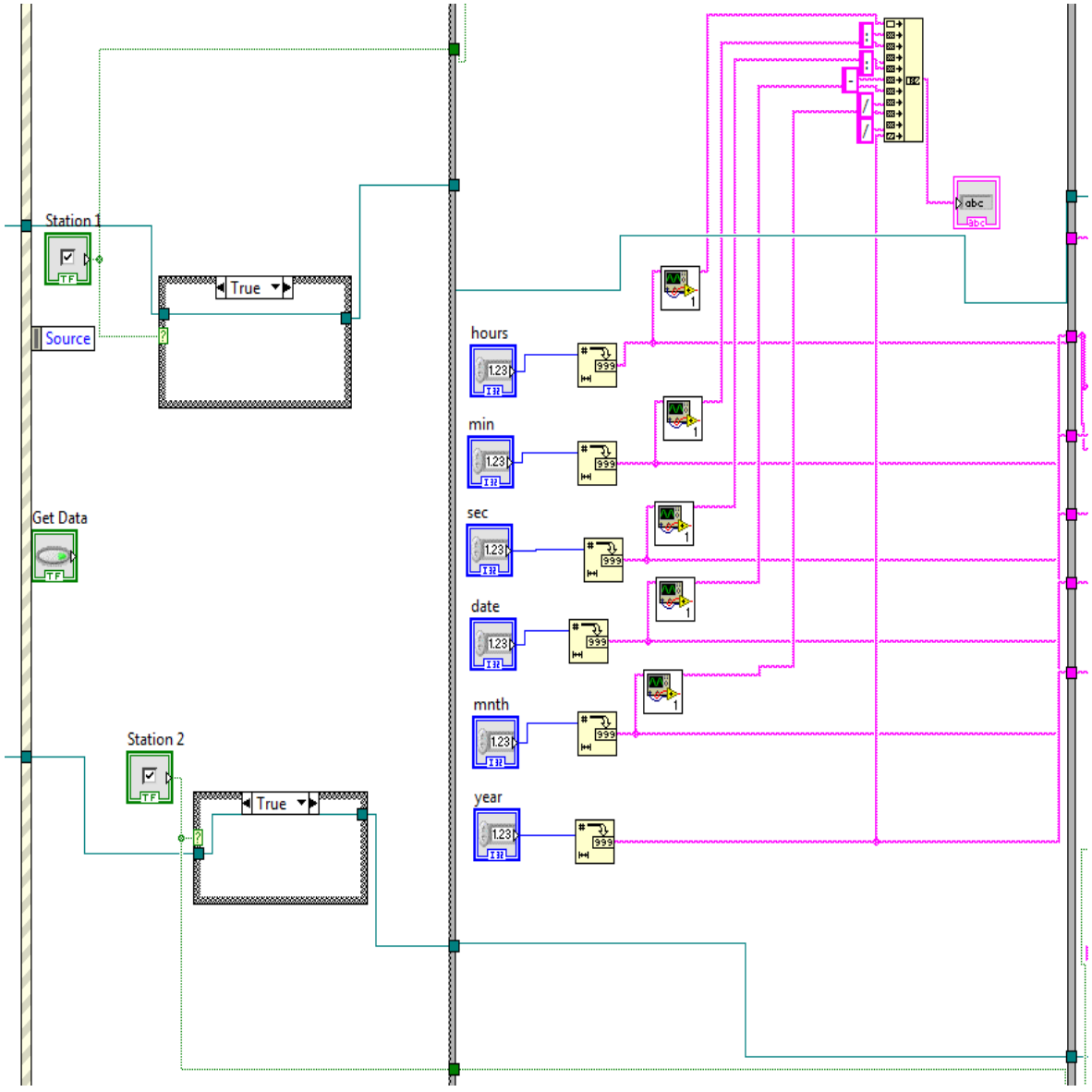


Σχήμα 5.4: Ο κώδικας για την σύνδεση στην βάση και την επιλογή των πινάκων

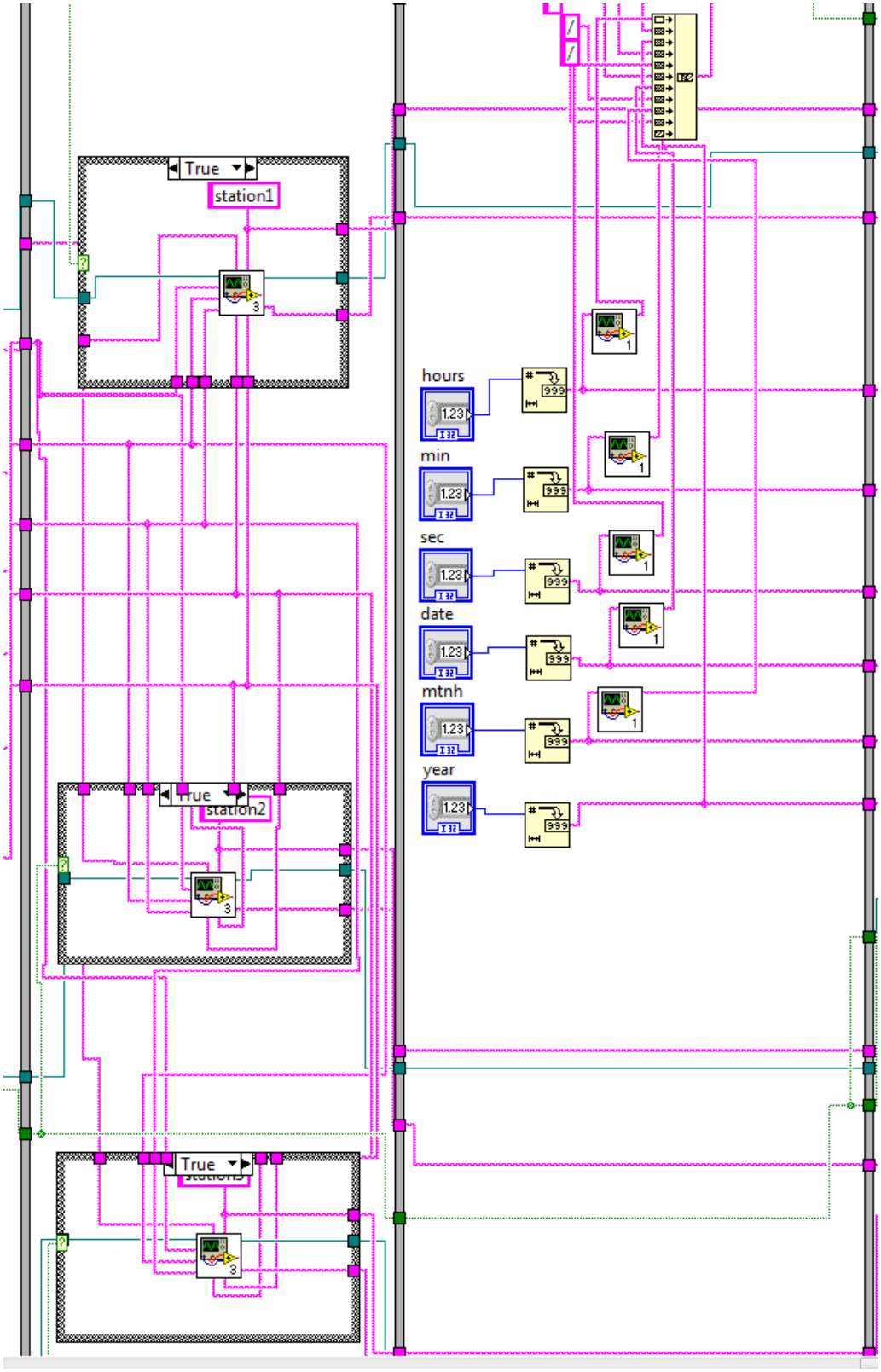
Κατά την σύνδεση στον MySQL server ανακτούνται από την βάση δεδομένων οι πληροφορίες για κάθε διαθέσιμο σταθμό από το table Systeminfo. Οι πληροφορίες αυτές περιλαμβάνουν για κάθε σταθμό το όνομα του table που αποθηκεύει τις μετρήσεις, την τοποθεσία που είναι εγκατεστημένο, το είδος των μετρήσεων. Η διαδικασία αυτή περιλαμβάνεται σε μία For Loop που εκτελείται μόνο την πρώτη φόρα. Επίσης η For Loop αυτή βρίσκεται μέσα στην While Loop που βρίσκεται και ο υπόλοιπος κώδικας του Block Diagram.

Εφόσον έχουν ληφθεί οι πληροφορίες που απαιτούνται στον πίνακα Systeminfo ο χρήστης θα πρέπει να επιλέξει από ποιο σταθμό επιθυμεί να ανακτήσει μετρήσεις καθώς και να επιλέξει το σύνολο των δεδομένων θέτοντας τα αντίστοιχα χρονικά όρια (Σχήμα 5.5). Στην συνέχεια εντοπίζονται μέσω της στηλης ID των αντίστοιχων tables τα δεδομένα που

πρόκειται να ανακτηθούν τα δεδομένα συμφώνα με τις επιλογές του χρήστη σύμφωνα με το τμήμα του κώδικα που παρουσιάζεται στο Σχήμα 5.6. Ο κώδικας που υλοποιεί την παραπάνω διαδικασία (Σχήματα 5.5 και 5.6) βρίσκεται μέσα σε μία περίπτωση της Event Structure η οποία βρίσκεται μέσα στην While Loop και εκτελείται κάθε φορά που πατιέται το κουμπί Get Data. Επομένως ο χρήστης κάθε φορά που επιθυμεί να ανακτήσει δεδομένα είτε από κάποιο άλλο σταθμό ή σταθμούς, είτε να αλλάξει την χρονική περίοδο θα πρέπει να πατήσει το κουμπί Get Data.

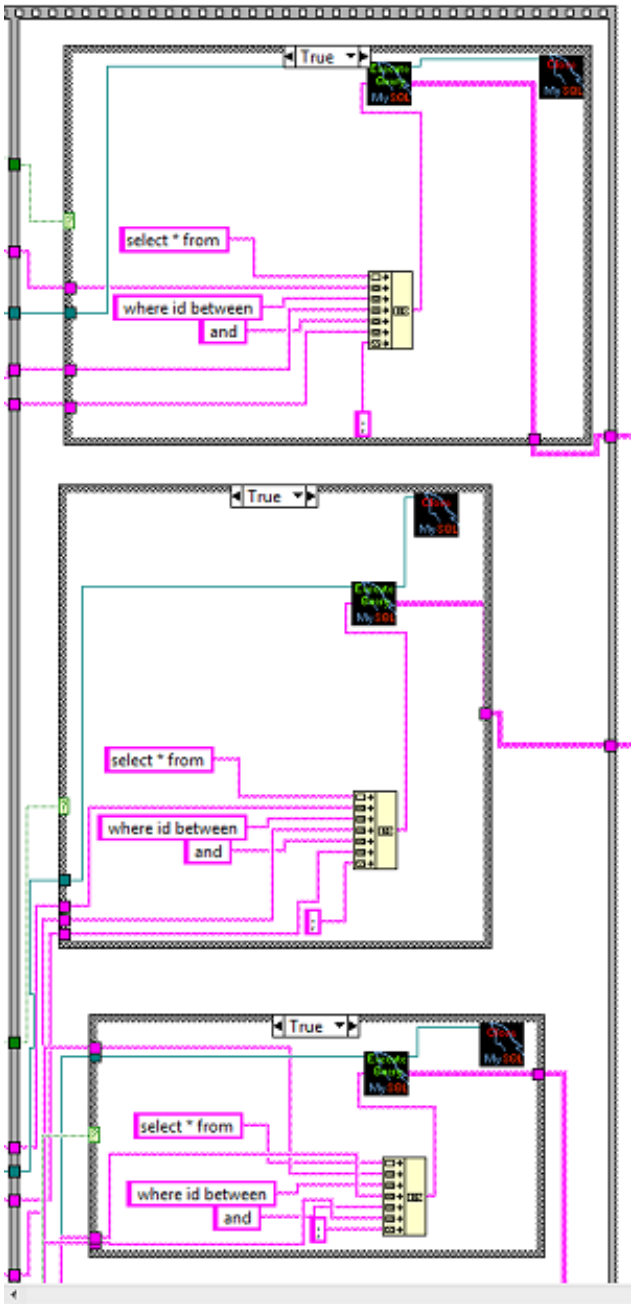


Σχήμα 5.5: Ο κώδικας για την επιλογή των σταθμών και των παραμέτρων ώρας και ημερομηνίας



Σχήμα 5.6: Ο κώδικας για τον εντοπισμό των προς ανάκτηση δεδομένων

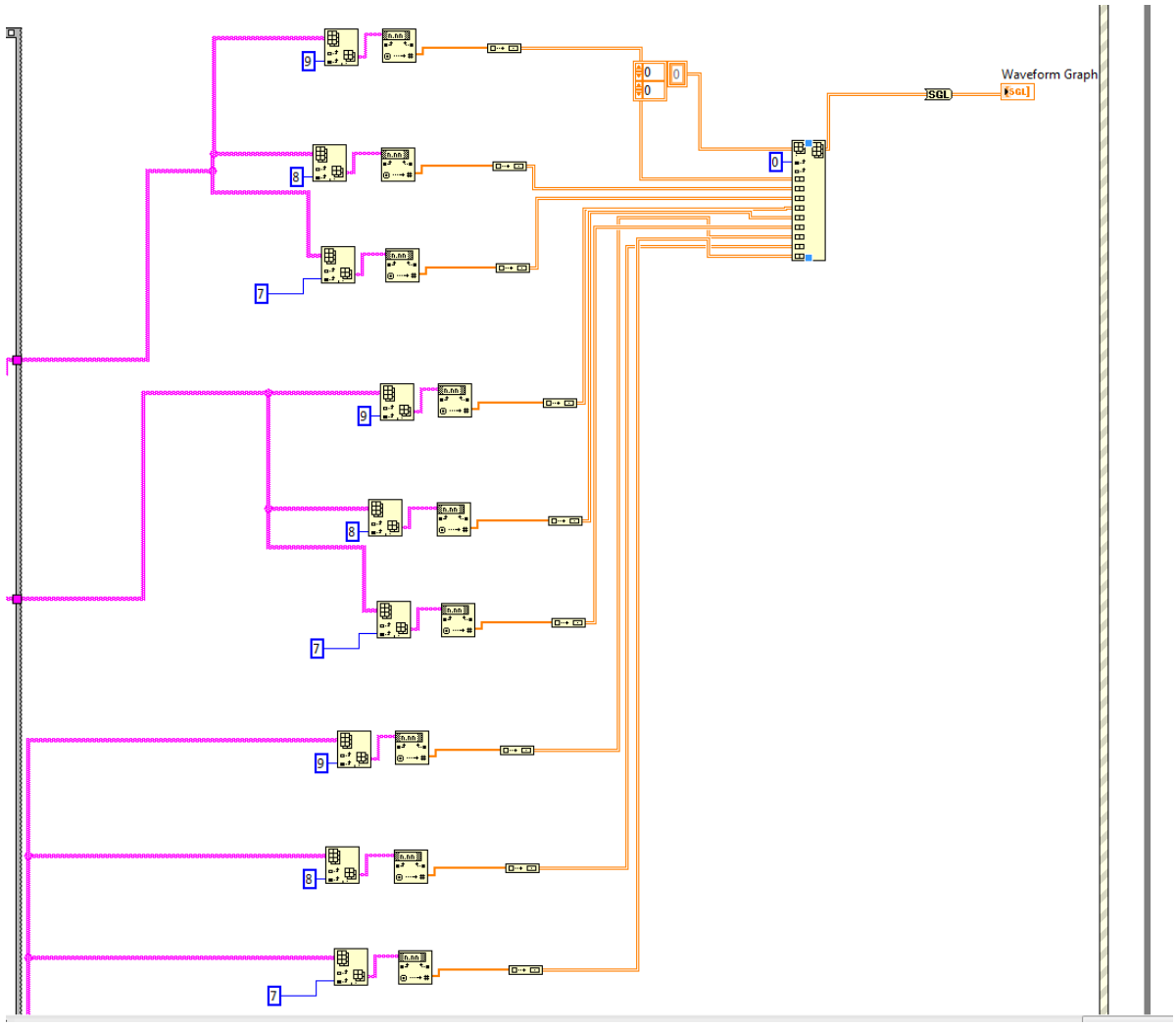
Στην παραπάνω περίπτωση της Event Structure περιλαμβάνεται επίσης και το τμήμα του κώδικα που παρουσιάζεται στο Σχήμα 5.7, το οποίο είναι υπεύθυνο για την ανάκτηση των δεδομένων που θα απεικονιστούν.



Σχήμα 5.7: Ο κώδικας για την ανάκτηση των δεδομένων

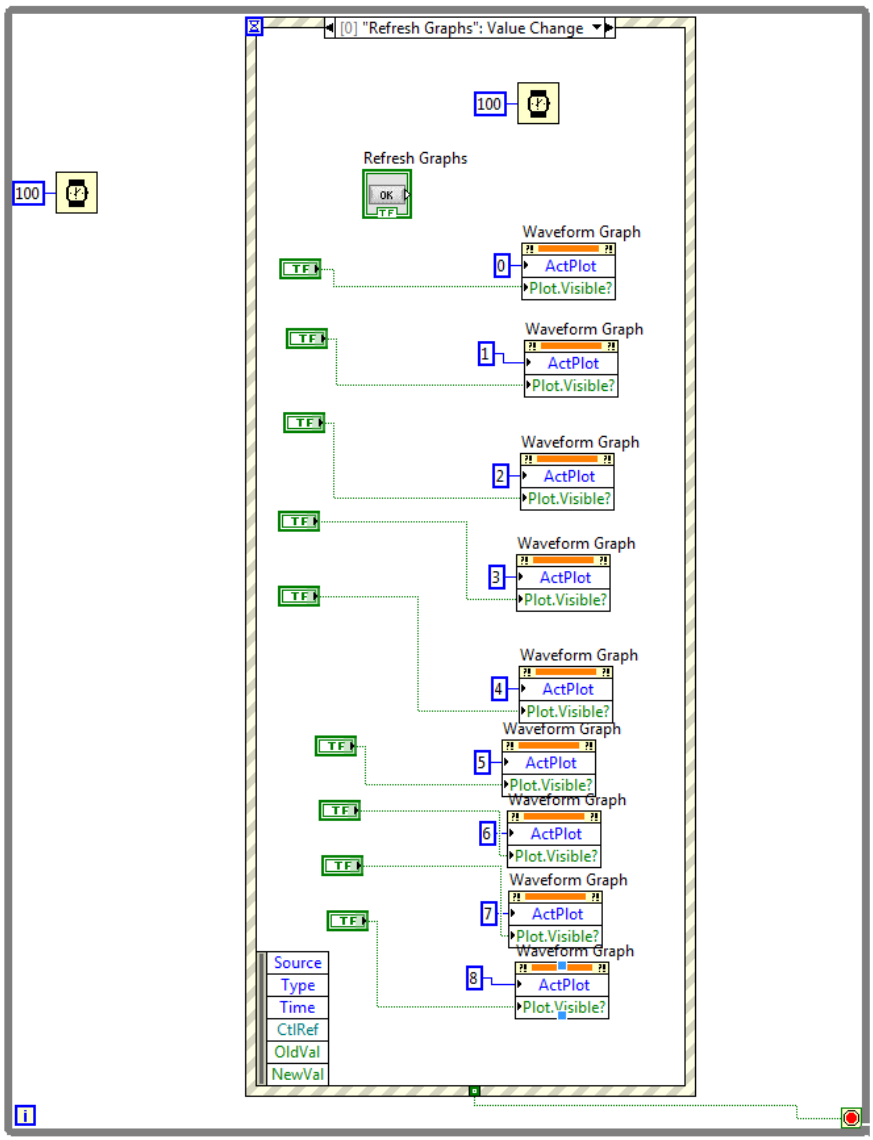
Στο τμήμα του κώδικα που παρουσιάζεται στο Σχήμα 5.8, αναλαμβάνει την οργάνωση και μορφοποίηση των δεδομένων ώστε να πραγματοποιηθεί η γραφική τους αναπαράσταση. Κατά την διαδικασία αυτή, που πραγματοποιείται μέσα στην προαναφερθείσα περίπτωση της Event Structure, επιλέγονται από κάθε πίνακα δεδομένων οι στήλες που περιέχουν τις μετρήσεις φωτεινότητας, θερμοκρασίας χώρου και θερμοκρασίας RTC. Στην συνέχεια, τα δεδομένα αφού διαχωριστούν και μετατραπούν σε πίνακες με αριθμούς κινητής

υποδιαστολής, οργανώνονται ως ένας πίνακας δύο διαστάσεων που οδηγείται στο στοιχείο Waveform Graph.



Σχήμα 5.8: Ο κώδικας για την δημιουργία των γραφημάτων από τα τις μετρήσεις όλων των σταθμών

Εκτός από την κεντρική While Loop που περιέχει τον βασικό κώδικα του συστήματος, υπάρχει και μία δεύτερη While Loop (Σχήμα 5.9) που περιέχει μία Event Structure που εκτελείται κάθε φορά που ο χρήστης πατά το κουμπί Refresh Graphs. Το τμήμα αυτού του κώδικα είναι ανεξάρτητο από το υπόλοιπο και δημιουργήθηκε ώστε ο χρήστης να μπορεί να επιλέγει τις γραφικές παραστάσεις των μεγεθών που επιθυμεί να εμφανίζονται στο Waveform Graph. Με τον τρόπο αυτό αποφεύγεται η εκ νέου ανάκτηση δεδομένων από την βάση δεδομένων που είναι μια χρονοβόρα διαδικασία, κάθε φορά που ο χρήστης επιθυμεί την εμφάνιση του γραφήματος κάποιου άλλου μετρούμενου μεγέθους. Η διαδικασία αυτή έχει κοινό stop με την κεντρική διαδικασία.

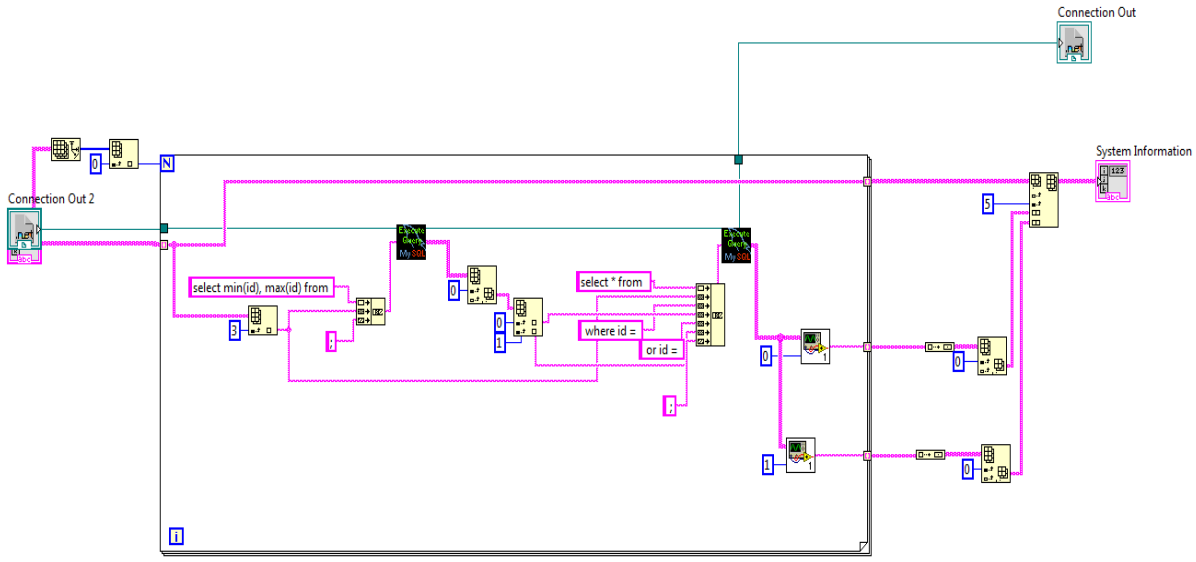


Σχήμα 5.9: Ο κώδικας διαχείρισης των γραφημάτων

Για την ανάπτυξη της εφαρμογής αξιοποιήθηκαν κάποια SubVIs που αναπτύχθηκαν για την υλοποίηση συγκεκριμένων διαδικασιών. Τα SubVIs που υλοποιήθηκαν είναι το Systeminfo.vi, Date format.vi, Fillzeros.vi, ID finder.vi. Η ανάλυση της λειτουργίας των SubVIs πραγματοποιείται παρακάτω.

Systeminfo.vi

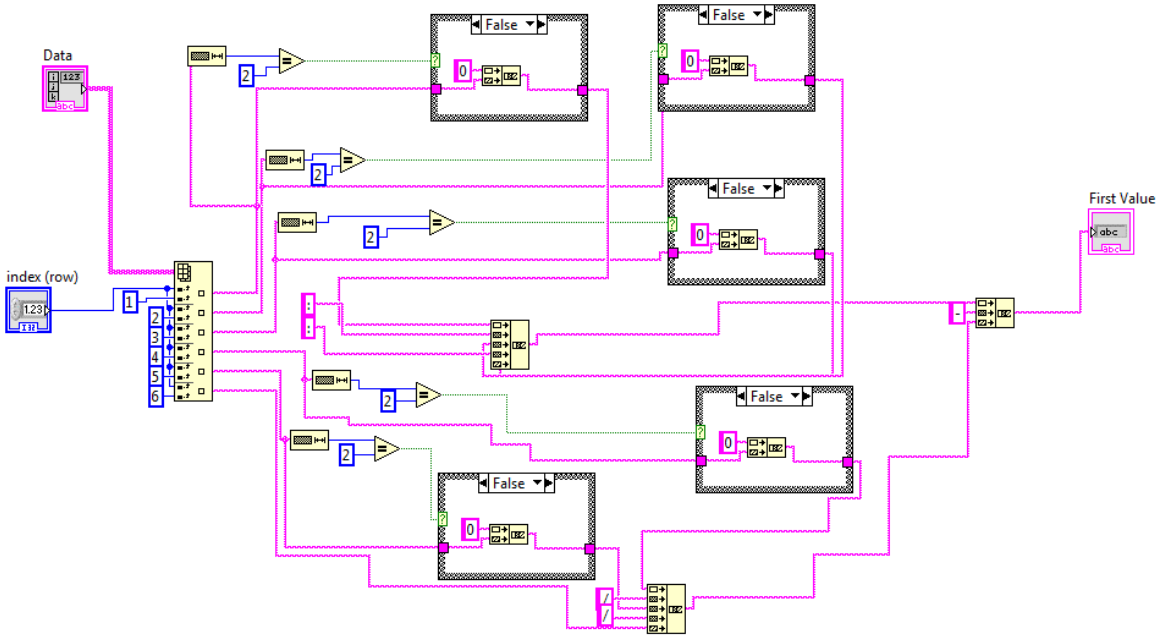
Το Systeminfo SubVI εμφανίζει στον πίνακα systeminformation την χρονική περίοδο κατά την οποία έχει καταγεγραμμένες μετρήσεις κάθε διαθέσιμος σταθμός. Δηλαδή εμφανίζει την ώρα και την ημερομηνία της πρώτης και της τελευταίας μέτρησης που περιέχει το αντίστοιχο table της βάσης δεδομένων.



Σχήμα 5.10: Ο κώδικας του Systeminfo

Date format.vi

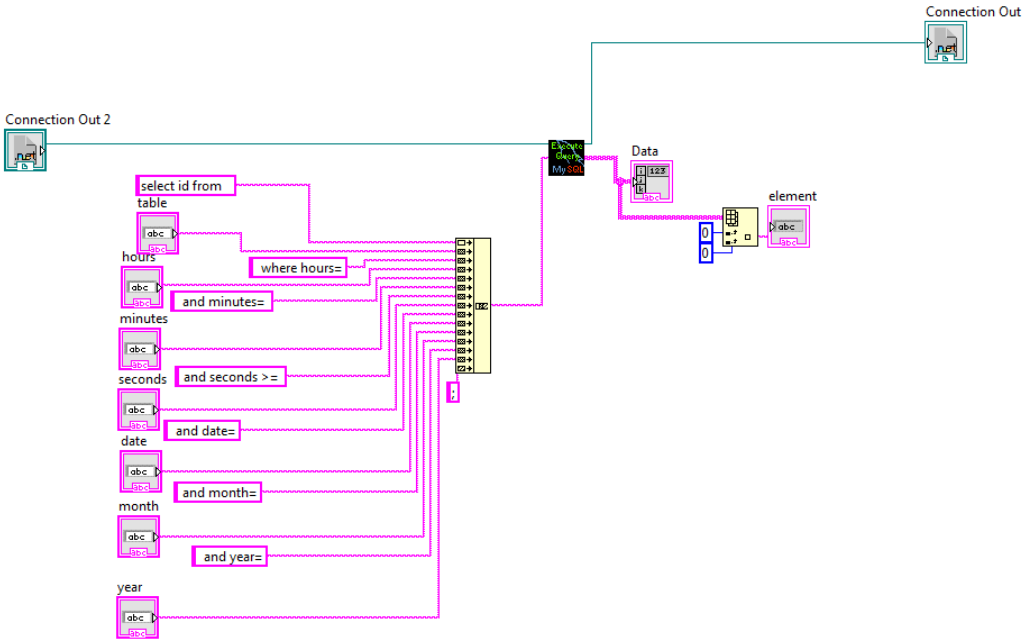
Το SubVI Date format αναλαμβάνει την μορφοποίηση της ώρας και ημερομηνίας μιας μέτρησης από τα δεδομένα που έχουν ανακτηθεί από την βάση δεδομένων, ώστε να έχει την μορφή hh:mm:ss-dd/mm/yyyy .



Σχήμα 5.11: Ο κώδικας του Date format

ID finder.vi

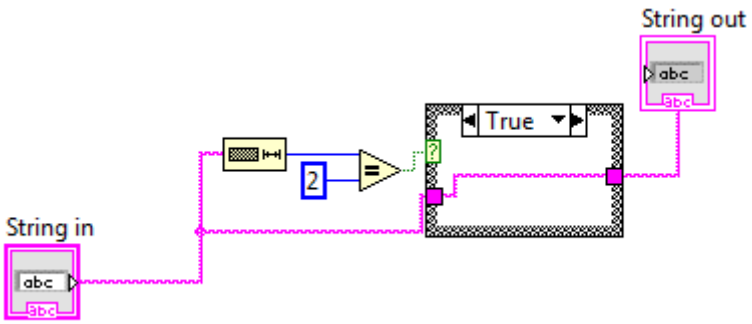
Η λειτουργία του ID finder SubVI είναι να εντοπίζει σύμφωνα με τις παραμέτρους της ώρας και ημερομηνίας που έχουν δοθεί, την τιμή του id στο αντίστοιχο table της βάση δεδομένων. Η τιμή αυτή χρησιμοποιείται για τον εντοπισμό της γραμμής του table που περιέχει τις μετρήσεις που αντιπροσωπεύουν οι παραπάνω παράμετροι ώστε να είναι εφικτή η ανάκτηση των συγκεκριμένων μετρήσεων.



Σχήμα 5.12: Ο κώδικας του ID finder

Fillzeros.vi

Η λειτουργία του Fillzeros SubVI είναι η μορφοποίηση των τιμών της ώρας και ημερομηνίας ώστε να εμφανίζεται με την μορφή hh:mm:ss-dd/mm/yyyy ελέγχοντας το μήκος των τιμών ώρας, λεπτών κλπ. Οι ημερομηνίες αυτές εμφανίζονται στα δύο άκρα του Waveform Graph και δείχνουν τα χρονικά όρια των μετρήσεων που εμφανίζονται.



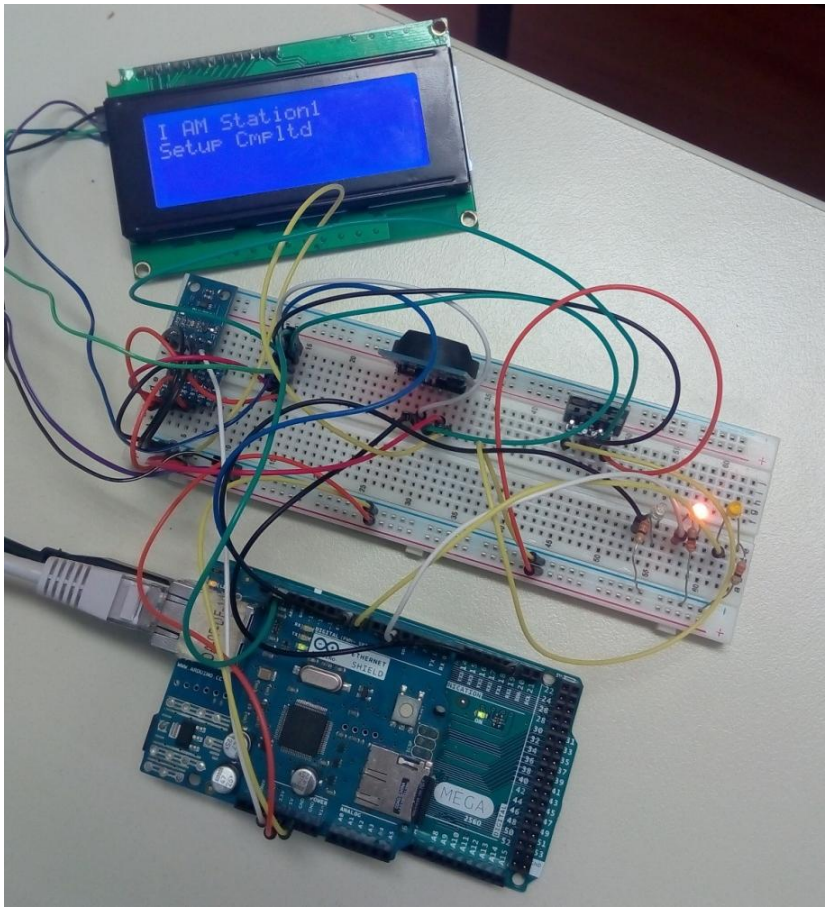
Σχήμα 5.13: Ο κώδικας του Fillzeros.vi

Κεφάλαιο 6

Αποτελέσματα-Συμπεράσματα

6.1 Εισαγωγή

Στο κεφάλαιο αυτό παρουσιάζονται τα αποτελέσματα του συστήματος που πραγματοποιήθηκε στην παρούσα πτυχιακή εργασία κατά την λειτουργία του σε τοπικό επίπεδο και κατά την λειτουργία σε απομακρυσμένο επίπεδο. Με σκοπό την επιβεβαίωση της λειτουργίας της εφαρμογής που αναπτύχθηκε υλοποιήθηκαν τρία όμοια συστήματα τα οποία τοποθετήθηκαν σε διαφορετικούς εργαστηριακούς χώρους του Τμήματος Ηλεκτρολόγων Μηχανικών Τ.Ε. του ΤΕΙ Δυτικής Ελλάδας. Τα παραπάνω συστήματα λειτούργησαν ταυτόχρονα κατά την χρονική περίοδο από 14:00:00 -05/07/2016 έως 14:00:00 -07/07/2016 και κατέγραψαν την θερμοκρασία του χώρου, την φωτεινότητά του καθώς και την θερμοκρασία της συσκευής RTC.



Σχήμα 6.1: Το σύστημα που αναπτύχθηκε

Η διάταξη που παρουσιάζεται στο Σχήμα 6.1 αποτελεί το ένα από τα τρία συστήματα που δημιουργήθηκαν. Εκτός από την οθόνη LCD που παρουσιάζει τα κατάλληλα μηνύματα σχετικά με την λειτουργία και τα στοιχεία του συστήματος, υπάρχουν και τρία ενδεικτικά Led. Το διάφανο Led αναβοσβήνει κάθε φορά που ενεργοποιείται το Alarm1. Το μεσαίο πορτοκαλί Led αναβοσβήνει κάθε φορά που πραγματοποιείται δειγματοληψία. Το τελευταίο

Led αναβοσβήνει κάθε φορά που πραγματοποιείται συγχρονισμός με τον επιλεγμένο NTP Server μέσω διαδικτύου. Η διάταξη αυτή μπορεί να τροφοδοτηθεί είτε μέσω της θύρας USB είτε από κατάλληλο εξωτερικό τροφοδοτικό. Το σύστημα όπως φαίνεται στο παραπάνω σχήμα συνδέεται στην θύρα USB που οδηγεί στον Η/Υ και με την θύρα RJ-45 του Ethernet Shield συνδέεται στο διαδίκτυο.

6.2 Η λειτουργία του συστήματος

Κατά την λειτουργία του συστήματος που ο χρήστης βρίσκεται παρών και παρακολουθεί από κοντά το σύστημα μπορεί να ενημερωθεί για την λειτουργία του από την LCD οθόνη που διαθέτει. Οι ενημερώσεις που μπορεί να λάβει μέσω της οθόνης παρουσιάζονται παρακάτω.

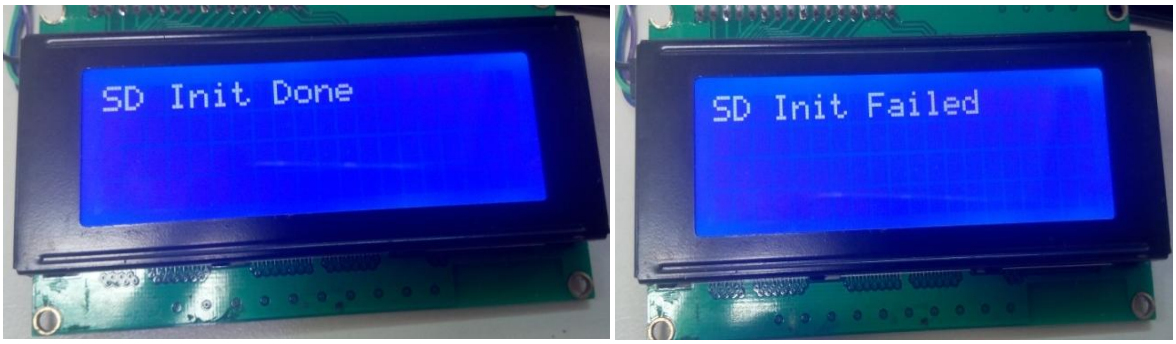
Το μήνυμα που παρουσιάζεται στο Σχήμα 6.2, εμφανίζεται στην LCD οθόνη κατά την εκκίνηση του συστήματος και δηλώνει την επιτυχή λήψη της δυναμικής IP διεύθυνσης ώστε να είναι εφικτή η σύνδεση του συστήματος στο διαδίκτυο.

Σχήμα 6.2: Μήνυμα για την επιτυχή λήψη της δυναμικής IP διεύθυνσης

Σχήμα 6.3: Μήνυμα για την αποτυχία σύνδεσης

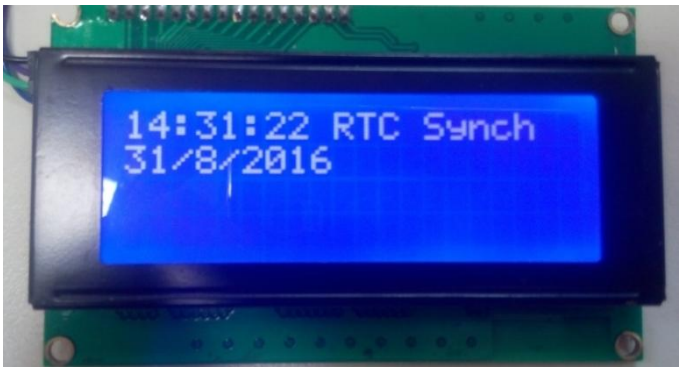
Στην συνέχεια το σύστημα επιχειρεί την σύνδεση με τον MySQL server. Αν δεν είναι δυνατή η σύνδεση με τον MySQL server εμφανίζεται το μήνυμα του Σχήματος 6.3 που δηλώνει πως η σύνδεση απέτυχε και το σύστημα σταματάει σε αυτό το σημείο. Εφόσον επιτευχθεί η σύνδεση με τον MySQL server το σύστημα αποσυνδέεται από τον MySQL server και συνεχίζει με την διαδικασία αρχικοποίησης της SD κάρτας που είναι τοποθετημένη την κατάλληλη υποδοχή που διαθέτει το Ethernet Shield. Με την ολοκλήρωση

της διαδικασίας αρχικοποίησης εμφανίζεται το αντίστοιχο μήνυμα επιτυχούς ή ανεπιτυχούς ολοκλήρωσης της συγκεκριμένης διαδικασίας στην LCD οθόνη (Σχήμα 6.4).



Σχήμα 6.4: Μήνυμα για την επιτυχία ή την αποτυχία της διαδικασίας αρχικοποίησης της SD κάρτας

Έπειτα επιχειρείται η σύνδεση του συστήματος με τον NTP Server για τον συγχρονισμό του RTC. Η διαδικασία αυτή εκτός από την αρχικοποίηση του συστήματος πραγματοποιείται σε τακτά χρονικά διαστήματα διάρκειας 5 λεπτών που καθορίζονται από το Alarm2 του RTC. Σε κάθε επιτυχή συγχρονισμό του συστήματος με τον NTP Server εμφανίζεται το μήνυμα του Σχήματος 6.5 που παρουσιάζει την ώρα και την ημερομηνία του πλέον πρόσφατου συγχρονισμού.



Σχήμα 6.5: Μήνυμα επιτυχούς συγχρονισμού του συστήματος με τον NTP Server

Το μήνυμα του Σχήματος 6.6 εμφανίζεται για να ενημερώσει τον χρήστη πως η διαδικασία αρχικοποίησης έχει ολοκληρωθεί. Επίσης ενημερώνει τον χρήστη με το όνομα του συγκεκριμένου σταθμού το οποίο είναι και το όνομα του table της βάσης δεδομένων στο οποίο καταγράφονται οι μετρήσεις.

Σχήμα 6.6: Μήνυμα για το όνομα του συστήματος

Κατά την λειτουργία του συστήματος τα δεδομένα καταγράφονται στην SD κάρτα σε αρχεία που περιλαμβάνουν μετρήσεις μίας ημέρας και έχουν την μορφή του αρχείου που παρουσιάζεται στο Σχήμα 6.7. Ο χρήστης έχει την δυνατότητα ανακτήσει και να επεξεργαστεί τα παραπάνω αρχεία αν αφαιρέσει την SD κάρτα από το Ethernet Shield και την συνδέσει μέσω κατάλληλης συσκευής στον Η/Υ του.

```
*******************************
Date:   05/07/2016
*******************************

Time          RtcTemp         EnvTemp          Lux
12:30:43    29.25         29.6250          476
12:30:53    29.25         29.6250          476
12:31:03    29.25         29.6250          496
12:31:13    29.25         29.6250          480
12:31:23    29.25         29.6250          480
12:31:33    29.25         29.5625          500
12:31:43    29.25         29.5000          472
12:31:53    29.25         29.5000          496
12:32:03    29.25         29.5625          492
12:32:13    29.25         29.5625          492
12:32:23    29.25         29.5625          488
12:32:33    29.25         29.5000          476
12:32:43    29.25         29.5000          492
12:32:53    29.25         29.5000          472
12:33:03    29.25         29.5000          476
12:33:13    29.25         29.5000          476
12:33:23    29.25         29.5000          476
12:33:33    29.25         29.5000          476
13:15:55    29.75         29.8125          356
13:16:05    29.75         29.8125          316
```

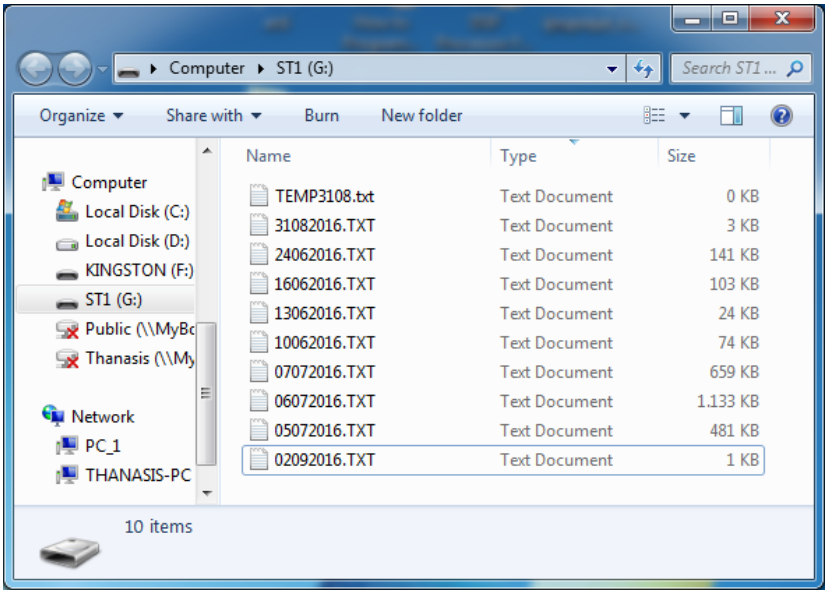
Σχήμα 6.7: Αρχείο μετρήσεων μιας ημέρας

Ο χρήστης επίσης μπορεί να έχει πρόσβαση και στα αρχεία προσωρινών δεδομένων (Σχήμα 6.8). Τα αρχεία αυτά δεν είναι εύκολα στην ανάγνωση από τον χρήστη επειδή περιέχουν τις SQL εντολές που πρέπει να εκτελεστούν για να είναι εφικτή η εγγραφή των προσωρινών δεδομένων στην βάση δεδομένων.

```
INSERT INTO test.station1 (hours, minutes, seconds, date, month, year, RTCtemp,EnvTemp,Lux)VALUES (14,50,56,31,8,2016,28.00,28.0625,640)
INSERT INTO test.station1 (hours, minutes, seconds, date, month, year, RTCtemp,EnvTemp,Lux)VALUES (14,51,6,31,8,2016,28.00,28.0625,644)
INSERT INTO test.station1 (hours, minutes, seconds, date, month, year, RTCtemp,EnvTemp,Lux)VALUES (14,51,16,31,8,2016,28.00,28.1250,284)
INSERT INTO test.station1 (hours, minutes, seconds, date, month, year, RTCtemp,EnvTemp,Lux)VALUES (14,51,26,31,8,2016,28.00,28.0625,624)
INSERT INTO test.station1 (hours, minutes, seconds, date, month, year, RTCtemp,EnvTemp,Lux)VALUES (14,51,36,31,8,2016,28.00,28.0625,640)
INSERT INTO test.station1 (hours, minutes, seconds, date, month, year, RTCtemp,EnvTemp,Lux)VALUES (14,51,46,31,8,2016,28.00,28.0625,632)
INSERT INTO test.station1 (hours, minutes, seconds, date, month, year, RTCtemp,EnvTemp,Lux)VALUES (14,51,56,31,8,2016,28.00,28.0625,632)
INSERT INTO test.station1 (hours, minutes, seconds, date, month, year, RTCtemp,EnvTemp,Lux)VALUES (14,52,6,31,8,2016,28.00,28.0625,640)
INSERT INTO test.station1 (hours, minutes, seconds, date, month, year, RTCtemp,EnvTemp,Lux)VALUES (14,52,16,31,8,2016,28.00,28.0625,616)
INSERT INTO test.station1 (hours, minutes, seconds, date, month, year, RTCtemp,EnvTemp,Lux)VALUES (14,52,26,31,8,2016,28.00,28.0625,616)
INSERT INTO test.station1 (hours, minutes, seconds, date, month, year, RTCtemp,EnvTemp,Lux)VALUES (14,52,36,31,8,2016,28.00,28.0625,628)
```

Σχήμα 6.8: Προσωρινό αρχείο μετρήσεων

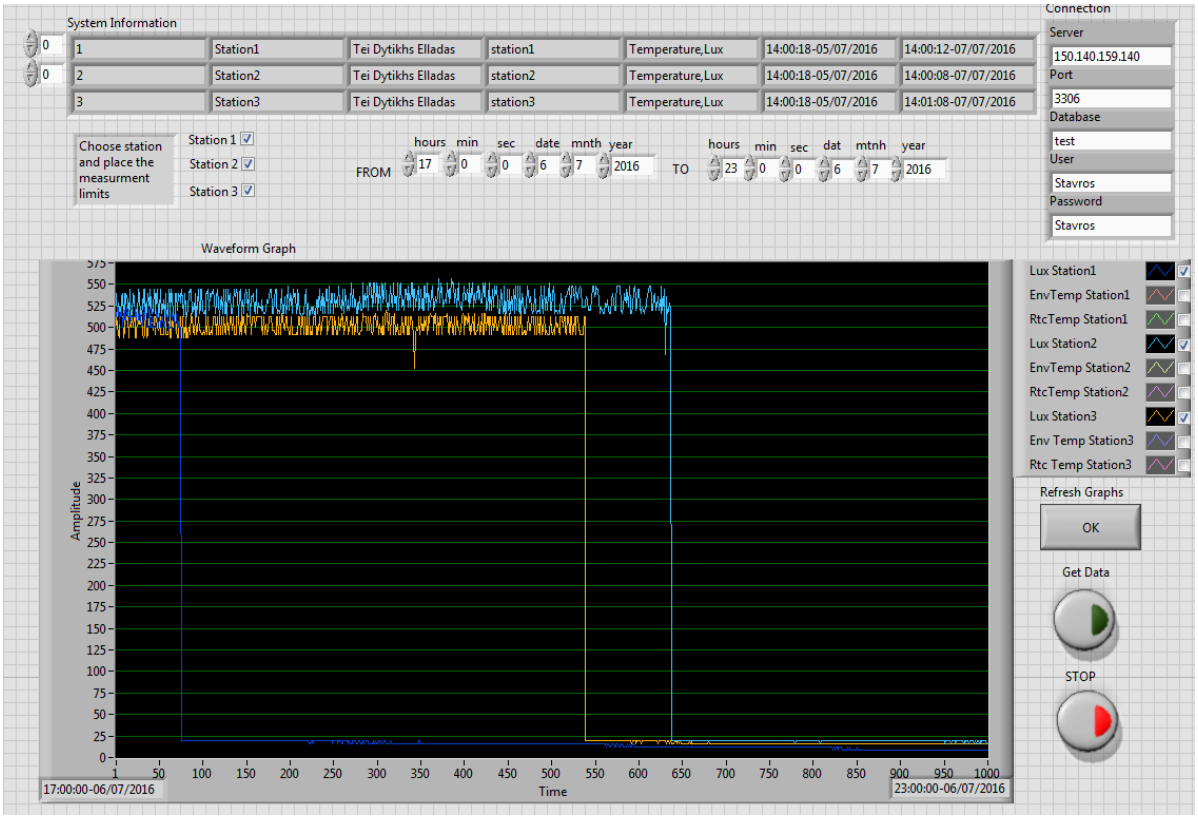
Κατά την ανάγνωση της SD κάρτας από τον υπολογιστή του χρήστη εμφανίζεται το παράθυρο του Σχήματος 6.9. Σε αυτό εμφανίζονται μια σειρά από .txt αρχεία που έχουν ως όνομα την ημερομηνία που δημιουργήθηκαν (πχ 06072016.txt για το αρχείο που δημιουργήθηκε στις 7/7/2016). Επίσης είναι πιθανόν να υπάρχουν και προσωρινά αρχεία δεδομένων των οποίων το όνομα ξεκινά με την λέξη TEMP.



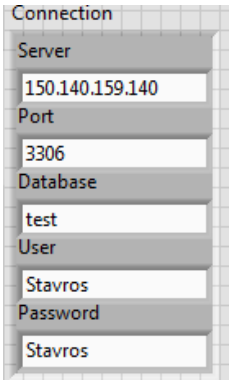
Σχήμα 6.9: Τα αρχεία στην SD κάρτα

6.3 Η λειτουργία του GUI

Η απομακρυσμένη λήψη των μετρήσεων των διαθέσιμων σταθμών πραγματοποιείται μέσω του GUI που αναπτύχθηκε με το LabVIEW και παρουσιάζεται στο Σχήμα 6.10. Αρχικά ο χρήστης στο πεδίο Connection (Σχήμα 6.11) θα πρέπει να καθορίσει την IP διεύθυνση του MySQL server, το Port, το όνομα της βάσης δεδομένων, το username και το password ώστε να είναι εφικτή η σύνδεση του GUI με την βάση δεδομένων.

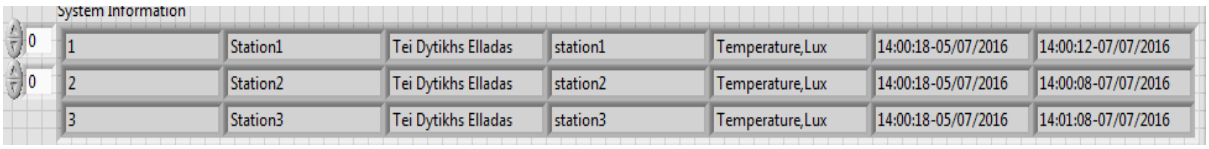


Σχήμα 6.10: Το GUI για την απομακρυσμένη λήψη και διαχείριση των μετρήσεων



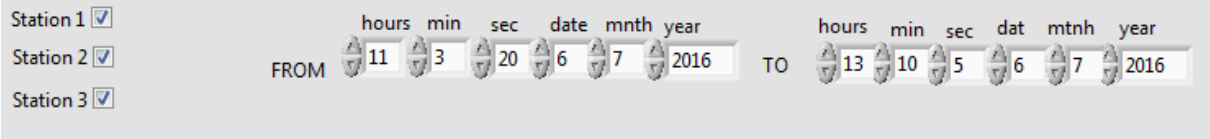
Σχήμα 6.11: Στοιχεία για την σύνδεση στην βάση

Εφόσον συμπληρωθούν τα παραπάνω στοιχεία και ξεκινήσει η εκτέλεση του GUI εμφανίζονται στον πίνακα του Σχήματος 6.12 πληροφορίες σχετικά με τα διαθέσιμα συστήματα. Οι πληροφορίες αυτές περιλαμβάνουν για κάθε σύστημα το όνομα του table που αποθηκεύει τις μετρήσεις, την τοποθεσία που είναι εγκατεστημένο, το είδος των μετρήσεων που λαμβάνει και την χρονική περίοδο κατά την οποία έχει καταγεγραμμένες μετρήσεις.



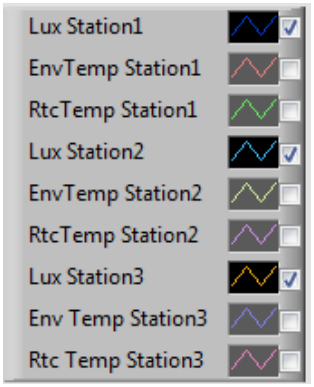
Σχήμα 6.12: Πίνακας πληροφοριών για όλα τα συστήματα

Ο χρήστης εφόσον γνωρίζει τα στοιχεία κάθε συστήματος έχει την δυνατότητα να επιλέξει αφενός τα την χρονική περίοδο των μετρήσεων που επιθυμεί και αφετέρου ένα ή περισσότερα από τα διαθέσιμα συστήματα (Σχήμα 6.13). Με τον τρόπο όταν πατήσει το κουμπί Get Data θα ανακτηθούν τα επιλεγμένα δεδομένα από την βάση δεδομένων.

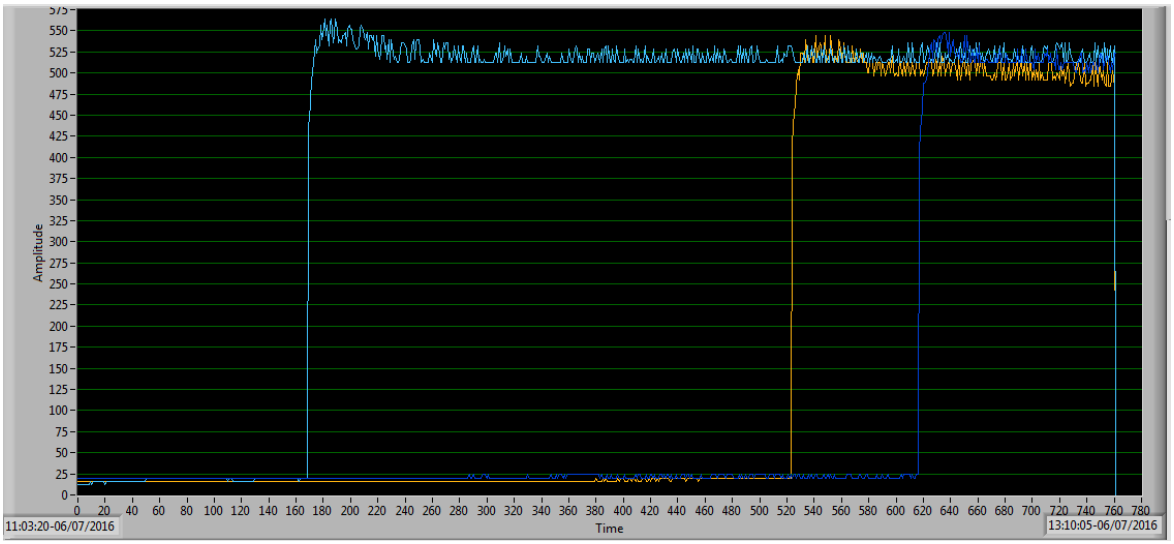


Σχήμα 6.13: Επιλογή συστήματος και χρονικής περιόδου για την ανάκτηση δεδομένων

Όταν ολοκληρωθεί η ανάκτηση των δεδομένων από την βάση δεδομένων, ο χρήστης μπορεί να επιλέξει τα γραφήματα που θέλει να δημιουργηθούν για κάθε σύστημα και για κάθε είδος μέτρησης μέσω των επιλογών που παρουσιάζονται στο Σχήμα 6.14. Η γραφική αναπαράσταση των μετρήσεων ενημερώνεται σύμφωνα με τις επιλογές του χρήστη όταν πατηθεί το κουμπί Refresh Graphs. Σύμφωνα με τις παραπάνω επιλογές θα δημιουργηθούν τα γραφήματα που παρουσιάζονται στο Σχήμα 6.15.



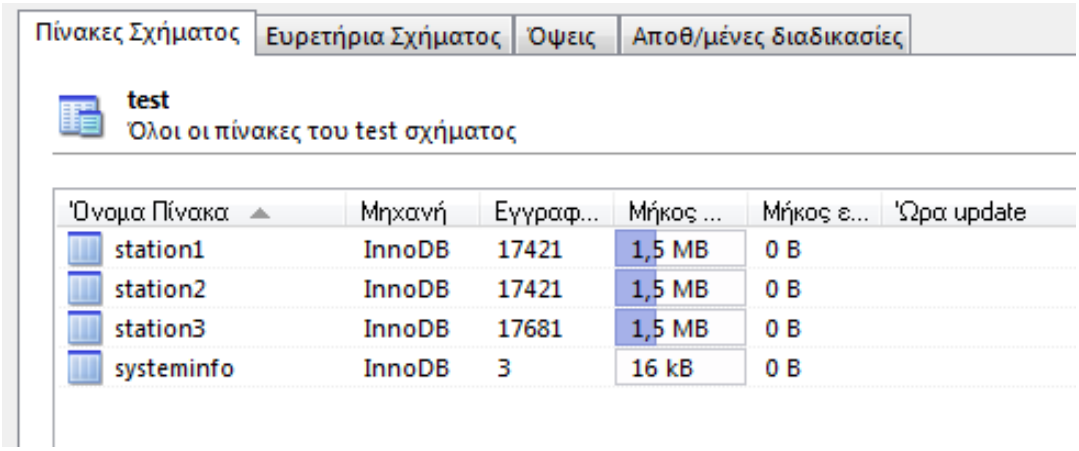
Σχήμα 6.14: Επιλογή γραφημάτων



Σχήμα 6.15: Προβολή γραφημάτων

6.4 Η ανάκτηση δεδομένων με το MySQL Administrator

Ο χρήστης έχει την δυνατότητα να ανακτήσει τα δεδομένα των μετρήσεων των διαθεσίμων σταθμών αξιοποιώντας το MySQL Administrator. Μέσω αυτού του περιβάλλοντος μπορεί να συνδεθεί στην βάση δεδομένων, να ανακτήσει και να διαχειριστεί τα δεδομένα που επιθυμεί. Στο περιβάλλον MySQL Administrator ο χρήστης επίσης μπορεί να διαμορφώσει εκ νέου την βάση δεδομένων, να επεξεργαστεί τους πίνακες, να διαχωρίσει τα δεδομένα κλπ. Το γεγονός αυτό οδήγησε στην ανάπτυξη του GUI που παρουσιάστηκε στην προηγούμενη παράγραφο ώστε οι απλοί χρήστες (εκτός των διαχειριστών) να μην έχουν την δυνατότητα να μεταβάλλουν την βάση δεδομένων είτε ως προς την μορφοποίηση της είτε ως προς το περιεχόμενο της γιατί αυτό θα είχε σημαντικές επιπτώσεις στην λειτουργία του συνολικού συστήματος.



Σχήμα 6.16: Τα tables του σχήματος test

Μόλις ο χρήστης συνδεθεί στην βάση δεδομένων και επιλέξει το σχήμα test εμφανίζονται τέσσερα tables (Σχήμα 6.16). Το table systeminfo περιέχει τα ονόματα των διαθέσιμων σταθμών, την τοποθεσία τους, το όνομα του table που καταγράφονται οι μετρήσεις και το είδος των μετρήσεων που συλλέγει κάθε σταθμός (Σχήμα 6.17).

id	StationName	SystemLocation	Table	Measurments Types
1	Station1	Tei Dytikhs Elladas	station1	Temperature,Lux
2	Station2	Tei Dytikhs Elladas	station2	Temperature,Lux
3	Station3	Tei Dytikhs Elladas	station3	Temperature,Lux

Σχήμα 6.17: Το περιεχόμενο του table systeminfo

Ανοίγοντας το table ενός σταθμού μετρήσεων (Σχήμα 6.18) ο χρήστης έχει την δυνατότητα να παρατηρήσει όλες τις μετρήσεις που έχουν ληφθεί από τον επιλεγμένο σταθμό και μπορεί να τις επεξεργαστεί χρησιμοποιώντας SQL εντολές.

Σύνολο Αποτελεσμάτων 1

id	hours	minutes	seconds	date	month	year	RTCtemp	EnvTemp	Lux
43	14	0	18	5	7	2016	30.5	30.6875	544
44	14	0	28	5	7	2016	30.5	30.6875	552
45	14	0	38	5	7	2016	30.5	30.6875	524
46	14	0	48	5	7	2016	30.5	30.6875	552
47	14	0	58	5	7	2016	30.5	30.6875	544
48	14	1	8	5	7	2016	30.5	30.6875	548
49	14	1	18	5	7	2016	30.5	30.6875	532
50	14	1	28	5	7	2016	30.5	30.6875	520
51	14	1	38	5	7	2016	30.5	30.75	552
52	14	1	48	5	7	2016	30.5	30.6875	524
53	14	1	58	5	7	2016	30.5	30.75	520
54	14	2	8	5	7	2016	30.5	30.75	540
55	14	2	18	5	7	2016	30.5	30.75	548
56	14	2	28	5	7	2016	30.5	30.6875	520
57	14	2	38	5	7	2016	30.5	30.75	520
58	14	2	48	5	7	2016	30.5	30.75	544
59	14	2	58	5	7	2016	30.5	30.6875	520
60	14	3	8	5	7	2016	30.5	30.75	540
61	14	3	18	5	7	2016	30.5	30.75	536
62	14	3	28	5	7	2016	30.5	30.75	536

Σχήμα 6.18: Το περιεχόμενο ενός table που καταγράφονται οι μετρήσεις

6.5 Προτάσεις-Συμπεράσματα

Η παρούσα πτυχιακή εργασία είχε ως αντικείμενο τον σχεδιασμό και την ανάπτυξη ενός συστήματος συλλογής, αποθήκευσης και επεξεργασίας περιβαλλοντικών μετρήσεων το οποίο παρέχει στον χρήστη την δυνατότητα να διαχειριστεί τις μετρήσεις απομακρυσμένα μέσω του GUI που δημιουργήθηκε είτε μέσω της σύνδεσής του στην βάση δεδομένων μέσω του MySQL Administrator. Η ανάπτυξη του συγκεκριμένου συστήματος στηρίχθηκε στην πλατφόρμα Arduino Mega2560 η οποία διαθέτει τον μικροελεγκτή ATmega 2560 αξιοποιώντας το Arduino IDE. Το GUI αναπτύχθηκε με το λογισμικό πακέτο LabVIEW ενώ για την διαμόρφωση της MySQL βάσης δεδομένων χρησιμοποιήθηκε το λογισμικό MySQL Administrator.

Όπως παρατηρήθηκε από τις μετρήσεις που πραγματοποιήθηκαν υπάρχει μία απώλεια χρόνου κατά την δειγματοληψία. Η απώλεια αυτή είναι ένα δευτερόλεπτο κάθε 8-9 ώρες και δικαιολογείται λόγο της χρονικής ακρίβειας του RTC και την χρονική καθυστέρηση που παρουσιάζει το σύστημα στο σύνολό του. Για την αντιμετώπιση της χρονικής καθυστέρησης του RTC θα μπορούσε να χρησιμοποιηθεί ένα RTC με μεγαλύτερη ακρίβεια.

Επίσης κατά την ανάπτυξη του συστήματος παρατηρήθηκε πως η αρχικοποίηση της LCD οθόνης μέσω των έτοιμων βιβλιοθηκών που παρέχονται θα πρέπει να προηγηθεί των αρχικοποιήσεων των υπόλοιπων συσκευών που κάνουν χρήση του TWI. Η παραπάνω δυσλειτουργία πιθανώς οφείλεται σε κάποια διένεξη που υπάρχει μεταξύ της συγκεκριμένης βιβλιοθήκης για την υποστήριξη της LCD οθόνης και των βιβλιοθηκών που δημιουργήθηκαν στα πλαίσια της παρούσας πτυχιακής εργασίας για την αξιοποίηση του TWI. Για την επίλυση αυτού του προβλήματος προτείνεται είτε ή αξιοποίηση κάποιας άλλης έτοιμης βιβλιοθήκης είτε η ανάπτυξη μιας νέας βιβλιοθήκης για την LCD οθόνη η οποία θα είναι βασισμένοι στις συναρτήσεις για την διαχείριση του TWI που παρουσιάστηκαν σε προηγούμενο κεφάλαιο.

Η δειγματοληψία των μετρήσεων στο συγκεκριμένο σύστημα πραγματοποιείται αξιοποιώντας ένα σήμα εξωτερική διακοπής που προέρχεται από το RTC. Κατά την ανάπτυξη του συστήματος παρατηρήθηκε ότι το σύστημα σταματούσε σε τυχαίες χρονικές

στιγμές κατά την εκτέλεση της αντίστοιχης ISR. Έπειτα από μια σειρά πειραμάτων προέκυψε το συμπέρασμα ότι το πρόβλημα αυτό οφείλεται στην καθυστέρηση απόκρισης του αισθητήρα φωτεινότητας όταν αυτό είχε ρυθμιστεί ώστε να λαμβάνει μετρήσεις με την μέγιστη ακρίβεια. Στην συγκεκριμένη υλοποίηση για να αποφευχθεί το παραπάνω πρόβλημα χρησιμοποιήθηκε η μικρότερη ακρίβεια στον συγκεκριμένο αισθητήρα με αποτέλεσμα την μεγάλη μείωση του χρόνου απόκρισης. Παρόλα αυτά μια πιο αποτελεσματική αντιμετώπιση του συγκεκριμένου προβλήματος θα ήταν η αξιοποίηση ενός δεύτερου μικρότερου μικροελεγκτή ο οποίος θα αναλάμβανε αποκλειστικά την διαχείριση των διακοπών του RTC ώστε να εξασφαλίζεται τόσο ο σταθερός ρυθμός δειγματοληψίας όσο και η αξιοπιστία του συστήματος ακόμη και αν προστεθεί μεγαλύτερος αριθμός αισθητήρων ή άλλων συσκευών.

Το σημαντικότερο πλεονέκτημα του συγκεκριμένου συστήματος είναι ότι με ελάχιστες τροποποιήσεις στον πρόγραμμα του μικροελεγκτή θα μπορούσαν να υποστηριχθούν αισθητήρες για την μέτρηση διαφορετικών φυσικών μεγεθών. Το γεγονός αυτό προσφέρει στο συγκεκριμένο σύστημα μια δυνατότητα ευελιξίας που έχει ως αποτέλεσμα την αξιοποίηση του σε διαφορετικές εφαρμογές.

Με σκοπό την εξέλιξη του συστήματος μελλοντικά θα μπορούσε να αναπτυχθεί ένας ή περισσότεροι κεντρικοί σταθμοί που θα αναλαμβάνουν την ανάκτηση και επεξεργασία των μετρήσεων με σκοπό την λήψη αποφάσεων και τον έλεγχο άλλων συστημάτων ή παραγωγικών διαδικασιών. Επίσης, θα μπορούσε να μελετηθεί η ανάπτυξη συστημάτων που θα υποστηρίζουν είτε την ασύρματη μεταφορά δεδομένων μέσω του δικτύου κινητής τηλεφωνίας είτε θα έχουν την δυνατότητα να επικοινωνούν ασύρματα μεταξύ τους μέσω πρωτοκόλλων ασύρματης δικτύωσης όπως το zigbee.

Βιβλιογραφία

[1] Atmel, "8-bit Atmel Microcontroller with 16/32/64KB In-System Programmable Flash", 2549Q–AVR, February 2014.

[2] Arduino Mega 2560, https://www.arduino.cc/en/Main/ArduinoBoardMega2560, τελευταία επίσκεψη, 3/7/2016.

[3] Arduino Ethernet Shield, https://www.arduino.cc/en/Main/ArduinoEthernetShield, τελευταία επίσκεψη, 3/7/2016.

[4] Arduino attachInterrupt(), https://www.arduino.cc/en/Reference/AttachInterrupt, τελευταία επίσκεψη, 4/7/2016.

[5] Αθανάσιος Καμπύλης, "Διαδραστικότητα Arduino με συσκευές Android", Διπλωματική Εργασία, Τμήμα Φυσικής, Παν. Πατρών, Σεπτέμβριος 2014.

[6] Maxim Integrated Products, "Extremely Accurate I2C-Integrated RTC /TCXO /Crystal", 19-5170, Rev 10, March 2015.

[7] ROHM Semiconductor, "Digital 16bit Serial Output Type Ambient Light Sensor IC", Rev.D. , November 2011.

[8] Microchip, "2-Wire Serial Temperature Sensor", DS21935D, 2010.

[9] Sparkfun, Using the Logic Level Converter, https://learn.sparkfun.com/tutorials/using-the-logic-level-converter, τελευταία επίσκεψη, 7/7/2016.

[10] Wikipedia, I2C, https://en.wikipedia.org/wiki/I%_C2%B2C, τελευταία επίσκεψη, 7/7/2016.

[11] EMBEDDS, Programming AVR I2C Interface, http://www.embedds.com/programming-avr-i2c-interface/, τελευταία επίσκεψη, 7/7/2016.

[12] Wikipedia, Πληροφοριακά συστήματα, https: //el.wikipedia.org/wiki/ Πληροφοριακά_συστήματα, τελευταία επίσκεψη, 20/7/2016.

[13] Wikipedia, Μικροελεγκτής, https://el.wikipedia.org/wiki/ %CE%9C%CE%B9%CE%BA%CF%81%CE%BF%CE%B5 %CE%BB%CE% B5%CE%B3%CE%BA%CF%84%CE%AE%CF% τελευταία επίσκεψη, 20/7/2016.

[14] Wikipedia, Arduino, https://el.wikipedia.org/wiki/Arduino, τελευταία επίσκεψη, 20/7/2016.

[15] Edn Netwrok, USART VS UART: Know the difference, http://www.edn.com/electronics-blogs/embedded-basics/4440395/USART-vs-UART--Know-the-difference, September 2015, τελευταία επίσκεψη, 20/7/2016.

[16] WhatIs.com, USART (Universal Synchronous/Asynchronous Receiver/Transmitter), http://whatis.techtarget.com/definition/USART-Universal-Synchronous-Asynchronous-Receiver-Transmitter, July 2012, τελευταία επίσκεψη, 22/7/2016.

[17] Ardenis Fejzaj, "Ανάπτυξη ενός Μονοφασικού Αναλυτή Ενέργειας με το ATmega328 με δυνατότητα επικοινωνίας με το LabVIEW μέσω Ethernet", Πτυχιακή Εργασία, Τμήμα Ηλεκτρολόγων Μηχανικών Τ.Ε, ΤΕΙ Δυτ. Ελλάδας, 2015.

[18] Α. Καλαντζόπουλος και Ε. Ζυγούρης, "Εισαγωγή στο LabVIEW", Διάλεξη σταπλαίσια του μαθήματος Συστήματα Επεξεργασίας Σημάτων με DSPs του ΔΠΜΣΗλεκτρονική και Επεξεργασία της Πληροφορίας, Πανεπιστήμιο Πατρών, 2010.

[19] Wikipedia, LabVIEW, https://en.wikipedia.org/wiki/LabVIEW, τελευταία επίσκεψη, 5/8/2016.

[20] National Instruments, "LabVIEW Fundamentals", 374029A-01, August 2005.

[21] National Instruments, LABVIEW IDEA EXCHANGE, http:// forums.ni. com/t5/ LabVIEW-Idea-Exchange /Direct-access-to-Plot- Visible-property-on-plot-legend/idi-p/1079931, τελευταία επίσκεψη, 5/8/2016.

[22] Tutorialspoint.com, MySQL Introduction, http://www. tutorialspoint.com/ mysql /mysql-introduction.htm, τελευταία επίσκεψη, 5/8/2016.

[23] Wikipedia, SQL, https://en.wikipedia.org/wiki/SQL, τελευταία επίσκεψη, 5/8/2016.

[24] W3schools.com, SQL Tutorial, http://www.w3schools.com/sql/default.asp, τελευταία επίσκεψη, 7/8/2016.

[25] A. Kalantzopoulos, D. Karageorgopoulos, and E. Zigouris, "A LabVIEW based Remote DSP Laboratory", 5th International Conference on Remote Engineering & Virtual Instrumentation (REV 2008), Dusseldorf, Germany, pp. 445-448, 23-25 June, 2008.

[26] A. Kalantzopoulos, D. Karageorgopoulos and E. Zigouris, "A LabVIEW based Remote DSP Laboratory", International Journal of Online Engineering (iJOE), Vol. 4, SI: REV2008, pp. 36-44, September 2008.