ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΗΛΕΚΤΡΙΚΗΣ ΙΣΧΥΟΣ

Έξυπνο Σπίτι: Έλεγχος και Επιτήρηση Ζεστού Νερού Χρήσης με τον Arduino

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

Ιωάννη Τσακαλάκη

Α.Μ.: 03109796

Επιβλέπων : Νικόλαος Χατζηαργυρίου
Καθηγητής Ε.Μ.Π.

Αθήνα, Μάρτιος 2017

Η σελίδα αυτή είναι σκόπιμα λευκή.

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΗΛΕΚΤΡΙΚΗΣ ΙΣΧΥΟΣ

Έξυπνο Σπίτι: Έλεγχος και Επιτήρηση Ζεστού Νερού Χρήσης με τον Arduino

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

Ιωάννη Τσακαλάκη

Επιβλέπων : Νικόλαος Χατζηαργυρίου
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την:

(Υπογραφή)	*(Υπογραφή)*	*(Υπογραφή)*
.....	.....	.....
Χατζηαργυρίου Νικόλαος Καθηγητής Ε.Μ.Π.	Παπαθανασίου Σταύρος Αναπληρωτής Καθηγητής Ε.Μ.Π.	Γεωργιλάκης Παύλος Επίκουρος Καθηγητής Ε.Μ.Π.

Αθήνα, Μάρτιος 2017

Ευχαριστίες

Θα ήθελα κατ’ αρχάς να ευχαριστήσω τον εισηγητή καθηγητή μου κ. Νικόλαο Χατζηαργυρίου για την ευκαιρία της πραγματοποίησης της εν λόγω εργασίας και την ερευνήτρια Dr. Χριστίνα Παπαδημητρίου για τις πολύτιμες οδηγίες και συμβουλές της και την πολλή όμορφη συνεργασία που είχαμε. Ξεχωριστά, θα ήθελα να ευχαριστήσω τους γονείς και τα αδέρφια μου για την στήριξη τους τα χρόνια των σπουδών μου, όλους τους συμφοιτητές και φίλους μου που μοιραστήκαμε κοινές αγωνίες και πολλές ώρες διαβάσματος στην αναζήτηση της γνώσης και όλα τα πρόσωπα που σιωπηλά ανεχτήκανε την απουσία μου από πολλά γεγονότα και πάντοτε με αγάπη και κατανόηση με δικαιολογούσαν. Ιδιαίτερα θα ήθελα να ευχαριστήσω τους παππούδες και τις γιαγιάδες μου, που ο καθένας με τον τρόπο του, με διδάξανε την αξία της γνώσης και το πόσο συναρπαστικό είναι το ταξίδι αναζήτησης της, όπως και το έζησα. Με τη περάτωση της εργασίας το ταξίδι δε σταματάει, απλά αλλάζει πλέον χώρο.

(Υπογραφή)

.....

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

Περιεχόμενα

Περίληψη

Η παρούσα διπλωματική εργασία καλείται να επικεντρωθεί σε ένα έξυπνο σύστημα ελέγχου και επιτήρησης του ζεστού νερού, ώστε να αποφεύγονται οι υπερκαταναλώσεις ηλεκτρικού ρεύματος και συνάμα οι καταναλωτές να εξυπηρετούνται ετοιμάζοντας ζεστό νερό προς άμεση χρήση. Δηλαδή, με συγκεκριμένο χρονικό πρόγραμμα, που ορίζεται βάσει των αναγκών και του οικονομικού συμφέροντος του καταναλωτή, γίνεται η θέρμανση του νερού. Η παρακολούθηση της δραστηριότητας γίνεται σε πραγματικό χρόνο μέσω του διαδικτύου και με τη χρήση βάσης δεδομένων καταχωρούνται οι δραστηριότητες σε αρχείο, παράγοντας ταυτόχρονα χρήσιμα στατιστικά στοιχεία προς βελτιστοποίηση της διαχείρισης. Η επίτευξη της εργασίας πραγματοποιείται με το μικροελεγκτή ATMega 2560 της εταιρείας ATMEL και του αναπτυξιακού πακέτου υλοποίησης Arduino που προσφέρει η ομώνυμη εταιρεία. Η βάση δεδομένων πραγματοποιείται με SQL Server 2014 της εταιρείας Microsoft. Το πρόγραμμα αλληλεπίδρασης χρήστη – εργασίας, έχει πραγματοποιηθεί σε Visual C# επίσης της εταιρείας Microsoft.

Η δομή της διπλωματικής εργασίας ξεκινάει με εισαγωγικές γνώσεις περί του Internet of Things (IoT) στο 1ο κεφάλαιο, περί των Smart Grid στο 2ο κεφάλαιο και της επικοινωνίας μεταξύ μηχανών (M2M Communication) στο 3ο κεφάλαιο. Το επόμενο κεφάλαιο (4ο κεφάλαιο) επικεντρώνεται στο μικροελεγκτή που χρησιμοποιήθηκε (Arduino Mega), ξεκινώντας με γενική αναφορά στους μικροελεγκτές, στον μικροελεγκτή της εργασίας και στο περιβάλλον ανάπτυξης προγραμμάτων με γενικά παραδείγματα χρήσης του. Το 5ο κεφάλαιο αναφέρεται στη γλώσσα SQL. Το τελευταίο κεφάλαιο (6ο κεφάλαιο) αναφέρεται ειδικά στον τρόπο λειτουργίας της διπλωματικής εργασίας, με σενάρια λειτουργίας.

Λέξεις Κλειδιά: << θέρμανση νερού, ATMega 2560, Arduino, SQL Server, Visual C#, Internet of Things >>

Η σελίδα αυτή είναι σκόπιμα λευκή.

Abstract

Present diploma thesis has the target to focus on a warm water control and monitoring system, to reduce the electrical current overconsumption and serve the consumer needs for instant warm water. Consumers has the ability to program a warm water schedule based on their economic interest. The monitoring of the procedure is coming in real time through the internet and the database use gives the opportunity to consumers to check their activity and at the same time view useful statistic results for an effective management. Thesis development have come true with ATMEL's microcontroller ATMega 2560 and Arduino's development platform. The thesis also uses the database tool SQL Server 2014 of Microsoft. The human machine interface has developed in Microsoft Visual C#.

In the first chapter there is an introduction to the term of "Internet of Things", in the second chapter we meet the Smart Grid term and in the third chapter we come across the topic for "Machine to Machine Communication". The fourth chapter introduces Arduino microcontroller and its development tools. The fifth chapter refers to SQL programming language and the sixth chapter focuses on the way that the diploma thesis project works.

Keywords: << warm water, ATMega 2560, Arduino, SQL Server, Visual C#, Internet of Things >>

Η σελίδα αυτή είναι σκόπιμα λευκή

Εισαγωγή

Τη σύγχρονη εποχή, η ανάπτυξη των ηλεκτρονικών (electronics), των ηλεκτρονικών ισχύος (power electronics) και γενικότερα των επιστημών περιβαλλοντικής τεχνολογίας (environmental science and technology) και τεχνικών διαχείρισης ενέργειας (energy management techniques), έχουν επιφέρει μία ζωτική υποστήριξη στην καταπολέμηση της υπερκατανάλωσης των φυσικών πόρων (overconsumption of natural resources), γεγονότος σημαντικού για τη συνέχιση της ύπαρξης του ανθρώπινου και όχι μόνο, είδους. Η υπερσυγκέντρωση πληθυσμών που παρατηρείται τα τελευταία χρόνια στα μεγάλα αστικά κέντρα, οδηγεί συνεπακόλουθα στην υπερκατανάλωση και στη μεγαλύτερη ανάγκη ενέργειας, όπου σχετίζονται με την απομάκρυνση του σύγχρονου ανθρώπου από τη φύση. Ενδεικτικά, το 72,4% του πληθυσμού των 28 κρατών μελών της Ευρωπαϊκής Ένωσης κατοικεί στα αστικά κέντρα με αυξητικές τάσεις αυτού του ποσοστού. Σύμφωνα με έκθεση του Ευρωπαϊκού Οργανισμού Περιβάλλοντος (Ε.Ο.Π.) οι σύγχρονες πόλεις χαρακτηρίζονται ως ζωντανοί οργανισμοί, που μεταβολίζουν υλικά και πόρους. Οι πόλεις χρειάζονται εισερχόμενες ροές από ενέργεια, ξυλεία, νερό, τρόφιμα, υλικά οικοδομών και υποδομών αλλά και χώρο. Με τη χρήση και το μετασχηματισμό τους, παράγονται εκροές με τη μορφή των αέριων ρύπων, υγρών λυμάτων, στερεών αποβλήτων, με αξιοσημείωτες επιπτώσεις στο περιβάλλον. Το νερό, ως ο σημαντικότερος παράγοντας που εισέρχεται στον ζωντανό οργανισμό της σύγχρονης πόλης, πρέπει να αξιοποιηθεί κατάλληλα για την οικιακή και τη βιομηχανική χρήση. Η περισσότερο ενεργοβόρα διαδικασία και με τεράστιο εύρος χρήσης στους παραπάνω τομείς είναι η παραγωγή ζεστού νερού (warm water production ή water heating). Έτσι, ταυτόχρονα με την τεχνολογική ανάπτυξη πραγματοποιούνται προσπάθειες σωστής χρήσης και κατανάλωσης του νερού, ώστε να μην εξαντλούνται τα φυσικά αποθέματα του, αλλά και ορθής χρήσης της παρεχόμενης ενέργειας που απαιτείται για τη θέρμανση του.

Ανά τους καιρούς, ένας σημαντικός παράγοντας που επηρέαζε πάντα την εξέλιξη του κάθε είδους και δη του ανθρώπινου, ήταν το νερό. Χαρακτηριστικές είναι οι μαζικές μετακινήσεις πληθυσμών από τα αρχαία χρόνια, σε μέρη όπου υπήρχε επάρκεια σε νερό. Πηγή ζωής και πηγή δραστηριοτήτων, αφού πέρα της ζωτικής του άμεσης σημασίας, χρησιμοποιείται σε κύκλους εργασιών απαραιτήτων για την επιβίωση του είδους. Ο αγροτικός τομέας και η βιομηχανία αποτελούν τις σημαντικότερες αναφορές εργασιών. Επίσης, η ατομική υγιεινή και καθαριότητα που έγκειται

στην καθημερινή απαίτηση του νερού, καθιστά εξίσου σημαντική την οικιακή του χρήση.

Πρέπει να σημειωθεί ότι νερό πόσιμο και εκμετάλλευσης είναι το “γλυκό” νερό, το οποίο υπάρχει σε μικρότερα ποσοστά. Αναφορικά, μόνο το 2,5%-3% του νερού της γης είναι “γλυκό», από αυτό το ποσοστό το 70% χρησιμοποιείται στην αγροτική παραγωγή και περίπου το 5% - 10% στην βιομηχανία (κυρίως σε χημικές ουσίες και σε εγκαταστάσεις βιομηχανικής ψύξης). Δυστυχώς, οι σημερινοί ρυθμοί ζωής και η ραγδαία αύξηση του πληθυσμού, επιβάλουν την μαζική κατανάλωση, την κατάχρηση, τη ρύπανση και μόλυνση των φυσικών υδάτινων πόρων. Η UNESCO σε έρευνα της το 2003, υπολογίζει ότι τα επόμενα χρόνια θα μειώνεται σταδιακά η ποσότητα νερού που αντιστοιχεί στον κάθε άνθρωπο μέχρι 30%. Αναλογιζόμενοι τον αριθμό και την εποχή, αυτά τα γεγονότα θα λάβουν χώρα σε βάρος αδύναμων ηπείρων και κρατών και όχι στον ανεπτυγμένο κόσμο. Άξιο λόγου είναι ότι το 40% του συνολικού κόσμου δεν έχει επαρκές νερό για στοιχειώδη υγιεινή. Όλες οι παραπάνω αναφορές αναδεικνύουν την ανάγκη ορθής διαχείρισης του πόσιμου νερού.

Η διαχείριση του νερού γίνεται αντιληπτό πως έχει σκοπό την εξοικονόμηση του και κατ’ επέκταση τη σωστή χρησιμοποίηση ηλεκτρικής ενέργειας όπου απαιτείται για την αξιοποίηση του. Αυτή η εξίσωση, εφόσον συνταχθεί σωστά, οδηγεί στη ελάττωση της κατάχρησης των φυσικών υδάτινων πόρων και σε εξοικονόμηση χρημάτων από τη μεριά του καταναλωτή. Προκύπτουν τομείς μελέτης που κατευθύνονται στην “έξυπνη διαχείριση του νερού” (smart water management) ή πιο απλά, όπως συναντάται στην αγγλική βιβλιογραφία, σαν “smart water”.

Στις σύγχρονες πλέον πόλεις, που κατοικούνται όλο και από περισσότερο πληθυσμό, φαντάζει απαραίτητη η εφαρμογή έξυπνων λύσεων για την αποφυγή υπερκαταναλώσεων πόρων (είτε ενεργειακών, είτε φυσικών πόρων) τα οποία θα οδηγούσαν πιο γρήγορα στην εξάντληση τους από τη γη. Γι’ αυτό κερδίζει έδαφος όλο και περισσότερο ο όρος “έξυπνη πόλη” (smart city). Οι ανά τον κόσμο μεγαλουπόλεις την δεκαετία που διανύεται (2010-2020), υπολογίζεται να έχουν ξοδέψει σε έρευνα, μέσω των κυβερνήσεων τους, ποσό της τάξεως των 110 δισεκατομμυρίων δολαρίων σε μελέτες υποδομών, όπως σε έξυπνους μετρητές, το smart grid, ενεργειακά αποδοτικά κτήρια με ανάλυση δεδομένων (άρθρο waterworld.com σε έρευνα της Navigant Research).

Οι έξι τομείς έρευνας που είναι απαραίτητο να διερευνηθούν εις βάθος και κατάλληλα να ενοποιηθούν, ώστε μία πόλη να γίνει περισσότερο “έξυπνη” και να είναι συνάμα κατάλληλη για κατοίκηση είναι: 1) έξυπνη διαχείριση

ενέργειας (smart energy), 2) έξυπνη ολοκλήρωση (smart integration), 3) έξυπνες δημόσιες παροχές (smart public services), 4) έξυπνες μετακινήσεις (smart mobility) 5) έξυπνα κτήρια (smart buildings) και 6) η έξυπνη διαχείριση του νερού (smart water).

Οι έξι τομείς προς ενοποίηση, μίας έξυπνης πόλης

Λόγω της άμεσης σχέσης με τη φύση και με τον άνθρωπο, το σημαντικότερο κομμάτι της αλυσίδας, είναι η ορθή διαχείριση του νερού. Σωστή διαχείριση του νερού σημειώνεται όχι μόνον κατά την κατανάλωση, αλλά και κατά τη διαδικασία οδήγησης των αποβλήτων του. Οπότε αρχικά, η υποδομή πρέπει να υλοποιείται με σύγχρονα κατασκευαστικά – τεχνικά πρότυπα, ώστε να διασφαλιστούν μηδενικές απώλειες κατά την μεταφορά στους καταναλωτές αλλά και σωστή οδήγηση των υδάτινων αποβλήτων ώστε να μην έρχονται σε επαφή με υγιές περιβάλλον αλλά και σωστή επεξεργασία τους ώστε να μπορούν να αξιοποιηθούν στη συνέχεια σε περιβαλλοντικές δραστηριότητες. Με την υλοποίηση της ιδανικής υποδομής, ακολουθεί η διαχείριση, όπου η τεχνολογία και οι επιστημονικές μέθοδοι συναντώνται για την πραγματοποίηση της.

1. Internet of Things (IoT)

Το Internet of Things (για συντομία IoT), αποτελεί μία νέα τάση και χωρίς υπερβολή μπορεί να χαρακτηριστεί το παρόν και το μέλλον της τεχνολογίας. Είναι το “πιο ραγδαία αναπτυσσόμενο θέμα” σύμφωνα με το περιοδικό “Forbes” και με μία έρευνα που διενήργησε εν έτει 2014, το 85- 90% των ερωτηθέντων, δεν γνώριζε κάτι σχετικό με τον όρο.

Πιο συγκεκριμένα, ο όρος αναφέρεται “*στο δίκτυο που δημιουργούν φυσικές συσκευές, οχήματα, κτήρια και άλλα αντικείμενα, στα οποία υπάρχουν ενσωματωμένα ηλεκτρονικά, λογισμικό, αισθητήρες και συνδεσιμότητα δικτύου και έτσι παρέχεται η δυνατότητα συλλογής και μετάδοσης δεδομένων από αυτά τα αντικείμενα*” (''Internet of Things Global Standards Initiative'', ITU, 26 June 2015). Πρόγονος του IoT μπορεί να χαρακτηριστεί το τραπεζικό ATM (Automated Teller Machine), όπου σαν τεχνολογία παρουσιάστηκε το 1974 (η διασυνδεδεμένη έκδοση με την τράπεζα).

Η μαζική εξάπλωση του IoT πραγματοποιήθηκε με την κυκλοφορία των έξυπνων τηλεφώνων (SmartPhones) και ήδη από το 2008 οι διασυνδεδεμένες συσκευές ξεπερνούν τον αριθμό των διασυνδεδεμένων χρηστών. Σύμφωνα με την Google για το 2016 υπολογίζονται οι διασυνδεδεμένες συσκευές σε περίπου 5 δισεκατομμύρια και το 2020 εκτιμάται ότι ο αριθμός αυτός θα φτάσει τις 50 δισεκατομμύρια (6,1 δισεκατομμύρια θα είναι μόνο τα έξυπνα τηλέφωνα).

Εικόνα 1: Το Internet of Things στην καθημερινότητά

Άξια αναφοράς αποτελεί επίσης η εκτίμηση της General Electrics ότι σαν τεχνολογία το IoT θα προσθέσει στο παγκόσμιο ΑΕΠ 10-15 τρισεκατομμύρια δολάρια τα επόμενα 20 χρόνια.

1.1. Εφαρμογές

Η ιδέα της εν λόγω τεχνολογίας δεν είναι κάτι νέο. Ήδη από τα μέσα της δεκαετίας του 1980 οι εταιρείες πληροφορικής συζητούσαν την ιδέα στα διάφορα συνέδρια της εποχής. Ο πυρήνας της ιδέας ήταν απλός, να συνδεθούν οι συσκευές στο διαδίκτυο και μέσω των εφαρμογών να συνομιλούν μεταξύ τους (Επικοινωνία μεταξύ μηχανών ή Machine to Machine Communication).

Πληθώρα εφαρμογών πραγματοποιείται σήμερα με το IoT. Αναφορικά:

- **Επιτήρηση του Περιβάλλοντος:** Χρησιμοποιούνται αισθητήρες για τη συλλογή μετρήσεων, βοηθώντας στην προστασία του περιβάλλοντος. Ενδεικτικές χρήσεις, για την ποιότητα του αέρα, για την ποιότητα του νερού, για την κατάσταση της ατμόσφαιρας, την κατάσταση του εδάφους, για επιτήρηση της φυσικής ζωής και προστασία της φύσης από επικίνδυνες καταστάσεις. Στις επικίνδυνες καταστάσεις, περιλαμβάνεται το γεγονός ότι με διάφορες στατιστικές μεθόδους από τη συλλογή δεδομένων που έχει προηγηθεί, μπορεί να γίνει έγκαιρη πρόβλεψη για σεισμούς ή ακόμα και για τσουνάμι.

- **Διαχείριση Υποδομών:** Πραγματοποιείται επιτήρηση και έλεγχος υπόγειων και επαρχιακών υποδομών, όπως γεφυρών, συρμών, απομακρυσμένων αιολικών πάρκων. Το IoT μπορεί και διαχειρίζεται γεγονότα και αλλαγές στην κατάσταση των εν λόγω υποδομών διασφαλίζοντας την ασφάλεια τους και μειώνοντας όποιο ρίσκο μπορεί να προκύψει λόγω της απομακρυσμένης ή δύσβατης τοποθεσίας που βρίσκονται. Επίσης αναλαμβάνει τις προγραμματισμένες συντηρήσεις ή μέσω της εποπτείας προλαμβάνει για όποιες έκτακτες επιδιορθώσεις. Έτσι η ποιότητα των εργασιών παραμένει υψηλή, μειώνοντας κόστη λειτουργίας χάρη στην έγκαιρη και σωστή επέμβαση και η διαχείριση γίνεται με αποτελεσματικό τρόπο.

- **Βιομηχανική Παραγωγή:** Ο δικτυακός έλεγχος και η διαχείριση της γραμμής παραγωγής μέσω του IoT αποτελεί πλέον πλεονέκτημα για τη βιομηχανία που λειτουργεί κατ' αυτόν τον τρόπο. Η "έξυπνη παραγωγή" επιτυγχάνεται μέσω ευφυών συστημάτων και αισθητήρων

που διοχετεύονται με μεγάλο ρυθμό στην αγορά προς χρήση από τις βιομηχανίες. Παράλληλα εξασφαλίζονται εργασίες όπως προληπτική ή προγραμματισμένη συντήρηση και στατιστική αξιολόγηση της παραγωγής για περαιτέρω βελτίωση της. Η “έξυπνη διαχείριση” μίας βιομηχανίας με το IoT πλέον αναπτύσσεται και σε πιο καίρια και δαπανηρά ζητήματα όπως η ενεργειακή βελτιστοποίηση σε πραγματικό χρόνο (real time energy optimization) εισάγοντας τον όρο του Smart Grid. Το βιομηχανικό πρότυπο του IoT είναι γνωστό ως Industrial Internet of Things (IIoT).Τα δεδομένα που εξάγονται κατά την παραγωγή είναι γνωστά ως “big data” (Lee, Jay, 2015,. Industrial Big Data. China: Mechanical Industry Press. ISBN 978-7-111-50624-9) και σύμφωνα με μελέτες, μπορούν να συνεισφέρουν, με σωστή χρήση τους, ως 30% στην μείωση του κόστους συντήρησης *(*Daugherty, Paul; Negm, Walid; Banerjee, Prith; Alter, Allan. "Driving Unconventional Growth through the Industrial Internet of Things", Accenture, 17 March 2016*).*

- **Ιατρική και Συστήματα Φροντίδας Υγείας:** Οι συσκευές IoT μπορούν να χρησιμοποιούνται πλέον, σαν μία ώριμη τεχνολογία, για εξ’ αποστάσεως επιτήρηση της υγείας και για σήμανση επειγόντων περιστατικών, ειδικά των ηλικιακά ευπαθών ομάδων. Οι συσκευές και τα συστήματα της κατηγορίας έχουν ένα μεγάλο εύρος χρήσης, από επίβλεψη των αρτηριών και της καρδιάς ως προχωρημένες τεχνολογικά συσκευές με δυνατότητα επίβλεψης μοσχευμάτων. Κατ’ αυτό τον τρόπο, χρόνια ασθενείς μπορούν να παρακολουθούν και να διαχειρίζονται την ιατροφαρμακευτική τους περίθαλψη, που δεν γίνεται πλέον κατ’ ανάγκη από νοσηλευτές. Γενικά είναι τέτοια η αποδοχή και από το ευρύτερο κοινό όπου χρησιμοποιούνται και φορητοί καρδιογράφοι και παλμογράφοι είτε μέσω εφαρμογών στα “έξυπνα τηλέφωνα” (SmartPhones), είτε σαν ανεξάρτητες συσκευές, ακόμα και κατά την άθληση.

- **Κατασκευές και Οικιακοί Αυτοματισμοί:** Μηχανολογικά, ηλεκτρολογικά και ηλεκτρονικά συστήματα που χρησιμοποιούνται σε πληθώρα κτηρίων (βιομηχανίες, δημόσια κτήρια, οικίες, ινστιτούτα κτλ) μπορούν να τα επιβλέπουν και να τα ελέγχουν μέσω των IoT συσκευών.

- **Μεταφορές:** Το IoT μπορεί να συμβάλει στην ανάπτυξη συστημάτων τηλεπικοινωνιών, ελέγχου και επεξεργασίας πληροφοριών στον τομέα των μεταφορών. Η δυναμική διαδραστικότητα μεταξύ των μέσων μεταφορών (είτε ιδιωτικών, είτε δημοσίων) μέσω των εξαρτημάτων που εισάγονται για την επίτευξη της ένταξης στο σύστημα, προσδίδουν

ευκολίες επικοινωνίας είτε μεταξύ οχήματος και οδηγού, είτε μεταξύ οδηγού και περιβάλλοντος. Επί παραδείγματι ο έξυπνος έλεγχος της κίνησης και η πρόταση εναλλακτικών αρτηριών, τα έξυπνα συστήματα παρκαρίσματος, τα έξυπνα συστήματα πληρωμής διοδίων ή και μεγαλύτερης κλίμακας υλοποιήσεις όπως η διαχείριση στόλου οχημάτων, συστήματα ενημέρωσης καυσίμου, ο απόλυτος οδηγικός έλεγχος αυτοκινήτου για πρόληψη ατυχήματος και συστήματα κάλυψης οδικής βοήθειας και ασφάλειας. Ένα μακροπρόθεσμο σχέδιο για τις μεγάλες πόλεις, που πιλοτικά εφαρμόζεται σε μικρότερες, είναι δημόσιες συγκοινωνίες χωρίς οδηγό (αυτοκινούμενες δημόσιες συγκοινωνίες).

- **Ανάπτυξη Μεγάλης Κλίμακας:** Μεγαλεπήβολα σχέδια υπάρχουν για άμεση και για μακροπρόθεσμη υλοποίηση, στηριζόμενα στο IoT. Αυτά τα σχέδια έχουν πρωταρχικό στόχο την ανάπτυξη “έξυπνων πόλεων” όπου η συστηματοποίηση της λειτουργίας τους θα βασίζεται στην “έξυπνη διαχείριση” τους. Η πρώτη πόλη που ήδη κατασκευάζεται να λειτουργεί εξ’ ολοκλήρου κατ’ αυτόν τον τρόπο είναι η πόλη Σόνγκντο (Songdo, 65 χιλιόμετρα νοτιοδυτικά της Σεούλ, πόλη αποκαλούμενη σαν “περιοχή διεθνούς εμπορίου”) της Νότιας Κορέας, η οποία είναι μία πλήρως διασυνδεδεμένη πόλη με εξοπλισμό και αισθητήρια που βασίζονται στο IoT και η ανάπτυξη του εγχειρήματος έχει κοστίσει μέχρι στιγμής περί τα 40 δισεκατομμύρια δολάρια. Σχολεία, νοσοκομεία, διαμερίσματα, γραφεία κτλ. έχουν σχεδιαστεί να λειτουργούν και να είναι ενταγμένα στο σχέδιο της “έξυπνης πόλης”, όπου συλλέγονται δεδομένα των κτηρίων και των λειτουργιών που σχετίζονται με αυτά. Έτσι, ζητήματα καθαριότητας, καταλληλότητας χρήσης, προληπτικής συντήρησης και φιλικότητας προς το περιβάλλον κτλ. πλέον δε θα τίθενται. Πιο απτό παράδειγμα είναι η έξυπνη διασύνδεση των σκαφών πλωτών μεταφορών της Νέας Υόρκης. Κατ’ αυτόν τον τρόπο διασφαλίζεται η ασφάλεια, η σωστή τήρηση των δρομολογίων, η σωστή ενεργειακή διαχείριση των πλωτών μέσων και η ψηφιοποίηση των εισιτηρίων.

- **Καταναλωτικές Εφαρμογές:** Η μεγαλύτερη μερίδα IoT συσκευών αναμφίβολα προορίζεται για μαζική καταναλωτική χρήση. Εφαρμογές όπως τα διασυνδεδεμένα αμάξια, εφαρμογές ψυχαγωγίας, τα έξυπνα σπίτια, εφαρμογές υγείας προσδίδουν νέα πεδία στην κατανάλωση.

1.2. Internet of Things και Ηθική

Ό, τι έχει ήδη αναφερθεί, δε συνεπάγεται αυτόματα ότι είναι και ηθικά αποδεκτό, παρά τις ευκολίες και τη χρηστικότητα που παρέχουν οι νέες τεχνολογίες. Πολλές φωνές διατυπώνουν την άποψη περί της ανάπτυξης νέων τεχνικών προγραμματισμού και δημιουργίας νέων συστημάτων, όπου θα βασίζονται στον ηθικό αντίκτυπο, γιατί πλέον με τη ραγδαία εξάπλωση των SmartPhones και των IoT συσκευών συλλέγονται και αξιοποιούνται δεδομένα από τις εταιρείες με σκοπό περαιτέρω κέρδος. Επίσης, λόγω ακριβώς ότι όλο και περισσότερα προσωπικά δεδομένα εμπιστεύονται από τους χρήστες στις εφαρμογές των IoT συσκευών, τίθεται το ζήτημα της ασφάλειας, καθώς δεν πρέπει να λησμονείται το γεγονός ότι η πλειοψηφία των χρηστών δεν είναι μηχανικοί ή ανώτερου μορφωτικού επιπέδου και χρησιμοποιούν με “εν λευκό” όρους την νέα τεχνολογία αδιαφορώντας για την ασφάλεια των προσωπικών δεδομένων και την ιδιωτικότητα τους. Αντιθέτως, οι νέες τάσεις μέσω της τεχνολογίας πολλές φορές υποσυνείδητα επιβάλλουν και τη δημόσια προβολή, καθιστώντας όλο και περισσότερους ανθρώπους σε “διαφημίσεις”, κατορθώνοντας έτσι οι αντίστοιχες διαφημιζόμενες εταιρείες με το ελάχιστο κόστος να έχουν μέγιστο κέρδος.

1.3. Σύνδεση με τη Διπλωματική Εργασία

Το Internet of Things αποτελεί μία υποδομή παγκοσμίου δικτύου, που συνδέει φυσικά και ψηφιακά αντικείμενα χρησιμοποιώντας την τεχνολογία cloud, την απεικόνιση δεδομένων και την επικοινωνία δικτύων. Επιτρέπει σε συσκευές να επικοινωνούν μεταξύ τους, να έχουν πρόσβαση στην πληροφορία μέσω διαδικτύου, να αποθηκεύουν και να ανακτούν δεδομένα και να είναι διαδραστικές με τους χρήστες, δημιουργώντας έξυπνα περιβάλλοντα για αυτό το σκοπό.

Μία τέτοια συσκευή είναι και ο Arduino με τα περιφερειακά του στην παρούσα διπλωματική εργασία, καθώς δύναται να έχει πρόσβαση στην παγκόσμια κοινότητα συσκευών. Ο ρόλος του στην εργασία μπορεί να θεωρηθεί ότι είναι ενός web server. Αποτελεί ένα ψηφιακό αντικείμενο έτοιμο προς παροχή ή ανταλλαγή πληροφοριών με άλλα ψηφιακά αντικείμενα και αλληλεπίδραση με φυσικά αντικείμενα που είναι συνδεδεμένα στην παγκόσμια αυτή κοινότητα. Οι πληροφορίες που συλλέγει με τα αισθητήρια του αποθηκεύονται στην βάση δεδομένων και από εκεί είναι διαθέσιμα προς επεξεργασία από τον χρήστη μέσω της μονάδος του ηλεκτρονικού του

υπολογιστή. Αναδεικνύοντας το γεγονός της επιτήρησης που έχουν οι συσκευές Internet of Things, υλοποιείται οθόνη παρακολούθησης παρέχοντας άμεσα κάποιες ενδεικτικές πληροφορίες σχετικές της εργασίας όπως την κατάσταση των ρελέ, δηλαδή αν είναι σε κατάσταση λειτουργίας ή όχι και τη θερμοκρασία που έχει το νερό ανά πάσα στιγμή.

2. Smart Grids (Έξυπνα Ηλεκτρικά Δίκτυα)

Ξεφεύγοντας από την πεζή καταναλωτική πλευρά του IoT και εστιάζοντας στην ενέργεια, συναντάται ο όρος “Smart Grid” (έξυπνα ή ευφυή ηλεκτρικά δίκτυα). Το Smart Grid γεννήθηκε με το IoT για να καλύψει το κενό που υπήρχε σχετικά με την “έξυπνη” κατανάλωση ενέργειας. Έτσι, μπορούν και *“περικλείονται μέσα σε ένα δικτύωμα ποικιλία λειτουργιών και ενεργειακών μετρήσεων, μέσω έξυπνων μετρητών, έξυπνων εφαρμογών παρακολούθησης, ανανεώσιμων πηγών ενέργειας και πηγών απόδοσης ενέργειας*” (Federal Energy Regulatory Commission Assessment of Demand Response & Advanced Metering", United States Federal Energy Regulatory Commission, United States Federal Energy Regulatory Commission).

Από δημιουργίας του υπάρχοντος ηλεκτρικού δικτύου, ελάχιστες τροποποιήσεις έχουν πραγματοποιηθεί, γεγονός που καθιστά επιτακτική την ανάγκη μέσω της τεχνολογικής εξέλιξης να βελτιωθούν, ή και να εξαλειφθούν, οι αδυναμίες που παρουσιάζονται. Τέτοιες αδυναμίες επιγραμματικά είναι:

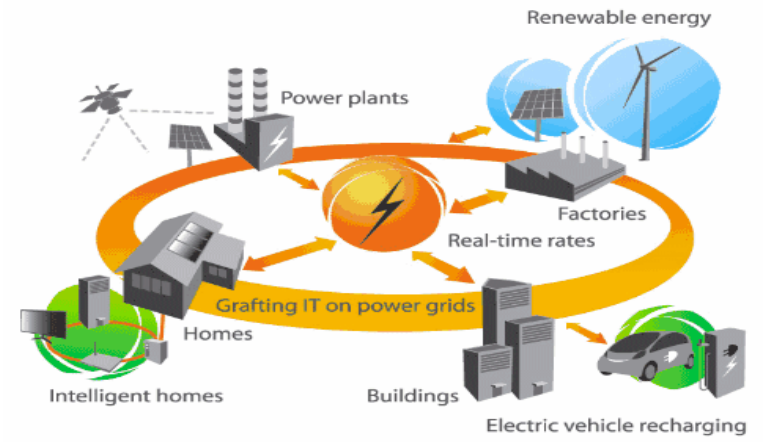
- Οι απρόσμενες διακοπές παροχής ρεύματος για απαγορευτικό χρονικό διάστημα, σε περιοχές όπου άμεση αποκατάσταση δεν παρέχεται. Έτσι κρίνεται αναγκαία η παρακολούθηση του δικτύου για άμεσο εντοπισμό βλαβών, Κατ’ αυτόν τον τρόπο οι εταιρίες ηλεκτρισμού γλυτώνουν το κόστος της ζημίας που θα είχαν οι ίδιες, όσο και οι πελάτες της.

- Οι νέες καταναλωτικές ανάγκες τόσο από τους ιδιώτες, όσο και από τις επιχειρήσεις, απαιτεί πλέον να υπάρχει μία δυναμική σχέση μεταξύ καταναλωτών και παρόχων ενέργειας, ώστε η ποσότητα παραγόμενης ενέργειας να μπορεί να καλύπτει τη μέγιστη ζήτηση παρέχοντας ταυτόχρονα και ένα πλεόνασμα ασφαλείας.

- Η όλο και αυξημένη γεωγραφική διασπορά των μονάδων παραγωγής ενέργειας που βασίζεται σε ανανεώσιμες πηγές, προκαλεί δυσκολίες στην ένταξη τους στο υπάρχον δίκτυο ηλεκτρικής ενέργειας.

- Η γνώση σε πραγματικό χρόνο της προσφοράς και ζήτησης της ηλεκτρικής ενέργειας.

- Έγκαιρη ενημέρωση για κλοπή ενέργειας, ηλεκτρολογικού εξοπλισμού κτλ.



Εικόνα 2: Τυπική διάταξη ενός μελλοντικού Smart Grid

Τις προαναφερθείσες αδυναμίες πλέον καλείται να καλύψει το Smart Grid, με την δυνατότητα ανταλλαγής δεδομένων σε πραγματικό χρόνο, καλύπτοντας όλη την έκταση του δικτύου ηλεκτρικής ενέργειας. Τα συστήματα έξυπνων μετρητών και έξυπνης παρακολούθησης που αναφέρθηκαν ανωτέρω, αποτελούν τα χαρακτηριστικότερα ζητήματα για την υλοποίηση ενός αξιόπιστου Smart Grid.

Σύμφωνα με τον μη κερδοσκοπικό οργανισμό με εδρεύει στις ΗΠΑ, τον ERPI (Electric Power Research Institute), το Smart Grid έχει κάποια χαρακτηριστικά:

- Η αμφίπλευρη ροή πληροφορίας και η αξιοποίηση της σε πραγματικό χρόνο
- Με την ροή πληροφορίας που παρέχεται μέσω έξυπνων μετρητών, αισθητήρων και συστημάτων παρακολούθησης, καθίσταται δυνατή η ευέλικτη τιμολόγηση, προς όφελος των καταναλωτών.
- Αυτόματη και άμεση αποκατάσταση βλαβών μέση της λειτουργίας της προβλεπτικότητας, χωρίς ανθρώπινη παρέμβαση. (Self healing)
- Βέλτιστη χρησιμοποίηση του Συστήματος Ηλεκτρικής Ενέργειας, καθώς χρησιμοποιείται βέλτιστα η ροή ισχύος του δικτύου που προέρχεται από τις ήδη υπάρχουσες πηγές ενέργειας, για την κάλυψη των καταναλωτικών αναγκών. Η κάλυψη μπορεί να επιτευχθεί “έξυπνα” και από την διείσδυση στο δίκτυο των Ανανεώσιμων Πηγών Ενέργειας.
- Η διείσδυση των Ανανεώσιμων Πηγών Ενέργειας μπορεί να φτάσει σε μέγιστο ποσοστό, με ελάχιστους κινδύνους αξιοπιστίας δικτύου,

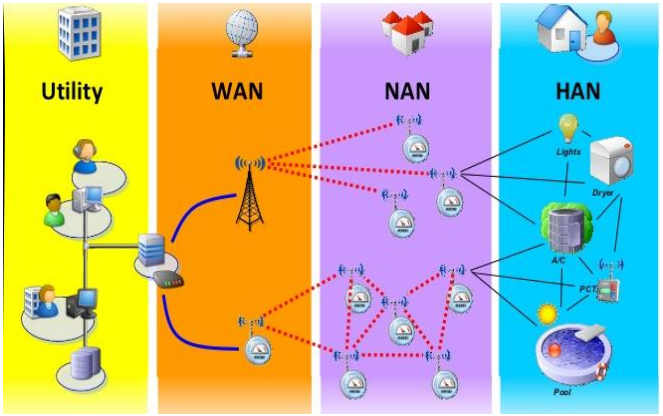
δυνατότητα διατήρησης ευστάθειας, δυνατότητα αποθήκευσης και ελέγχου της ζήτησης ενέργειας.

2.1. Έξυπνη (ή Ευφυής) Παρακολούθηση

Ο σκοπός της έξυπνης παρακολούθησης είναι η πρόληψη και η άμεση αποκατάσταση βλαβών, μέσω της επιτήρησης του Smart Grid. Αισθητήρες τεχνολογίας IoT συλλέγουν δεδομένα, όπως μετρήσεις τάσης, ρεύματος, ισχύος, συντελεστή ισχύος, ανά τακτά και καίρια σημεία του δικτύου και έτσι κατορθώνεται να εξάγονται χρήσιμα συμπεράσματα για την κατάσταση του δικτύου αλλά και την ποιότητα ενέργειας. Οι αισθητήρες λόγω του μικρού κόστους και της εύκολης εγκατάστασης που πλέον διαθέτουν, αποτελούν την αποτελεσματικότερη λύση επιτήρησης. Επιπρόσθετα σε ένα δίκτυο, τοποθετούνται κάμερες καταγραφής αλλά και δίκτυα τηλεπικοινωνιών για τη μετάδοση των δεδομένων. Οι κάμερες καταγραφής επιφορτίζονται με τη διαδικασία φύλαξης του εξοπλισμού και αποφυγής κλοπής ρεύματος.

2.2. Δίκτυα Επικοινωνίας

Γίνεται αντιληπτό πως ένας ολοένα και αυξανόμενος αριθμός συσκευών, με τη λειτουργία και τη διασύνδεση του, πρέπει να υποστηρίζεται από μία υποδομή επικοινωνιών ώστε να ικανοποιούνται στο έπακρο οι απαιτήσεις επίδοσης. Προς αυτήν την κατεύθυνση είναι οργανωμένη μία ιεραρχική υποδομή, όπου με βάση το εύρος κάλυψης διαιρούνται σε επιμέρους υποδίκτυα. Έτσι τα δίκτυα επικοινωνιών ταξινομούνται στα “Μεγάλης Περιοχής Δίκτυα” (Wide Area Networks- WAN), στα “Δίκτυα Περιοχής Πεδίου” (Field Area Networks- FAN ή NAN- Neighborhood Area NetWorks) και στα “Οικιακά Δίκτυα” (Home Area Networks- HAN).



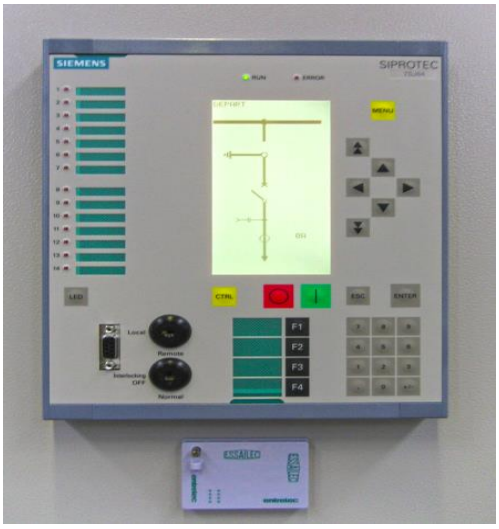
Εικόνα 3: Τα δίκτυα επικοινωνίας

2.2.1. Wide Area NetWorks- WAN

Η σύνδεση των κατανεμημένων και των μικρότερων δικτύων που εξυπηρετούν τα συστήματα ηλεκτρικής ενέργειας σε διάφορες θέσεις, γίνεται επάνω στην “κεντρική αρτηρία”, το δίκτυο WAN. Μέσω του WAN μεταφέρονται αμφίδρομα μεταξύ κέντρων ελέγχου και ηλεκτρικών συσκευών, μετρήσεις πραγματικού χρόνου που λαμβάνονται από τις ηλεκτρικές συσκευές και εντολές λειτουργίας από τα κέντρα ελέγχου.

Για βέλτιστη ανάγνωση των συνθηκών σε ένα WAN, χρειάζονται πληροφορίες σχετικά με την κατάσταση του ηλεκτρικού δικτύου (Regional Transmission Operator- RTO). Η ανάγνωση αυτή επιτυγχάνεται με τη χρήση στους υποσταθμούς γρήγορων, χρονικά σφραγισμένων και πραγματικού χρόνου πληροφοριών για το σύστημα, που προέρχονται από ηλεκτρικούς αισθητήρες (μονάδες PMU - Phasor Measurement Unit). Οι συσκευές PMU καταγράφουν πληροφορίες για το διάνυσμα της συχνότητας του ρεύματος και της τάσης. Οι πληροφορίες κατόπιν χρησιμοποιούνται από τα συστήματα διαχείρισης ενέργειας στα κέντρα ελέγχου για βελτιωμένη εκτίμηση, παρακολούθηση, έλεγχο και προστασία της κατάστασης λειτουργίας.

Σαν “κεντρική αρτηρία” τα WAN συντελούν στην επικοινωνία μεταξύ των IEDs (Intelligent Electronic Devices- Έξυπνες Ηλεκτρονικές Συσκευές**)** και των κέντρων ελέγχου. Τα IEDs εγκαθίστανται κατά μήκος των γραμμών μεταφοράς και στους υποσταθμούς, για να καταγράφουν τις πληροφορίες από τα συστήματα SCADA (Supervisory Control And Data Acquisition- Εποπτεία, Έλεγχος και Συλλογή Δεδομένων) και να ενεργούν βάσει εντολών ελέγχου και προστασίας που λαμβάνονται από τα κέντρα ελέγχου.



Εικόνα 4: Παράδειγμα IED, ψηφιακό ρελέ προστασίας με μικροεπεξεργαστή

Στην παρούσα χρονική στιγμή, οι υποσταθμοί επικοινωνούν με τα κέντρα ελέγχου χρησιμοποιώντας τηλεφωνικές ή μικροκυματικές ζεύξεις. Προφανώς, υπό την απουσία ενός υψηλής ταχύτητας δικτύου, τα ψηφιακά δεδομένα από τις συσκευές PMU περιορίζονται εντός των υποσταθμών και δεν μπορούν να χρησιμοποιηθούν αποτελεσματικά από τα κέντρα ελέγχου, γεγονός που σημειώνει την ανάγκη ενός υψηλού επιπέδου και εύρους ζώνης δικτύου WAN.

2.2.2. Field Area Networks- FAN ή NAN (Neighborhood Area Networks)

Τα δίκτυα FAN συνιστούν τη μονάδα επικοινωνίας για τα συστήματα διανομής ηλεκτρικής ενέργειας. Οι κύριες πηγές πληροφοριών προς επιτήρηση και έλεγχο από τα DMS (Distribution Management System- Κατανεμημένα Συστήματα Διαχείρισης) προς τα κέντρα ελέγχου είναι οι ηλεκτρικοί αισθητήρες στα τροφοδοτικά και τους μετασχηματιστές διανομής, τα IEDs (έξυπνες ηλεκτρικές συσκευές) που εκτελούν εντολές ελέγχου από τα συστήματα DMS, οι κατανεμημένοι ενεργειακοί πόροι (Distributed Energy Resources –DER) στα συστήματα διανομής, οι σταθμοί φόρτισης ηλεκτρικών οχημάτων και οι έξυπνοι μετρητές στις εγκαταστάσεις των πελατών. Οι εφαρμογές του τομέα διανομής ηλεκτρικής ενέργειας χρησιμοποιούν τα δίκτυα FAN για να ανταλλάσουν πληροφορίες. Δύο κατηγορίες εφαρμογών συναντώνται, εφαρμογές με “βάση τον τομέα”, που είναι αυτές που σχετίζονται με τις γραμμές μεταφοράς, τους αισθητήρες, τους ρυθμιστές τάσης κτλ και οι εφαρμογές με “βάση τους καταναλωτές”, όπου σχετίζονται με τον τελικό καταναλωτή, όπως οικίες, κτήρια, βιομηχανίες κτλ.

Οι εν λόγω κατηγορίες εφαρμογών έχουν διαφορετικές κρίσιμες απαιτήσεις. Οι “βασισμένες στους καταναλωτές” εφαρμογές (υποδομές προηγμένων μετρήσεων, μηχανισμοί διαχείρισης της ζήτησης κτλ) απαιτούν ένα ευέλικτα επεκτάσιμο δίκτυο επικοινωνίας μεταξύ του παρόχου ενέργειας και του καταναλωτή, γεγονός που θα επέτρεπε την προσθήκη περισσότερων εφαρμογών και καταναλωτών στο μέλλον, ενώ η ευαισθησία ως προς το χρόνο (γήρανση εξοπλισμού στο χρόνο) αποτελεί δευτερεύον θέμα. Οι “βασισμένες στον τομέα” εφαρμογές (εφαρμογές SCADA, παρακολούθηση και έλεγχος των DER κτλ) έχουν μεγαλύτερη ευαισθησία ως προς το χρόνο.

2.2.3. Home Area Networks (HAN)

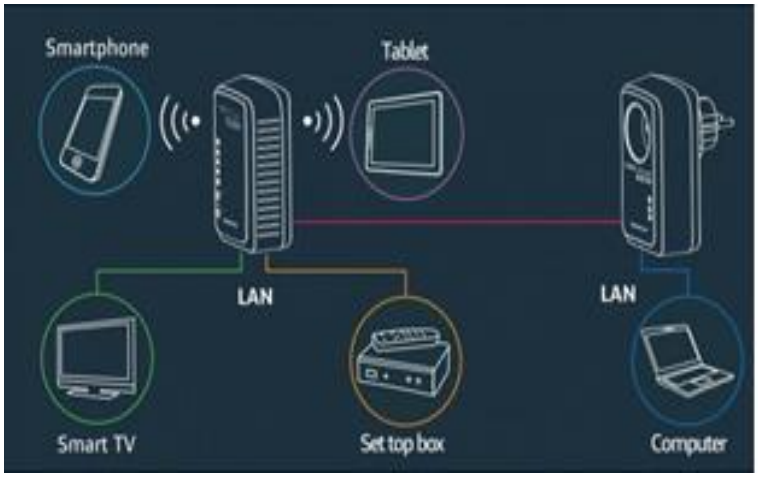
Τα οικιακά δίκτυα λόγω του ότι είναι πιο “εμφανή” προς τον καταναλωτή είναι και πιο γνωστά προς το ευρύ κοινό. Λαμβάνουν χώρα εντός του χώρου του καταναλωτή για την παρακολούθηση και τον έλεγχο των “έξυπνων διασυνδεδεμένων συσκευών”. Λειτουργίες όπως ή διαχείριση της απόκρισης ζήτησης ενέργειας και των προηγμένων έξυπνων μετρήσεων, πραγματοποιούνται μόνον εφόσον προηγηθεί οι οικιακή δικτύωση. Με την επικράτηση του ενσύρματου προτύπου Ethernet και του ασύρματου προτύπου Wi-Fi 802.11, έγινε πιο εύκολο και απλό να αναπτυχθούν όλες οι έξυπνες συσκευές και οι λειτουργίες τους.

Οι τεχνικές απαιτήσεις ενός οικιακού δικτύου είναι διαφορετικές από των προηγούμενων δικτύων, καθώς υπάρχει ανάγκη αδιάλειπτης επικοινωνίας. Συσκευές όπως θερμοστάτες, συστήματα θέρμανσης, ψύξης και εξαερισμού, συστήματα οικιακού αυτοματισμού και διαχείρισης ενέργειας, μετρητές ηλεκτρικού ρεύματος και υδροδότησης, διασυνδέονται και επικοινωνούν επιτρέποντας στους καταναλωτές την βέλτιστη διαχείριση και κατανάλωση ενέργειας.

Τα πρότυπα δικτύωσης όπως έχουν ήδη αναφερθεί είναι στις ενσύρματες επικοινωνίες το Ethernet και στις ενσύρματες το WiFi 802.11.

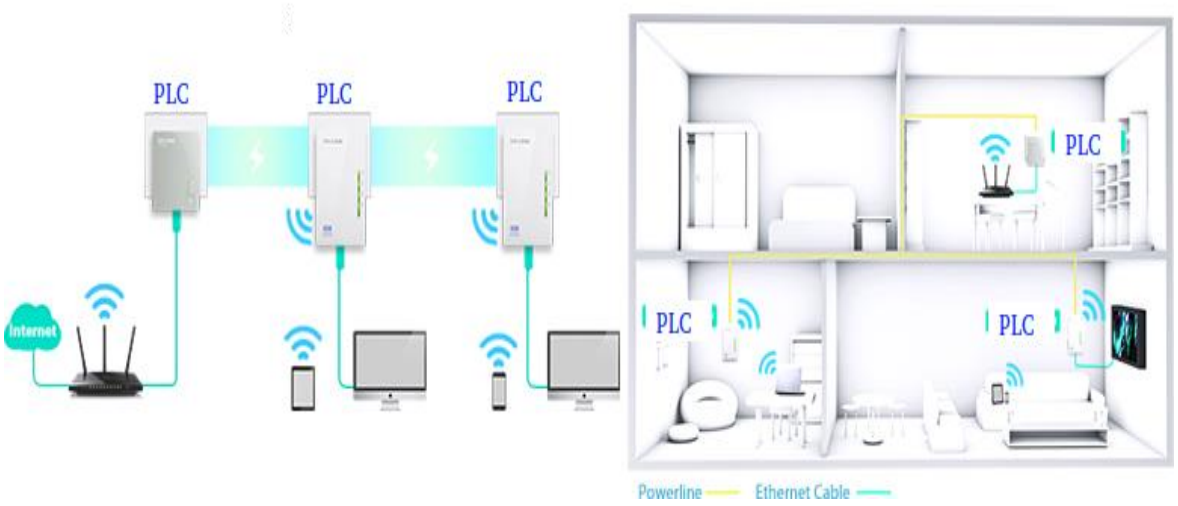
Στην περίπτωση της ενσύρματης επικοινωνίας, το Ethernet έχει το πλεονέκτημα ότι μπορεί να χρησιμοποιήσει την υπάρχουσα καλωδίωση αντί να προηγηθεί εγκατάσταση νέου δικτύου, χρησιμοποιώντας τις υπάρχουσες τηλεφωνικές γραμμές και γραμμές ηλεκτρικού δικτύου. Προς αυτήν την κατεύθυνση έχουν αναπτυχθεί τεχνολογίες όπως το HomePlug για επικοινωνία μέσω ηλεκτρικών γραμμών (το PowerLine Connection) και το HomePNA για διασύνδεση μέσω της τηλεφωνικής γραμμής ή ομοαξονικού καλωδίου. Επιπλέον υπάρχει το G.hn, ένα πρότυπο κατά την διεθνή ένωση τηλεπικοινωνιών (ITU - International Telecommunication Union) για δικτύωση μέσω γραμμών ηλεκτρικού ρεύματος, τηλεφωνικών γραμμών, ομοαξονικών καλωδίων με υψηλούς ρυθμούς δεδομένων έως 1Gbps (Giga bit per second).

Εικόνα 5: Το καλώδιο Ethernet



Εικόνα 6: Η δικτύωση με PLC

Στην περίπτωση της ασύρματης επικοινωνίας, η ασύρματη διασύνδεση των συσκευών είναι μία τάση που κερδίζει συνεχώς έδαφος. Τα κυριότερα ασύρματα πρότυπα είναι το 802.11 (το Wi- Fi) και το 802.15.4 (που συναντάται στο ZigBee και γενικότερα σε μικροελεγκτές PIC).



Εικόνα 7: Παράδειγμα σύνδεσης οικιακού δικτύου

2.3. Έξυπνοι (ή Ευφυείς) Μετρητές

Ο σημαντικότερος μηχανισμός που διαφυλάττει την εύρυθμη λειτουργία του Smart Grid είναι ανεπιφύλακτα οι έξυπνοι μετρητές. Καταγράφουν την κατανάλωση ηλεκτρικής ενέργειας, δημιουργώντας μία αμφίδρομη επικοινωνία μεταξύ του μετρητή και ενός κεντρικού συστήματος (της εταιρίας παροχής ενέργειας), καλύπτοντας όποια αύξηση (ή και μείωση) στη ζήτηση ενέργειας και αποδίδοντας μία άμεση γνώση της κατανάλωσης ενέργειας. Η ζήτηση μπορεί και καταγράφεται σε μία βάση δεδομένων, δημιουργώντας ένα

ιστορικό κατανάλωσης ενέργειας, ώστε σαν μία επέκταση αυτοματισμού, να είναι δυνατή στο μέλλον η απόδοση του μεγέθους ενέργειας που είχε ζητηθεί κατά το αντίστοιχο διάστημα. Βάσει της γνώσης που αποκτά το Smart Grid, μπορεί και αναπτύσσει δυναμική σχέση μεταξύ καταναλωτή και παραγόμενης ενέργειας, ώστε να αποφεύγεται άσκοπη παραγωγή ενέργειας

Συνοπτικά, τα πλεονεκτήματα εγκατάστασης έξυπνων μετρητών είναι:

- Ενημέρωση σε πραγματικό χρόνο, όπου οι πληροφορίες κατανάλωσης παρέχονται άμεσα στον χρήστη από συσκευές ευρείας και καθημερινής χρήσης (εφαρμογή σε έξυπνο τηλέφωνο, λογαριασμός στον υπολογιστή, οικιακή οθόνη πολλαπλών ενδείξεων κτλ). Μέσω γραφημάτων και πινάκων παρουσιάζεται το ενεργειακό αποτύπωμα του καταναλωτή και μέσω κατάλληλων εργαλείων προτείνεται η περαιτέρω βελτίωση του.

Εικόνα 8: Η χρήση των έξυπνων μετρητών από την ΔΕΗ

- Η αμφίδρομη επικοινωνία, δηλαδή η αποστολή και λήψη δεδομένων για την προαναφερθείσα λειτουργία, αλλά και για ενημέρωση από τη μεριά του παρόχου προς τον καταναλωτή σχετικά με ζητήματα

ασφαλούς κατανάλωσης ενέργειας (έκτακτη διακοπή ρεύματος και προφύλαξη συσκευών από πιθανή ζημία που μπορεί να προκληθεί) ή ζητήματα που αφορούν τον λογαριασμό κατανάλωσης ρεύματος. Επιπρόσθετα, οι έξυπνοι μετρητές σε συνδυασμό με μετρητικές διατάξεις που είναι τοποθετημένες στο δίκτυο ηλεκτρικής ενέργειας, μπορούν και παρέχουν λειτουργίες εξομάλυνσης της τάσης, διόρθωσης της συχνότητας και προστασία από υπερτάσεις και υπερεντάσεις επιτυγχάνοντας βελτίωση ποιότητας ισχύος.

- Η δυνατότητα λήψης εντολών στους έξυπνους μετρητές μπορεί να βοηθήσει στην περαιτέρω εξοικονόμηση ενέργειας προτείνοντας ή και πράττοντας ενεργοβόρες προγραμματισμένες διαδικασίες σε περιόδους χαμηλής ζήτησης ενέργειας από το δίκτυο (φόρτιση ηλεκτρικού αυτοκινήτου, φόρτιση σωμάτων θέρμανσης σε περίοδο νυχτερινού τιμολογίου κτλ). Επίσης μπορεί να πραγματοποιηθεί εκκίνηση ή διακοπή σύνδεσης με το υπάρχον δίκτυο ή μετάβαση σε εναλλακτικό πάροχο ηλεκτρικής ενέργειας με άμεσο τρόπο.

- Οι έξυπνοι μετρητές μπορούν να συμβάλουν στην ευκολότερη ενσωμάτωση μικρών μονάδων ανανεώσιμων πηγών ενέργειας.

- Εξάλειψη της δαπάνης της καταμέτρησης των συμβατικών μετρητών κατανάλωσης ενέργειας, εξοικονομώντας τα χρήματα μετακίνησης και υπαλλήλων που μπορούν να τοποθετηθούν σε άλλους τομείς.

2.3.1. Τεχνικά χαρακτηριστικά Έξυπνων Μετρητών

Η υλοποίηση ενός έξυπνου μετρητή πραγματοποιείται κατά κύριο λόγο με έναν μετρητή τάσης, μετρητή ρεύματος, μία μητρική πλακέτα (motherboard) με ενσωματωμένο μικροεπεξεργαστή ο οποίος έχει επίσης ενσωματωμένη τη λειτουργία μετατροπής σήματος από αναλογικό σε ψηφιακό (Analog to Digital Converter- ADC).

Όπως γίνεται αντιληπτό ο μετρητής τάσης πραγματοποιεί την μέτρηση της μονοφασικής ή τριφασικής τάσης. Ο μορφοτροπέας τάσης (Voltage Transducer) μετατρέπει την υψηλή τιμή της τάσης σε μετρούμενο μέγεθος τάσης προς χρήση από τους έξυπνους μετρητές ή του επιτηρητές τάσης.

Αντίστοιχα λειτουργεί και ο μορφοτροπέας ρεύματος (Current Transducer). Κατ' αυτόν τον τρόπο, μεγάλης κλίμακας αναλογικά μεγέθη μετατρέπονται σε μικρής κλίμακας αναλογικά μεγέθη και έπειτα σε ψηφιακά

μεγέθη με τον μετατροπέα αναλογικού- ψηφιακού σήματος. Όπως προαναφέρθηκε, η λειτουργία ADC πραγματοποιείται στον μικροεπεξεργαστή, ο οποίος κατόπιν κάνει τη συλλογή και επεξεργασία των δεδομένων που του παρέχονται. Η μητρική πλακέτα παρέχει τις θύρες επικοινωνίας (USB, RS232, ethernet κτλ), τις θέσεις αποθήκευσης (μνήμη SD συνήθως ή μνήμη flash) και τις αναλογικές/ψηφιακές εισόδους/εξόδους.

Τέλος, στα βασικά χαρακτηριστικά ενός έξυπνου μετρητή αξίζει να αναφερθεί ότι υπάρχουν τουλάχιστον 4 αναλογικές είσοδοι για τη δειγματοληψία του ρεύματος και τουλάχιστον 3 αναλογικές είσοδοι για κάθε φάση της τάσης.

Εικόνα 9: Ηλεκτρικός έξυπνος μετρητής

Εικόνα 10: Ηλεκτρομηχανικός μετρητής

2.3.2. Ζητήματα από τη Χρήση των Έξυπνων Μετρητών

Ζητήματα που χρήζουν περαιτέρω μελέτη, δημιουργεί η εκτεταμένη χρήση και αποδοχή των έξυπνων μετρητών στην καθημερινότητα των καταναλωτών. Αναφορικά, τέτοια ζητήματα είναι:

- Προστασία προσωπικών δεδομένων, όπου για τη διασφάλιση της εμπιστευτικότητας απαιτείται η πρόσβαση σε αυτά από εξουσιοδοτημένα άτομα και η κρυπτογράφηση τους με ειδικούς αλγορίθμους προστασίας. Επιπρόσθετα ένα ισχυρό νομοθετικό πλαίσιο που θα αποτρέπει την παραβίαση και την πώληση- διαρροή τους για περαιτέρω σκοπούς.
- Προστασία των δεδομένων από κυβερνοεπιθέσεις. Η όλο και μεγαλύτερη χρήση του διαδικτύου και μικρότερου εύρους δικτύων για την ροή πληροφοριών, καθιστά ευάλωτα πολλά ευαίσθητα στοιχεία.

- Προστασία από αυθαιρεσίες και μη επιτρεπτών πρωτοβουλιών εκ μέρους των παρόχων ηλεκτρικής ενέργειας, όπως λανθασμένοι ή αλλοιωμένοι λογαριασμοί, μετάβαση σε διαφορετικά προγράμματα σύνδεσης κτλ., λόγω της ολοένα και μεγαλύτερης συγκέντρωσης στοιχείων.

2.4. Κοινωνικά Οφέλη

Τα κοινωνικά οφέλη που προκύπτουν από την χρήση έξυπνων μετρητών είναι:

- Σημαντική εξοικονόμηση ενέργειας λόγω βελτίωσης του προφίλ των καταναλωτών.
- Μείωση των εκπομπών αέριων ρύπων, που οφείλεται στην ελαχιστοποίηση παραγωγής πλεονάζουσας ενέργειας, λόγω της δυναμικής συμπεριφοράς και της αμφίδρομης επικοινωνίας με τους έξυπνους μετρητές.
- Ευκολότερη παροχή ενέργειας σε ευπαθείς κοινωνικά ομάδες, λόγω της καταγραφής των συγκεκριμένων αναγκών που έχουν και παροχής αντίστοιχων υπηρεσιών.
- Εξομάλυνση της καμπύλης φορτίου του συστήματος, λόγω μετάθεσης ενεργοβόρων διαδικασιών σε περιόδους χαμηλής ζήτησης ρεύματος. Έτσι αποφεύγεται η υπερφόρτωση του ηλεκτρικού δικτύου στις ώρες αιχμής.

2.5. Σύνδεση με τη Διπλωματική Εργασία

Ο Arduino με τη συλλογή δεδομένων που καλύπτει η βάση δεδομένων, μπορεί και μετράει την κατανάλωση ρεύματος που προκαλεί η ενεργοβόρα διαδικασία της θέρμανσης του νερού. Το προφίλ του καταναλωτή μπορεί να βελτιωθεί καθώς η θέρμανση του νερού μπορεί να γίνεται οικονομικά είτε κατά την ηλιοφάνεια είτε κατά το νυχτερινό τιμολόγιο που ορίζει ο πάροχος ηλεκτρικής ενέργειας. Από τη στιγμή που ο Arduino έχει πρόσβαση στο διαδίκτυο σαν συσκευή Internet of Things, τα δεδομένα του μπορούν να παρέχονται και εξ' αποστάσεως στον χρήστη προς επεξεργασία.

Ο Arduino επίσης καλύπτεται από τα δίκτυα επικοινωνίας σε τοπικό και σε απομακρυσμένο επίπεδο, είτε εντός τοπικού δικτύου είτε σαν συσκευή ενός μεγάλου εύρους δικτύου (μέλος του Internet of Things) που είναι ορατό στην παγκόσμια κοινότητα συσκευών.

3. Επικοινωνία Μεταξύ Συσκευών (Machine to Machine Communication)

Η τεχνολογία επικοινωνίας μεταξύ των συσκευών (Machine to Machine Communication ή M2M) επιτρέπει σε συσκευές όπως υπολογιστές, αισθητήρες, ενσωματωμένα συστήματα, έξυπνα κινητά τηλέφωνα κτλ τα οποία είναι διασυνδεδεμένα είτε ασύρματα είτε ενσύρματα να επικοινωνούν μεταξύ τους και να λαμβάνουν αποφάσεις με την ελάχιστη ανθρώπινη παρέμβαση.

Οι συσκευές έχουν την ευφυΐα – νοημοσύνη να αποφασίζουν αυτόνομα βάσει των δεδομένων που συλλέγουν οι ίδιες ή άλλες συσκευές για λογαριασμό τους.

Ένας αισθητήρας, ένας μετρητής ή μία συσκευή (για παράδειγμα ένας αναλογικός αισθητήρας θερμοκρασίας) συλλέγει πληροφορίες από το περιβάλλον (πχ την θερμοκρασία) και την αναμεταδίδει μέσω του δικτύου (ασύρματου ή ενσύρματου ή υβριδικού) σε μία εφαρμογή (πχ το λογισμικό της θέρμανσης ή της ψύξης του χώρου αφού πρώτα την μετατρέψει σε ψηφιακό δεδομένο) και πραγματοποιείται μία απόφαση (θέρμανση ή ψύξη του χώρου). Το γεγονός μπορεί να καταγραφεί σε μία βάση δεδομένων και με βάση αυτό να εξαχθεί μία χρήσιμη πληροφορία (πχ κατανάλωση ενέργειας) η οποία να χρησιμοποιηθεί για ανατροφοδότηση του συστήματος.

Εικόνα 11: Επικοινωνία μεταξύ μηχανών, με μία ευρύτερη έννοια

Παραδείγματα εφαρμογών χρήσης είναι:

- Σύνδεση μηχανών – συσκευών μεταξύ τους, είτε απομακρυσμένα είτε εντός τοπικού δικτύου

- Σύνδεση μηχανών – συσκευών με κέντρα υπηρεσιών π.χ. αυτοκίνητα με ενημέρωση προς το συνεργείο για την προγραμματισμένη συντήρηση, ή διαχείριση και τοποθεσία στόλου αυτοκινήτων.

- Σύνδεση μηχανών – συσκευών με κέντρα υπηρεσιών για ενημέρωση των αποθεμάτων.

Η M2M επικοινωνία είναι μία εξελιγμένη ιδέα με βάση την τηλεμετρία - τηλεματική, η οποία χρησιμοποιείται για την μετάδοση και μέτρηση δεδομένων εξ' αποστάσεως είτε ενσύρματα είτε ασύρματα. Αναλυτικά, αισθητήρες και απομακρυσμένες συσκευές συλλέγουν πληροφορίες όπου αποστέλλονται σε ένα κεντρικό σημείο για ανάλυση. Η M2M επικοινωνία διαφοροποιείται εφαρμόζοντας σύγχρονες τεχνολογίες δικτύωσης. Τεχνολογίες όπως τα ασύρματα δίκτυα των αισθητήρων, το διαδίκτυο και τα ενσωματωμένα συστήματα συνεργάζονται και δημιουργούν την εν λόγω τεχνολογία.

3.1. Διαφορές Μεταξύ Τηλεμετρίας και M2M Επικοινωνίας

Αναφορικά, οι διαφορές τηλεμετρίας με τη machine to machine επικοινωνία είναι:

- Τα συστήματα τηλεμετρίας μεταδίδουν ένα τυχαίο ραδιόσημα, ενώ η M2M επικοινωνία χρησιμοποιεί τα υπάρχοντα ασύρματα ή ενσύρματα δίκτυα.

- Οι αισθητήρες στις επικοινωνίες τηλεμετρίας ήταν απόλυτα εξειδικευμένοι στη λειτουργία που επιτελούσαν και η τροφοδοσία τους ήταν ένα δύσκολο και σοβαρό θέμα, καθώς απαιτούσαν μεγάλη κατανάλωση ισχύος για τη μετάδοση των δεδομένων. Οι σύγχρονοι αισθητήρες συνδεόμενοι στα γνωστά δίκτυα επικοινωνίας, με μικρή κατανάλωση ισχύος, κάνουν πολλαπλές εργασίες.

- Στα συστήματα τηλεμετρίας η συλλογή των δεδομένων ήταν ανομοιογενής εάν κάποιος αισθητήρας τοποθετούνταν σε "δύσκολο σημείο" από την άποψη εμβέλειας σήματος, καθώς η ροή των δεδομένων είχε προβλήματα. Με τα σύγχρονα δίκτυα επικοινωνίας η επεκτασιμότητα είναι πιο εύκολα πραγματοποιήσιμη δίνοντας μεγάλη αξιοπιστία στη ροή και στην ποιότητα των δεδομένων.

3.2. Τρόπος Λειτουργίας M2M Επικοινωνίας

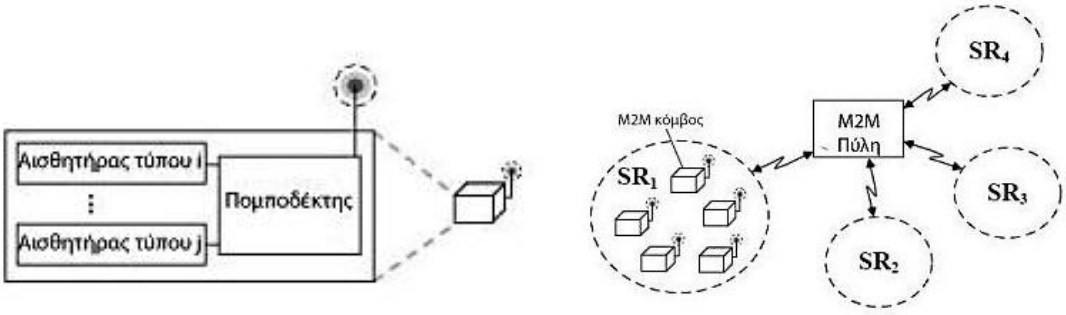
Σαν πρώτο βήμα, πρέπει να γίνει μελέτη για την τοποθέτηση των αισθητηρίων σε καίρια σημεία. Οι αισθητήρες ανά περιοχή δημιουργούν ένα τοπικό δίκτυο. Από αυτούς λαμβάνονται δεδομένα πραγματικού χρόνου και μέσω του τοπικού δικτύου, που είναι συνδεδεμένο στο διαδίκτυο, καταλήγουν στους μηχανικούς είτε σε αυτοματοποιημένα συστήματα που κάνουν την παρακολούθηση της εισερχόμενης ροής δεδομένων.

Εικόνα 12: Τα βασικά στοιχεία της M2M επικοινωνίας

- Οι Αισθητήρες, είναι χαμηλής κατανάλωσης ισχύος και “plug and play”, δηλαδή με το που τροφοδοτούνται ξεκινάει η λειτουργία τους.
- Η Επικοινωνία χαρακτηρίζεται από τη δυνατότητα σύνδεσης μεγάλου αριθμού σταθερών και κινητών συσκευών με πολύ χαμηλή ενεργειακή κατανάλωση.
- Ο Υπολογισμός έγκειται στο γεγονός ότι έχουν καταγραφεί τα διάφορα σενάρια, οπότε οι απαντήσεις είναι άμεσα διαθέσιμες με το που έρχονται οι ερωτήσεις. Η ευφυΐα των συσκευών δεν είναι ατομική αλλά καταμερίζεται ανά δίκτυο μέσω της τεχνολογίας cloud.
- Οι μηχανές παρέχουν τις πολύτιμες Υπηρεσίες τους, έχοντας επικοινωνία με τους ανθρώπους μέσω των διεπαφών.

Ένα M2M αποτελείται από ένα σύνολο M2M κόμβων και M2M πυλών. Όπως παρατηρείται και στην επόμενη εικόνα ένας M2M κόμβος διαθέτει πολλαπλούς αισθητήρες για τη συλλογή διαφορετικών τύπων δεδομένων (πχ θερμοκρασία, υγρασία κτλ) και έναν πομποδέκτη για τη μετάδοση των δεδομένων σε μία M2M πύλη μέσω πρωτοκόλλων επικοινωνίας πχ WiFi, WiMAX, ZigBee κτλ. Μέσω του συστήματος προσδιορισμού θέσης (GPS) ένας κόμβος M2M μπορεί να λάβει πληροφορίες για τη θέση του. Μία M2M πύλη συνήθως είναι εφοδιασμένη με μόνιμη παροχή ρεύματος οπότε έχει τη δυνατότητα υπολογισμού και μετάδοσης δεδομένων εκτελώντας το βασικό της καθήκον.

Η πιο απαιτητική υλοποίηση στις M2M επικοινωνίες είναι η παρακολούθηση πραγματικού χρόνου. Το M2M δίκτυο επικοινωνίας διαιρείται σε περιοχές ανίχνευσης (Sensing Regions-Srs). Σε κάθε περιοχή υπάρχουν ένας ή περισσότεροι τύποι δεδομένων προς συλλογή ανά χρονική περίοδο.



Εικόνα 13: Η αρχιτεκτονική του δικτύου της M2M επικοινωνίας. Αριστερά φαίνονται τα στοιχεία ενός M2M κόμβου και στην εικόνα δεξιά φαίνονται οι M2M κόμβοι, μία M2M πύλη και SRs

Κάθε κόμβος M2M έχει διαφορετικές ή και τις ίδιες ικανότητες ανίχνευσης, ανάλογα το είδος αισθητήρων που συλλέγουν δεδομένα. Οι καταστάσεις λειτουργίας των κόμβων είναι η “Ενεργή” (Active) και η “Αδρανής” (Sleep). Ως “Ενεργή” κατάσταση καλείται η κατάσταση κατά τη συλλογή και αποστολή δεδομένων και ως “Αδρανής” καλείται η κατάσταση κατά την οποία υπάρχει αδράνεια λειτουργίας και το σύστημα προς βελτιστοποίηση της κατανάλωσης ενέργειας μεταβαίνει σε αυτήν. Η δυσκολία που συναντάται σε αυτό το κομμάτι είναι ότι παρόλο που το ατομικό κόστος αγοράς των αισθητήρων δεν είναι υψηλό, λόγω του μεγάλου όγκου χρησιμοποίησης για αξιοπιστίας δεδομένων δημιουργείται τελικά θέμα κόστους. Σημαντικό γεγονός αποτελεί και η συντήρηση και η αντικατάσταση τους λόγω γήρανσης. Επίσης, ένας σημαντικός ανασταλτικός παράγοντας είναι ο τρόπος και το κόστος αντικατάστασης της μπαταρίας τους. Αυτές είναι και οι βασικές προκλήσεις που αντιμετωπίζουν οι σχεδιαστές αυτού του εξοπλισμού.

Στο επόμενο βήμα, πρέπει να γίνει η μετάδοση της πληροφορίας. Αυτό γίνεται συνήθως ασύρματα μέσω WiFi, Bluetooth, ZigBee, κυψελωτού δικτύου κινητής τηλεφωνίας και άλλων ασύρματων τεχνολογιών. Η πρόκληση που αντιμετωπίζει αυτό το κομμάτι είναι το ολοένα και αυξανόμενο πλήθος των διασυνδεδεμένων συσκευών αν θα μπορεί να συμμετέχει ισάξια και ποιοτικά

στο διαμοιρασμό της πληροφορίας, καθώς όταν υπάρχει μεγάλος αριθμός ταυτόχρονων χρηστών κάποιος δε θα λάβει υπηρεσία.

Η μεγάλη ποσότητα των δεδομένων απαιτεί την ανάγκη γρηγορότερων και ικανότερων υπολογισμών, δηλαδή καλύτερων προγραμμάτων ανάλυσης με υψηλότερη ευφυΐα, όπου κάθε νέο δεδομένο που θα λαμβάνεται να μπορεί να ενσωματώνεται άμεσα στο σύστημα. Ο τομέας της πληροφορικής είναι αυτός που θα αναλάβει αυτόν το σημαντικό ρόλο.

Τέλος μέσω της αλληλεπίδρασης ανθρώπου – μηχανής πραγματοποιείται η κατάλληλη δράση, είτε από τους χρήστες είτε από τις μηχανές.

3.3. Εφαρμογές

Οι αυξημένες υπολογιστικές ικανότητες που προσφέρει η M2M επικοινωνία, προσφέρει πληθώρα εφαρμογών. Έτσι συναντάται (ή προσφέρεται προς εφαρμογή):

- Σε επιχειρήσεις κοινής ωφέλειας, για την τιμολόγηση των πελατών βάσει συλλογής ενεργειακών δεδομένων από την κατανάλωση πετρελαίου, φυσικού αερίου, αλλά και πρότασης νέων τιμολογίων ανάλογα του προφίλ του πελάτη.

- Η ρύθμιση της κυκλοφορίας βάσει του δυναμικού και αμφίδρομου χαρακτήρα που έχει η επικοινωνία των αισθητήρων με τα κέντρα ελέγχου. Έτσι ρυθμίζονται κατάλληλα οι σηματοδότες ώστε να δίνεται αντίστοιχη προτεραιότητα σε δρόμους με περισσότερη ένταση κίνησης, ή οι ενημερωτικές πινακίδες πληροφόρησης της κυκλοφορίας ώστε να μετατοπίζεται η κίνηση προς άλλους δρόμους ή ακόμη και ο φωτισμός στους δρόμους.

- Η τηλεϊατρική χρήση, όπου ασθενείς με προβλήματα υγείας φέρουν ειδικές συσκευές παρακολούθησης της πορείας της υγείας τους (πχ παρακολούθηση καρδιακής λειτουργίας). Τα δεδομένα ανά τακτά ή ανά κρίσιμα χρονικά διαστήματα στέλνονται είτε σε επιβλέποντα ιατρό είτε σε εμφυτευμένες στον ασθενή συσκευές όπου σε επείγοντα περιστατικά μπορούν να αντιδράσουν και να βοηθήσουν την κατάσταση (πχ αισθητήρας με αντλία για χορήγηση ινσουλίνης).

- Για επιχειρηματικούς λόγους, όπου η M2M επικοινωνία βοηθά σε θέματα παρακολούθησης των αποθεμάτων και ρύθμισης παραγγελιών

σε πραγματικό χρόνο (πχ αυτόματοι πωλητές, μηχανήματα τζόγου κτλ).

- Σε θέματα ασφαλείας, όπου παρακολουθείται ένας χώρος για διατήρηση της τάξης.
- Σε γεωργικά ζητήματα, για μετρήσεις και ελέγχους που αφορούν την παραγωγή (δοσολογία λιπασμάτων, υγρασία χώματος κτλ)

Οι τομείς εφαρμογών συνεχώς επαναπροσδιορίζονται με συνδυασμό των παραπάνω και της δημιουργίας νέων κατηγοριών. Εφόσον υπάρχει μία σαφής μελέτη και ανάλυση για το πεδίο εφαρμογής μίας νέας M2M επικοινωνίας, είναι εφικτή η δημιουργία και η εγκατάσταση της με ευέλικτο και αποδεκτό τρόπο.

3.4. Σύνδεση με τη Διπλωματική Εργασία

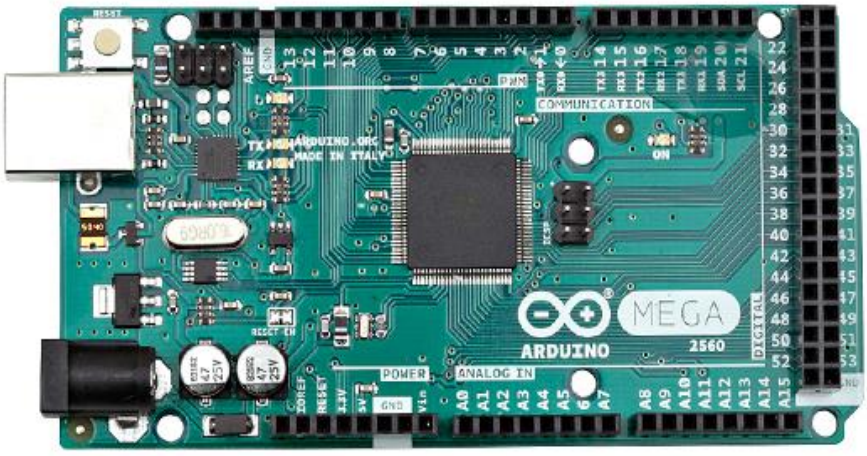
Οι περιφερειακές συσκευές της διπλωματικής εργασίας πέρα από τη συλλογή των δεδομένων, έχουν μία δομή στη λειτουργία τους. Δρουν σαν είσοδοι στον Arduino δίνοντας τα δεδομένα τους προς επεξεργασία. Οπότε υπάρχει επικοινωνία μηχανών κ το ρόλο συντονιστή παρέχει ο Arduino. Βέβαια, θα μπορούσαν να εμπλακούν και άλλες συσκευές Arduino με τα αισθητήρια τους, που λόγω της σωστής αρχικοποίησης της όλης δομής δε θα υπήρχε σύγχυση ρόλων.

Ακολουθεί ένα πιθανό σενάριο λειτουργίας: η ανάγνωση της θερμοκρασίας του νερού που γίνεται από το αισθητήριο θερμοκρασίας είναι συνεχής και χρησιμοποιείται πάντα σαν στοιχείο στην παρακολούθηση της κατάστασης του νερού μέσω του διαδικτύου. Όταν υπάρχει εντολή για εκκίνηση της θέρμανσης του νερού χωρίς αντίστοιχη εντολή παύσης μετά από κάποιο χρόνο, η παύση γίνεται με αυτόματο τρόπο σε μία κρίσιμη θερμοκρασία που παρέχει το αισθητήριο θερμοκρασίας για λόγους ασφαλείας. Οπότε τρία περιφερειακά, το κουμπί εκκίνησης, η οθόνη παρακολούθησης μέσω διαδικτύου και το αισθητήριο θερμοκρασίας αλληλεπιδρούν μεταξύ τους υπό την κεντρική καθοδήγηση του Arduino. Σε αυτό το σενάριο θα μπορούσε να εμπλακεί και η κάρτα μνήμης παρέχοντας μία επιπλέον είσοδο, δηλαδή μία εντολή για εκκίνηση της θέρμανσης του νερού σε προγραμματισμένο χρόνο χωρίς όμως να υπάρχει λόγος αφού το νερό είναι ήδη ζεστό. Αρά πλέον εμπλέκονται τέσσερεις περιφερειακές συσκευές.

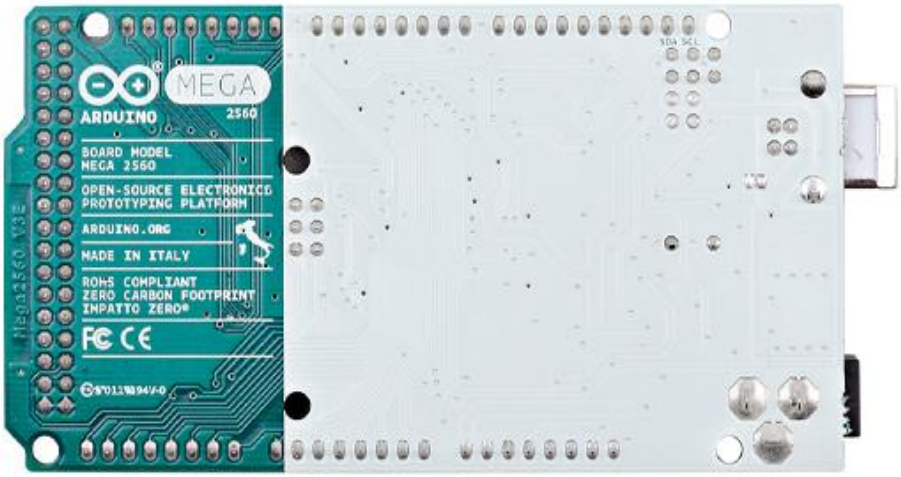
4. Ο Μικροελεγκτής Arduino

Το Arduino είναι μία πλατφόρμα με “σχέδιο ανοιχτού κώδικα” (open source project), το οποίο έχει δημιουργηθεί να λειτουργεί βάσει των δυνατοτήτων του μικροελεγκτή που αποτελεί την εκάστοτε πλατφόρμα. Η οικογένεια μικροελεγκτών είναι ATmega της εταιρείας Atmel και όλα τα σχέδια και το απαραίτητο λογισμικό για τη λειτουργία του διανέμονται δωρεάν, ώστε η υλοποίηση να είναι εφικτή από όποιον το επιθυμεί (γι’ αυτό χαρακτηρίζεται και σαν “σχέδιο ανοικτού κώδικα”). Αφού γίνει η υλοποίηση, η συμπεριφορά του είναι σαν ενός “μικρού υπολογιστή”, καθώς μπορούν να συνδεθούν πολλαπλές μονάδες εισόδου/εξόδου και ο μικροελεγκτής να προγραμματιστεί κατάλληλα ώστε να δέχεται δεδομένα από τις μονάδες εισόδου, να τις επεξεργάζεται και κατόπιν να στέλνονται αντίστοιχες εντολές στις μονάδες εξόδου.

Σίγουρα, δεν είναι ο μοναδικός ούτε και ο καλύτερος τρόπος υλοποίησης μίας διαδραστικής ηλεκτρονικής συσκευής. Το κύριο όμως πλεονέκτημα είναι η τεράστια κοινότητα που το υποστηρίζει και έχει δημιουργήσει μία τεράστια διαδικτυακή γνωσιακή βάση. Έτσι, παρότι ένας έμπειρος ηλεκτρονικός μηχανικός μπορεί να προτιμήσει διαφορετική πλατφόρμα υλοποίησης ή εξαρτήματα, το Arduino με το εκτενές υλικό με παραδείγματα και υλοποιήσεις (documentation) κερδίζει κοινό στο οποίο οι γνώσεις είναι περιορισμένες σε εκπαιδευτικό και όχι επαγγελματικό επίπεδο.



Εικόνα 14: Άνω όψη της πλατφόρμας Arduino Mega 2560, με τον μικροελεγκτή ATmega2560, της διπλωματικής εργασίας



Εικόνα 15: Κάτω όψη της πλατφόρμας Arduino Mega 2560, με τον μικροελεγκτή ATmega2560, της διπλωματικής εργασίας

Ακριβώς λόγω του ότι απευθύνεται σε μη επαγγελματίες, κυκλοφορούν έτοιμες προκατασκευασμένες πλακέτες Arduino με μικρό κόστος. Υπάρχουν, όπως προαναφέρθηκε, διάφορες επίσημες εκδόσεις ανάλογα τον μικροελεγκτή που χρησιμοποιείται και αναφορικά είναι οι Nano, Uno, Mega, Yun, Leonardo κτλ.

Στην παρούσα εργασία χρησιμοποιείται η πλατφόρμα Arduino Mega 2560 με τον μικροελεγκτή ATmega2560 έναν 8bit RISC (Reduced Instruction Set Computer), όπου χρονίζει στα 16MHz. Ο ATmega2560 διαθέτει μνήμη τριών τύπων:

- 8 kb μνήμης SRAM, η ονομαζόμενη ωφέλιμη μνήμη για να αποθηκεύονται μεταβλητές, πίνακες κτλ κατά το runtime (την εκτέλεσή του). Αυτή η μνήμη χάνει τα δεδομένα με τη διακοπή παροχής ρεύματος.

- 4 kb μνήμης EEPROM, η οποία χρησιμοποιείται για “επίκτητη” αποθήκευση δεδομένων, δηλαδή δε χάνει τα περιεχόμενα της με διακοπή της παροχής ρεύματος.

- 256 kb μνήμης Flash, από τα οποία 8 kb είναι το firmware του Arduino (ή bootloader στην ορολογία του κατασκευαστή) και τα υπόλοιπα χρησιμοποιούνται για την εγκατάσταση προγραμμάτων, αφού πρώτα μεταγλωττιστούν στον προσωπικό Η/Υ. Η μνήμη Flash όπως και η

EEPROM δεν χάνουν τα δεδομένα τους με απώλεια τροφοδοσίας ή με reset.

Στα χαρακτηριστικά επίσης συγκαταλέγονται τα εξής:

- Ότι διαθέτει 54 ψηφιακές εισόδους/εξόδους (εκ των οποίων οι 15 για PWM έξοδο, δηλαδή μπορούν να προσομοιώσουν αναλογικές εξόδους)
- 16 αναλογικές εισόδους

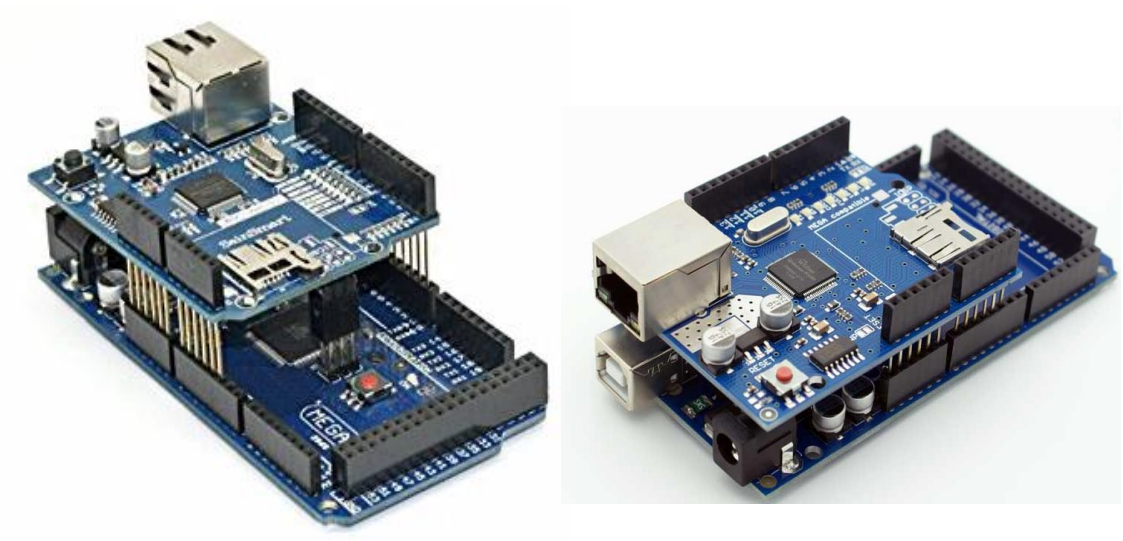
Αναλυτικά τα τεχνικά χαρακτηριστικά του Arduino Mega (Εικόνα 16):

Microcontroller	ATmega2560
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	54 (of which 15 provide PWM output)
Analog Input Pins	16
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	256 KB of which 8 KB used by bootloader
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz

Εικόνα 16: Τα τεχνικά χαρακτηριστικά του Arduino Mega

Η διασύνδεση της πλατφόρμας Arduino με τον Η/Υ γίνεται μέσω της θύρας USB, η οποία θύρα του παρέχει και σταθερή τροφοδοσία 5 Volts. Η επέκταση των δυνατοτήτων μίας πλατφόρμας (Ethernet, GPS, Wireless, SD, LCD Display κτλ) επιτυγχάνεται με τυποποιημένες πλακέτες επέκτασης που ονομάζονται “shields” και η εγκατάσταση τους γίνεται εύκολα κουμπώνοντας στο πάνω μέρος της υπάρχουσας πλακέτας Arduino (Εικόνα 17).

Στην παρούσα εργασία χρησιμοποιήθηκε το Ethernet Shield με το chip W5500.

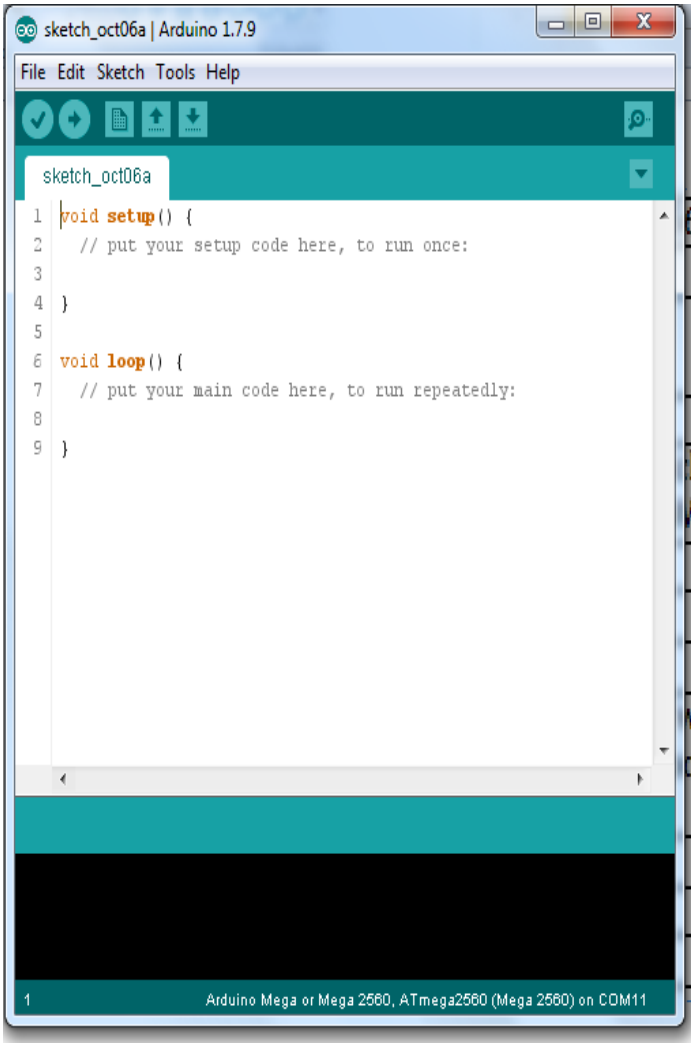


Εικόνα 17: Arduino Mega με εγκατεστημένο το Ethernet Shield πάνω του. Έτσι δίνεται η δυνατότητα επέκτασης των δυνατοτήτων του Arduino για πρόσβαση στο διαδίκτυο καθώς και χρήσης SD κάρτας μνήμης

4.1. Το Περιβάλλον Ανάπτυξης

Για να προγραμματιστεί η μονάδα Arduino, χρειάζεται το περιβάλλον προγραμματισμού Arduino IDE. Εκεί γράφεται κώδικας σε γλώσσα Wiring, μία παραλλαγή της C/C++ και ακολούθως μεταγλωττίζεται και μεταφορτώνεται στην πλατφόρμα του Arduino. Το Arduino IDE (Εικόνα 18) υπάρχει σε εκδόσεις για Windows, Linux και Mac και διατίθεται δωρεάν από την επίσημη ιστοσελίδα http://www.arduino.org/downloads. Το περιβάλλον διαθέτει επίσης εξελληνισμένο μενού και πληθώρα έτοιμων παραδειγμάτων χρήσης βασικών εντολών και λειτουργιών (στο File => Examples).

Οι θύρες εισόδου/εξόδου (i/o pins) είναι αναλογικές και ψηφιακές. Σαν ψηφιακή είσοδο λογίζονται 0 Volts ή 5 Volts, με τον χαρακτηρισμό LOW ή HIGH. Παρόμοια συμβαίνει και σε οποιαδήποτε ψηφιακή έξοδο, όπου η έξοδος μπορεί να είναι 0 Volts ή 5 Volts με τον χαρακτηρισμό επίσης LOW ή HIGH. Στις αναλογικές εισόδους διαβάζονται τιμές ρεύματος που δίνονται αναλόγως του εύρους τάσης 0 έως 5Volts και για αναλογική έξοδο χρησιμοποιούνται οι PWM ψηφιακές θύρες, οι οποίες δίνουν ρεύμα εξόδου όποια τιμής επιθυμείται από εύρος τάσης 0 έως 5 Volts.

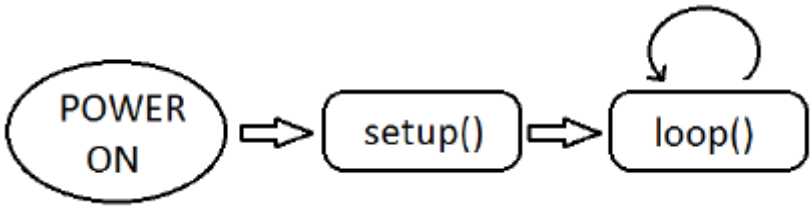


Εικόνα 18: Το περιβάλλον Arduino IDE. Διακρίνονται οι δύο βασικές συναρτήσεις setup() και loop()

Η λογική του περιβάλλοντος είναι απλή και βασίζεται σε δύο βασικές συναρτήσεις, τη setup() και τη loop(), οι οποίες δουλεύουν ως εξής:

- setup() : τοποθετούνται όλες οι εντολές που πρέπει να τρέξουν μία φορά, όταν ενεργοποιείται η πλατφόρμα (δηλαδή όταν τροφοδοτείται με ρεύμα ή πατηθεί το πλήκτρο reset που υπάρχει πάνω στον Arduino) Συνήθως τοποθετούνται αρχικοποιήσεις τιμών μεταβλητών και οι χαρακτηρισμοί των εισόδων/εξόδων που χρησιμοποιούνται (δηλαδή αν κάποια θύρα είναι είσοδος ή έξοδος).

- loop() : γράφεται το κυρίως πρόγραμμα. Οι εντολές που υπάρχουν θα τρέξουν και όταν φτάσουν στο τέλος ενεργοποιείται ξανά η loop(), συνεχίζοντας από την αρχή της και ξανά. Αυτό συμβαίνει συνεχώς έως ότου διακοπεί η τροφοδοσία του Arduino ή πατηθεί το κουμπί reset. Στην περίπτωση του reset, ξανατρέχει η συνάρτηση setup() μία φορά και ακολούθως η loop() όπως περιγράφτηκε.



Εικόνα 19: Η ροή εκτέλεσης ενός προγράμματος

Οπότε ένα τυπικό πρόγραμμα έχει την παρακάτω δομή:

```
void setup() {
  /* οι εντολές εδώ θα τρέξουν μόνο στην ενεργοποίηση ή μετά από Reset */
}
void loop() {
/* οι εντολές εδώ θα τρέχουν ξανά και ξανά,
μέχρι να απενεργοποιηθεί ή να πατηθεί το Reset */
}
```

Εικόνα 20: Δομή προγράμματος

4.1.1. Τύποι Μεταβλητών

Οι τύποι μεταβλητών που υποστηρίζονται στο Arduino είναι οι γνωστοί όπως στις γλώσσες προγραμματισμού C/C++. Επιγραμματικά:

- boolean, με τιμές 0 και 1 (ή False και True)
- byte, με τιμές 0 έως και 255
- int, ακέραιος με εύρος τιμών -32768 έως και 32767
- long, ακέραιος με εύρος τιμών -2147483648 έως και 2147483647
- float, δεκαδικοί αριθμοί
- char, ένας χαρακτήρας (μεγέθους ενός Byte)
- string, πίνακας χαρακτήρων

Ένα παράδειγμα δήλωσης μεταβλητών ακολουθεί:

```
int ledPin = 12;        // ορίζεται ακέραια μεταβλητή ledPin και
                        // αρχικοποιείται η τιμή της σε 12

float SinValue;         // ορίζεται δεκαδική μεταβλητή SinValue
```

4.1.2. Συναρτήσεις Διαχείρισης Θυρών Εισόδου/Εξόδου

Η κύρια λειτουργία του μικροελεγκτή είναι ο έλεγχος των θυρών που διαθέτει, είτε δίνοντας είτε λαμβάνοντας ρεύμα από αυτές. Στη συνάρτηση setup() που γίνεται η αρχικοποίηση κάθε προγράμματος, χαρακτηρίζονται οι θύρες (Pins) που θα χρησιμοποιηθούν ως είσοδοι ή έξοδοι.

Η συνάρτηση pinMode(Pin, Mode) χρησιμοποιείται με το όνομα της και με ορίσματα τον αριθμό θύρας (Pin) και την κατάσταση λειτουργίας (Mode) που χαρακτηρίζεται με τη λέξη INPUT (είσοδος) ή OUTPUT (έξοδος).

Όπως έχει ήδη αναφερθεί διατίθενται 54 ψηφιακές θύρες (Pins) εκ των οποίων 15 είναι δυνατότητας προσομοίωσης PWM, με ονόματα 0 έως 53 και 16 αναλογικές θύρες με ονόματα A0 έως A15.

Ακολουθεί παράδειγμα χρήσης της συνάρτησης:

```
pinMode(12,OUTPUT);          // ορίζεται η ψηφιακή θύρα 12 ως
                             // έξοδος

pinMode(ledPin, OUTPUT);     // η μεταβλητή του προηγούμενου
                             // παραδείγματος ledPin, ορίζεται ως
                             // έξοδος

pinMode(A2,INPUT);           // η αναλογική θύρα 2 ορίζεται ως
                             // είσοδος, προαιρετική εντολή
                             // καθώς πάντα οι αναλογικές θύρες
                             // είναι είσοδοι
```

4.1.3. Συναρτήσεις Εισόδου/Εξόδου Ρεύματος

Για να διαχειριστεί το ρεύμα μίας θύρας (Pin), πρέπει αρχικά να γίνει ορισμός της θύρας ως είσοδος ή έξοδος, όπως εξηγήθηκε στην προηγούμενη παράγραφο. Στη συνέχεια πρέπει να δοθεί έξοδος στις ψηφιακές θύρες (Pins) εξόδου, δηλαδή τάση 0 Volts ή 5 Volts. Αυτό πραγματοποιείται με τη χρήση της συνάρτησης digitalWrite(Pin,Value), όπου το όρισμα Pin αναφέρεται στον αριθμό θύρας στην οποία δίνεται τάση εξόδου ίση με Value όπως υποδεικνύει το επόμενο όρισμα και είναι LOW για 0 Volts και HIGH για 5 Volts. Ακολουθεί πλήρες παράδειγμα:

```
int ledPin = 12;                  // ορίζεται ακέραια μεταβλητή ledPin

pinMode(ledPin, OUTPUT);          // αρχικοποιείται σαν έξοδος στη
                                  // setup() υποχρεωτικά

digitalWrite(ledPin, HIGH);       // δίνεται τάση εξόδου 5 Volts
```

Όλες οι ψηφιακές είσοδοι του Arduino δουλεύουν και ως ψηφιακές είσοδοι, δηλαδή μπορούν να “διαβάσουν” τάση τιμής 0 Volts ή 5 Volts. Το γεγονός αυτό επιτυγχάνεται με χρήση της συνάρτησης digitalRead(Pin), όπου το όρισμα Pin αναφέρεται στον αριθμό θύρας εισόδου. Η τιμή της τάσης εισόδου είναι 0 Volts ή 5 Volts, οι οποίες τιμές αναπαρίστανται καθορισμένα με LOW και HIGH αντίστοιχα. Ακολουθεί πλήρες παράδειγμα:

```
int inPin = 12;                   // ορίζεται ακέραια μεταβλητή inPin

pinMode(inPin, INPUT);            // αρχικοποιείται σαν είσοδος στη
                                  // setup() υποχρεωτικά

digitalRead(inPin);               // διαβάζεται τάση εισόδου

boolean Val = digitalRead(inPin);       // εκχωρείται η κατάσταση
                                        // εισόδου σε μία μεταβλητή
                                        // τύπου Boolean, για χρήση
                                        // μέσα στο πρόγραμμα
```

Έχει αναφερθεί πιο πριν ότι κάποιες από τις 15 θύρες (Pins) του Arduino Mega έχουν την ένδειξη PWM, δηλαδή μπορούν να προσομοιώσουν την αναλογική έξοδο μέσω της παλμοκωδικής διαμόρφωσης. Πρακτικά, με ακέραιες τιμές από 0 έως 255 προσομοιώνεται κατά αναλογία το εύρος τάσης 0 Volts έως 5 Volts. Αυτό επιτυγχάνεται με τη χρήση της συνάρτησης analogWrite(Pin, Value), όπου το όρισμα Pin αναφέρεται στον αριθμό θύρας που δίνεται η τάση εξόδου, ενώ το όρισμα Value λαμβάνει τιμές από 0 (για 0 Volts) έως 255 (για 5 Volts) στην προαναφερθείσα αναλογική θύρα εξόδου. Αν δοθεί κάποια ενδιάμεση τιμή π.χ. 122 θα ληφθεί τάση 2,5 Volts. Ακολουθεί παράδειγμα διατύπωσης:

```
int ledPin = 7;                  // ορίζεται ακέραια μεταβλητή ledPin

pinMode(ledPin, OUTPUT);         // αρχικοποιείται σαν έξοδος στη
                                 // setup() υποχρεωτικά

analogWrite(ledPin, 122);        // δίνεται τάση εξόδου 2,5 Volts
```

Αν το παραπάνω παράδειγμα υλοποιηθεί με μία φωτοδίοδο (led) και με διαφορετικές τιμές στην analogWrite(), ξεκινώντας έστω από analogWrite(ledPin, 100) και σταδιακά αυξάνοντας έως το analogWrite(ledPin, 255), θα παρατηρηθεί ότι η φωτεινότητα του ενδεικτικού led θα αυξάνεται σταδιακά.

Η πλατφόρμα Arduino Mega διαθέτει 16 αναλογικές εισόδους, όπου χαρακτηρίζονται με τα σύμβολα A0, A1, A2, A3, …., A15. Σε αυτές τις αναλογικές εισόδους μπορεί να συνδεθεί ένα αναλογικό εξάρτημα, όπως ένα ποτενσιόμετρο και να διαβαστεί ως είσοδος. Η υλοποίηση γίνεται με τη χρήση της συνάρτησης analogRead(Pin), όπου το όρισμα Pin αναφέρεται στον αριθμό της θύρας στην οποία λαμβάνεται η είσοδος. Η τιμή της εισόδου κυμαίνεται από 0 έως 1023 και συνήθως χρησιμοποιείται μία ακέραια (integer) μεταβλητή για να καταχωρείται η τιμή. Για παράδειγμα:

```
pinMode (A1, INPUT);             // αρχικοποιείται σαν είσοδος στη
                                 // setup() υποχρεωτικά

int potValue = analogRead(A1);   // καταχώρηση στην potValue της
                                 // τιμής της αναλογικής εισόδου A1
```

4.1.4. Συναρτήσεις Χρόνου

Σε πληθώρα προγραμμάτων χρειάζεται να γίνει διαχείριση ή καταγραφή του χρόνου. Προς τούτου υπάρχουν αντίστοιχες συναρτήσεις χρόνου που βοηθούν.

Η συνάρτηση καθυστέρησης delay(time), ορίζει καθυστέρηση που διαρκεί το γεγονός. Στο όρισμα time αποδίδεται χρόνος σε msec (1/100 sec). Στο σημείο που αναγράφεται η ανωτέρω εντολή, σταματά η εκτέλεση του προγράμματος για χρόνο ίσο με time. Για παράδειγμα:

```
delay(1000);        // σταματά η εκτέλεση του προγράμματος
                    // για 1000 msec = 1 sec

delay(500);         // σταματά η εκτέλεση του προγράμματος για
                    // 500 msec = 0,5 sec
```

Εντελώς παρόμοια με πριν, απλά με διαφορετική τάξη μεγέθους χρόνου, χρησιμοποιείται η εντολή delayMicroseconds(time). Στο όρισμα αποδίδεται χρόνος σε μsec ($1/10^6$ sec). Για παράδειγμα:

```
delayMicroseconds(10);    // σταματά η εκτέλεση του προγράμματος
                          // για 10 μsec = 1/100000 sec
```

Στο Arduino Mega υπάρχει ενσωματωμένο ρολόι, το οποίο μετράει το χρόνο από τη στιγμή που ενεργοποιείται. Η πληροφορία αυτή είναι διαθέσιμη σε κάθε σημείο με κλήση της συνάρτησης millis() και επιστρέφει το χρόνο σε milliseconds που έχει περάσει από την ενεργοποίηση του Arduino. Για παράδειγμα:

```
float lastPress = millis();              // ξεκινάει η μέτρηση

if (lastPress – millis() > 1000) {       // αν η μέτρηση μέχρι το σημείο
                                         // αυτό έχει ξεπεράσει το 1 sec τότε
                                         // να εκτελεστεί το block κώδικα

}
```

4.1.5. Συνάρτηση Αντιστοίχησης Τιμών

Πολλές φορές χρειάζεται να αντιστοιχηθούν κάποιες τιμές που αντιστοιχούν σε κάποιο πεδίο τιμών με άλλες τιμές που ανήκουν σε άλλο πεδίο τιμών. Η μαθηματική πράξη είναι μία απλή και γνωστή διαδικασία, αλλά υπάρχει η δυνατότητα από το Arduino να γίνεται απευθείας καλώντας μία συνάρτηση για να την εκτελέσει. Αυτή είναι η:

```
map(<τιμή>, <κάτω_όριο_A>, <άνω_όριο_A>, <κάτω_όριο_B>, <άνω_όριο_B>)
```

Όπου:

- <τιμή> είναι η μεταβλητή προς μετατροπή
- <κάτω_όριο_A> είναι το κάτω όριο του πεδίου τιμών της μεταβλητής A
- <άνω_όριο_A> είναι το άνω όριο του πεδίου τιμών της μεταβλητής A
- <κάτω_όριο_B> είναι το κάτω όριο του πεδίου τιμών της μεταβλητής B (προς μετατροπής)
- <άνω_όριο_B> είναι το άνω όριο του πεδίου τιμών της μεταβλητής B (προς μετατροπής)

Ακολουθεί αναλυτικό παράδειγμα:

```
val = map(val, 0, 1023, 0, 179);     // μετατρέπεται μία αναλογική
                                     // τιμή που διαβάζεται από ένα
                                     // ποτενσιόμετρο (0 έως 1023)
                                     // σε μία τιμή για ένα σέρβο (0
                                     // έως 179)
```

4.1.6. Η Σειριακή Θύρα Επικοινωνίας

Το Arduino παρέχει σειριακή θύρα επικοινωνίας μεταξύ της πλακέτας και του Η/Υ. Για αυτό το σκοπό χρησιμοποιείται η σύνδεση μέσω του καλωδίου USB. Η ενεργοποίηση της σειριακής θύρας γίνεται εντός της συνάρτησης setup() με την εντολή Serial.begin(BaudRate), όπου το όρισμα BaudRate εκφράζει το ρυθμό μετάδοσης των bits πληροφορίας. Η τιμή ορίσματος που εξασφαλίζει λειτουργία χωρίς προβλήματα είναι 9600.

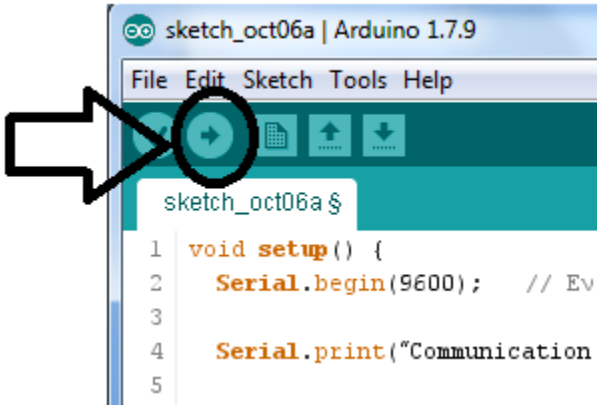
```
Serial.begin(9600); // Ενεργοποίηση της σειριακής θύρας με ροή bits
                    // 9600, εντός της setup()
```

Η σειριακή θύρα χρησιμοποιείται από τις εφαρμογές για αμφίδρομη επικοινωνία, δηλαδή να στέλνονται και να λαμβάνονται δεδομένα. Απλή περίπτωση χρήσης της επικοινωνίας είναι για εκσφαλμάτωση (debugging) των προγραμμάτων, δηλαδή να παρακολουθούνται οι τιμές των μεταβλητών μέσω της οθόνης σειριακής επικοινωνίας. Η εντολή που βοηθάει προς τούτο είναι η print(), που εκτυπώνει ένα μήνυμα ή τιμές, ή η println() που λειτουργεί με ίδιο τρόπο αλλάζοντας όμως γραμμή μετά την εκτύπωση. Για παράδειγμα:

```
Serial.begin(9600); // Ενεργοποίηση της σειριακής θύρας με ροή bits
                    // 9600, εντός της setup()


Serial.print("Communication begins!");   // εμφανίζεται το παρόν μήνυμα
                                         // χωρίς αλλαγή γραμμής μετά

/* γράφοντας εντός της loop() */

int distance = 100;                      // δηλώνεται ακέραια μεταβλητή
                                         // distance ίση με 100

Serial.print("Distance is ");            // εμφανίζεται το μήνυμα χωρίς αλλαγή
                                         // γραμμής

Serial.println(distance);                // εμφανίζεται η τιμή distance, δηλαδή
                                         //100 και ακολουθεί αλλαγή γραμμής
```

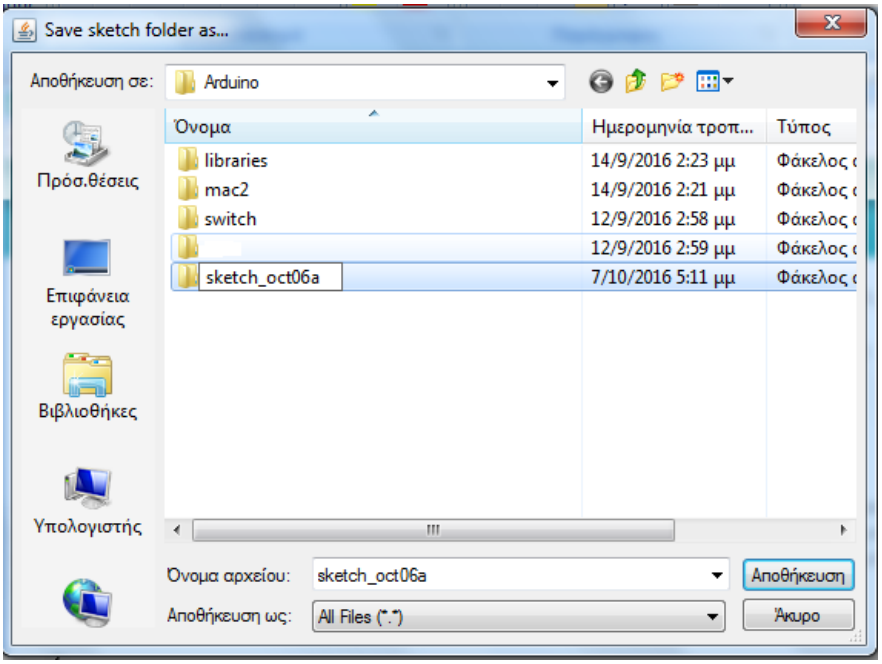
Αφού συνδεθεί η πλατφόρμα του Arduino μέσω του USB καλωδίου με τον Η/Υ και γραφτεί το πρόγραμμα στην κονσόλα του, για να ανοιχθεί η σειριακή οθόνη πρέπει να ενεργοποιηθεί από το περιβάλλον του Arduino. Αυτό γίνεται αφού προηγηθούν τα εξής βήματα:

1. Γίνεται μεταφόρτωση πρώτα του προγράμματος στον Arduino πατώντας το κουμπί που υποδεικνύεται στην παρακάτω εικόνα (Εικόνα 21).



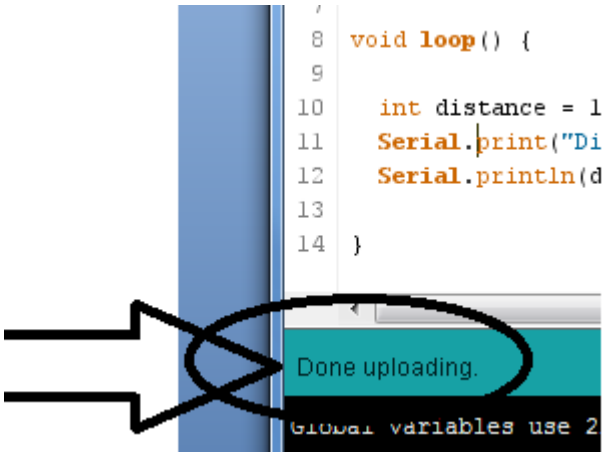
Εικόνα 21: Κουμπί μεταφόρτωσης του Arduino IDE

2. Ακολούθως γίνεται αποθήκευση σε φάκελο ομότιτλο με το όνομα του προγράμματος (Εικόνα 22).



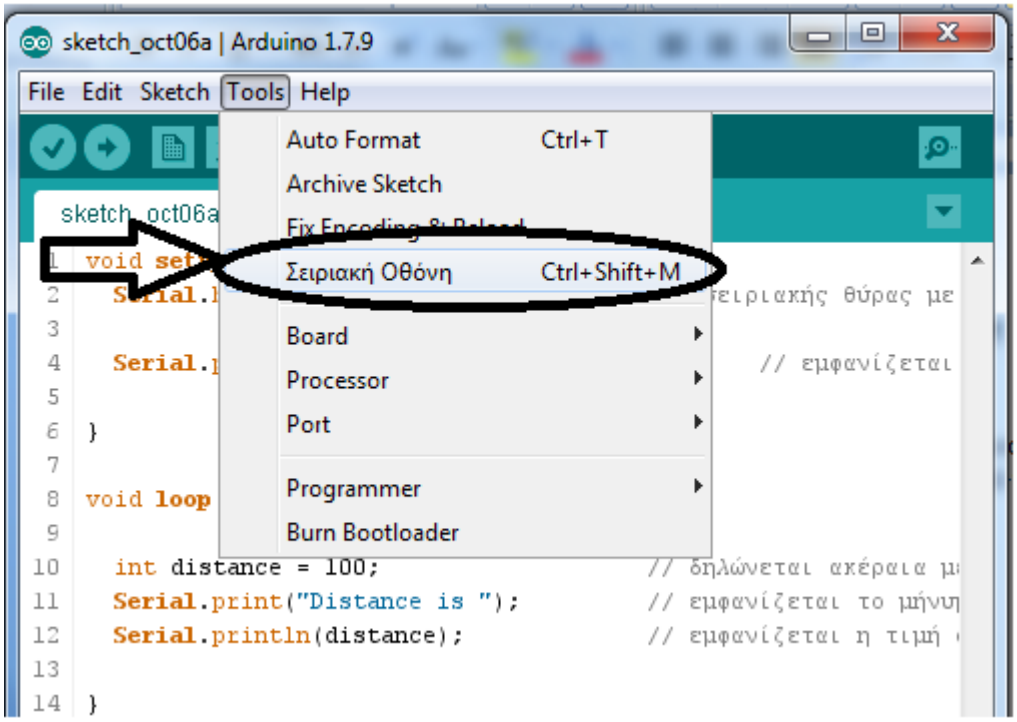
Εικόνα 22: Αποθήκευση του προγράμματος

3. Μικρή αναμονή έως ότου ο Arduino IDE επιβεβαιώσει ότι έγινε επιτυχώς η μεταφόρτωση (Εικόνα 23)



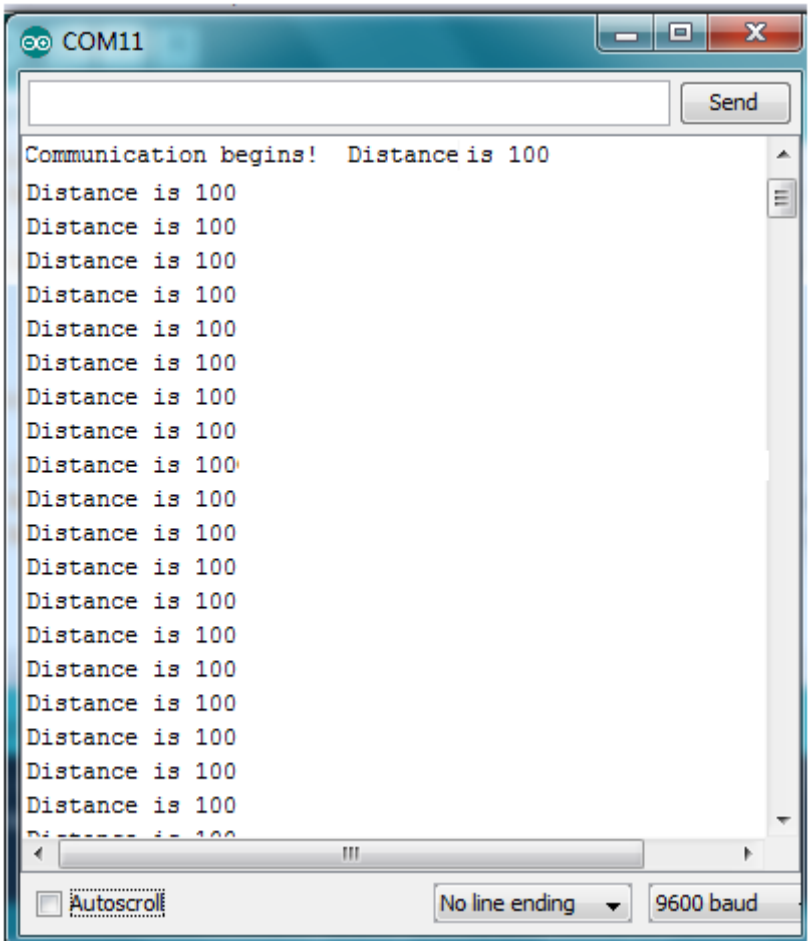
Εικόνα 23: Επιτυχής μεταφόρτωση

4. Η σειριακή οθόνη εμφανίζεται είτε με ταυτόχρονο πάτημα των κουμπιών πληκτρολογίου Shift + Control +m είτε από το Menu bar με επιλογή Tools => Σειριακή Οθόνη (Εικόνα 24)



Εικόνα 24: Εμφάνιση Σειριακής Οθόνης

Η εμφάνιση της σειριακής οθόνης με τα αποτελέσματα ακολουθεί στην Εικόνα 25.



Εικόνα 25: Αποτελέσματα στη σειριακή οθόνη

.

4.1.7. Συνθήκη Επιλογής if

Στον προγραμματισμό είναι αναγκαίο πολλές φορές να ελέγχεται κάποια συνθήκη, πριν αποφανθεί το πρόγραμμα ποιο τμήμα κώδικα θα εκτελέσει. Αυτό επιτυγχάνεται με τη χρήση της συνθήκης επιλογής if, η οποία συντάσσεται:

```
if <συνθήκη> { <τμήμα κώδικα 1> }

else { <τμήμα κώδικα 2> }
```

Όπου <συνθήκη> είναι ο έλεγχος που γίνεται, συνήθως χρησιμοποιώντας τους τελεστές σύγκρισης (>, <, ==, <>, >=, <=). Η συνθήκη αυτή μπορεί να γίνει και πιο σύνθετη χρησιμοποιώντας λογικούς τελεστές (|| για τη λογική πράξη Ή, && για τη λογική πράξη ΚΑΙ).

Στο κομμάτι {<τμήμα κώδικα 1>} εκτελούνται οι εντολές εφόσον η συνθήκη ικανοποιείται, αλλιώς εκτελούνται οι εντολές {<τμήμα κώδικα 2>}. Σε κάθε περίπτωση το else{<τμήμα κώδικα 2>} δεν είναι απαραίτητο να υπάρχει.

4.1.8. Δομή Επανάληψης for

Υπάρχει εντολή επανάληψης με σκοπό να επαναλαμβάνει ένα κομμάτι κώδικα όσες φορές χρειάζεται μετρώντας τις επαναλήψεις με κάποιο σταθερό βήμα που ορίζεται από τον προγραμματιστή και ελέγχοντας κάθε φορά μία συνθήκη. Αυτή είναι η δομή επανάληψης for η οποία έχει την ακόλουθη σύνταξη:

```
for (<αρχική_τιμή>; <συνθήκη_τερματισμού>; <βήμα>)
{
        <εντολές>
}
```

Όπου:

<αρχική_τιμή> δίνεται τιμή εκκίνησης μίας μεταβλητής π.χ. i = 0

<βήμα> είναι το βήμα επανάληψης της δομής π.χ. i++ = i+1 δηλαδή η τιμή i από 0 να γίνει 1 στην επόμενη επανάληψη

<συνθήκη_τερματισμού> η συνθήκη που όσο ισχύει εκτελείται η δομή επανάληψης, όταν πάψει να ισχύει η συνθήκη ο κώδικας φεύγει από τη δομή for και συνεχίζει στον επόμενο κώδικα.

Ακολουθεί παράδειγμα υλοποίησης:

```
setup() {
        int ledPin = 12;
        pinMode(ledPin,OUTPUT);
}
loop() {
        for (int i=0;i<10;i++){
                digitalWrite(ledPin,HIGH);
                digitalWrite(ledPin,LOW);
        }
}
```

Στο ανωτέρω παράδειγμα θα πραγματοποιηθούν συνολικά 10 αναβοσβησίματα ενός led που έχει συνδεθεί στην ψηφιακή έξοδο 12 (αφού προηγηθεί κατάλληλη συνδεσμολογία στην πλατφόρμα του Arduino Mega).

Εντολές επανάληψης υπάρχουν και χωρίς προκαθορισμένο αριθμό βημάτων, ελέγχοντας συνεχώς την ισχύ της συνθήκης. Αυτές είναι:

- Επαναληπτικός τύπος while

```
        while <συνθήκη> {
                <εντολές>
        }
```

Μόνο όταν πάψει η ισχύς της συνθήκης ο κώδικας φεύγει από τη δομή επανάληψης while.

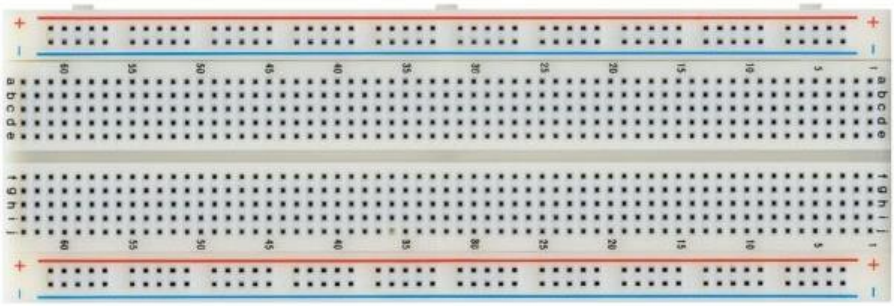
- Επαναληπτικός τύπος repeat

```
repeat {
        <εντολές>
}
until <συνθήκη>
```

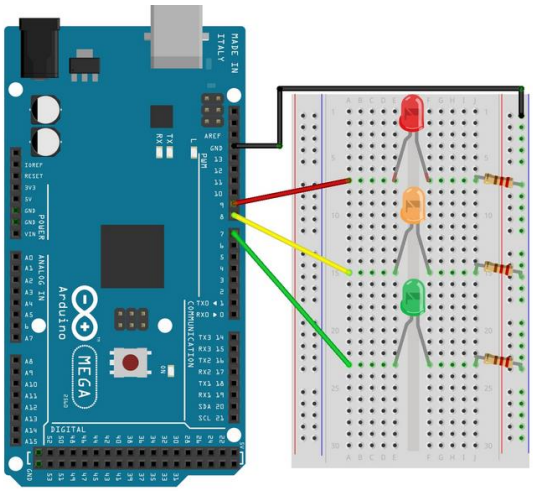
Οι εντολές της επαναληπτικής δομής εκτελούνται όσο δεν είναι αληθής η συνθήκη.

4.2. Χρήση Breadboard

Τα κυκλώματα που υλοποιούνται συνήθως περιλαμβάνουν αρκετά ηλεκτρονικά στοιχεία (φωτοδιόδοι led, αναλογικοί ή ψηφιακοί αισθητήρες ή και τα δύο ταυτόχρονα, αντιστάσεις, πυκνωτές κτλ.), οπότε κρίνεται αναγκαία η χρήση του breadboard για τη δημιουργία πρωτότυπων πειραματικών κυκλωμάτων, χωρίς τη χρήση κολλήσεων. Τα καλώδια και τα ηλεκτρονικά στοιχεία εισάγονται στις ειδικές οπές που απέχουν μεταξύ τους 0,1 ίντσες (2,54 mm) και βραχυκυκλώνονται κάθετα ώστε να διευκολύνεται η σύνδεση όλων των εξαρτημάτων ακόμη και χωρίς καλώδια. Στο ακριανό μέρος ενός breadboard διακρίνονται οι γραμμές «+» και «–» που είναι οριζόντια βραχυκυκλωμένες και συνήθως συνδέεται στο «+» τροφοδοσία από τα 5 Volts που διαθέτει ο Arduino και στο «-» η γείωση από το GND του Arduino. Στην εικόνα 26 φαίνεται ένα τυπικό breadboard και στην εικόνα 27 ένα απλό παράδειγμα συνδεσμολογίας κυκλώματος.



Εικόνα 26: Ένα τυπικό breadboard



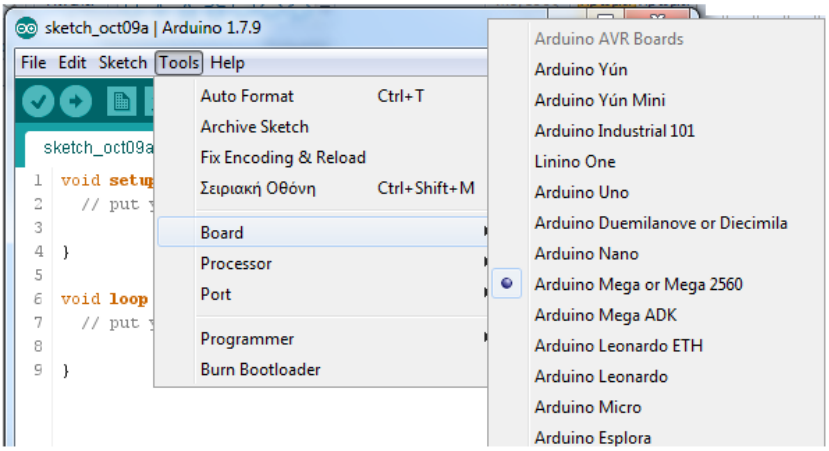
Εικόνα 27: Παράδειγμα υλοποίησης εφαρμογής φωτεινού σηματοδότη

Αναλυτικότερη πληροφόρηση μπορεί να βρεθεί και στον ιστότοπο της εταιρείας SparkFun στην αγγλική γλώσσα http://learn.sparkfun.com/tutorials/how-to-use-a-breadboard

4.3. Μεταφόρτωση Προγράμματος στον Arduino

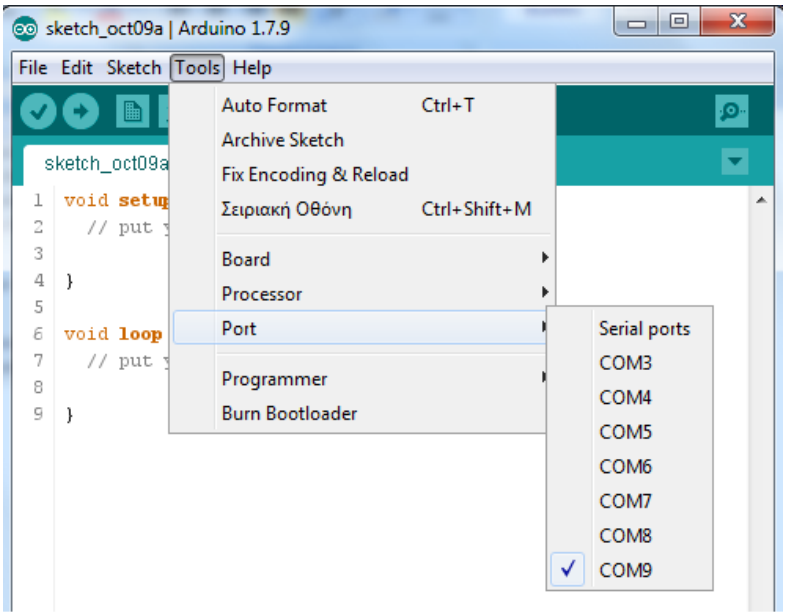
Για να μεταφορτωθεί κάποιο πρόγραμμα στον Arduino, θα πρέπει να έχει προηγηθεί σύνδεση με τον Η/Υ μέσω USB. Ο Η/Υ αναγνωρίζει την σειριακή θύρα διασύνδεσης του Arduino, γεγονός που επαληθεύεται και από την "Διαχείριση Συσκευών" του "Πίνακα Ελέγχου" του Η/Υ.

Αρχικά, από το Menu Bar στην επιλογή Tools => Board επιλέγεται η μονάδα εργασίας Arduino (Εικόνα 28).



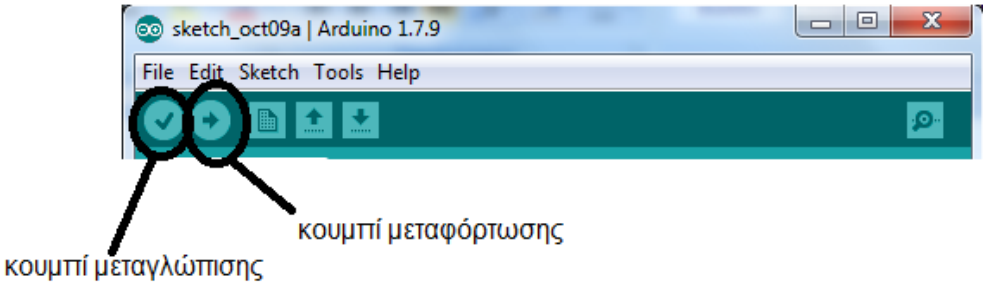
Εικόνα 28: Επιλογή μονάδας Arduino

Εν συνεχεία επιλέγεται η σειριακή θύρα από το Menu Bar, Tools=>Port και η σειριακή θύρα COM που αντιστοιχεί στον Arduino (είναι ορατή όπως προαναφέρθηκε από τη “Διαχείριση Συσκευών” του “Πίνακα Ελέγχου”).



Εικόνα 29: Ορισμός σειριακής θύρας επικοινωνίας

Αφού εκτελεστούν τα προαναφερθέντα βήματα και έχει γραφτεί κάποιο πρόγραμμα στο Arduino IDE, μεταγλωττίζεται πατώντας το κατάλληλο κουμπί από το Menu Bar (Εικόνα 30) και εφόσον ευρεθεί χωρίς λάθη μεταφορτώνεται, με το αντίστοιχο κουμπί, στην μονάδα Arduino (Εικόνα 30).



Εικόνα 30: Το κουμπί μεταγλώττισης και το κουμπί μεταφόρτωσης ενός προγράμματος προς τη μονάδα Arduino

4.4. Παραδείγματα Υλοποιήσεων με Arduino

Στην παρούσα ενότητα παρουσιάζεται η υλοποίηση τριών ενδεικτικών ολοκληρωμένων παραδειγμάτων χρήσης της μονάδος Arduino και του Arduino IDE, ώστε να γίνει περισσότερο αντιληπτή η υλοποίηση μίας διάταξης.

4.4.1. Παράδειγμα 1. Φωτοδίοδος (led) που Αναβοσβήνει

Στο πρώτο πρόγραμμα που παρουσιάζεται, πραγματοποιείται το αναβοσβήσιμο ενός led ανά ένα δευτερόλεπτο. Δηλαδή από το led περνάει ρεύμα για ένα δευτερόλεπτο και το επόμενο δευτερόλεπτο δεν υπάρχει διέλευση. Η διαδικασία επαναλαμβάνεται συνεχώς.

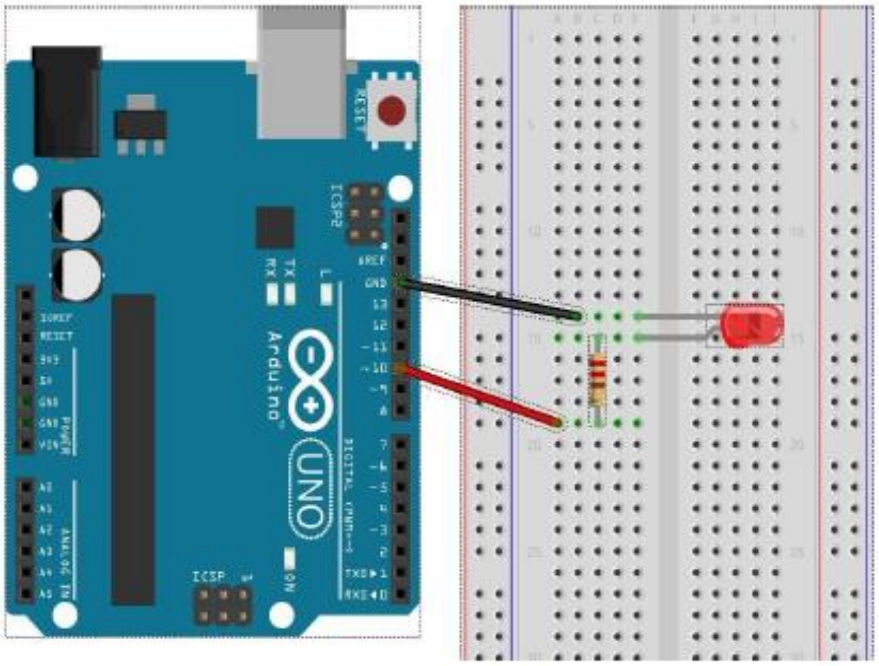
Για την υλοποίηση του κυκλώματος απαιτείται ένα led, μία αντίσταση 220 Ω και ένα breadboard. Η κατάλληλη αντίσταση υπολογίζεται βάσει του τύπου:

$$R = (V_s - V_{led}) / I$$

Όπου:

- V_s είναι η τάση τροφοδοσίας από τον Arduino προς το κύκλωμα και είναι 5 Volts
- V_{led} είναι η τροφοδοσία ενός led και είναι ίση με 2 Volts
- I είναι το ρεύμα γραμμής ή ρεύμα του led ή το ρεύμα του κυκλώματος. Για να λειτουργήσει σωστά ένα μικρό led είναι αποδεκτό ρεύμα 15mA έως 20 mA.

Τυπικά, μία αντίσταση της τάξης 220 Ω καλύπτει τα περισσότερα led, ώστε να προστατεύονται από κάποια υπέρταση. Στην εικόνα που ακολουθεί (Εικόνα 31) παρουσιάζεται το προς υλοποίηση κύκλωμα.



Εικόνα 31: Κύκλωμα 1ου παραδείγματος

Η έξοδος του ρεύματος στο εν λόγω κύκλωμα είναι στη θύρα 10, ενώ πρέπει να προσεχθεί η σύνδεση της αντίστασης να γίνει ανάμεσα στην έξοδο της θύρας και το led. Πρέπει να σημειωθεί ότι το μακρύτερο πόδι από το led συνδέεται προς τη θετική πολικότητα και το μικρότερο προς τη γείωση, όπως και συμβαίνει.

Ακολούθως, απομένει ο προγραμματισμός του κυκλώματος σύμφωνα με όσα έχουν ήδη αναφερθεί. Η μορφή των βημάτων θα είναι:

1. Να δηλωθεί μία μεταβλητή που αντιπροσωπεύει τη θύρα 10΄
2. Να οριστεί η θύρα 10 ως έξοδος
3. Θα πρέπει για ένα δευτερόλεπτο να υπάρχει διέλευση ρεύματος, κατά τη διάρκεια του οποίου θα ανάβει το led
4. Στο επόμενο δευτερόλεπτο διακόπτεται η τροφοδοσία προς τη θύρα 10

Τα βήματα 3,4 επαναλαμβάνονται συνεχώς.

```
int ledPin = 10;                    // η μεταβλητή ledPin αντιπροσωπεύει
                                    // τη ψηφιακή θύρα 10

void setup(){

        pinMode(ledPin,OUTPUT);     // η θύρα 10 ορίζεται σαν έξοδος

}

void loop(){

        digitalWrite(ledPin, HIGH); // ενεργοποιείται η τροφοδοσία
                                    // της θύρας 10

        delay(1000);                // και παραμένει για ένα
                                    // δευτερόλεπτο

        digitalWrite(ledPin,LOW);   // απενεργοποιείται η
                                    // τροφοδοσία της θύρας 10

        delay(1000);                // και παραμένει ως έχει για ένα
                                    // δευτερόλεπτο

}                                   // η loop() θα ξαναεκτελεστεί
```

4.4.2. Παράδειγμα Χρήσης της Σειριακής Οθόνης και της Δομής Επανάληψης for

Στο παρόν παράδειγμα που παρουσιάζεται, χρησιμοποιείται η σειριακή οθόνη στέλνοντας μηνύματα και τιμές μεταβλητών από ένα πρόγραμμα που έχει φορτωθεί στη μονάδα του Arduino.

Η ανάλυση του αλγορίθμου έχει ως εξής:

1. Ενεργοποιείται η σειριακή θύρα στη συνάρτηση setup() με ρυθμό δεδομένων 9600.
2. Στη συνάρτηση loop(), σε κάθε επανάληψή της, στέλνεται ένα μήνυμα της μορφής “Communication Started: ” χωρίς αλλαγή γραμμής.
3. Με μία επαναληπτική διαδικασία δίνονται τιμές σε μία μεταβλητή από το 1 μέχρι το 20 και οι τιμές αυτές στέλνονται στη σειριακή οθόνη για εκτύπωση ανά γραμμή.

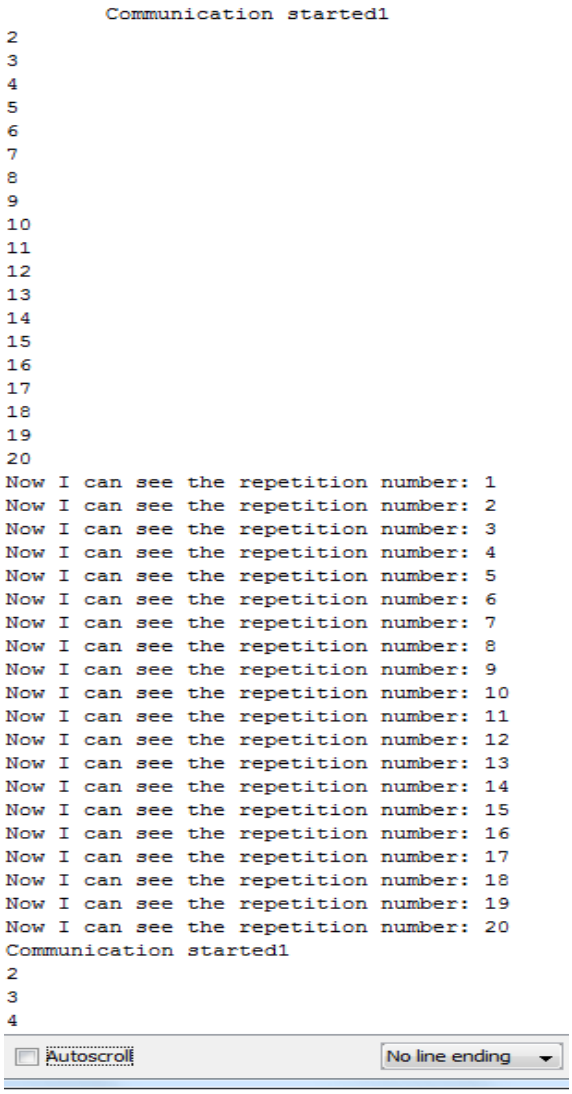
4. Στη συνέχεια φτιάχνεται ένα πιο σύνθετο μήνυμα, χρησιμοποιώντας κείμενο και τιμή μεταβλητής και γίνεται εκτύπωση του στη σειριακή οθόνη με και χωρίς αλλαγή οθόνης π.χ. “Now, I can see the repetition number: X” όπου X η αντίστοιχη τιμή της μεταβλητής σε κάθε επανάληψη.

Αφού συνδεθεί η μονάδα του Arduino και επιλεχτεί ο τύπος της μονάδας και η θύρα επικοινωνίας, μεταγλωττίζεται και μεταφορτώνεται το πρόγραμμα. Ακολούθως, ενεργοποιείται η σειριακή οθόνη και παρακολουθούνται τα μηνύματα που εμφανίζονται σε αυτήν. Ακολουθεί ο κώδικας με τα σχόλια του:

```
int i;          // ορίζεται ακέραια μεταβλητή i


void setup(){

    Serial.begin(9600);          // εκκίνηση σειριακής οθόνης

}

void loop(){

    Serial.print(“Communication started”);   // εμφάνιση μηνύματος
                                             // στην σειριακή οθόνη

    for(i=1;i<=20;i++){

        Serial.println(i);              // τυπώνεται η τιμή του I, με
                                        // αλλαγή γραμμής

        delay(100);          // σταματάει η εκτέλεση του
                             //προγράμματος για 100 msec

    }

    for(i=1;i<=20;i++){

        Serial.print(“Now I can see the repetition number: “);
/*τύπωμα στη σειριακή οθόνη μηνύματος χωρίς αλλαγή γραμμής*/

        Serial.println(i);   // όπως παραπάνω στο αντίστοιχο
                             // μήνυμα
```

```
                delay(200);    // σταματάει η εκτέλεση του κώδικα για
                               // 200 msec

        }

}
```

Στην ακόλουθη εικόνα (Εικόνα 32) φαίνεται η σειριακή οθόνη με τα μηνύματα της.



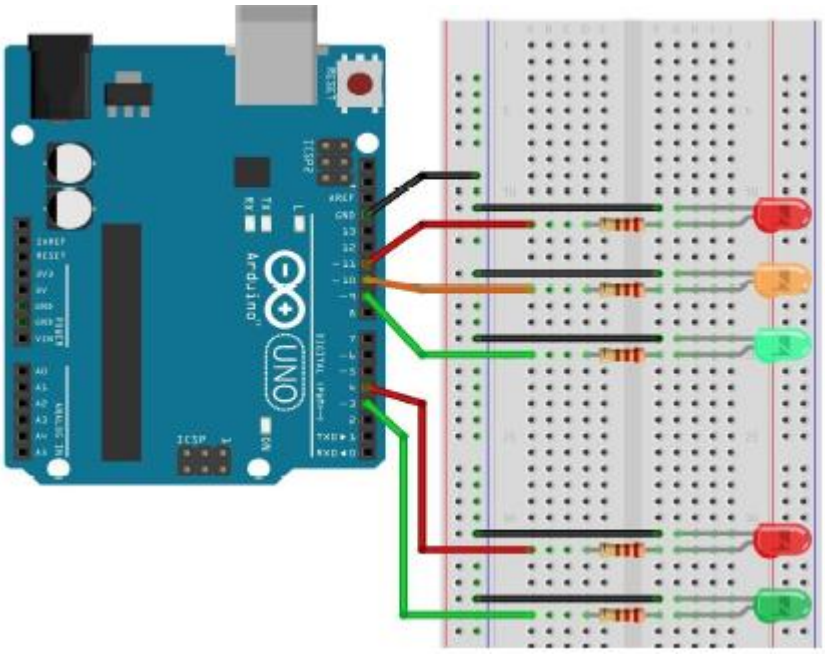
Εικόνα 32: Τα μηνύματα στη σειριακή οθόνη του Arduino

4.1.3. Τα Φανάρια Κυκλοφορίας

Στο παρόν παράδειγμα προσομοιώνεται η λειτουργία των φανών κυκλοφορίας. Δίνοντας κατάλληλα χρονικά διαστήματα, ανάβουν διαδοχικά τα τρία φανάρια κυκλοφορίας σε κατάλληλο συνδυασμό με τα φανάρια των πεζών.

Για την υλοποίηση του φαναριού που υποδεικνύει την κίνηση στα αυτοκίνητα χρειάζονται τρία led, ένα κόκκινο, ένα πράσινο και ένα πορτοκαλί και για το φανάρι των πεζών δύο led, ένα πράσινο και ένα κόκκινο. Όλα τα led χρειάζεται να συνοδεύονται από μία αντίσταση προστασίας τους, σε σύνδεση σε σειρά, των 220 Ohm.

Παρατίθεται στην παρακάτω εικόνα το κύκλωμα (Εικόνα 33).



Εικόνα 33: Κύκλωμα παραδείγματος

Ακολουθεί ο κώδικας προγραμματισμού:

```
int ledRed = 11;          // η κόκκινη σήμανση στο φανάρι αυτοκινήτων
                          // αντιστοιχεί στη ψηφιακή θύρα 11

int ledOrange = 10;       // η πορτοκαλί σήμανση στο φανάρι αυτοκινήτων
                          // αντιστοιχεί στη ψηφιακή θύρα 10
```

```
int ledGreen = 9;          // η πράσινη σήμανση στο φανάρι αυτοκινήτων
                           // αντιστοιχεί στη ψηφιακή θύρα 9

int pedRed = 4;            // η κόκκινη σήμανση στο φανάρι πεζών,
                           // αντιστοιχεί στη ψηφιακή θύρα 4

int pedGreen = 3;          // η πράσινη σήμανση στο φανάρι πεζών,
                           // αντιστοιχεί στη ψηφιακή θύρα 3

void setup() {

        pinMode(ledRed, OUTPUT);      // ορίζεται η ψηφιακή θύρα 11 σαν
                                      // έξοδος

        pinMode(ledOrange, OUTPUT);   // ορίζεται η ψηφιακή θύρα 10 σαν
                                      // έξοδος

        pinMode(ledGreen, OUTPUT);    // ορίζεται η ψηφιακή θύρα 9 σαν
                                      // έξοδος

        pinMode(pedRed, OUTPUT);      // ορίζεται η ψηφιακή θύρα 4 σαν
                                      // έξοδος

        pinMode(pedGreen, OUTPUT);    // ορίζεται η ψηφιακή θύρα 3 σαν
                                      // έξοδος

}

void loop() {

        /* κόκκινο για 3 δευτερόλεπτα, πράσινο στους πεζούς */

        digitalWrite(ledRed, HIGH);       // ενεργοποιείται η ψηφιακή θύρα 11

        digitalWrite(ledOrange, LOW);     // απενεργοποιείται η ψηφιακή θύρα
                                          // 10

        digitalWrite(ledGreen, LOW);      // απενεργοποιείται η ψηφιακή θύρα
                                          // 9

        digitalWrite(pedRed, LOW);        // απενεργοποιείται η ψηφιακή θύρα
                                          // 4

        digitalWrite(pedGreen, HIGH);     // ενεργοποιείται η ψηφιακή θύρα 3

        delay(3000);                      // σταματάει η εκτέλεση του
                                          // προγράμματος για 3 sec
```

```
        /* κόκκινο στους πεζούς, αναμονή για 1 δευτερόλεπτο πριν δοθεί
        πράσινο στα αμάξια */
        digitalWrite(pedRed, HIGH);
        digitalWrite(pedGreen, LOW);
        delay(1000);
        /* πράσινο κυκλοφορίας για 5 δευτερόλεπτα */
        digitalWrite(ledRed, LOW);
        digitalWrite(ledOrange, LOW);
        digitalWrite(ledGreen, HIGH);
        delay(5000);
        /* πορτοκαλί για 1 δευτερόλεπτο */
        digitalWrite(ledRed, LOW);
        digitalWrite(ledOrange, HIGH);
        digitalWrite(ledGreen, LOW);
        delay(1000);
}
```

5. Η Γλώσσα SQL

Στην παρούσα διπλωματική εργασία σημαντικό ρόλο κατέχει η χρήση του SQL Server 2014. Είναι ένα πρόγραμμα το οποίο έχει αναπτυχθεί από την εταιρεία λογισμικού Microsoft και λειτουργεί ως “σύστημα διαχείρισης σχεσιακών βάσεων δεδομένων”.

Με τον όρο “βάση δεδομένων”, εννοείται μία συλλογή από εγγραφές και αρχεία, τα οποία είναι οργανωμένα με τέτοιο τρόπο ώστε να εξυπηρετείται ένας συγκεκριμένος σκοπός. Τέτοιες εγγραφές μπορεί να είναι οικονομικά στοιχεία από πελάτες ή προμηθευτές ή καταχωρήσεις από βιβλία ισολογισμών ή συλλογή από επιστολές που κρατούνται και στοιχειοθετούνται με βάση τον παραλήπτη κτλ. Μία πιο οργανωτική προσέγγιση είναι οι εκατοντάδες αυτές εγγραφές να αποθηκεύονται μέσα σε φακέλους ή και υποφακέλους. Ο χρήστης που οργανώνει κατά αυτόν τον τρόπο λογίζεται ως ο “διαχειριστής” της βάσης δεδομένων. Όταν όμως ο όγκος δεδομένων είναι τεράστιος και “ξεφεύγει” των δυνατοτήτων του διαχειριστή, είναι αναγκαία η χρησιμοποίηση ενός “συστήματος διαχείρισης των βάσεων δεδομένων”.

Ο όρος “σχεσιακή” βάση δεδομένων απευθύνεται στην συγγένεια που μπορεί να έχουν δεδομένα από διαφορετικούς πίνακες αποθήκευσης αυτών των δεδομένων ή εγγραφών. Έτσι μία μεταβολή που μπορεί να συμβεί στα στοιχεία της μίας βάσης δεδομένων επηρεάζει ανάλογα και τα στοιχεία με τις εγγραφές μίας άλλης βάσης δεδομένων που “σχετίζεται” με αυτήν. Ο βασικός λόγος που χρησιμοποιούνται τέτοιου είδους συστήματα από τις εταιρείες και τους οργανισμούς είναι η οργάνωση και η ανάκτηση δεδομένων όποια στιγμή ζητηθεί.

Η ανάκτηση των δεδομένων από τη βάση δεδομένων, γίνεται με τη μορφή ερωτήσεων (queries) που υποβάλλονται από τον χρήστη. Οι ερωτήσεις συντάσσονται χρησιμοποιώντας τη γλώσσα SQL (Structured Query Language). Στον SQL Server 2014 υπάρχει η T-SQL μία μικρή παραλλαγή της ANSI SQL (της κατά πρότυπο γλώσσας SQL).

Πρόγονος της SQL ήταν η γλώσσα SEQUEL (Structured English QUEry Language) που αναπτύχθηκε από την εταιρεία IBM στα μέσα της δεκαετίας του 1970. Ο λόγος ανάπτυξης της δε διαφέρει από τους σημερινούς λόγους ύπαρξης της, δηλαδή ώστε να χειρίζονται και να ανακτώνται τα αποθηκευμένα δεδομένα από τη σχεσιακή βάση δεδομένων της εταιρείας την System R. Αργότερα έγινε η μετονομασία σε SQL λόγω του ότι η ονομασία SEQUEL ήταν κατοχυρωμένη από μία βρετανική εταιρεία αεροσκαφών.

5.1. Χρήση Ερωτημάτων (Queries)

Ακολουθεί ένα παράδειγμα χρήσης ερωτημάτων με μία θεωρητική βάση δεδομένων Students που κατέχει στοιχεία τριών μαθητών (Εικόνα 34).

Students

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@eecs	18	3.2
53650	Smith	smith@math	19	3.8

Εικόνα 34: Η βάση δεδομένων, με 5 στήλες, τη στήλη μητρώου, τη στήλη επιθέτου, τη στήλη login, τη στήλη ηλικίας και τη στήλη μέσου όρου βαθμού

Για να βρεθούν οι μαθητές ηλικίας 18 ετών διατυπώνεται το παρακάτω ερώτημα:

```
SELECT *                    // επιλογή όλων των στοιχείων
FROM STUDENTS S             // από το όνομα βάσης STUDENTS
                            // και που ορίζεται μεταβλητή STUDENTS S
WHERE S.age = 18            // όπου έχουν ηλικία = 18
```

Στην ακόλουθη εικόνα (Εικόνα 35) φαίνονται τα πλήρη αποτελέσματα (με όλα τα διαθέσιμα στοιχεία) όσων έχουν τη ζητούμενη ηλικία:

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@ee	18	3.2

Εικόνα 35: Αποτελέσματα ερωτήματος

Αν αντί για πλήρη αποτελέσματα, ο χρήστης ζητήσει μόνον τα ονόματα και το login, το ερώτημα διατυπώνεται ως εξής:

```
SELECT S.NAME, S.LOGIN      // επιλογή των στοιχείων όνομα και
                            // login
FROM STUDENTS S             // από το όνομα βάσης STUDENTS
                            // και που ορίζεται μεταβλητή STUDENTS S
WHERE S.age = 18            // όπου έχουν ηλικία = 18
```

Εμφανίζεται το παρακάτω αποτέλεσμα (Εικόνα 36).

name	login
Jones	jones@cs
Smith	smith@ee

Εικόνα 36: Αποτελέσματα ερωτήματος

Έστω ότι διατίθεται προς παράλληλη χρήση με την προηγούμενη βάση δεδομένων και μία άλλη βάση δεδομένων με εγγραφές βαθμών μαθητών σε διάφορα μαθήματα (Εικόνα 37).

Enrolled

sid	cid	grade
53831	Carnatic101	C
53831	Reggae203	B
53650	Topology112	A
53666	History105	B

Εικόνα 37: Βάση δεδομένων με εγγραφές

Ζητούμενο είναι να βρεθεί ποιος είναι ο μαθητής που ο αριθμός μητρώου του ταυτίζεται και στις δύο βάσεις δεδομένων (Εικόνα 34 και Εικόνα 37), έχει βαθμό B και σαν αποτέλεσμα να εμφανιστούν το όνομα του και το όνομα του μαθήματος. Οπότε η διατύπωση του ερωτήματος (query) θα είναι ως εξής:

```
SELECT S.name, E.cid          // να εμφανίζεται όνομα μαθητή και
                              // όνομα μαθήματος
FROM Student S, Enrolled E    // από το όνομα βάσης STUDENTS
                              // και που ορίζεται μεταβλητή STUDENTS S
                              // και από το όνομα βάσης ENROLLED
                              // που ορίζεται μεταβλητή ENROLLED E
WHERE S.sid = E.sid AND E.grade = 'A' // όπου ο αριθμός μητρώου
                                      // υπάρχει και στις δύο βάσεις
                                      // και έχει βαθμό 'A'
```

Στην Εικόνα 38, φαίνονται τα αποτελέσματα του ερωτήματος:

S.name	E.cid
Smith	Topology112

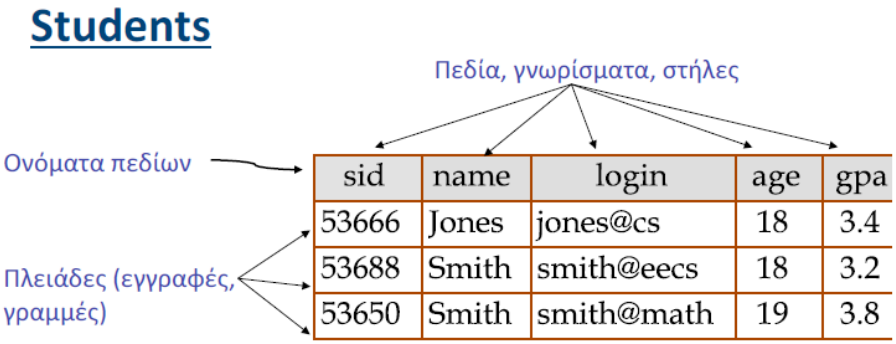
Εικόνα 38: Αποτέλεσμα ερωτήματος

5.2. Δημιουργία Πινάκων και Βασικές Εντολές

Έχει ήδη αναφερθεί πως με τον όρο "Σχεσιακή βάση Δεδομένων" ορίζεται η σχέση ενός συνόλου πινάκων. Η λέξη κλειδί της προηγούμενης πρότασης είναι η "σχέση", καθώς μέσα της περικλείει 2 συστατικά.

1. Το Στιγμιότυπο, ένας πίνακας από γραμμές και στήλες
2. Το Σχήμα, που προσδιορίζει το όνομα κάθε πίνακα (ή σχέσης) και τα ονόματα και τους τύπους κάθε στήλης, όπως για παράδειγμα στην Εικόνα 34, όπου υπάρχει η βάση δεδομένων των μαθητών Students και προσδιορίζονται οι τύποι της κάθε στήλης ως εξής: Students (sid:string, name string, login string, age: Integer, gpa: real)

Μία σχέση πρακτικά θεωρείται το σύνολο των γραμμών που αποτελούν τον πίνακα, καθώς οι γραμμές είναι διαφορετικές μεταξύ τους. Στο σχήμα 39 φαίνονται τα προαναφερθέντα:



Εικόνα 39: Στιγμιότυπο της σχέσης (ή του πίνακα) Student

Η δημιουργία του παραπάνω πίνακα στη γλώσσα SQL γίνεται ως εξής:

```
CREATE TABLE Students          // εντολές για τη δημιουργία (CREATE)
                               // της σχέσης (TABLE) που ονομάζεται
                               // Student
    (sid CHAR(20),             // το πεδίο student id δέχεται μέχρι 20
                               // χαρακτήρες (char)
    name CHAR(20),             // το πεδίο name δέχεται μέχρι 20
                               // χαρακτήρες (char)
    login CHAR(10),            // το πεδίο login δέχεται μέχρι 10
                               // χαρακτήρες (char)
    age INTEGER,               // το πεδίο age δέχεται ακέραιους
                               // (integers) αριθμούς
    gpa REAL)                  // το πεδίο gpa δέχεται πραγματικούς
                               // (real) αριθμούς
```

Κατά παρόμοιο τρόπο γίνεται και η δημιουργία του πίνακα (ή σχέσης) Enrolled (Εικόνα 37), που είναι ο πίνακας με τα μαθήματα που επιλέγουν οι μαθητές.

```
CREATE TABLE ENROLLED
    (sid CHAR(20),
    cid CHAR(20),              // πεδίο course id
    grade CHAR(2) )            // πεδίο βαθμού
```

Αφού παρουσιάστηκαν οι εντολές και ο τρόπος δημιουργίας πίνακα (ή σχέσεων), ακολούθως παρατίθενται κάποιες βασικές εντολές σχετικά με ακύρωση και μεταβολή των πινάκων (σχέσεων).

Η εντολή ακύρωσης μίας σχέσης:

```
DROP TABLE <Όνομα_Πίνακα>
```

όπου η πληροφορία του πίνακα και οι γραμμές του διαγράφονται. Με βάση το παράδειγμα της ενότητας, μπορεί να συνταχθεί:

```
DROP TABLE Students
```

Σαν μεταβολή του πίνακα εννοείται το γεγονός να προστεθεί μία στήλη (ή ένα πεδίο), το οποίο θα λάβει αρχικά κενή τιμή (ή null).

```
ALTER TABLE <Όνομα_Πίνακα>          // μεταβολή στον πίνακα

ADD COLUMN <Πεδίο_Πίνακα> <Τύπος Πεδίου>      // προσθέτοντας
                                                // πεδίο
```

με βάση το υπάρχον παράδειγμα η σύνταξη θα είναι:

```
ALTER TABLE Students

ADD COLUMN Year integer
```

όπου δηλώνεται το έτος γέννησης ενός μαθητή στον πίνακα Students.

Παρόμοια, μπορεί να καταργηθεί (ακυρωθεί) ένα πεδίο από τον πίνακα. Για αυτό το λόγο η γνωστή εντολή DROP συντάσσεται ως εξής:

```
ALTER TABLE <Όνομα_Πίνακα>

DROP COLUMN <Όνομα Στήλης>
```

και βάσει του παραδείγματος:

```
ALTER TABLE Students

DROP COLUMN Year
```

Πέραν των στηλών, υπάρχει η δυνατότητα μεταβολής και των γραμμών ενός πίνακα, δηλαδή να διαγραφεί ή να προστεθεί μία γραμμή με στοιχεία.

Η πρόσθεση μίας επιπλέον γραμμής γίνεται με την εντολή INSERT INTO, η οποία συντάσσεται ως εξής:

```
INSERT INTO <Όνομα_Πίνακα> (Πεδίο1, Πεδίο 2, …)

/* να προστεθεί στον πίνακα <Όνομα Πίνακα> με τα ήδη υπάρχοντα πεδία */

VALUES (Τιμή1, Τιμή 2, …)

/* γραμμή με τιμές αντίστοιχες των πεδίων */
```

Σαν παράδειγμα αναφέρεται:

```
INSERT INTO Students (sid, name, login, age, gpa)

VALUES (53688, 'Smith', 'smith@ee', 18, 3.2)
```

Σαν ερώτηση (query) μπορεί να συνταχθεί και η εντολή διαγραφής μίας γραμμής υπό συνθήκη, η οποία συντάσσεται ως εξής:

```
DELETE FROM <Όνομα_Πίνακα> <Μεταβλητή_Πίνακα>

WHERE <Μεταβλητή_Πίνακα>.<Πεδίο_Πίνακα> = <Συνθήκη>
```

Παραδείγματος χάριν:

```
DELETE FROM Students S

WHERE S.name = 'Smith'
```

5.3. Ο Ρόλος του Κλειδιού στην SQL

Σε κάθε πίνακα μίας σχεσιακής βάσης δεδομένων, είναι υποχρεωτικός ο ορισμός ενός “πρωτεύοντος κλειδιού” (primary key), ο οποίος είναι αυτός που χαρακτηρίζει τον πίνακα και τα στοιχεία του κατά μοναδικό τρόπο. Για να γίνει η προηγούμενη πρόταση αντιληπτή γίνεται αναδρομή στον πίνακα Students της Εικόνας 34, όπου εμφανίζονται οι εγγραφές των μαθητών. Στον πίνακα υπάρχουν 2 μαθητές με το ίδιο επίθετο, το Smith, οπότε κρίνεται πως το επίθετο (name στήλη στον πίνακα) δεν μπορεί με μοναδικό τρόπο να χαρακτηρίσει τον πίνακα. Κρίνεται πως μόνον ο αριθμός μητρώου είναι η στήλη που μπορεί να χαρακτηρίσει με μοναδικό τρόπο κάθε φοιτητή, ακόμη και σε περίπτωση συνωνυμίας επιθέτου, ονόματος, ονόματος πατέρα κτλ. Έτσι στο υπάρχον παράδειγμα ρόλο πρωτεύοντος κλειδιού (primary key) έχει το πεδίο sid (student id).

Στην SQL, όταν επιθυμείται να γίνει σύνδεση του αρχικού πίνακα δεδομένων με έναν πίνακα εξαρτώμενο από τον αρχικό πίνακα στοιχείων, χρησιμοποιείται το “ξένο κλειδί” (foreign key). Για να γίνει κατανοητή η προαναφερθείσα πρόταση εισάγεται ένα νέο παράδειγμα (Εικόνα 40).

ΕΙΔΟΣ

ΚΩΔΙΚΟΣ	ΟΝΟΜΑΣΙΑ	ΔΙΑΘΕΣΙΜΗ ΠΟΣΟΤΗΤΑ	ΚΩΔΙΚΟΣ ΠΡΟΜΗΘΕΥΤΗ
1111	ΜΟΛΥΒΙ	1002	02
2222	ΓΟΜΑ	200	06
3333	ΞΥΣΤΡΑ	3234	02
4444	ΣΤΥΛΟ	2378	01
5555	ΤΕΤΡΑΔΙΟ	567	01

ΠΡΟΜΗΘΕΥΤΗΣ

ΚΩΔΙΚΟΣ	ΠΟΛΗ_ ΠΟΥ_ ΕΔΡΕΥΕΙ	ΤΗΛΕ ΦΩΝΟ	ΤΑΧ ΚΩΔΙΚΑΣ	ΑΡΙΘ FAX
01	Αθήνα	3233677	11521	3233678
02	Αθήνα	2333766	11245	2366777
03	Λάρισα	888867	10453	887879
04	Πάτρα	556768	12345	557879
05	Πάτρα	557899	10934	557888
06	Αθήνα	6898777	11257	6899777

Εικόνα 40: Σύστημα παρακολούθησης παραγγελιών σε ένα κατάστημα λιανικής πώλησης σχολικών ειδών

Ένα κατάστημα πώλησης σχολικών ειδών διαθέτει μία βάση δεδομένων που αποτελείται από δύο πίνακες. Ο πρώτος πίνακας, που αποτελεί και τον αρχικό πίνακα, διαθέτει τα είδη που πουλάει το κατάστημα, δίνοντας τους ο καταστηματάρχης ένα πεδίο με μοναδικό κωδικό (το “πρωτεύον κλειδί” ή “primary key” είναι το πεδίο “ΚΩΔΙΚΟΣ”). Ο αρχικός πίνακας συνδέεται με έναν πίνακα προμηθευτών αυτών των ειδών (τον εξαρτώμενο πίνακα “ΠΡΟΜΗΘΕΥΤΗΣ”). Για να γίνει η σύνδεση, στον αρχικό πίνακα υπάρχει πεδίο “ΚΩΔΙΚΟΣ ΠΡΟΜΗΘΕΥΤΗ” το οποίο πεδίο δεν έχει μοναδικά στοιχεία, όμως στον εξαρτώμενο πίνακα “ΠΡΟΜΗΘΕΥΤΗΣ”,

χρησιμοποιείται κατά τρόπο μοναδικό και αποτελεί το “ξένο κλειδί” (“foreign key”) καθώς είναι το πεδίο που συσχετίζει τους δύο πίνακες.

Η σχέση μεταξύ των πινάκων (αρχικού και εξαρτημένου) χωρίζεται σε διάφορες κατηγορίες. Αναφορικά, η σχέση του προηγούμενου παραδείγματος μεταξύ των πινάκων “ΕΙΔΟΣ” και “ΠΡΟΜΗΘΕΥΤΗΣ” ονομάζεται σχέση “πολλά προς ένα” (many to one relationship). Σε διαφορετική περίπτωση θα ήταν δυνατόν κάθε τιμή του πρωτεύοντος κλειδιού του αρχικού πίνακα να αντιστοιχεί μόνο μία τιμή για το ξένο κλειδί ενός άλλου πίνακα (του εξαρτώμενου πίνακα). Η σχέση του αρχικού πίνακα με τον εξαρτώμενο, είναι σχέση “ένα προς ένα” (one to one relationship). Με βάσει αυτήν τη λογική οι σχέσεις μεταξύ των πινάκων δύναται να χωριστούν στις εξής κατηγορίες:

- Ένα προς ένα (one to one)
- Ένα προς N (one to many)
- N προς ένα (many to one)
- M προς N (many to many)

Όπου M, N, συμβολίζουν την ύπαρξη της έννοιας “πολλά” στο συσχετισμό των πινάκων.

6. Παρουσίαση της Εργασίας

Στο παρόν κεφάλαιο θα παρουσιαστεί η εργασία, οι τομείς που αποτελούν την εργασία παρουσιάζοντας το ρόλο του κάθε κομματιού στη σύνθεση της και η τεχνική υλοποίηση.

Η εργασία έχει σκοπό να παρουσιάσει μία πτυχή υλοποίησης του “έξυπνου σπιτιού” και ευρύτερα των έξυπνων αυτοματισμών που βρίσκουν εφαρμογή επίσης σε οποιοδήποτε χώρο πέραν της οικίας. Όπως έχει ήδη αναφερθεί, μία από τις πιο ενεργοβόρες διαδικασίες μίας οικίας και ενός χώρου, υπό την ευρύτερη έννοια, είναι η θέρμανση του νερού. Επί τούτου, πρέπει να γίνεται επιτήρηση του κόστους της θέρμανσης του νερού και να αυτοματοποιείται η εν λόγω διαδικασία ώστε να πραγματοποιείται κατά ώρες όπου υπάρχει ενεργό το οικονομικό τιμολόγιο του παρόχου ηλεκτρικής ενέργειας. Βέβαια, σαν πρώτη επιλογή υπάρχει η θέρμανση του νερού από τον ηλιακό θερμοσίφωνα, όπου έχει μηδενικό κόστος και ύστερα υπάρχει η επιλογή της θέρμανσης με ηλεκτρικό ρεύμα. Τελευταία επιλογή είναι η θέρμανση του νερού στο ημερήσιο τιμολόγιο του παρόχου, που είναι και το πιο ακριβό ως προς το κόστος.

Η καινοτομία βέβαια της εργασίας δεν έγκειται τόσο στον έλεγχο, όσο στη δυνατότητα που υπάρχει ως προς την αποθήκευση των δεδομένων που είναι σχετικά με τη θέρμανση, σε βάση δεδομένων και την κλήση τους υπό συνθήκες. Έτσι ο χρήστης συνδεόμενος στο πρόγραμμα που αλληλοεπιδρά με την εφαρμογή, παραμένει ενήμερος σχετικά με το κόστος που έχει ήδη επιβαρυνθεί, χωρίς να απαιτείται η μεσολάβηση του παρόχου ηλεκτρικής ενέργειας για την ενημέρωση του. Επιπλέον, η εργασία δύναται να επιτηρεί την κατάσταση θέρμανσης του νερού μέσω διαδικτύου ώστε ανά πάσα στιγμή να μπορεί να ενημερώνει για τη θερμοκρασία του νερού και για το αν είναι ενεργή η διαδικασία θέρμανσης.

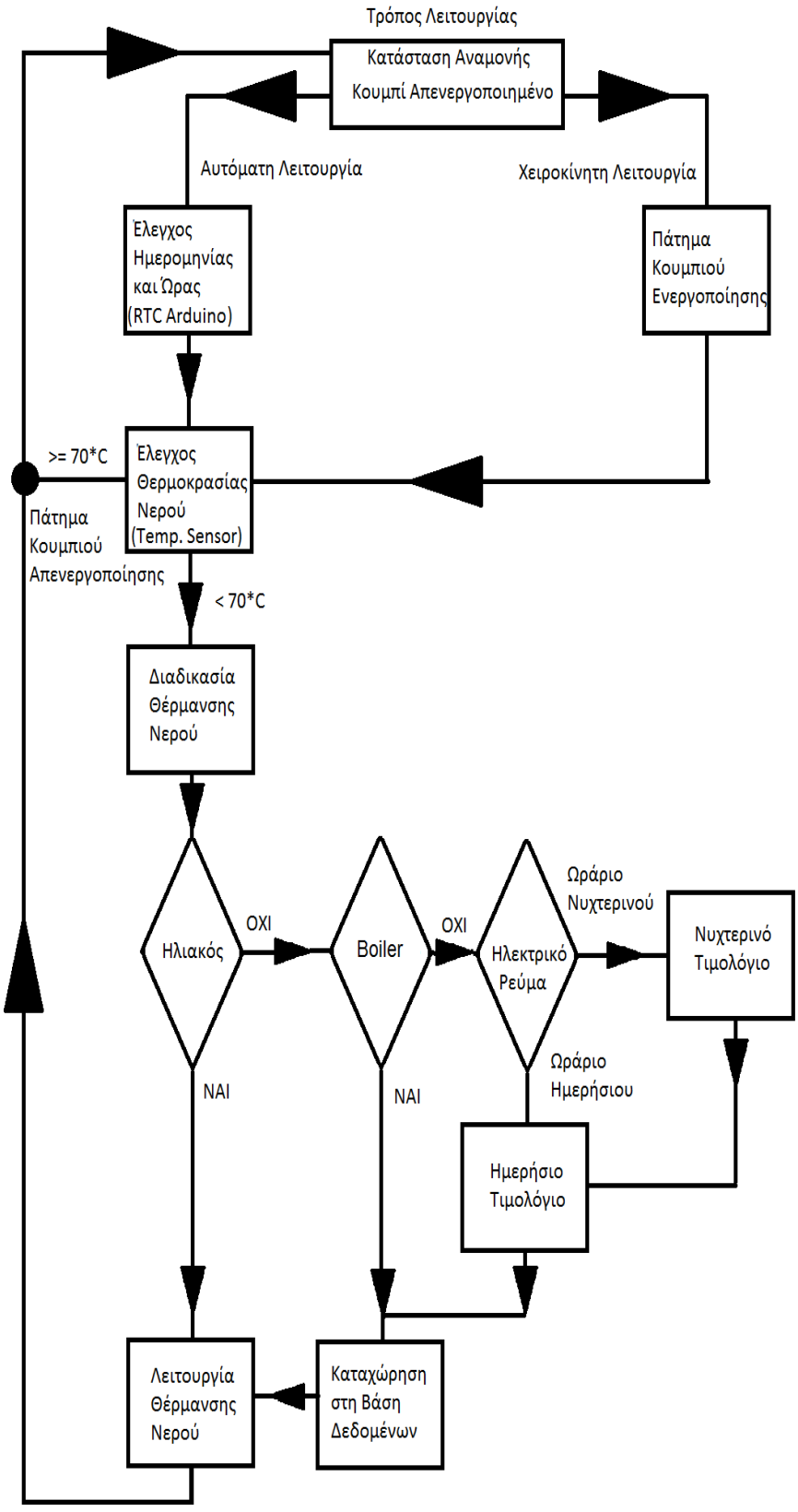
Η αυτοματοποίηση των εκκινήσεων της θέρμανσης του νερού γίνεται αφού προηγηθεί η εγγραφή ενός απλού κειμένου, με τις επιθυμητές μέρες και ώρες ενεργοποίησης της διαδικασίας και του χρόνου θέρμανσης. Το κείμενο γράφεται στον προσωπικό υπολογιστή του χρήστη και κατόπιν αποθηκεύεται σε μία απλή κάρτα μνήμης. Η κάρτα μνήμης με τη σειρά της τοποθετείται στο σύστημα της εργασίας. Αν ο χρήστης βέβαια επιθυμεί, ενώ έχει ήδη τοποθετήσει την κάρτα μνήμης, να εκκινήσει μία χειροκίνητη ενεργοποίηση θέρμανσης του νερού μία τυχαία στιγμή της ημέρας όπου έχει ανάγκη νερό και να την απενεργοποιήσει ο ίδιος, αυτό είναι εφικτό. Όλες οι ενεργοποιήσεις, είτε είναι αυτοματοποιημένες είτε χειροκίνητες έχουν σαν δικλείδα ασφαλείας

μία κρίσιμη θερμοκρασία, στην εργασία θεωρήθηκαν οι 70 βαθμοί κελσίου, όπου πέραν αυτής της θερμοκρασίας σταματάει η διαδικασία θέρμανσης.

Το κεντρικό σύστημα, η καρδιά του όλου εγχειρήματος, είναι η πλακέτα υλοποιήσεων Arduino με τα περιφερειακά του και την πλατφόρμα προγραμματισμού Arduino IDE. Η οθόνη αλληλεπίδρασης με το χρήστη έχει υλοποιηθεί σε Visual C# της Microsoft και η βάση δεδομένων σε SQL Server επίσης της Microsoft.

Αναφορικά, στην ενότητα 6.1. θα παρουσιαστεί η σύνδεση των επιμέρους πλακετών και περιφερειακών του Arduino και του προγραμματισμού στην πλατφόρμα Arduino IDE. Στις ενότητες 6.2. και 6.3. θα παρουσιαστούν τα μέρη σχετικά με τον προγραμματισμό σε Visual C# και στον SQL Server (ο κώδικας του SQL Server παρατίθεται στο παράρτημα) και ό,τι αφορά τη μεταξύ τους συνεργασία. Τέλος, στην ενότητα 6.4., θα γίνει αναφορά στον τρόπο λειτουργίας της όλης εργασίας με αναλυτικά σενάρια. Πρέπει να αναφερθεί ότι ο προγραμματισμός έχει γίνει κατά τρόπο πρότυπο, χωρίς βάση από άλλες εργασίες.

Αρχικά, πριν την είσοδο στις εν λόγω ενότητες, θα παρουσιαστεί στην επόμενη σελίδα ένα λογικό διάγραμμα (Εικόνα 41) που περιγράφει τη λειτουργία της εργασίας, για να γίνει κατανοητή η σύνδεση των επιμέρους κομματιών της εργασίας. Σημειώνεται πως όλη η διαδικασία που σημειώνεται στο λογικό διάγραμμα επιτηρείται διαδικτυακά.

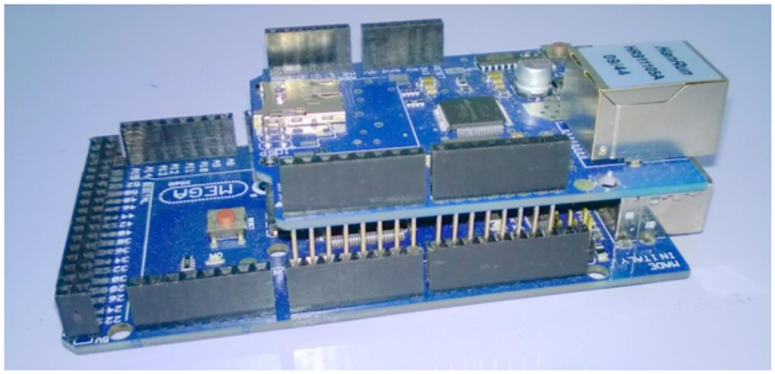


Εικόνα 41: Το λογικό διάγραμμα της εργασίας

6.1. Το Κομμάτι του Arduino

6.1.1. Το Υλικό

Το υλικό που αποτελεί το κομμάτι σύνθεσης του Arduino είναι το Ethernet Shield της εταιρείας και το Arduino Mega 2560 με τον μικροελεγκτή ATmega 2560 της εταιρείας ATMEL. Η τοποθέτηση τους γίνεται προσεκτικά, κουμπώνοντας τα δύο υλικά μεταξύ τους όπως φαίνεται στην Εικόνα 42-α.



Εικόνα 42-α: Τοποθέτηση των δύο υλικών μεταξύ τους

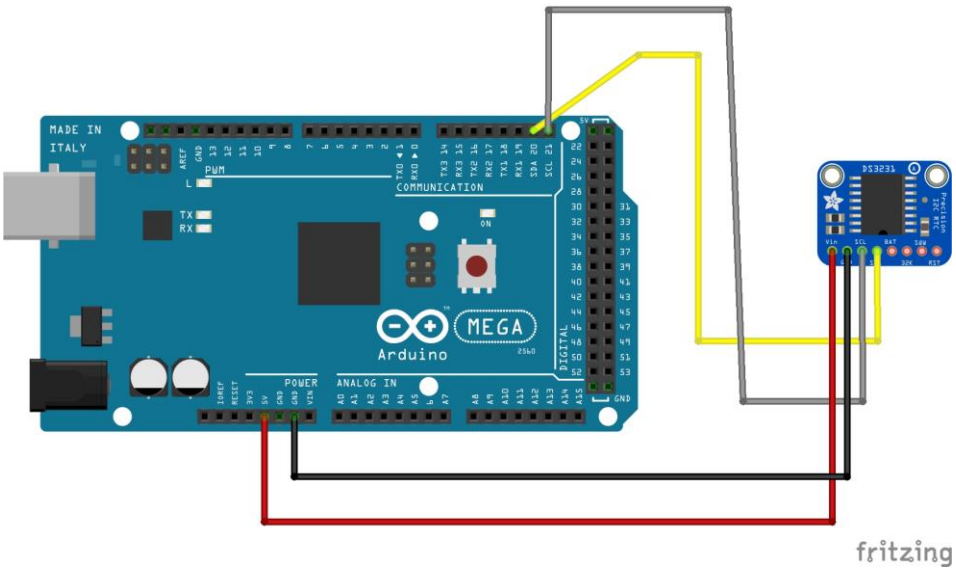
Το επόμενο υλικό που αποτελεί την εργασία είναι η πλακέτα με τα ρελέ (relay module). Η πλακέτα διατίθεται συναρμολογημένη στο εμπόριο και είναι κατάλληλη για εργαστηριακά πειράματα (Εικόνα 42-β). Διαθέτει 4 ρελέ και μπορεί να οδηγήσει μέχρι ρεύμα έντασης 10 Ampere. Η τροφοδοσία της είναι 5 Volts και γίνεται από την πλακέτα του Arduino.

Εικόνα 42-β: Η πλακέτα με τα ρελέ (relay module)

Το υλικό αποτελείται επίσης από το ρολόι του Arduino που ονομάζεται Real Time Clock (RTC) και είναι το μοντέλο DS3231 (Εικόνα 43) που διατίθεται στο εμπόριο και συνδέεται εξωτερικά μέσω καλωδίωσης με τον Arduino (Εικόνα 44).

Εικόνα 43: Το RTC DS3231

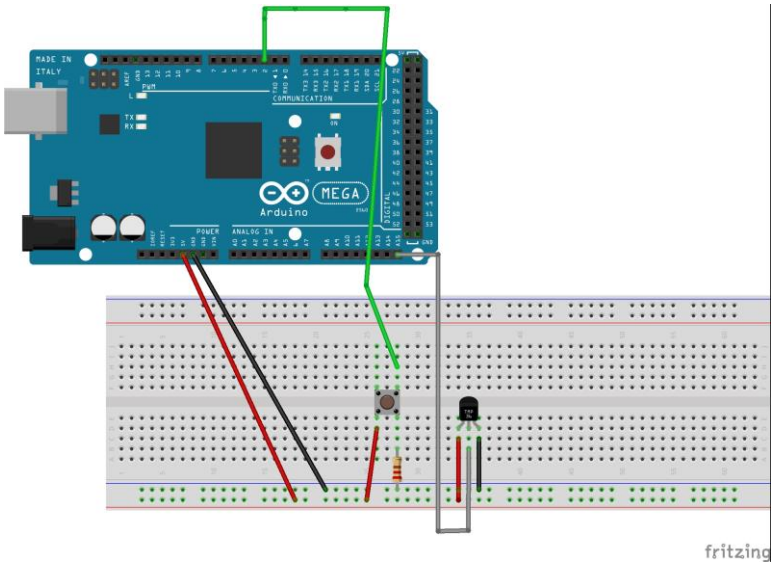
Η σύνδεση γίνεται συνδέοντας τη θύρα VCC του ρολογιού με την τροφοδοσία 5 Volts του Arduino, τη θύρα GND με τη γείωση του Arduino και τις θύρες SCL και SDA με τις αντίστοιχες θύρες που υπάρχουν επάνω στην πλακέτα του Arduino Mega.



Εικόνα 44: Η σύνδεση του RTC με τον Arduino Mega

Το τελικό κομμάτι σύνθεσης του υλικού είναι η συνδεσμολογία που πραγματοποιήθηκε στο breadboard για την πρόσθεση ενός κουμπιού (button)

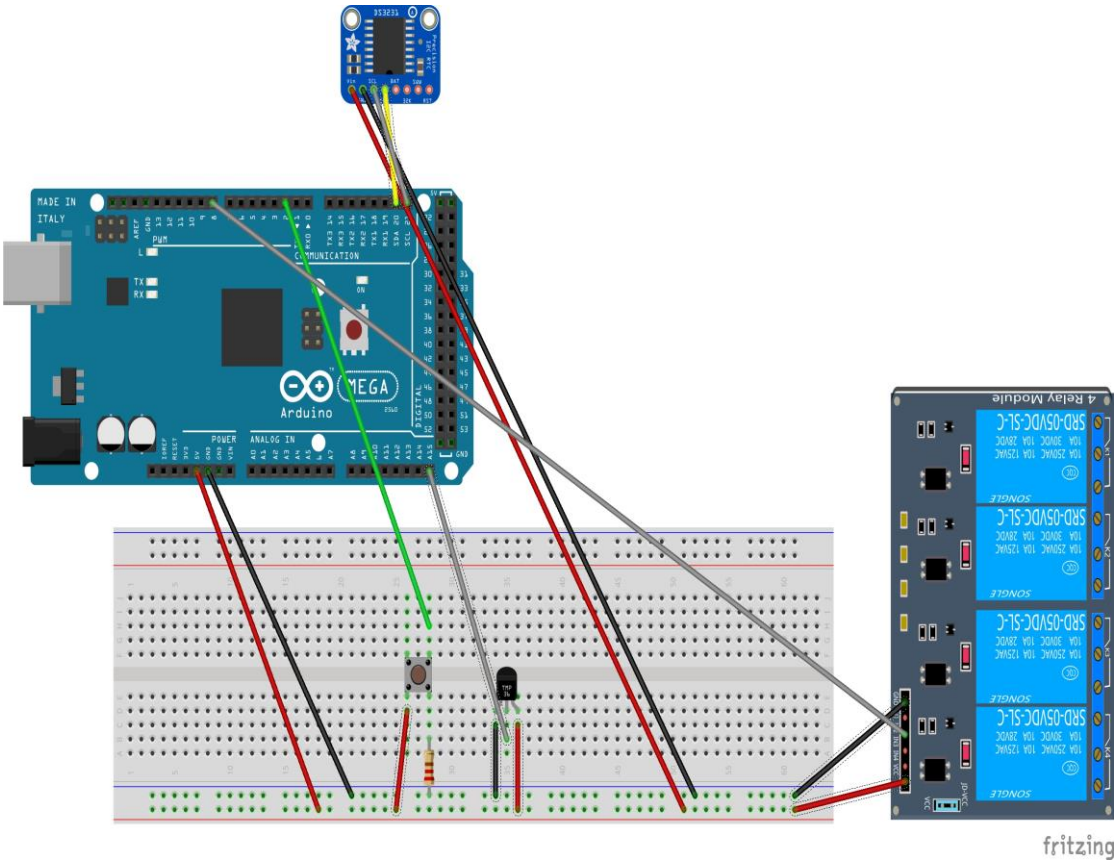
που ενεργοποιεί ή απενεργοποιεί την λειτουργία θέρμανσης του νερού (Εικόνα 45) και από το αδιάβροχο αναλογικό αισθητήριο θερμοκρασίας LM35 (Εικόνα 46). Το αδιάβροχο περίβλημα περιλαμβάνει μέσα του το αισθητήριο.



Εικόνα 45: Η σύνδεση του διακόπτη και του αισθητηρίου θερμοκρασίας με τον Arduino Mega

Εικόνα 46: Το αδιάβροχο αισθητήριο θερμοκρασίας

Στην Εικόνα 47 παρουσιάζεται μία συνολική σύνθεση όλου του υλικού.



Εικόνα 47: Η ολική σύνθεση της εργασίας

6.1.2. Το Λογισμικό του Arduino

Το λογισμικό του Arduino πραγματοποιήθηκε στην κονσόλα Arduino IDE που διατίθεται δωρεάν από τον φορέα του Arduino στο διαδικτυακό του τόπο: www.arduino.org. Αφού πραγματοποιηθεί η εγκατάσταση του, πρέπει να βρεθούν στο διαδίκτυο οι βιβλιοθήκες που απαιτούνται για την εύρυθμη λειτουργία του υλικού. Συνήθως διατίθενται στον διαδικτυακό τόπο www.arduino.cc. Στην περίπτωση της εργασίας ήταν οι εξής:

SD.h: βιβλιοθήκη για την περιφερειακή λειτουργία της κάρτας μνήμης SD

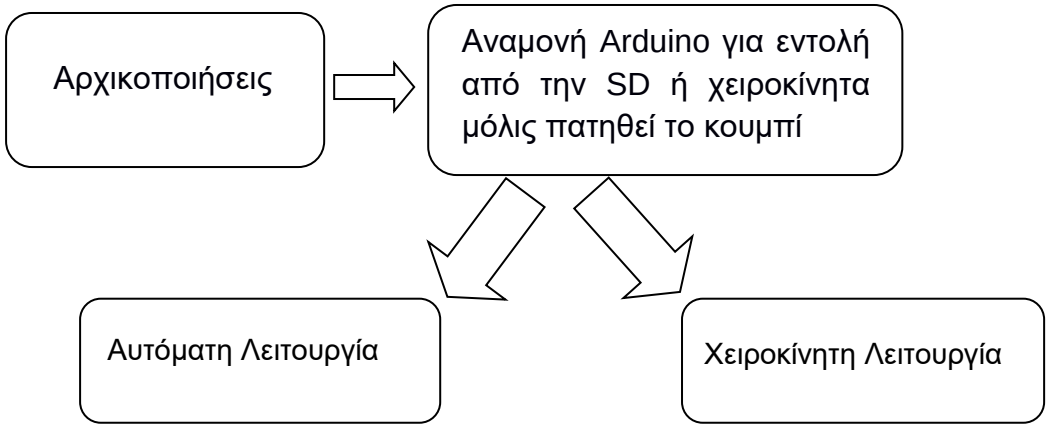
Ethernet2.h: βιβλιοθήκη απαραίτητη για τη περιφερειακή λειτουργία του Arduino ως web-server

DS3231.h: βιβλιοθήκη για τη λειτουργία του ρολογιού του Arduino που συνδέεται σαν πρόσθετο εξωτερικό περιφερειακό

SPI.h: Serial Peripheral Interfaces, βιβλιοθήκη για τη συνεργασία όλων των περιφερειακών του Arduino χωρίς εμπλοκή μεταξύ τους

Επιπροσθέτως, έχουν γραφτεί στα πλαίσια της εργασίας πρόσθετες βιβλιοθήκες για να εξασφαλίσουν την πραγματοποίηση κάποιας σύνθετης λειτουργίας. Αυτές οι βιβλιοθήκες θα παρουσιαστούν στη συνέχεια της εργασίας στα σημεία που γίνεται απαραίτητο.

Αρχικά, θα παρουσιαστεί ένα διάγραμμα της λογικής λειτουργίας του προγράμματος.

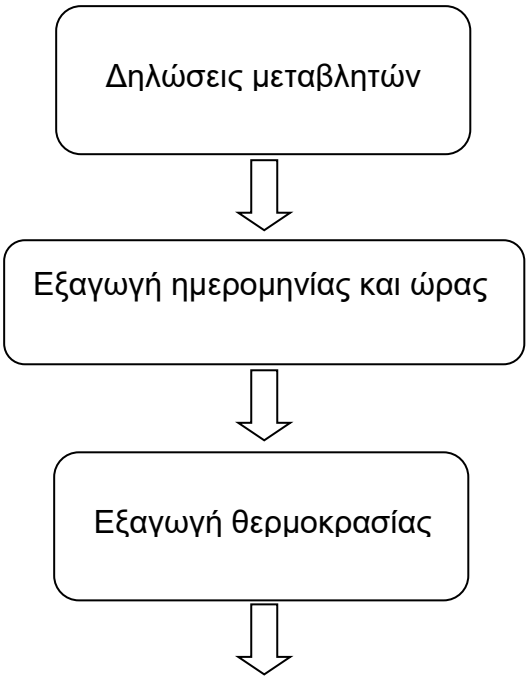


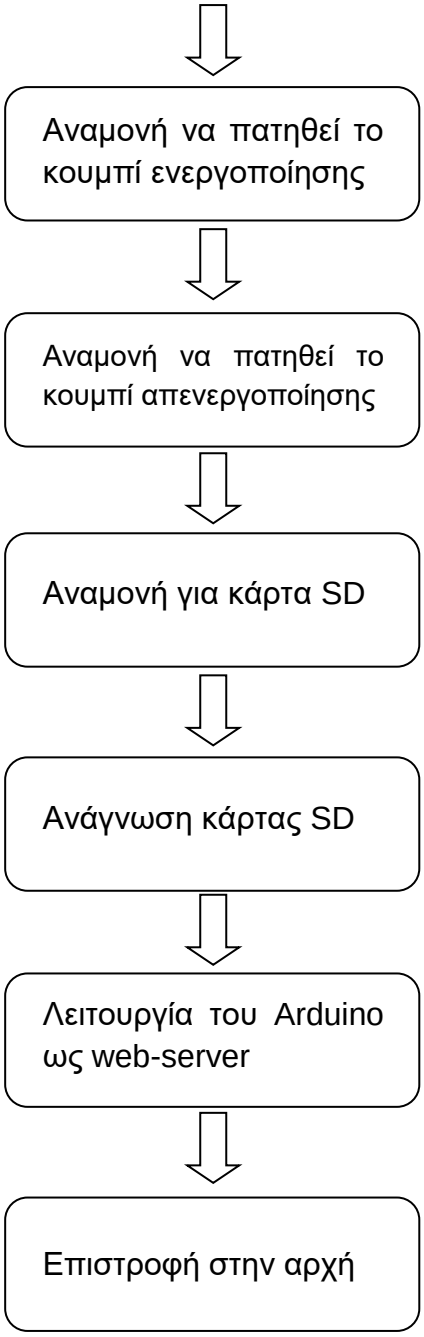
Με τον όρο “Αυτόματη Λειτουργία” εννοείται μία προγραμματισμένη προς εκτέλεση λειτουργία κατά συγκεκριμένη ημέρα, ώρα και διάρκεια μέσα στην εβδομάδα. Όλες οι προγραμματισμένες λειτουργίες αποθηκεύονται σε μία κάρτα μνήμης τεχνολογίας micro SD στο αρχείο hours.txt. Μία προγραμματισμένη λειτουργία απενεργοποιείται υπό κάποιες συνθήκες. Αυτές είναι:

- Να τελειώσει φυσιολογικά η θέρμανση του νερού βάσει χρόνου που έχει οριστεί από τον χρήστη.
- Να διακοπεί λόγω του ότι η θερμοκρασία του νερού έχει φτάσει την κρίσιμη τιμή της (στην εργασία έχει οριστεί ότι είναι 70 βαθμοί κελσίου).
- Να διακοπεί η θέρμανση του νερού χειροκίνητα πατώντας του κουμπί, πριν βεβαίως φτάσει την κρίσιμη θερμοκρασία.

Αναφερόμενοι στην “Χειροκίνητη Λειτουργία“, εννοείται μία λειτουργία που ενεργοποιείται και απενεργοποιείται σύμφωνα με τη βούληση του χρήστη, πατώντας το κουμπί. Βέβαια, υπάρχει η περίπτωση μία χειροκίνητη ενεργοποίηση να προηγηθεί χρονικά (με μικρή απόκλιση) μίας προγραμματισμένης ή αυτόματης εκκίνησης. Τότε η απενεργοποίηση πραγματοποιείται υπό τις παραπάνω συνθήκες.

Στην συνέχεια της παρουσίασης του λογισμικού του Arduino θα δοθεί ένα λογικό διάγραμμα αναφερόμενο στη ροή του προγράμματος.





Ο κώδικας που αναφέρεται, ανά κομμάτια, στις παραπάνω λειτουργίες είναι:

- Δηλώσεις μεταβλητών και ταυτόχρονη εξαγωγή ημερομηνίας ώρας και θερμοκρασίας

```
res->dtLogDate = rtc.getDateStr();          // String ημερομηνίας από τη
                                            //βιβλιοθήκη DS3231.h

res->dtLogTime = rtc.getTimeStr();          // String ώρας από τη βιβλιοθήκη
                                            //DS3231.h


time = rtc.getTime();          // μεταβλητή αποθήκευσης ώρας, από τη
                               //βιβλιοθήκη DS3231.h


int dow = time.dow;                         // μεταβλητή αποθήκευσης ημέρας της
                                            //εβδομάδος από τη βιβλιοθήκη DS3231.h


res->temperature = currentTemperature;      // θερμοκρασία


currentTemperature = WaterTemperature();    //μεταβλητή αποθήκευσης
                                            //θερμοκρασίας, καλείται από την
                                            //βιβλιοθήκη sensors.h


int relayStatusInput = digitalRead(Relay_Switch);   // ορισμός του κουμπιού
                     //ενεργοποίησης σαν είσοδος και αποθήκευση της κατάστασης του
                     //στην ακέραια μεταβλητή relayStatusInput. Η μεταβλητή
                     //Relay_Switch έχει οριστεί στη βιβλιοθήκη variables.h

int relayStatusOutput = digitalRead(Relay_Mode);    // ορισμός της θύρας
                     //εξόδου και αποθήκευση της κατάστασης στην ακέραια μεταβλητή
                     //relayStatusOutput. Η μεταβλητή Relay_Mode έχει οριστεί στη
                     //βιβλιοθήκη variables.h
```

- Αναμονή να πατηθεί το κουμπί ενεργοποίησης, να εισαχθεί η κάρτα μνήμης και η υλοποίηση των συνθηκών ανά περίπτωση

```
res->ButtonDown = (relayStatusInput == 1);

RelayIsOpen = (relayStatusOutput != 0);// το ρελέ είναι ενεργοποιημένο όταν η
                                       //έξοδος είναι 1 (μεταβλητή RelayIsOpen)

 if ( (RelayIsOpen) && (currentTemperature >= Critical_Temperature) )

 {

   RelayClose();

 }      // Αν το ρελέ είναι ενεργοποιημένο και η θερμοκρασία >= κρίσιμης
        //θερμοκρασίας, τότε το ρελέ κλείνει

 else   //      Αλλιώς όσο δεν έχει ξεπεραστεί η κρίσιμη θερμοκρασία, αναμονή
        //      μήπως εισέλθει κάρτα SD

{       // Ακολουθεί δήλωση μεταβλητών

params = new paramsInfo();

readSD(res, params);    // Ανάγνωση παραμέτρων SD από την βιβλιοθήκη card.h

Day curParams;

ResultOfCompare = false;

// Ακολουθεί σύγκριση των παραμέτρων της SD με τις τρέχουσες παραμέτρους
//ημερομηνίας και ώρας

for (int curDay=0; curDay < 7; curDay++)        // έχει οριστεί ότι Sunday=0 και
                                                //Saturday = 6

{

for (int i = 0; i < params->myDayInfo[curDay].counter; i++)

{

curParams = params->myDayInfo[curDay].myDay[i];

if ( (!ResultOfCompare) && (curDay == dow)) {

ResultOfCompare = compare2SD(time, curParams.hour, curParams.minute,
curParams.seconds, curParams.duration);

}

}

}
```

```
//"Check if Relay is open"
res->SwitchSrc = evntSrcNONE;
res->RelayIsOn = RelayIsOpen;
res->RelaySwitch = false;
if (!RelayIsOpen) // Αν δεν είναι ενεργοποιημένο
{
//"Relay is open"
bool toOpen = false;
int evntSrc = evntSrcNONE;
//"Is SD Present and active schedule?"
if (ResultOfCompare)     // Αποτέλεσμα σύγκρισης
{
//"YES SD IS  Present and active schedule"
toOpen = true;
evntSrc = evntSrcSD ;
}
else // "NO: SD IS NOT Present or not active schedule. CHECK Button"
{
if (relayStatusInput != 0)
{
//"Button Pressed"
toOpen = true;
evntSrc = evntSrcButton;
}
}
if (toOpen){
```

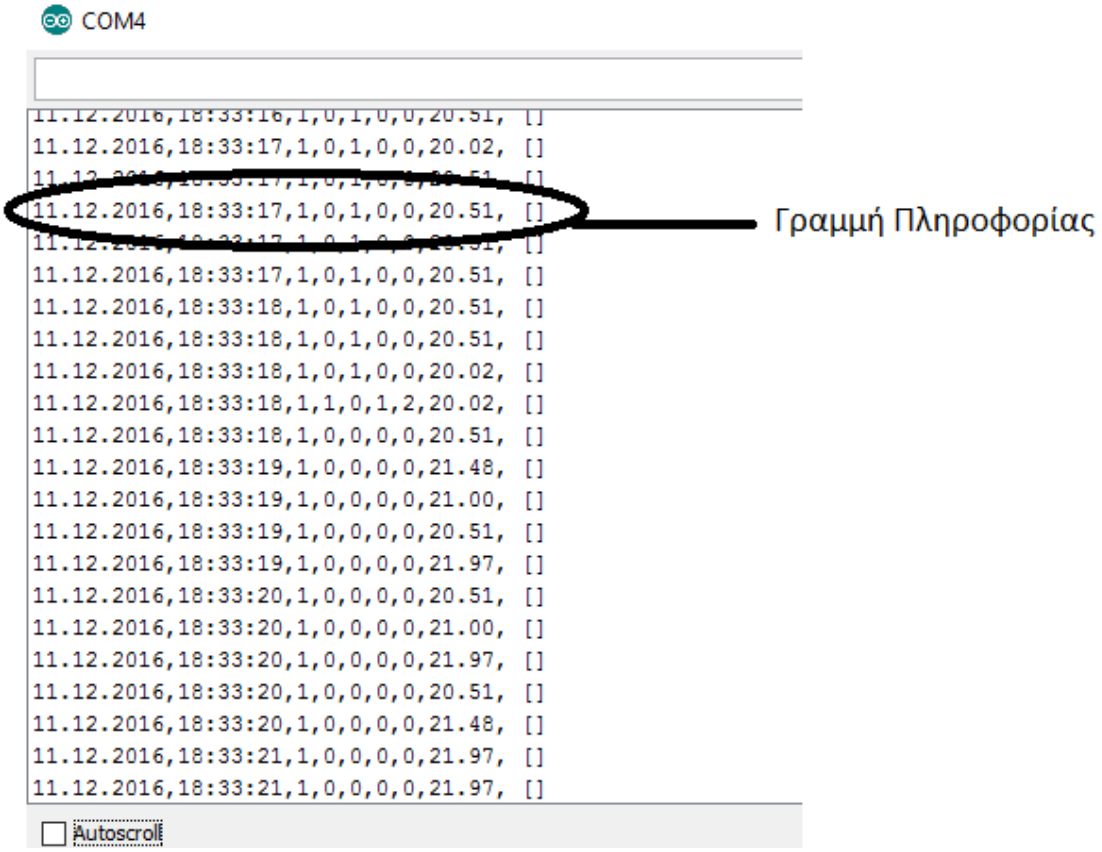
```
RelayOpen(evntSrc);
res->SwitchSrc = evntSrc;
res->RelayIsOn = true;
res->RelaySwitch = true;
}
}
else // RelayIsOpen
{
// "RELAY is ON. Should I turn it OFF?. First Check SD"
bool toClose = false;
if ( (LastEventSrc == evntSrcButton) && (ResultOfCompare) ){
LastEventSrc = evntSrcSD;
}
else
{
if ( (LastEventSrc == evntSrcSD) && (!ResultOfCompare) )
{
// "SD caused Relay to open and now OUT of active schedule. Turn it off"
toClose = true;
}
else
{          //"Button Pressed. Turn it OFF");
if ( (relayStatusInput != 0) && (LastEventSrc == evntSrcButton) ) toClose = true;
}
}
```

```
if (toClose) { // εντολή να κλείσει το ρελέ

RelayClose();

res->SwitchSrc = LastEventSrc;

res->RelayIsOn = false;

res->RelaySwitch = true;

}

}


free(params); // οι δυναμικές μεταβλητές απελευθερώνουν μνήμη του Arduino

}


// Τυπώνονται τα δεδομένα στη σειριακή

Serial.println(res->ToString(0));

// καλείται η συνάρτηση για να γίνει ο Arduino web server από τη βιβλιοθήκη //webserver.h

showClientStatusForBrowser(relayStatusInput, relayStatusOutput, rtc, server);

// τοποθετείται ακολούθως μία μικρή χρονική καθυστέρηση

delay(LOOP_DELAY);

}

// η ροή του κώδικα επιστρέφει στην αρχή
```

Στην ενότητα “Παράρτημα” δίνονται οι βιβλιοθήκες sensors.h, variables.h, webserver.h, card.h.

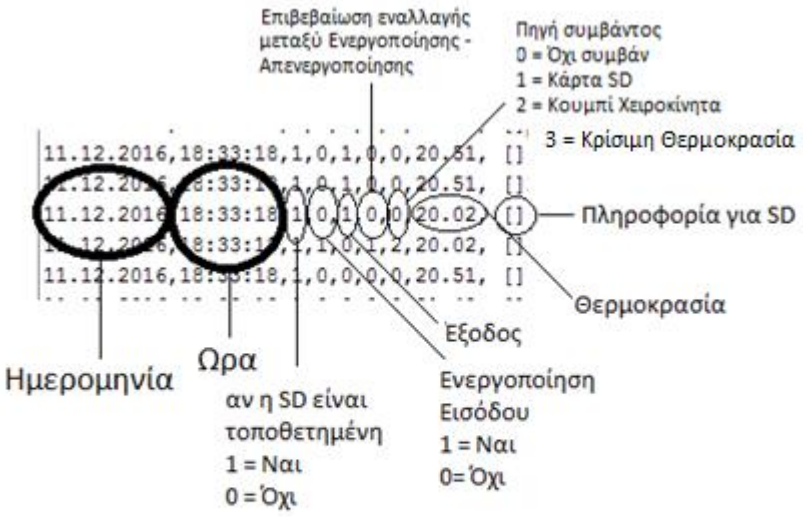
6.2. Το Λογισμικό Ανάγνωσης της Σειριακής

Το παρόν κεφάλαιο αναφέρεται στον τρόπο που επιτυγχάνεται η ανάγνωση της σειριακής εξόδου του Arduino από μία εφαρμογή που έχει γραφτεί και σχεδιαστεί με τη γλώσσα προγραμματισμού Visual C# της εταιρείας Microsoft. Στην Εικόνα 48 παρουσιάζεται η μορφή της σειριακής εξόδου από το Arduino.



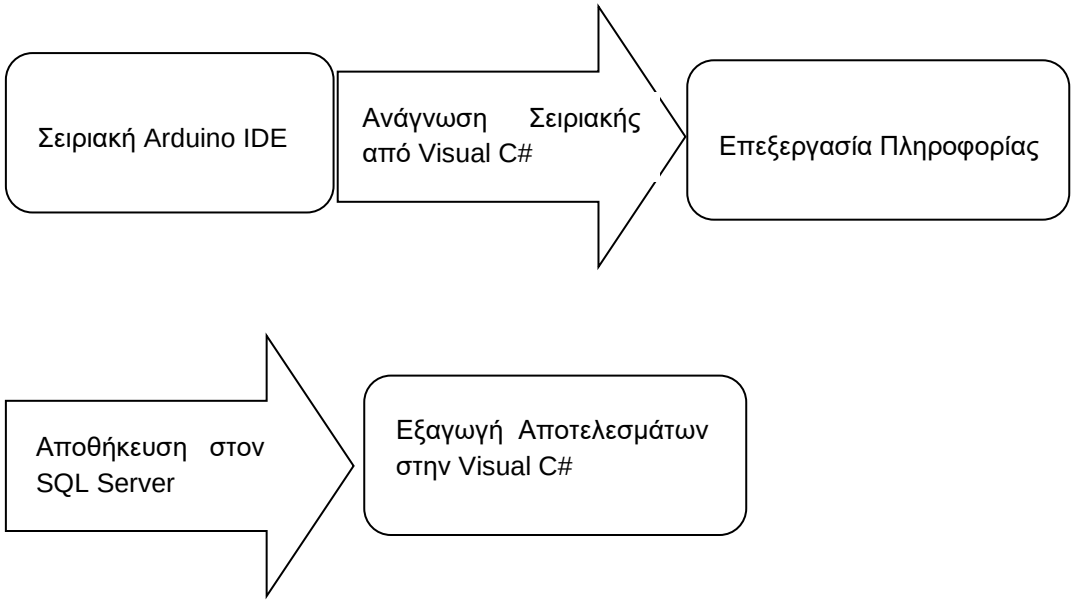
Εικόνα 48: Η σειριακή οθόνη από το Arduino IDE

Το πρόγραμμα αφού κάνει την ανάγνωση της σειριακής, ακολούθως την κάθε γραμμή πληροφορίας τη χωρίζει σε κομμάτια πληροφορίας (Εικόνα 49). Αυτά τα κομμάτια πληροφορίας χωρίζονται μεταξύ τους με κόμμα (,) και μεταβιβάζονται και αποθηκεύονται σε μία βάση δεδομένων, στον SQL Server.



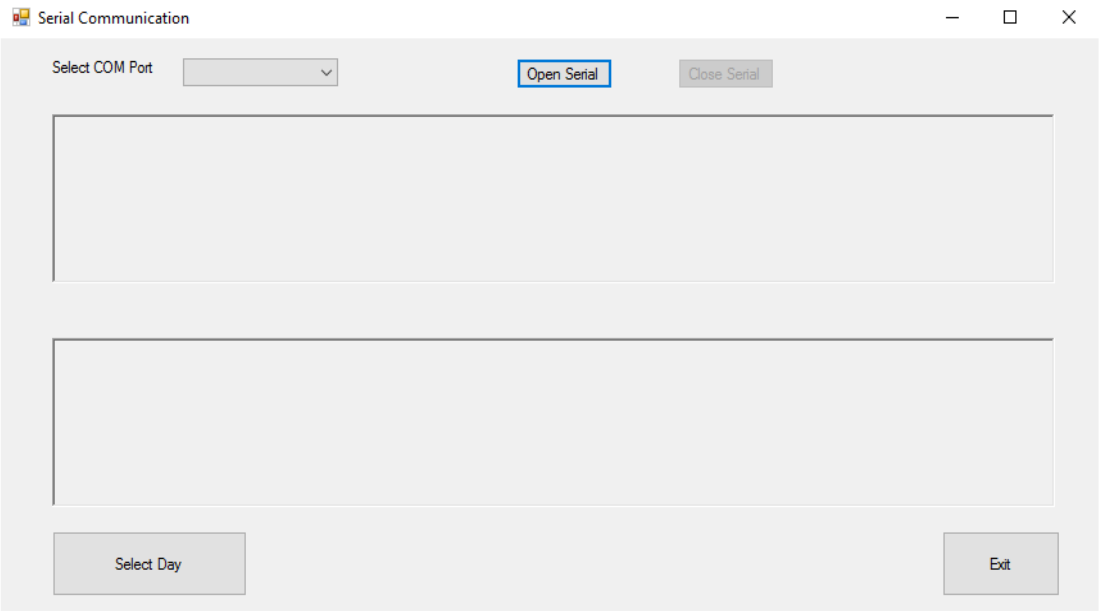
Εικόνα 49: Τα κομμάτια πληροφορίας

Από τον SQL Server, μέσω του προγράμματος της Visual C#, ανακτώνται κάθε φορά χρήσιμες πληροφορίες σχετικά με τις ώρες λειτουργίας της εργασίας και τα αντίστοιχα κόστη. Στην Εικόνα 50 φαίνεται ένα διάγραμμα που παρουσιάζει την εν λόγω λογική.



Εικόνα 50: Το διάγραμμα της πληροφορίας

Ακολουθεί η εφαρμογή και ο κώδικας για την ανάγνωση της σειριακής θύρας του Arduino. Η εφαρμογή ονομάζεται SerialTest2.exe. Στην Εικόνα 51 φαίνεται το περιβάλλον εργασίας.

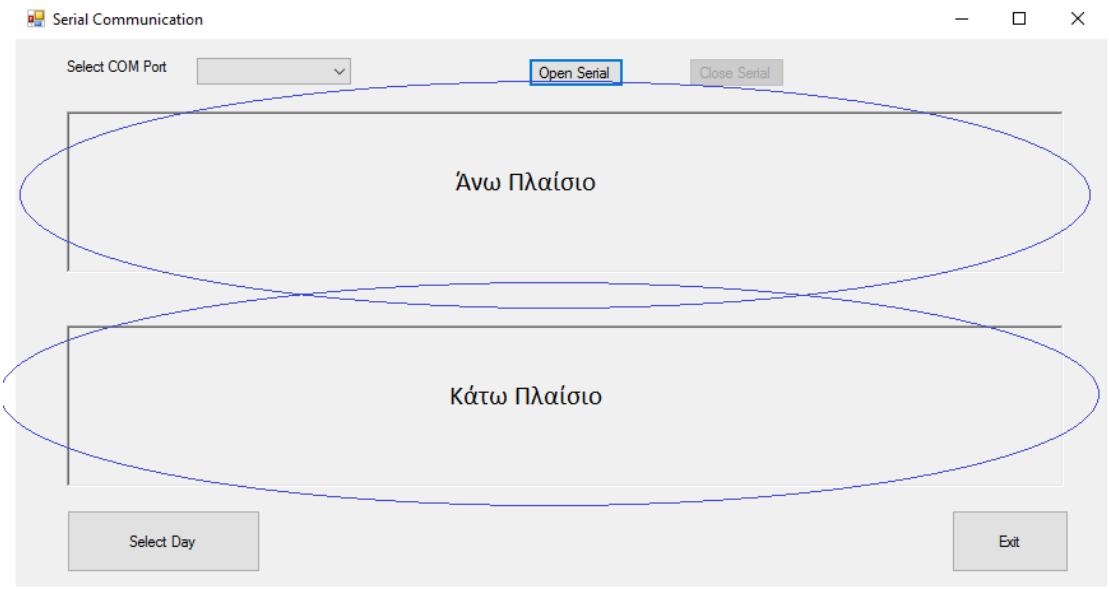


Εικόνα 51: Το περιβάλλον εργασίας του SerialTest2

Στο περιβάλλον εργασίας διακρίνονται κάποια πεδία. Αυτά είναι:

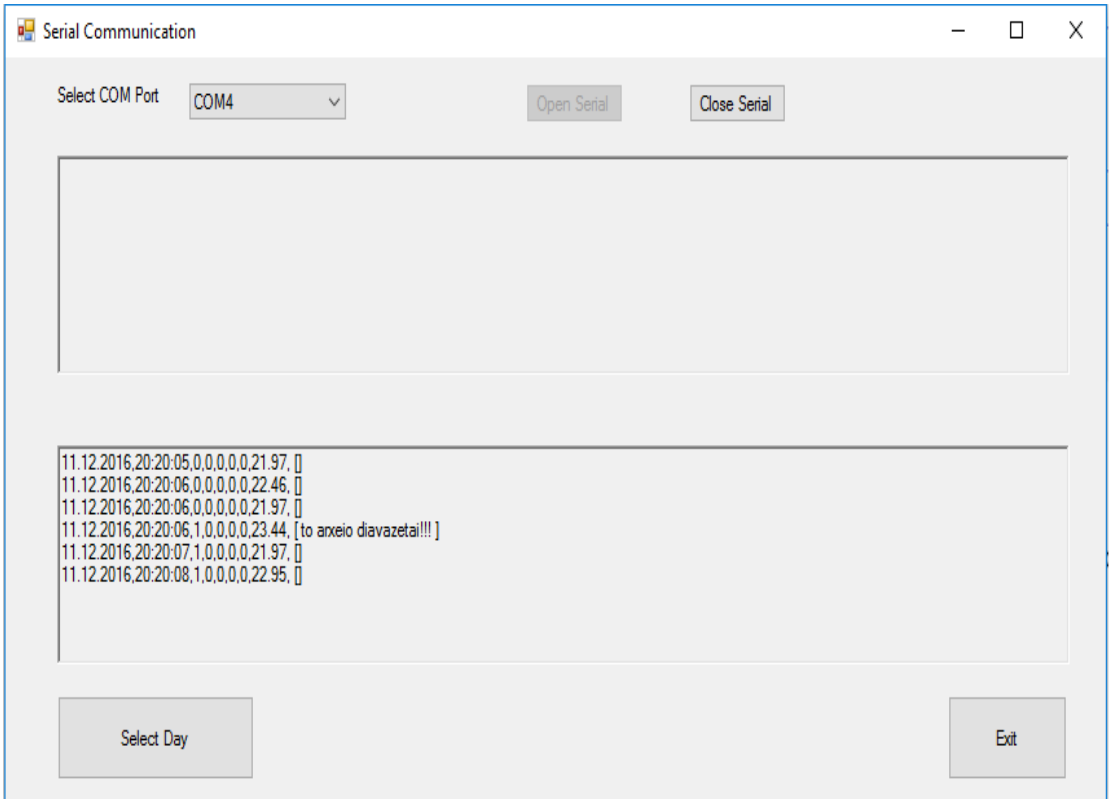
- Το πεδίο “Select COM Port” όπου γίνεται επιλογή της σειριακής θύρας.
- Τα κουμπιά “Open Serial” και το “Close Serial” που εκτελούν τις λειτουργίες ανοίγματος και κλεισίματος της σειριακής θύρας.
- Το κουμπί “Select Day” το οποίο πατώντας το μεταφέρει τον χρήστη στο νέο παράθυρο της επεξεργασίας δεδομένων (θα παρουσιαστεί αργότερα εκτεταμένα).
- Το κουμπί “Exit”
- Τα πλαίσια της σειριακής (Εικόνα 52)

Τα πλαίσια της σειριακής για λόγους κατανόησης χαρακτηρίζονται ως “Άνω Πλαίσιο” και “Κάτω Πλαίσιο”



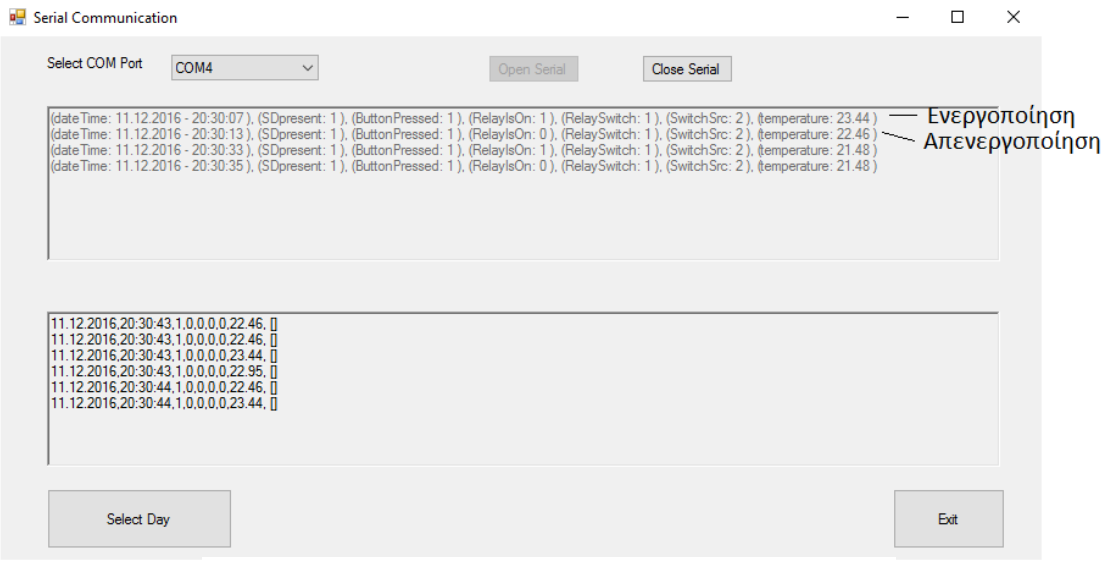
Εικόνα 52: Τα πλαίσια της σειριακής θύρας

Το Κάτω Πλαίσιο είναι το πλαίσιο εμφάνισης της γνωστής σειριακής θύρας, ακριβώς όπως από το Arduino IDE (Εικόνα 53).



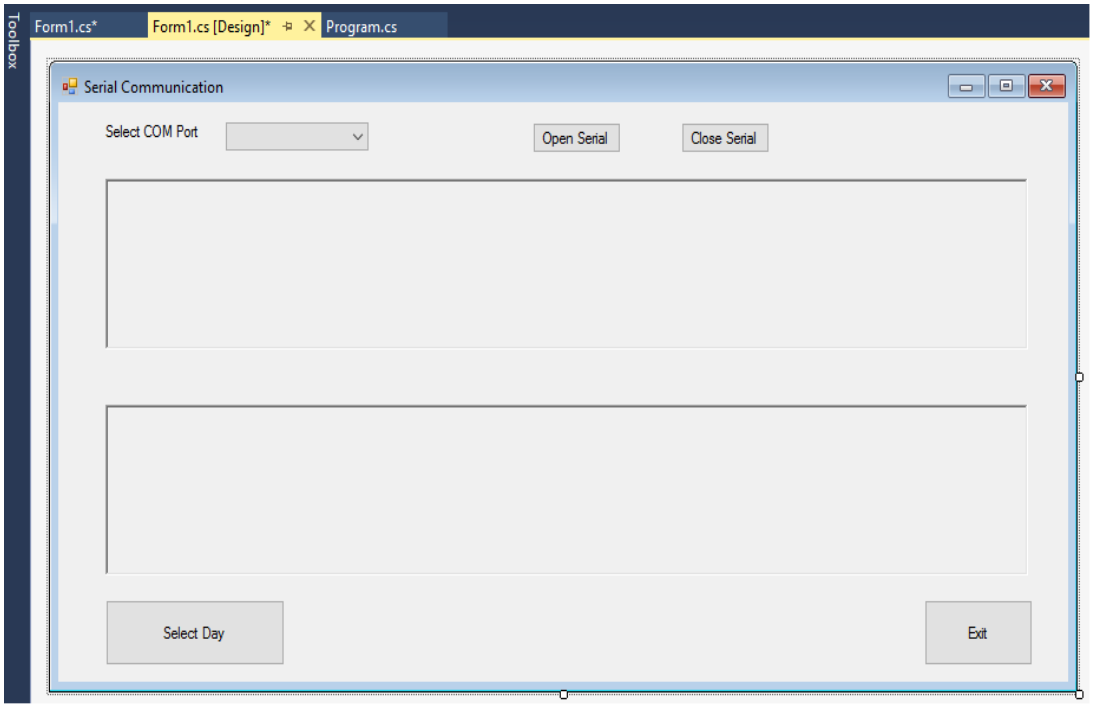
Εικόνα 53: Τα Κάτω Πλαίσιο της σειριακής θύρας

Το Άνω Πλαίσιο (Εικόνα 54) είναι το πλαίσιο εμφάνισης των Ενεργοποιήσεων – Απενεργοποιήσεων του ρελέ είτε από τα προγραμματισμένα γεγονότα που είναι καταγεγραμμένα στην κάρτα SD είτε χειροκίνητα από τον χρήστη πατώντας το κουμπί.



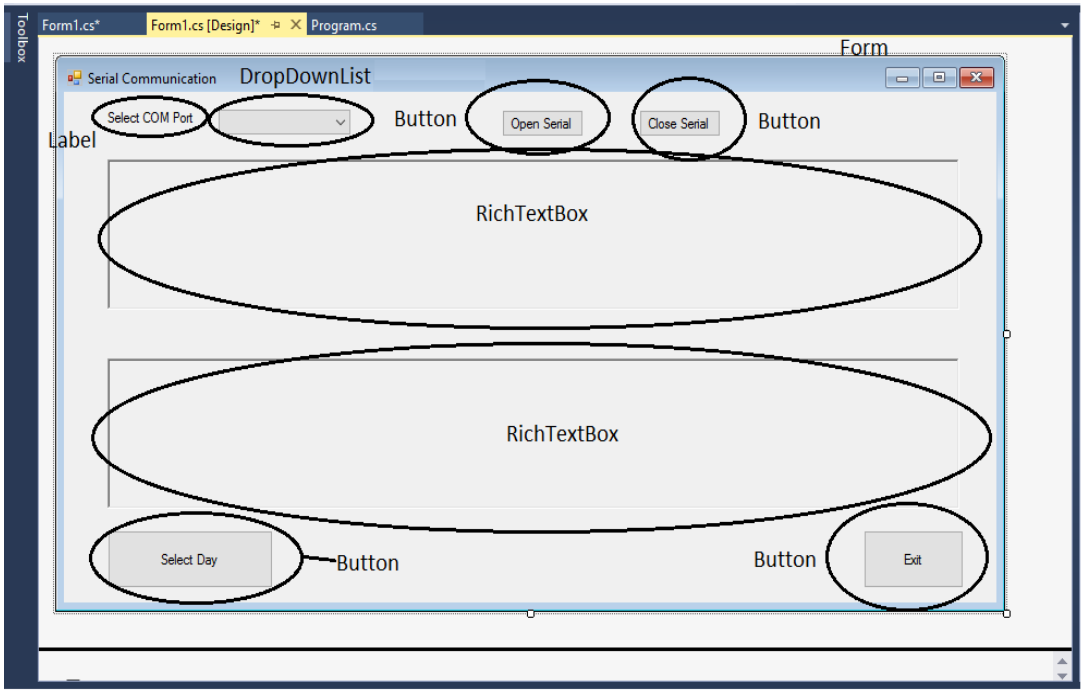
Εικόνα 54: Τα Άνω Πλαίσιο της σειριακής θύρας

Η πληροφορία από το άνω πλαίσιο είναι αυτή που στέλνεται προς αποθήκευση στον SQL Server. Ακολουθεί ο κώδικας και ο σχολιασμός του. Αρχικά, παρουσιάζεται η φόρμα σχεδιασμού του προγράμματος (Εικόνα 55). Υπενθυμίζεται ότι η γλώσσα γραφής και σχεδιασμού της φόρμας είναι η Visual C#.



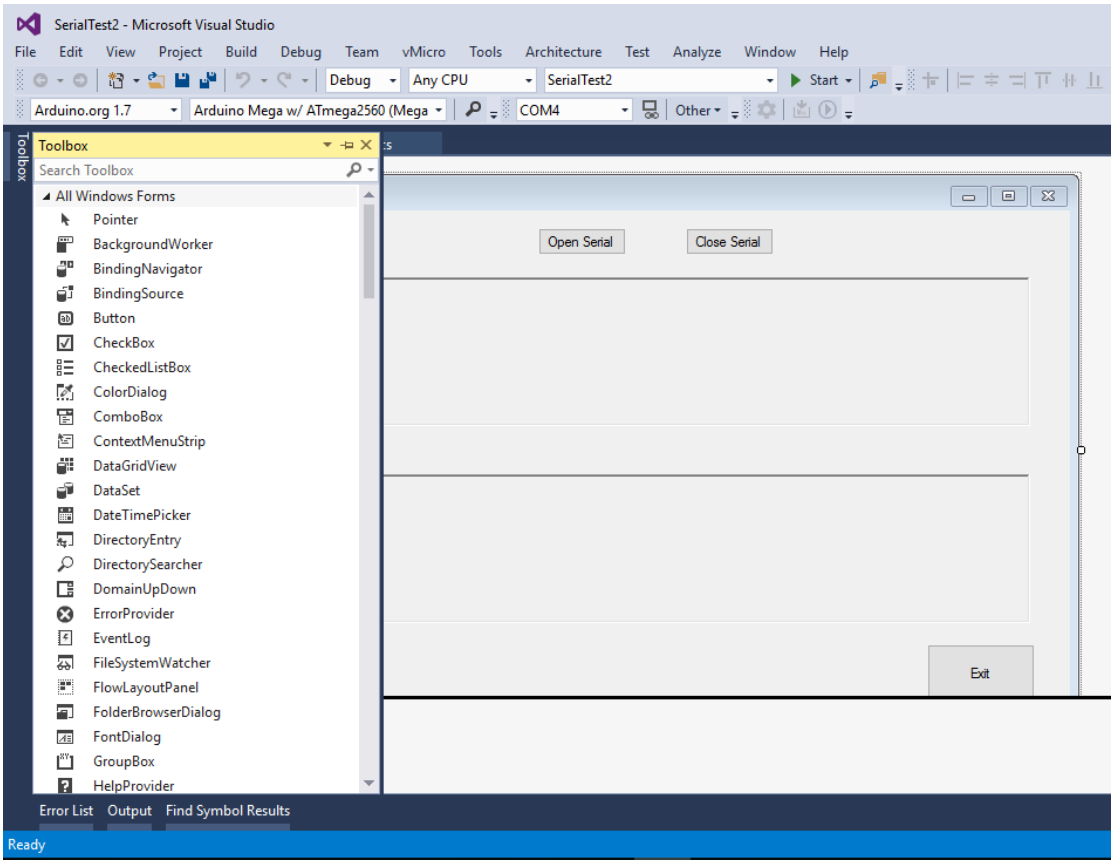
Εικόνα 55: Η φόρμα του προγράμματος

Στην Εικόνα 56 αναλύονται τα δομικά στοιχεία της Visual C# και που αποτελούν τη φόρμα.



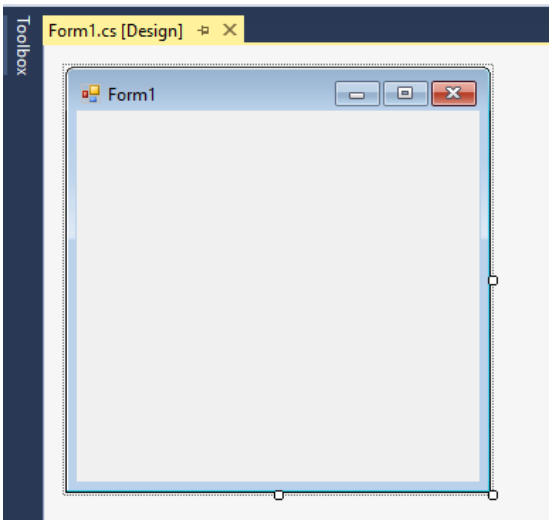
Εικόνα 56: Τα δομικά στοιχεία του προγράμματος

Όλα τα δομικά στοιχεία επιλέγονται από την “Εργαλειοθήκη” (Toolbox) της φόρμας.



Εικόνα 57: Η εργαλειοθήκη της Visual C#

Επιλέγονται ένα προς ένα και εισάγονται στην αρχικά κενή φόρμα σχεδίασης.



Εικόνα 58: Η αρχικά κενή φόρμα της Visual C#

Αφού ολοκληρωθεί η εισαγωγή των δομικών στοιχείων, έρχεται η σειρά της γραφής κώδικα και του σχολιασμού του.

```
// θεμελιώδεις και βασικές κλάσσεις δηλώνονται με το using Namespace

using System; // Για πρόσβαση σε τύπους στο System Namespace από τη C#

using System.Collections.Generic;  //The System.Collections.Generic
//namespace contains interfaces and classes that define generic
//collections, which allow users to create strongly typed collections
//that provide better type safety and performance than non-generic
//strongly typed collections.
using System.ComponentModel; // The System.ComponentModel namespace
//provides classes that are used to implement the run-time and design-
//time behavior of components and controls. This namespace includes the
//base classes and interfaces for implementing attributes and type
//converters, binding to data sources, and licensing components.
using System.Data; //The System.Data namespace provides access to
//classes that represent the ADO.NET architecture. ADO.NET lets you
//build components that efficiently manage data from multiple data
//sources.
using System.Drawing; //The System.Drawing namespace provides access to
//GDI+ basic graphics functionality.
using System.Linq; //The System.Linq namespace provides classes and
//interfaces that support queries that use Language-Integrated Query
//(LINQ).
using System.Text; //The System.Text namespace contains classes that
//represent ASCII and Unicode character encodings; abstract base classes
//for converting blocks of characters to and from blocks of bytes; and a
//helper class that manipulates and formats String objects without
//creating intermediate instances of String.
using System.Threading.Tasks; //The System.Threading.Tasks namespace
//provides types that simplify the work of writing concurrent and
//asynchronous code.
using System.Windows.Forms; //The System.Windows.Forms namespace
//contains classes for creating Windows-based applications that take
//full advantage of the rich user interface features available in the
//Microsoft Windows operating system.
using System.IO.Ports; //The System.IO.Ports namespace contains classes
//for controlling serial ports. The most important class, SerialPort,
//provides a framework for synchronous and event-driven I/O, access to
//pin and break states, and access to serial driver properties. It can
//be used to wrap Stream objects, allowing the serial port to be
//accessed by classes that use streams.
using System.Data.SqlClient; //The System.Data.SqlClient namespace is
//the.NET Framework Data Provider for SQL Server.
using System.Diagnostics; // The System.Diagnostics namespaces contain
//types that enable you to interact with system processes, event logs,
//and performance counters.
```

```
namespace SerialTest2 //Η δεσμευμένη λέξη namespace χρησιμοποιείται για
//να δηλώσει μία περιοχή η οποία περιλαμβάνει πλήθος σχετικών
//αντικειμένων. Αυτή η περιοχή ονομάζεται SerialTest2.
{
    public partial class Form1 : Form
    {
        StringBuilder sb;
        Queue<string> Info;
        SqlConnection cnn = new SqlConnection("Integrated
Security=SSPI;Persist Security Info=False;User ID=\"\";Initial
Catalog=Thesis;Data Source=DESKTOP-07RBA1T;Initial File Name=\"\";");
        SqlCommand cmd; //Αναπαριστά μία ανοιχτή σύνδεση σε μία βάση
//δεδομένων στον SQL

        int counter = 0;
        public Form1()
        {
            InitializeComponent(); //Αρχικοποίηση συστατικών μερών χώρου
            getAvailablePorts(); //Διαθέσιμες θύρες
            Info = new Queue<string>();

            cnn.Open(); //Ανοίγει η σύνδεση με τη βάση δεδομένων
            cmd = cnn.CreateCommand(); //Δημιουργεί και επιστρέφει το
//αντικείμενο (object) που συνεργάζεται με την SQL σύνδεση

//Παραμετροποίηση και αποθήκευση στη βάση δεδομένων
            cmd.CommandType = CommandType.StoredProcedure;
            cmd.CommandText = "AddLogInfo";

//Ακολουθεί η δήλωση των παραμέτρων που υπάρχουν σε μορφή πίνακα στην
//βάση δεδομένων
            cmd.Parameters.Add("@LogInfo", SqlDbType.VarChar, 2048);
            cmd.Parameters.Add("@dtLog", SqlDbType.DateTime);
            cmd.Parameters.Add("@RelayIsOn", SqlDbType.Bit);
            cmd.Parameters.Add("@RelaySwitch", SqlDbType.Bit);

            cmd.Parameters.Add("@ButtonPressed", SqlDbType.Bit);
            cmd.Parameters.Add("@EventSrc", SqlDbType.Int);
            cmd.Parameters.Add("@SDRead", SqlDbType.Bit);

            cmd.Parameters.Add("@Temperature", SqlDbType.Float);
        }
```

```
//Διαθέσιμες θύρες
void getAvailablePorts()
        {
            String[] ports = SerialPort.GetPortNames();
            comboBox1.Items.AddRange(ports);
        }
//Επιλογή θύρας από το αναδυόμενο μενού comboBox1
        private void button1_Click(object sender, EventArgs e)
        {
            try{    // Αν μείνει κενή η επιλογή
                if (comboBox1.Text == ""){
                    MessageBox.Show("Please Select COM Port");
                }
                else      //Όταν επιλέγεται κάτι
                {
                    serialPort1.PortName = comboBox1.Text;
                    serialPort1.Open();
                    richTextBox1.Enabled = true;
                    buttonOpen.Enabled = false;
                    buttonClose.Enabled = true;
                }
            }
            catch (UnauthorizedAccessException) //Σε περίπτωση σφάλματος
            {
                MessageBox.Show("Unauthorized Access!!");
            }
        }



//Κουμπί για εκκίνηση σειριακής θύρας

        private void button2_Click(object sender, EventArgs e)
        {
            serialPort1.Close();
            buttonOpen.Enabled = true;
            buttonClose.Enabled = false;
            richTextBox1.Enabled = false;
        }



//Κουμπί για κλείσιμο σειριακής επικοινωνίας
        private void button1_Click_1(object sender, EventArgs e)
        {
            this.Close();
        }
```

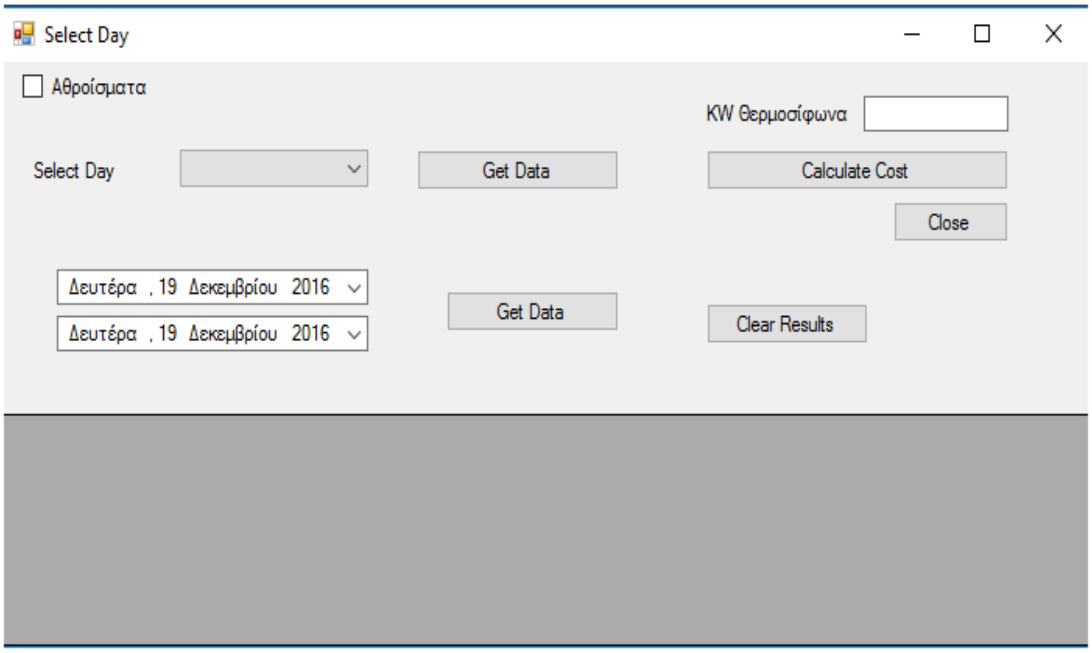
```
//Έλεγχος της σειριακής θύρας
        private     void     serialPort1_DataReceived(object     sender,
SerialDataReceivedEventArgs e)
        {
            this.Invoke(new EventHandler(textshow));
        }
//Μορφή εμφάνισης της σειριακής θύρας
        private void textshow(object sender, EventArgs e)
        {
//Ανάγνωση της σειριακής θύρας
            string str = serialPort1.ReadLine();
            string formattedStr = "";
            if (str.Trim().Length > 0 )
            { // & counter > 20
                writeLog(str, out formattedStr);
            } else
            {
                counter += 1;
            }

           if (Info.Count > 5)
            {
                Info.Dequeue();
            }
            Info.Enqueue(formattedStr);
            richTextBox1.Text  =  QueueToStr();  //εμφάνιση  της  σειριακής
//θύρας
        }

//Προσάρτηση κειμένου από τη σειριακή θύρα στην ουρά
        private string QueueToStr()
        {
            StringBuilder sb = new StringBuilder();
            foreach (var str in Info)
            {
                sb.Append(str);
            }
            return sb.ToString();
        }
```

```
//”Σπλιτάρισμα” του κειμένου της σειριακής θύρας ανά γραμμή
        private void writeLog(String log, out String formatted)
        {
            //Αναζητείται το σύμβολο (‘,’) για να γίνει το κόψιμο
            string[] parts = log.Split(',');
            if ((parts.Length >= 8) && (parts[5].Trim() == "1"))
            {
//Ενδεικτικά έχω 7 “σπλιταρίσματα”
//       0       1         2               3           4              5
6                7 αντιστοιχούν σε:
//    27.11.2016,14:15:23,(SDpresent    :1),(ButtonDown    :0),(RelayIsOn
:0),(RelaySwitch :0),(SwitchSrc :0),(temperature :28.81), []

//Αποθηκεύω σε κάθε στήλη παραμέτρων της SQL τα αντίστοιχα δεδομένα
                cmd.Parameters[0].Value = log;  //0: @LogInfo
                cmd.Parameters[1].Value = DateTime.Now; //1: @dtLog
                cmd.Parameters[2].Value   =   (parts[4]   ==   "1");   //2:
//@RelayIsOn
                cmd.Parameters[3].Value   =   (parts[5]   ==   "1");   //3:
//@RelaySwitch
                cmd.Parameters[4].Value   =   (parts[3]   ==   "1");   //4:
//@ButtonPressed
                cmd.Parameters[5].Value  =  parts[6];                 //5:
//@EventSrc
                cmd.Parameters[6].Value   =   (parts[2]   ==   "1");   //6:
@SDRead
//Δήλωση θερμοκρασίας
                float temperature;
                float.TryParse(parts[7],out temperature);

                cmd.Parameters[7].Value = temperature;

                cmd.ExecuteNonQuery();  //Εκτελεί  και  επιστρέφει  τις
στήλες από τη βάση δεδομένων
```

```
// Η ενημέρωση των στηλών της βάσης δεδομένων
                sb.AppendFormat("(dateTime: {0} - {1} ), ", parts[0],
parts[1]);
                sb.AppendFormat("(SDpresent: {0} ), ", parts[2]);
                sb.AppendFormat("(ButtonPressed: {0} ), ", parts[3]);
                sb.AppendFormat("(RelayIsOn: {0} ), ", parts[4]);
                sb.AppendFormat("(RelaySwitch: {0} ), ", parts[5]);
                sb.AppendFormat("(SwitchSrc: {0} ), ", parts[6]);
                sb.AppendFormat("(temperature: {0} )", parts[7]);
                sb.AppendLine(); //τερματίζεται η γραμμή

                formatted = sb.ToString();
                DBLogInfo.AppendText(formatted);
                sb.Clear();
            }
            else
            {
                formatted = log;
            }
//Στο παρακάτω path αποθηκεύεται εφεδρικό αρχείο .txt με όλα τα δεδομένα
//της σειριακής θύρας

System.IO.File.AppendAllText(@"C:\Users\werty\Desktop\projectDB\WriteLin
es.txt", log + Environment.NewLine);
        }

// Όταν πατηθεί το κουμπί Select Day εκτελείται το πρόγραμμα επιλογής
//ημέρας που είναι αποθηκευμένο στο παρακάτω μονοπάτι

        private void button1_Click_2(object sender, EventArgs e)
        {
            Process.Start(@"C:\Users\werty\Documents\Visual       Studio
2015\Projects\giannis\giveDay\selectDay\selectDay\bin\Debug\selectDay.ex
e");
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            sb = new StringBuilder();
        }
    }
}
```

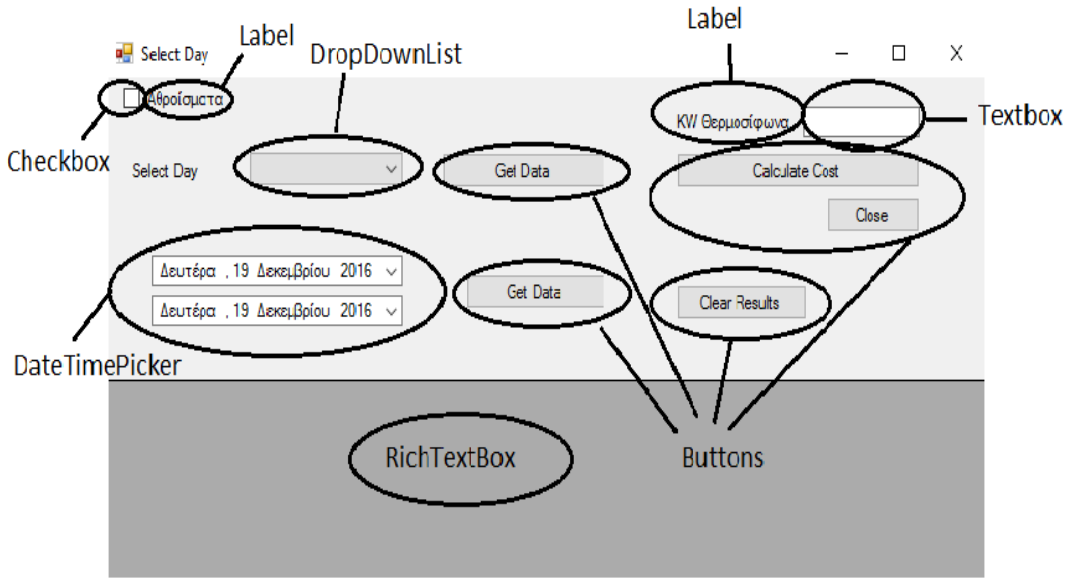
6.3. Η Συνεργασία των Λογισμικών της Εργασίας

Η συνεργασία των λογισμικών, δηλαδή μεταξύ της Visual C# και του SQL Server επιτυγχάνεται με τη δημιουργία του προγράμματος SelectDay σε γλώσσα Visual C#. Στις Εικόνες 55 και 56 στο κάτω αριστερά σημείο της φόρμας διακρίνεται το κουμπί Select Day, όπου ο χρήστης πατώντας το μεταφέρεται σε ένα δεύτερο πρόγραμμα που εκτελείται ταυτόχρονα με την ανάγνωση της σειριακής θύρας. Η φόρμα του προγράμματος έχει την ακόλουθη μορφή (Εικόνα 59).



Εικόνα 59: Η φόρμα του προγράμματος Select Day

Παρατηρούνται πολλά πεδία, τα οποία όμως έχει το καθένα το δικό του ρόλο. Ακολουθεί η Εικόνα 60 με μία περιγραφή των δομικών στοιχείων που αποτελούν την φόρμα του προγράμματος.



Εικόνα 60: Τα δομικά στοιχεία του προγράμματος Select Day

Στη συνέχεια παρατίθεται ο κώδικας και ο σχολιασμός του:

```
//Η δεσμευμένη λέξη namespace χρησιμοποιείται για να δηλώσει μία
//περιοχή η οποία περιλαμβάνει πλήθος σχετικών αντικειμένων. Αυτή η
//περιοχή ονομάζεται selectDay.

namespace selectDay
{
    public partial class Form1 : Form
    {
//Αναπαριστά μία ανοιχτή σύνδεση σε μία βάση δεδομένων στον SQL
        SqlConnection cnn = new SqlConnection("Integrated
Security=SSPI;Persist Security Info=False;User ID=\"\";Initial
Catalog=Thesis;Data Source=DESKTOP-07RBA1T;Initial File Name=\"\";");
        SqlCommand cmd;
        public Form1()
        {
            InitializeComponent(); //Αρχικοποίηση συστατικών μερών χώρου
        }

// Κουμπί εξαγωγής δεδομένων Get Data
        private void btnGetData_Click(object sender, EventArgs e)
        {
            GetData(!cbGrouped.Checked);
        }
```

```
//     Εξαγωγή δεδομένων από την βάση δεδομένων
private void GetData(bool detail = false)
        {
            try
            {
                cnn.Open();
                cmd = cnn.CreateCommand();
                cmd.CommandType = CommandType.Text;
                cmd.CommandText = "select " +
                                        (detail ? " * " : " Count(*) as
Counter, Convert(varchar, dateAdd(second, Sum(TimeDay), 0), 114) as TimeDay,
Convert(varchar, dateAdd(second, Sum(TimeNight), 0), 114) as TimeNight,
Sum(CostNight) as CostNight, Sum(CostDay) as CostDay ")
                                        + " from [dbo].OpenClose where
datePart(weekday, opened) = " + (cbDay.SelectedIndex + 1).ToString();
                SqlDataAdapter da = new SqlDataAdapter(cmd);
                DataSet ds = new DataSet("myData");
                da.Fill(ds);
                gvData.DataSource = ds.Tables[0];

            }

// Σε περίπτωση σφάλματος, εμφάνιση αντίστοιχου μηνύματος

            catch (Exception ex)
            {
                MessageBox.Show("Error: " + ex.Message, "Error retreiving
data", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
            finally
            {
                if (cnn != null)
                {
                    if (cnn.State == ConnectionState.Open)
                    {
                        cnn.Close();
                    }
                }
            }
        }
```

```
        //     Κουμπί για κλείσιμο προγράμματος
         private void button1_Click(object sender, EventArgs e)
         {
             this.Close();
         }

        // Προγραμματισμός για ζήτηση εύρους ημερών

         private void btnGiveData_Click(object sender, EventArgs e)
         {
             getData4Range(!cbGrouped.Checked);
         }

         private void getData4Range(bool detail = false)
         {
             try
             {
                 cnn.Open();

// Εξαγωγή αποτελεσμάτων από τη βάση δεδομένων

                 cmd = cnn.CreateCommand();
                 cmd.CommandType = CommandType.Text;
                 cmd.CommandText = "select " +
                                   (detail ? " * " : " Count(*) as Counter,
Convert(varchar,  dateAdd(second,  Sum(TimeDay),  0),  114)  as  TimeDay,
Convert(varchar,  dateAdd(second,  Sum(TimeNight),  0),  114)  as  TimeNight,
Sum(CostNight)  as  CostNight,  Sum(CostDay)  as  CostDay  ")  +  "  from
[dbo].OpenClose where opened >= @dateFrom and opened < @dateTo";

//Ημέρα ΑΠΟ
                 cmd.Parameters.Add("@dateFrom",    SqlDbType.Date).Value    =
dtpFrom.Value;

//Ημέρα Έως
                 cmd.Parameters.Add("@dateTo",     SqlDbType.Date).Value     =
dtpTo.Value;

                 SqlDataAdapter da = new SqlDataAdapter(cmd);
                 DataSet ds = new DataSet("myData");
                 da.Fill(ds);
                 gvData.DataSource = ds.Tables[0];

             }
             catch
             {
             }
             finally
             {
                 if (cnn != null)
                 {
                     if (cnn.State == ConnectionState.Open)
                     {
                         cnn.Close();
                     }
                 }
             }
         }
}
```

```
//Υπολογισμός κόστους

private void Cost()
        {
            try
            {
//Εξαγωγή δεδομένων από τη βάση δεδομένων
                cnn.Open();
                cmd = cnn.CreateCommand();
                cmd.CommandType = CommandType.StoredProcedure;
                cmd.CommandText = "Process";
                cmd.Parameters.Add("@Power",    SqlDbType.Decimal).Value    =
Convert.ToDecimal(txtPower.Text);
                cmd.ExecuteNonQuery();
                MessageBox.Show("Ο  υπολογισμός  ολοκληρώθηκε",  "Υπολογισμός
κόστους", MessageBoxButtons.OK, MessageBoxIcon.Information);
            }

//Περίπτωση σφάλματος

            catch (Exception ex)
            {
                MessageBox.Show("Ο  υπολογισμός  απέτυχε:  "  +  ex.Message,
"Υπολογισμός κόστους", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
            finally
            {
                if (cnn != null)
                {
                    if (cnn.State == ConnectionState.Open)
                    {
                        cnn.Close();
                    }
                }
            }
        }


//     Κουμπί καθαρισμού πλαισίου αποτελεσμάτων

         private void btnClearResults_Click(object sender, EventArgs e)
         {
             this.gvData.DataSource = null;
         }
    }
}
```

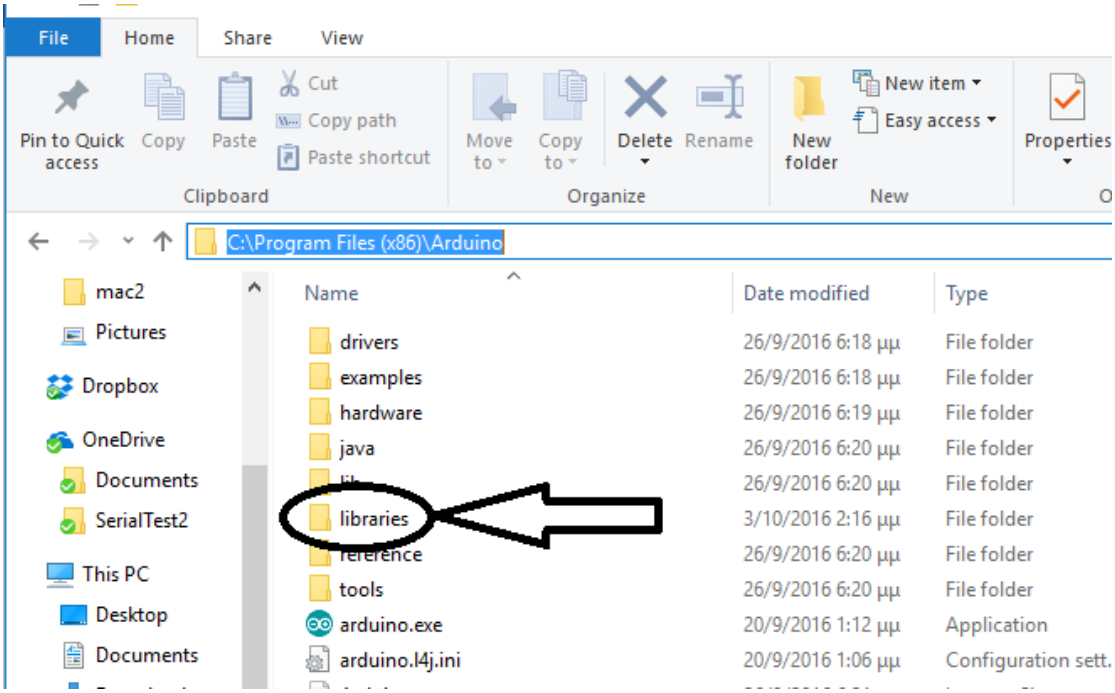
Στο επόμενο κεφάλαιο θα δίνει εκτενής παρουσίαση του τρόπου λειτουργίας του εκάστοτε προγράμματος.

6.4. Η Λειτουργία της Εργασίας

6.4.1. Η Προεγκατάσταση

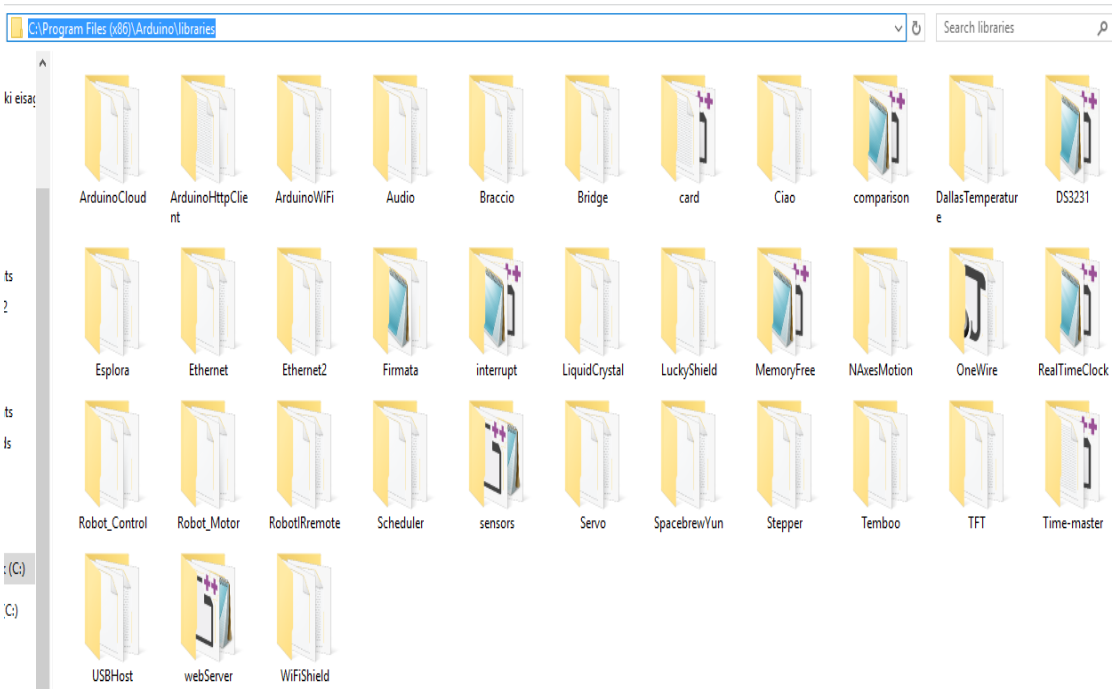
Στο παρόν κεφάλαιο θα γίνει αναλυτική παρουσίαση της συνδεσμολογίας των εξαρτημάτων της εργασίας και του τρόπου λειτουργίας των προγραμμάτων.

Σαν πρώτο βήμα πρέπει να γίνει η εγκατάσταση των βιβλιοθηκών του Arduino στην τοποθεσία εγκατάστασης του Arduino IDE. Στην παρούσα περίπτωση είναι: C:\Program Files (x86)\Arduino. Στην Εικόνα 61 διακρίνεται ο φάκελος libraries που τοποθετούνται οι κάτωθι βιβλιοθήκες (Εικόνα 62).



Εικόνα 60: Ο φάκελος εγκατάστασης του Arduino IDE και ο φάκελος libraries

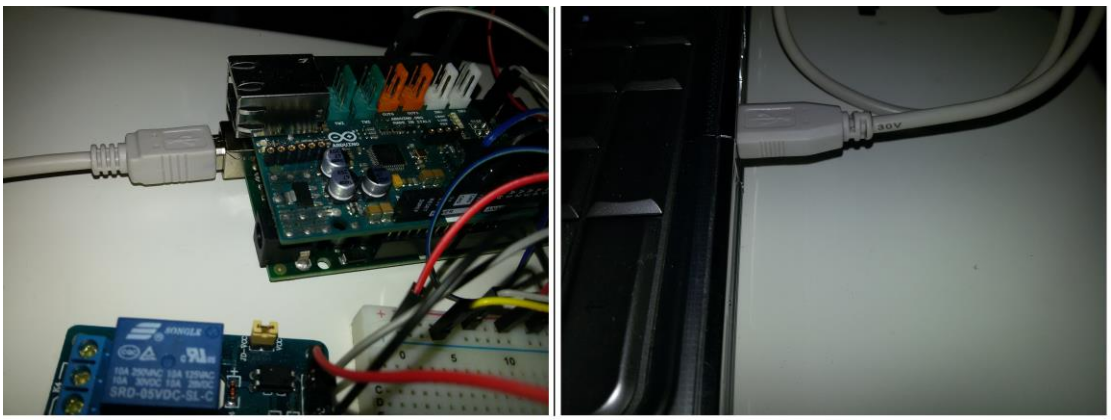
Ο χρήστης καλείται να επιβεβαιώσει αν διαθέτει όλες τις απαραίτητες βιβλιοθήκες που φαίνονται στην Εικόνα 61. Οι βιβλιοθήκες διατίθενται είτε από τον διαδικτυακό χώρο www.arduino.cc είτε από το CD της εργασίας για τις βιβλιοθήκες που γράφτηκαν για την εργασία πιο συγκεκριμένα.



Εικόνα 61: Οι απαραίτητες βιβλιοθήκες της εργασίας

Οι απαραίτητες βιβλιοθήκες είναι οι εξής: webserver.h, SD.h, card.h, DS3231.h, Ethernet2.h, interrupt.h, RealTimeClock.h,sensors.h, variables.h, sensors.h, debug.h, result.h, SPI.h.

Κατόπιν, πραγματοποιείται η σύνδεση του Aduino. Ο Arduino συνδέεται μέσω καλωδίου USB με τον προσωπικό υπολογιστή του χρήστη (Εικόνα 62).



Εικόνα 62: Η σύνδεση του Arduino με τον προσωπικό υπολογιστή

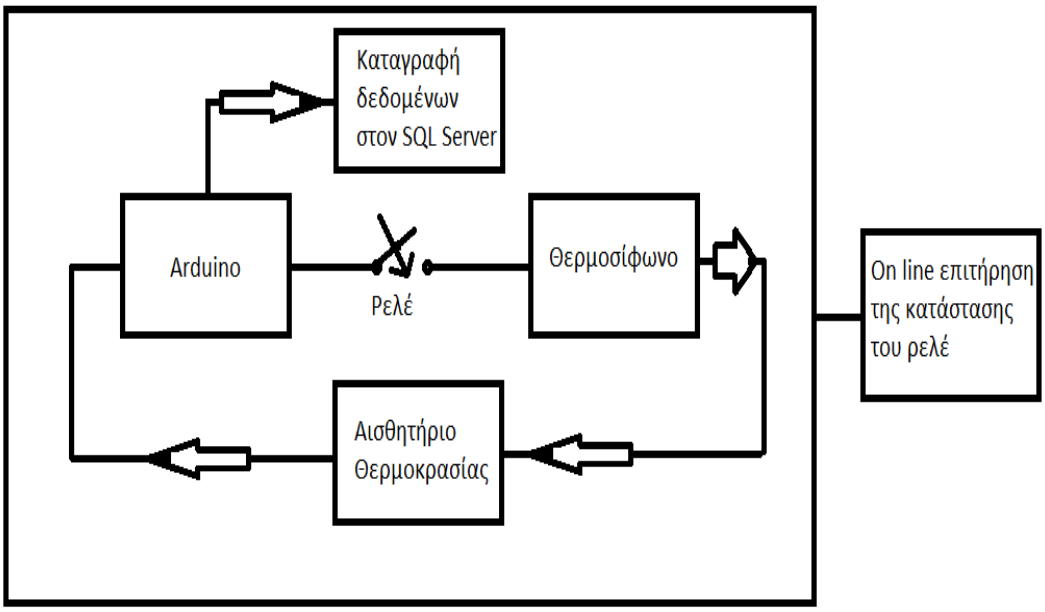
Η σύνδεση υπό άλλες συνθήκες μπορεί να πραγματοποιηθεί άπαξ για να μεταφορτωθεί το πρόγραμμα στον Arduino έχοντας συνδεδεμένη σταθερή τροφοδοσία και κατόπιν να αποσυνδεθεί, όμως υπό τις παρούσες συνθήκες η σύνδεση πρέπει να είναι σταθερή για να μεταφέρονται τα δεδομένα στη βάση δεδομένων. Ακολουθεί η σύνδεση του πίνακα προστασίας (Εικόνα 63) με το ρελέ του Arduino. Η πραγματοποίηση του κρίθηκε αναγκαία καθώς είναι απαραίτητη η προστασία της όλης κατασκευής (Arduino και βραστήρα) από αυξομειώσεις της έντασης του ρεύματος λόγω της ευαισθησίας και του κόστους που έχουν.

Εικόνα 63: Η σύνδεση του Arduino με τον πίνακα προστασίας και το βραστήρα

Τέλος, το αισθητήριο θερμοκρασίας του Arduino βυθίζεται μέσα στο βραστήρα που παίζει το ρόλο του θερμοσίφωνα (Εικόνα 64).

Εικόνα 64: Η βύθιση του αισθητηρίου θερμοκρασίας του Arduino στο βραστήρα

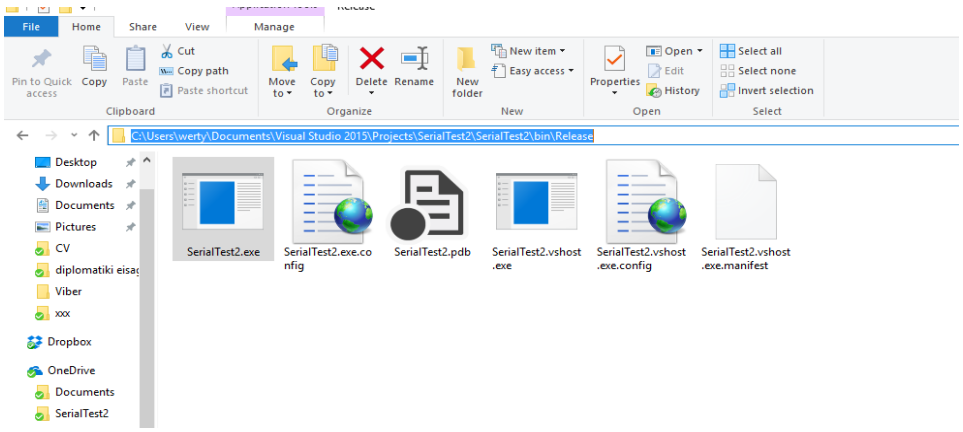
Ακολουθεί ένα διάγραμμα της λογικής της υλοποίησης της συνδεσμολογίας (Εικόνα 65).



Εικόνα 65: Το διάγραμμα της λογικής της συνδεσμολογίας

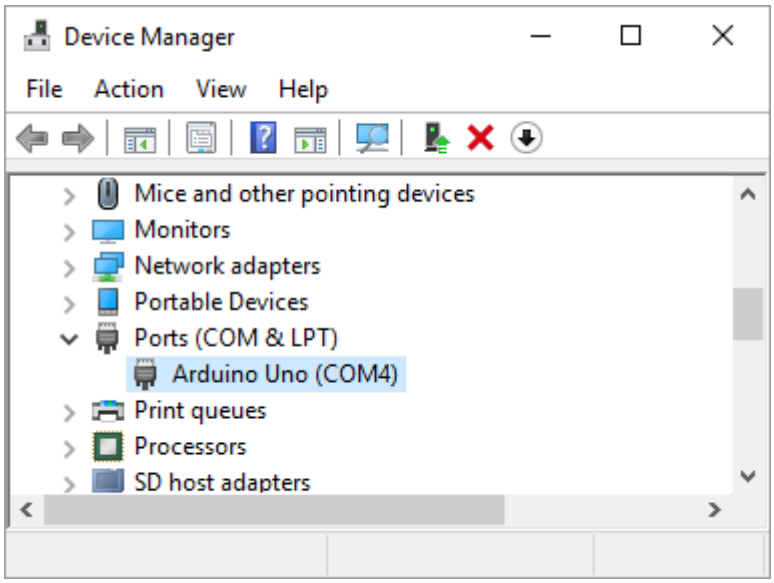
6.4.2. Η Εκκίνηση της Λειτουργίας

Αφού πραγματοποιηθεί η συνδεσμολογία, είναι έτοιμη η εκκίνηση της εφαρμογής που γίνεται εκτελώντας το πρόγραμμα SerialTest2.exe. Το πρόγραμμα υπάρχει στο φάκελο C:\Visual Studio 2015\Projects\SerialTest2\SerialTest2\bin\Release (Εικόνα 66).



Εικόνα 66: Το προς εκτέλεση πρόγραμμα

Με την εκτέλεση του προγράμματος εμφανίζεται η οθόνη εργασίας, όπου παρακολουθείται η σειριακή έξοδος του Arduino. Οι Εικόνες 51 έως 56 έδωσαν μία σαφή αναφορά. Ο χρήστης έχει σαν αρχική υποχρέωση να ορίσει την COM Port όπου θα παρακολουθείται η σειριακή έξοδος του Arduino. Για να ευρεθεί αυτό το ζητούμενο ακολουθούνται τα εξής βήματα: This PC -> System Properties -> Device Manager και από εκεί αναζητείται το αναδυόμενο μενού Ports (COM & LPT) όπως φαίνεται στην Εικόνα 67. Κάνοντας ένα απλό κλικ, εμφανίζεται η COM (στην περίπτωσή της εργασίας η COM4) που χρησιμοποιεί ο Arduino.

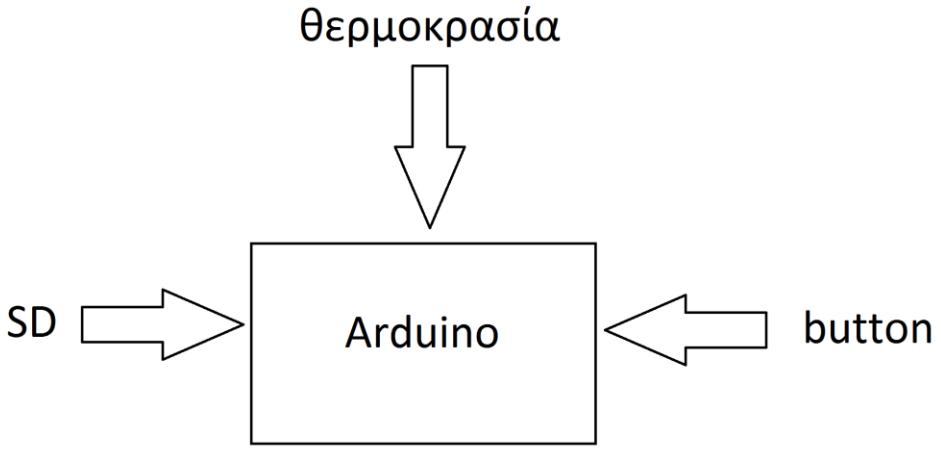


Εικόνα 67: Η COM Port του Arduino

Για τη συνέχεια, κλείνουν τα προηγούμενα παράθυρα αναζήτησης και απλά συμπληρώνεται η απαιτούμενη COM Port στην οθόνη εργασίας του προγράμματος. Η διαδικασία της σειριακής θύρας εκκινεί πατώντας το κουμπί “Open Serial”. Όπως έχει σχολαστικά αναφερθεί σε προηγούμενο κεφάλαιο, τότε ξεκινά η ανάγνωση της σειριακής θύρας του Arduino. Η σειριακή θύρα του Arduino για να γίνει απολύτως κατανοητό, είναι ο “καθρέπτης” της εργασίας. Δηλαδή, ό,τι ερεθίσματα δέχεται ο Arduino από το περιβάλλον όπως θερμοκρασία, ανοιγοκλείσιμο του διακόπτη λειτουργίας, είσοδος – έξοδος της κάρτας SD, όλα αυτά τα χρήσιμα δεδομένα αντανακλώνται στην σειριακή θύρα (Εικόνα 52, “Κάτω Πλαίσιο”). Βέβαια, όλα τα δεδομένα δεν είναι εξίσου απαραίτητα. Τη βάση δεδομένων την ενδιαφέρουν τα δεδομένα κατά τη διάρκεια που είναι σε λειτουργία ο θερμοσίφωνας. Αυτά τα δεδομένα

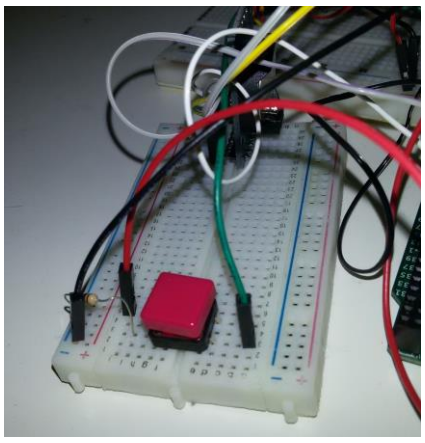
διαχωρίζονται και κρατούνται σε ξεχωριστή οθόνη (στο “Άνω Πλαίσιο”), όπως έχει παρουσιαστεί στην Εικόνα 52.

Με το πάτημα του κουμπιού “Open Serial” ξεκινά η ανάγνωση της κατάστασης του Arduino και μεταφέρεται στη σειριακή. Ο Arduino βρίσκεται σε μία συνεχή κατάσταση αναμονής ερεθισμάτων από το περιβάλλον. Τα ερεθίσματα από το περιβάλλον είναι: η θερμοκρασία, το κουμπί ενεργοποίησης – απενεργοποίησης και τα προγραμματισμένα γεγονότα από την SD (Εικόνα 68).



Εικόνα 68: Οι είσοδοι του Arduino

Πατώντας ο χρήστης απλά το κουμπί ενεργοποίησης – απενεργοποίησης καταφέρνει να θέτει και να σβήνει διαδοχικά τη λειτουργία του θερμοσίφωνα. Στο άνω πλαίσιο καταχωρούνται τα πλήρη στοιχεία της διάρκειας ενεργοποίησης, τα οποία παράλληλα καταχωρούνται και στη βάση δεδομένων.

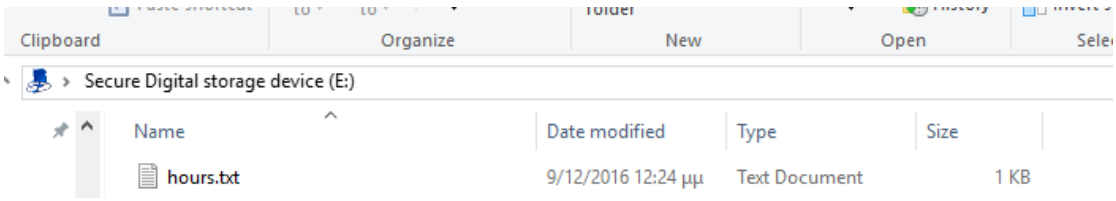


Εικόνα 69: Το κουμπί ενεργοποίησης - απενεργοποίησης

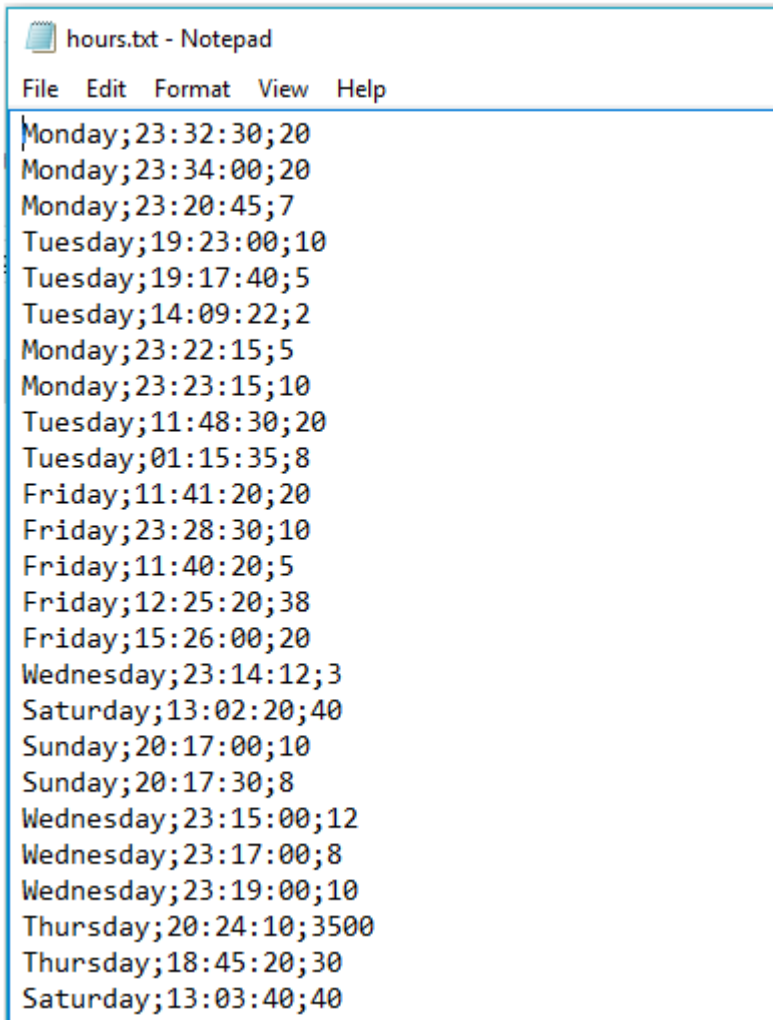
Πέρα από τον χειροκίνητο τρόπο ενεργοποίησης – απενεργοποίησης ενός συμβάντος, υπάρχει και ο προγραμματισμένος από τον χρήστη. Ένας κατάλογος με προγραμματισμένες ενεργοποιήσεις καταχωρείται στην κάρτα SD στο φάκελο hours.txt (Εικόνα 71).

Εικόνα 70 (από αριστερά προς τα δεξιά): Η κάρτα micro SD, ο προσαρμογέας (adaptor) για τον προσωπικό υπολογιστή και η τοποθέτηση τους στην αντίστοιχη θύρα.

Κάνοντας ένα διπλό κλικ επάνω στο εικονίδιο του αρχείου hours.txt και ανοίγοντας το αρχείο με έναν απλό κειμενογράφο, παρουσιάζονται τα προγραμματισμένα γεγονότα προς ενεργοποίηση του θερμοσίφωνα (Εικόνα 72). Τα γεγονότα που παρουσιάζονται είναι ενδεικτικά και μπορούν να τροποποιηθούν σύμφωνα με τις απαιτήσεις τους εκάστοτε χρήστη. Η σύνταξη της κάθε γραμμής γίνεται με ορισμένο τρόπο και παρουσιάζεται στη συνέχεια.



Εικόνα 71: Το αρχείο hours.txt

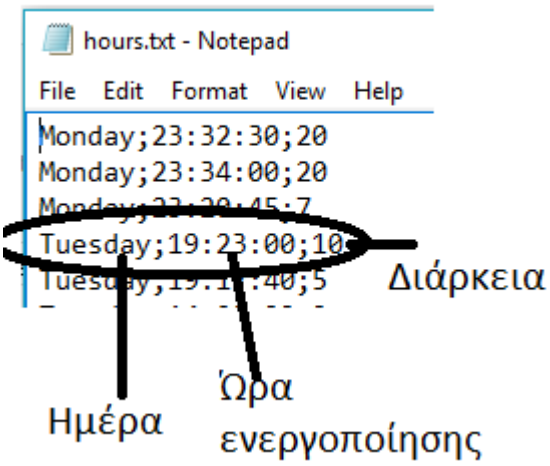


Εικόνα 72: Τα προγραμματισμένα γεγονότα ενεργοποίησης του θερμοσίφωνα

Η σύνταξη της γραμμής έχει συγκεκριμένο τρόπο. Όπως παρατηρείται πρώτα αναγράφεται η ημέρα (στα αγγλικά), μετά με ελληνικό ερωτηματικό διαχωρίζεται η επόμενη γραφή που είναι η ακριβής ώρα ενεργοποίησης και τέλος με ελληνικό ερωτηματικό διαχωρίζεται η επόμενη γραφή που είναι η διάρκεια εκτέλεσης, σε δευτερόλεπτα, της ενεργοποίησης του θερμοσίφωνα (Εικόνα 73).

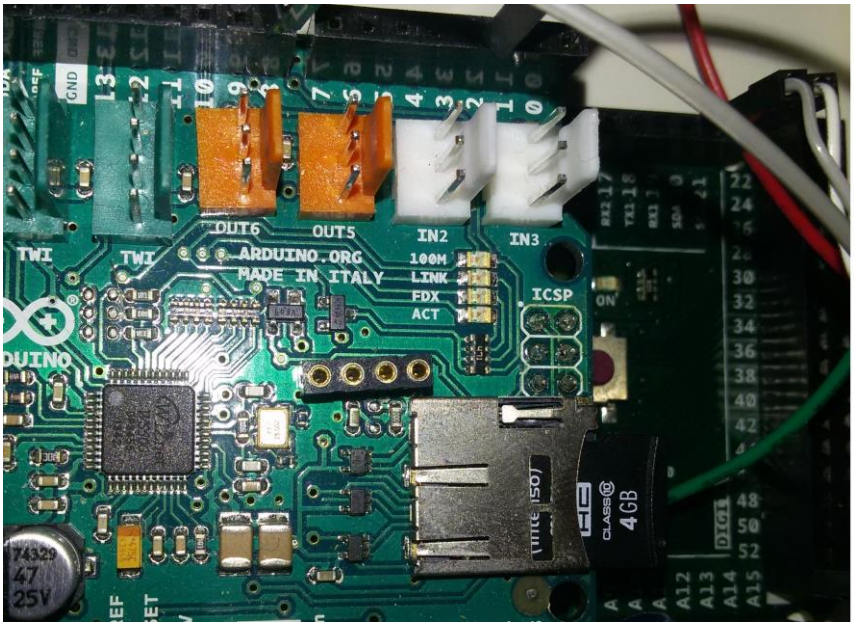
Πρέπει να τονιστεί ότι, είτε στα αυτοματοποιημένα είτε στα χειροκίνητα συμβάντα υπάρχει η κρίσιμη θερμοκρασία των 70 βαθμών κελσίου που απενεργοποιείται αυτόματα η θέρμανση του νερού. Η επιλογή της συγκεκριμένης θερμοκρασίας έχει γίνει βάσει του ότι στην εργασία χρησιμοποιείται ένας απλός βραστήρας (το δοχείο του νερού είναι μικρό και η θερμοκρασία ανακτάται και χάνεται σχετικά γρήγορα) και του ότι είναι μία θερμοκρασία που ικανοποιεί της απαιτήσεις μίας πειραματικής διαδικασίας

(δηλαδή οι συμβαίνουσες αποκλίσεις από την αναγραφόμενη θερμοκρασία μας εξασφαλίζουν την ασφαλή διεξαγωγή του πειράματος).



Εικόνα 73: Η γραμμή με τα δεδομένα ενεργοποίησης ενός συμβάντος. Μεταξύ των δεδομένων διακρίνεται το ελληνικό ερωτηματικό που τα διαχωρίζει

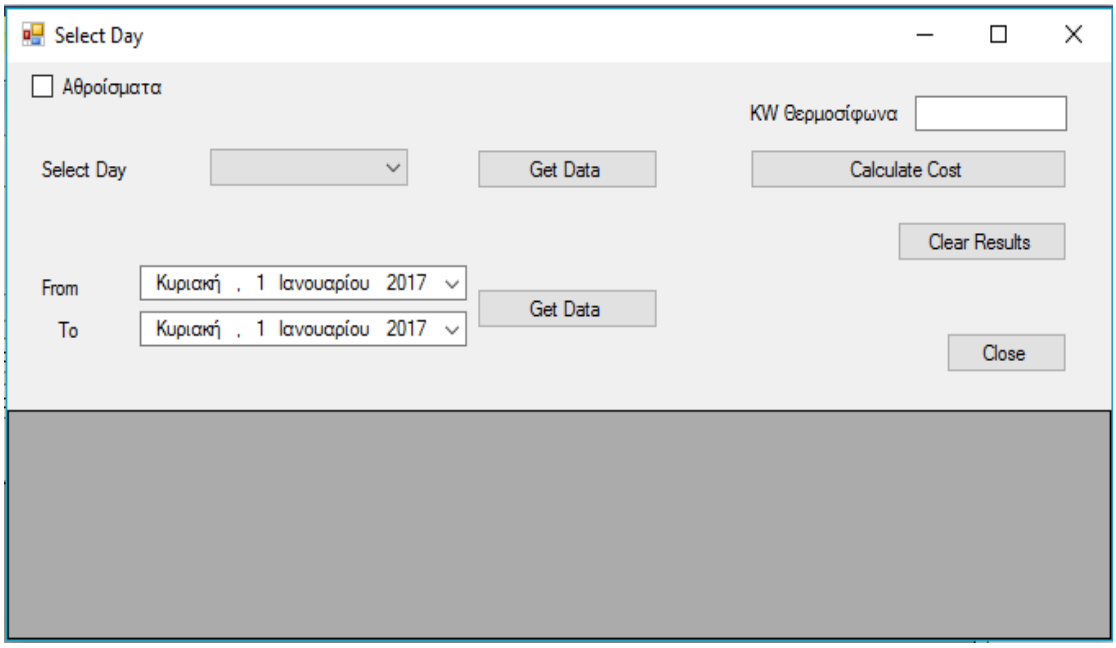
Όταν ολοκληρωθεί η σύνταξη του αρχείου hours.txt, ο χρήστης καλείται να κάνει αποθήκευση του αρχείου, ασφαλή κατάργηση της micro SD από τον προσωπικό του υπολογιστή και κατόπιν να τοποθετήσει την κάρτα στον Arduino (Εικόνα 74), όπου βρίσκεται σε ενεργή κατάσταση αναμονής. Χωρίς τοποθετημένη την κάρτα SD, ο Arduino μπορεί να δουλέψει μόνο χειροκίνητα.



Εικόνα 74: Τοποθέτηση της κάρτας SD στον Arduino

Τα γεγονότα που αναγράφονται στο αρχείο hours.txt, εκτελούνται σε εβδομαδιαία βάση και επ' άπειρον, εφόσον η κάρτα SD δεν αφαιρεθεί από τη θύρα του Arduino

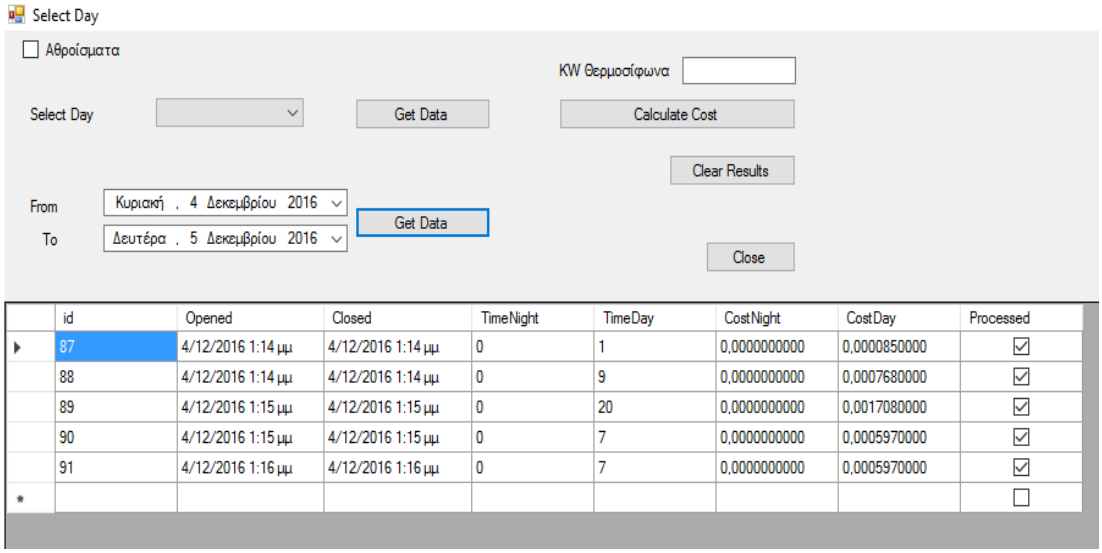
Ενώ το SerialTest2.exe εκτελείται, ο χρήστης μπορεί να ανακτήσει πληροφορίες της λειτουργίας του θερμοσίφωνου. Αυτές οι πληροφορίες ανακτώνται πατώντας το κουμπί Select Day που βρίσκεται κάτω αριστερά στο περιβάλλον εργασίας. Με την εκκίνηση του εκτελείται το πρόγραμμα selectDay.exe. Το εν λόγω πρόγραμμα κάνει τους υπολογισμούς και την ανάκτηση δεδομένων από τη βάση δεδομένων στον SQL Server. Χάριν απλούστευσης στην επόμενη εικόνα (Εικόνα 75), παρουσιάζεται η φόρμα εργασίας του χρήστη.



Εικόνα 75: Η φόρμα εργασίας του χρήστη

Αρχικά συμπληρώνονται τα kiloWatts του θερμοσίφωνα. Στην περίπτωση της εργασίας είναι 2 kW. Στην συνέχεια, μόλις γίνουν όλες οι λειτουργίες του θερμοσίφωνα, πατιέται το κουμπί "Calculate Cost". Σαν πρώτο ζητούμενο, μπορεί να προκύψει το ερώτημα της ημερήσιας κατανάλωσης. Για να προκύψουν αποτελέσματα αρκεί στο πεδίο "From" (Από) να συμπληρωθεί η ζητούμενη ημερομηνία, για παράδειγμα η ημέρα "Κυριακή 4 Δεκεμβρίου 2016" και στο πεδίο "Το" (Έως) να τοποθετηθεί η επόμενη ημέρα δηλαδή "Δευτέρα 5 Δεκεμβρίου 2016" (Εικόνα 76). Δηλαδή, τα αποτελέσματα λαμβάνονται με τη ανισοτική λογική στο πεδίο "From" του "μεγαλύτερου ίσου" και στο πεδίο "Το" με τη λογική του "μικρότερου".

Συμπληρώνοντας την ημέρα ακολούθως πατιέται το κουμπί “Get Rates” που βρίσκεται πλησίον των ημερομηνιών και λαμβάνονται τα αποτελέσματα. Παρατηρώντας την Εικόνα 76, διακρίνονται τα ημερήσια αποτελέσματα της “Κυριακής 4 Δεκεμβρίου 2016”.



id	Opened	Closed	TimeNight	TimeDay	CostNight	CostDay	Processed
87	4/12/2016 1:14 μμ	4/12/2016 1:14 μμ	0	1	0,0000000000	0,0000850000	☑
88	4/12/2016 1:14 μμ	4/12/2016 1:14 μμ	0	9	0,0000000000	0,0007680000	☑
89	4/12/2016 1:15 μμ	4/12/2016 1:15 μμ	0	20	0,0000000000	0,0017080000	☑
90	4/12/2016 1:15 μμ	4/12/2016 1:15 μμ	0	7	0,0000000000	0,0005970000	☑
91	4/12/2016 1:16 μμ	4/12/2016 1:16 μμ	0	7	0,0000000000	0,0005970000	☑
*							☐

Εικόνα 76: Τα ζητούμενα αποτελέσματα

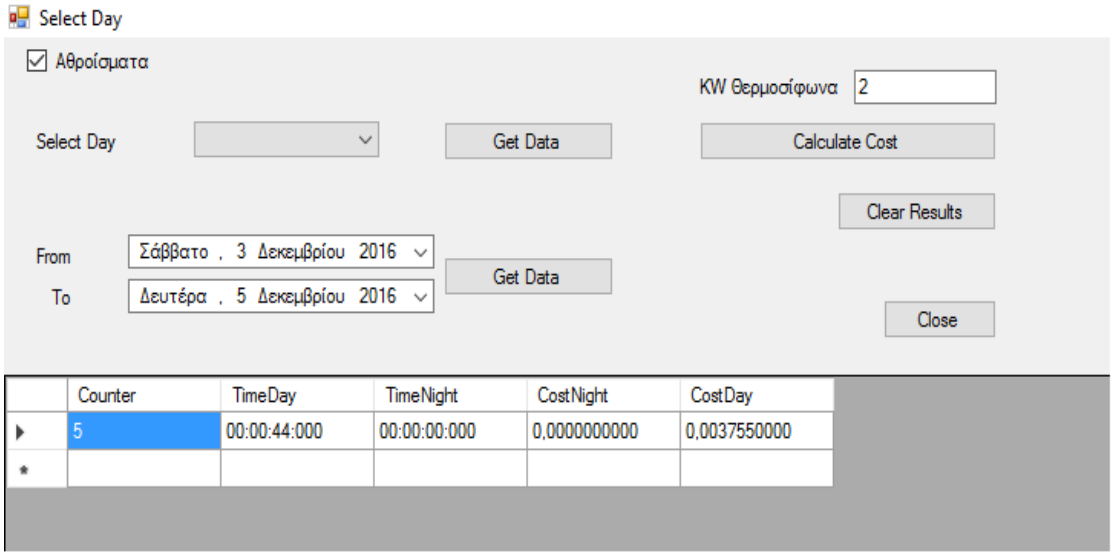
Τα πεδία που δίνουν πληροφορίες για τα αποτελέσματα είναι τα εξής (από τα αριστερά προς τα δεξιά στην Εικόνα 76): id, Opened, Closed, TimeNight, TimeDay, CostNight, CostDay Processed.

Αναλυτικά, το “id” είναι ο αύξων αριθμός της εγγραφής μέσα στη βάση δεδομένων και είναι μοναδικός ανά εγγραφή. Το “Opened” δίνει την πληροφορία του πότε ξεκίνησε να λειτουργεί ο θερμοσίφωνας (ημερομηνία και ώρα με ακρίβεια λεπτού) και το πεδίο “Closed” δίνει την πληροφορία του πότε σταμάτησε η λειτουργία του. Τα πεδία “TimeNight” και “TimeDay” δίνουν την πληροφορία του χρόνου σε δευτερόλεπτα που λειτούργησε ο θερμοσίφωνας και αν η λειτουργία του ανήκει στο νυκτερινό ή το ημερήσιο τιμολόγιο αντίστοιχα. Για παράδειγμα, παρατηρώντας την εγγραφή 89, προκύπτει το συμπέρασμα ότι ο θερμοσίφωνας λειτούργησε 20 δευτερόλεπτα στο ημερήσιο τιμολόγιο. Οπότε και η τιμολόγηση θα γίνει βάσει του ημερήσιου τιμολογίου του παρόχου ηλεκτρικής ενέργειας (στην εργασία έχει ληφθεί υπόψιν το τιμολόγιο της ΔΕΗ και αναφέρεται στο Παράρτημα) και παρουσιάζεται στα πεδία "CostNight” και “CostDay”. Άρα για την εγγραφή 89, αφού ανήκει στο ημερήσιο τιμολόγιο το κοστολόγιο παρουσιάζεται στο πεδίο “CostDay” και είναι 0,001708 ευρώ. Το τελευταίο πεδίο “Processed” δηλώνει ότι η εγγραφή έχει καταχωρηθεί επιτυχώς στη βάση δεδομένων του χρήστη. Ο χειρισμός της

πληροφορίας γίνεται παρόμοια και για ένα εύρος ημερών. Επί παραδείγματι, αν ο χρήστης θέσει ζητούμενο για την λειτουργία του θερμοσίφωνα το Σαββατοκύριακο 3 και 4 Δεκεμβρίου 2016, αρκεί να πατήσει το κουμπί “Clear Results” για να καθαρίσει η οθόνη αποτελεσμάτων και να θέσει στα πεδία “From” την “3η Δεκεμβρίου 2016” και στο “Το” την “5η Δεκεμβρίου 2016” μην ξεχνώντας την ανισοτική λογική που αναφέρθηκε στην προηγούμενη παράγραφο.

Το εύρος των ημερών δεν έχει όριο και δύναται να υπολογιστεί εύρος καθημερινών ημερών, εύρος μήνα, εύρος εποχής, εύρος έτους και γενικά ό,τι επιθυμείται από τον χρήστη, αρκεί για τους σωστούς υπολογισμούς να τηρείται η ανισοτική λογική που είναι η σύμβαση της εργασίας.

Το δεύτερο ζητούμενο ενός χρήστη μπορεί να είναι η απαίτηση να μη βλέπει αποσπασματικά αλλά αθροιστικά τα αποτελέσματα στην οθόνη. Δηλαδή, για την περίπτωση του Σαββατοκύριακου 3 και 4 Δεκεμβρίου 2016 να παρουσιάζεται ο συνολικός χρόνος ανά τιμολόγιο που λειτούργησε ο θερμοσίφωνας και τα αντίστοιχα συνολικά κοστολόγια (Εικόνα 77). Για την περίπτωση αυτή αρκεί να επιλεχτεί η επιλογή “Αθροίσματα” που βρίσκεται στο επάνω δεξιά μέρος της φόρμας εργασίας και να πατηθεί το κουμπί “Get Rates” που βρίσκεται πλησίον των ημερομηνιών.



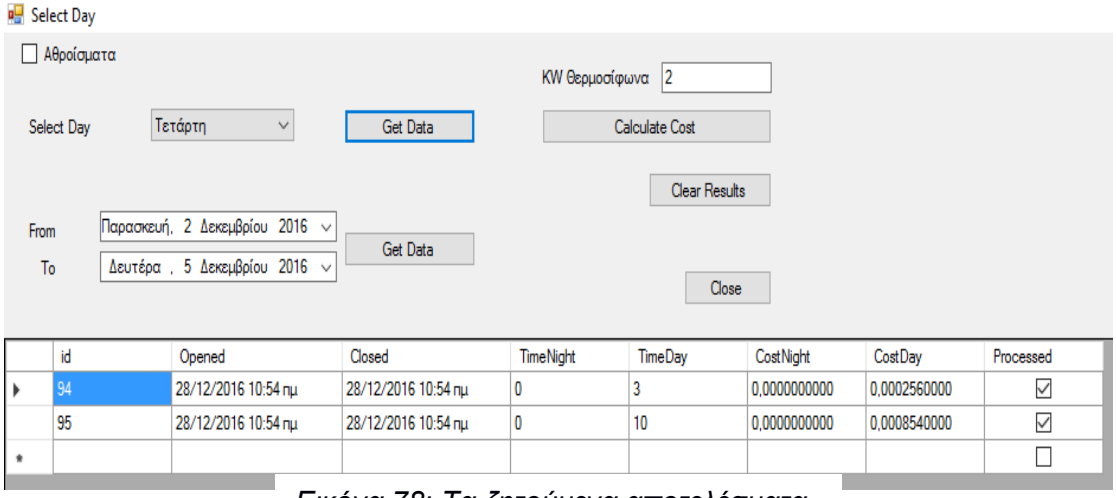
Εικόνα 77: Τα αθροιστικά αποτελέσματα

Πλέον εμφανίζονται τα εξής πεδία αποτελεσμάτων (από αριστερά προς τα δεξιά): τα πεδίο “Counter”, “TimeDay”, “TimeNight”, “CostNight” και “CostDay”.

Το πεδίο “Counter” δηλώνει πόσες συνολικά ενεργοποιήσεις της λειτουργίας έγιναν την εν λόγω χρονική περίοδο, τα πεδία “TimeDay” και “TimeNight” το συνολικό χρόνο ανά ζώνη τιμολόγησης (ημερήσια ή νυκτερινή αντίστοιχα). Τέλος, τα πεδία “CostNight” και “CostDay” αναφέρουν τα αντίστοιχα αθροιστικά κόστη ανά περιοχή τιμολόγησης. Στην Εικόνα 77 το “Counter” έχοντας τιμή 5 δηλώνει ότι υπήρχαν 5 ενεργοποιήσεις της λειτουργίας του θερμοσίφωνου, το “TimeDay” αναγράφοντας 00:00:44:000 δηλώνει ότι συνολικά 44 δευτερόλεπτα λειτούργησε ο θερμοσίφωνας τη ζητούμενη περίοδο και το “TimeNight” με 00:00:00:000 δηλώνει μηδενική λειτουργία, γεγονότα που αντικατοπτρίζονται και στα αντίστοιχα πεδία με τα κόστη “CostDay” με 0,003755 ευρώ και “CostNight” με μηδενικό κόστος.

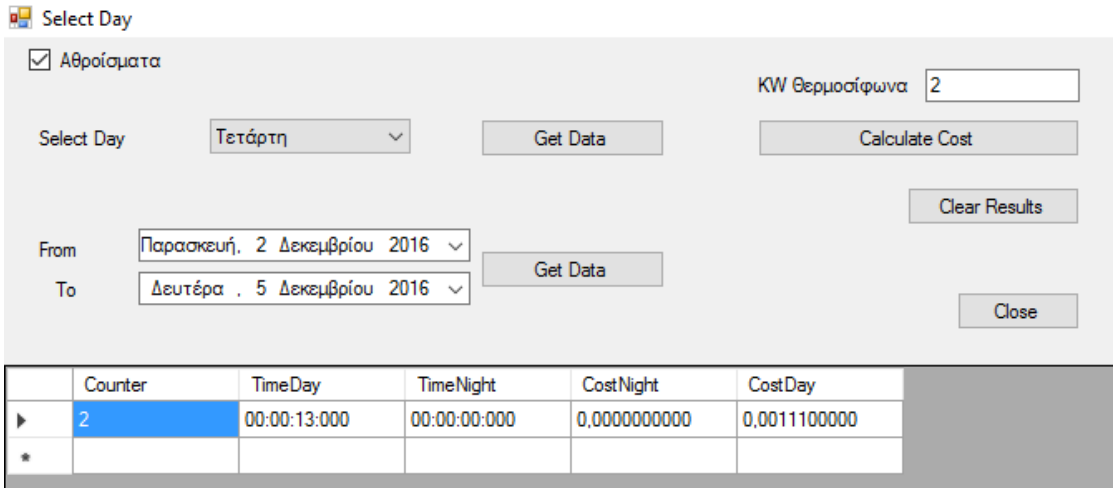
Σε περίπτωση που ο χρήστης θέλει να επανέλθει στην προηγούμενη εικόνα με τα αποσπασματικά και αναλυτικά αποτελέσματα των ενεργοποιήσεων του θερμοσίφωνου, αρκεί να καταργήσει την επιλογή “Αθροίσματα” και να πατήσει το κουμπί “Get Rates” πλησίον των ημερομηνιών.

Η επόμενη λειτουργία που παρέχεται από το πρόγραμμα προς το χρήστη είναι αποτελέσματα που αφορούν μία συγκεκριμένη ημέρα της εβδομάδος. Για παράδειγμα, να αναφέρονται αποτελέσματα που αφορούν μόνο την ημέρα “Δευτέρα”. Έτσι ο χρήστης θα μπορεί να φτιάξει μία εικόνα του καταναλωτικού του προφίλ ανά ημέρα της εβδομάδος και να διακρίνει τα κόστη και το χρόνο λειτουργίας που διαθέτει για τη θέρμανση του νερού. Στην επόμενη εικόνα (Εικόνα 78) φαίνεται μία τέτοια εφαρμογή, αφού προηγουμένως έχει πατηθεί το κουμπί “Clear Results”. Το παράδειγμα της ακόλουθης εικόνας αναφέρεται στην ημέρα “Τετάρτη”. Την εν λόγω ημέρα φαίνεται ότι έχουν γίνει συνολικά μόνο 2 ενεργοποιήσεις, έχουν αύξοντα αριθμό 94 και 95 στη βάση δεδομένων και τα λοιπά πεδία έχουν τα στοιχεία όπως παρουσιάστηκαν παραπάνω. Για την λήψη των αποτελεσμάτων πατιέται το κουμπί “Get Rates” που βρίσκεται στο ύψος του πεδίου “Select Day”.



Εικόνα 78: Τα ζητούμενα αποτελέσματα

Τα στοιχεία της ημέρας της εβδομάδος μπορούν να εμφανιστούν και με αθροιστικό χαρακτήρα, ενεργοποιώντας το πεδίο “Αθροίσματα” στο επάνω αριστερά μέρος της φόρμας εργασίας και πατώντας το κουμπί “Get Rates” που βρίσκεται στο ύψος του πεδίου “Select Day”(Εικόνα 79).



Εικόνα 79: Τα αθροιστικά αποτελέσματα

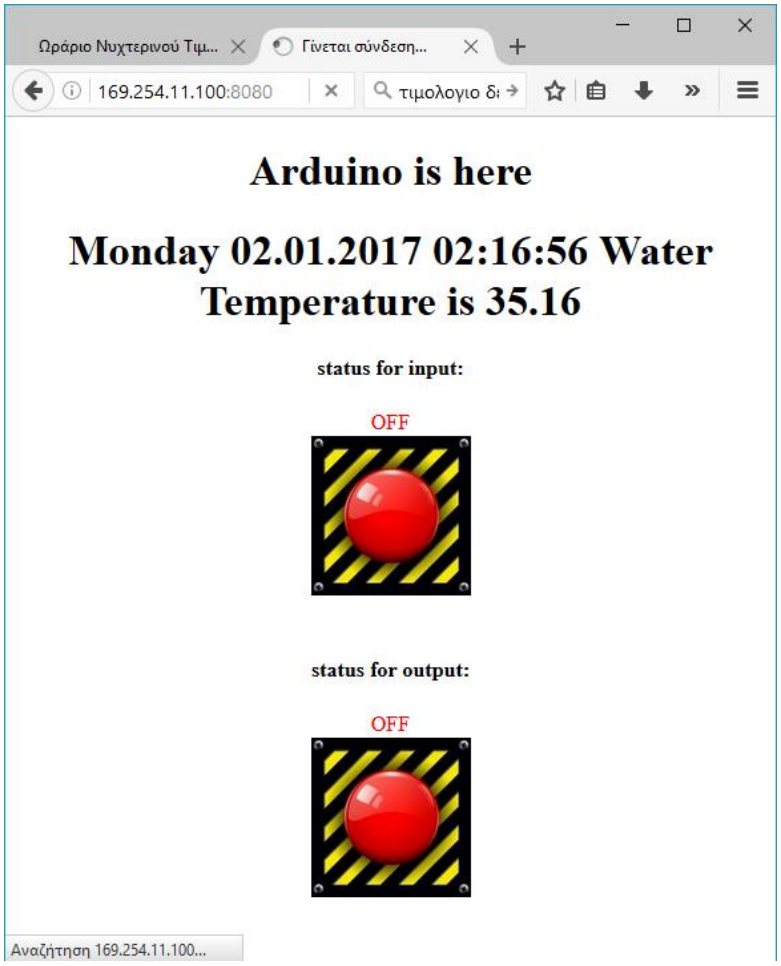
Τα πεδία αναφέρονται όπως και στην προηγούμενη περίπτωση των αθροιστικών αποτελεσμάτων.

Απενεργοποιώντας το πεδίο “Αθροίσματα” και πατώντας το κουμπί “Get Data” που βρίσκεται στο ύψος του πεδίου “Select Day” ο χρήστης επανέρχεται στα αναλυτικά αποτελέσματα.

Το κουμπί “Close” κλείνει τη φόρμα εργασίας “Select Day”, αφήνοντας το πρόγραμμα SerialTest2 να εκτελείται. Για να απενεργοποιηθεί και το SerialTest2 πατιέται το κουμπί “Exit” του εν λόγω προγράμματος.

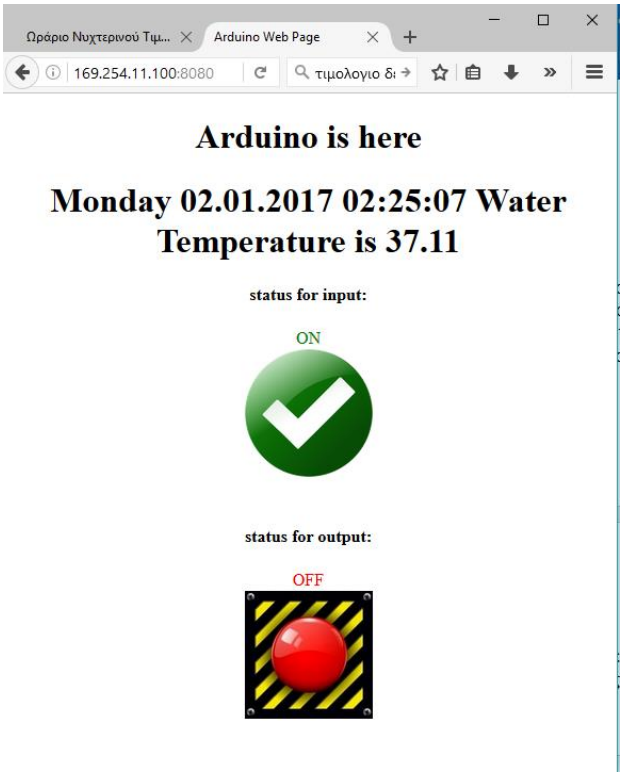
6.4.3. Η Επιτήρηση της Λειτουργίας

Πρέπει να αναφερθεί πριν το κλείσιμο της παρουσίασης της λειτουργίας της εργασίας ότι όσο λειτουργεί ο Arduino και τα δεδομένα του φαίνονται στην σειριακή του θύρα, η παρακολούθηση της κατάστασης του θερμοσίφωνα δηλαδή αν βρίσκεται σε λειτουργία ή όχι και η θερμοκρασία του νερού, γίνεται σε πραγματικό χρόνο μέσω IP. Οπότε, πληκτρολογώντας την IP του Arduino σε ένα internet browser (στην περίπτωση της εργασίας και του συγκεκριμένου προσωπικού υπολογιστή είναι 169.254.11.100:8080) ο χρήστης μεταφέρεται στην σελίδα του Arduino όπως φαίνεται στην Εικόνα 80.



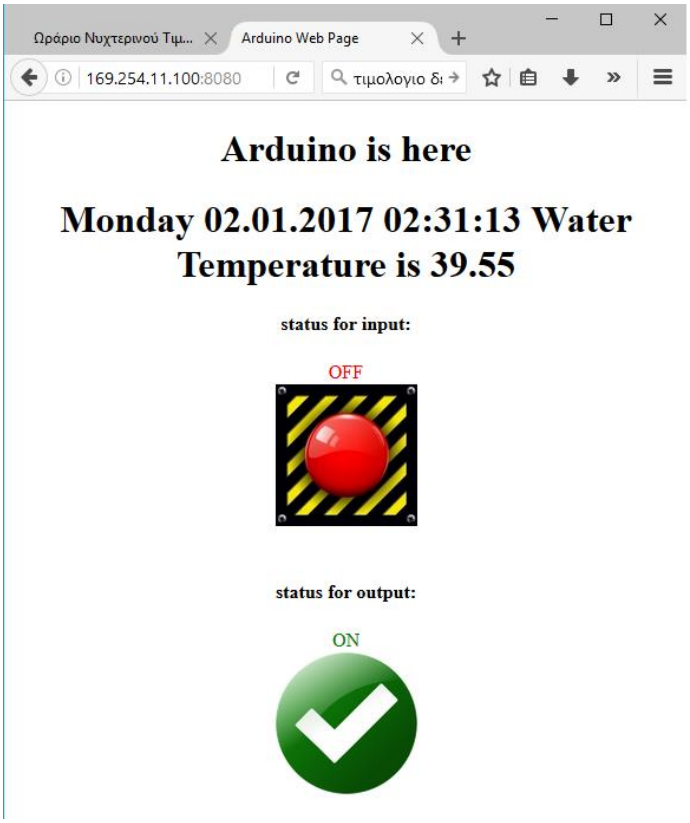
Εικόνα 80: Η σελίδα επιτήρησης λειτουργίας

Παρατηρείται η παρούσα ημερομηνία, ώρα και θερμοκρασία του νερού σε βαθμούς κελσίου. Επίσης, αν έχει ενεργοποιηθεί η είσοδος (με ερέθισμα είτε από το κουμπί χειροκίνητα είτε προγραμματισμένα από την SD) και η κατάσταση της εξόδου (αν λειτουργεί δηλαδή η θέρμανση του νερού). Ο ρυθμός ανανέωσης της σελίδας πληροφοριών είναι ανά 0,2 δευτερόλεπτα ήτοι 5 φορές μέσα σε ένα δευτερόλεπτο. Σε περίπτωση ενεργοποίησης της εισόδου η ειδοποίηση που φαίνεται στη σελίδα είναι σαν της Εικόνας 81 (το εικονίδιο “status for input” αλλάζει στιγμιαία).



Εικόνα 81: Η στιγμή ενεργοποίησης της εισόδου

Έκτοτε, ενεργοποιείται η έξοδος δηλαδή η λειτουργία του θερμοσίφωνα (εικονίδιο “status for output”) και παρουσιάζεται στην Εικόνα 82 η νέα μορφή της σελίδας.



Εικόνα 82: Η ενεργοποίηση του θερμοσίφωνου

6.5. Προτάσεις Επέκτασης της Εργασίας

Πληθώρα μελλοντικών επεκτάσεων μπορούν να συμβούν στην παρούσα εργασία. Αρχικά πρέπει να αναφερθεί ότι η εργασία δίνει τη δυνατότητα να προσαρμοστεί οποιαδήποτε ηλεκτρική συσκευή ενδιαφέρει το χρήστη προς επιτήρηση, είτε άμεσα κατά αντικατάσταση του βραστήρα και προσαρμογή νέας συσκευής είτε κατά παραλληλία με μικρή επέμβαση στην όλη κατασκευή. Στην περίπτωση της παραλληλίας λειτουργίας συσκευών, ορθό θα ήταν να δημιουργηθεί ανά συσκευή και μία νέα βάση δεδομένων όπου θα αποθηκεύονται τα εκάστοτε στοιχεία της συσκευής. Επίσης πρέπει ανά συσκευή να προσαρμοστεί και ένα ρελέ ελέγχου από τον Arduino και ο πίνακας ελέγχου να έχει τα αντίστοιχα ρελέ και διακόπτες προστασίας. Τέλος, για ορθή επιτήρηση των συσκευών, η ιστοσελίδα επιτήρησης του Arduino θα αναφέρει την κατάσταση τους.

Μία σημαντική πρόταση, που βρίσκει πρόσφορο έδαφος προς υλοποίηση, είναι και η ιδέα η ιστοσελίδα να μην παρέχει μόνο τη δυνατότητα επιτήρησης αλλά να προσφέρει και έλεγχο των συσκευών. Δηλαδή, με τη μεταφορά του χρήστη στην ιστοσελίδα του Arduino να υπάρχει δυνατότητα διαδραστικότητας μέσω μία πλήρης λίστας των ελεγχόμενων συσκευών της οικίας ή του χώρου και να ενεργοποιούνται και να απενεργοποιούνται με ένα απλό πάτημα κουμπιού. Έτσι θα προσφέρεται και η δυνατότητα του ελέγχου εξ’ αποστάσεως.

Επεκτείνοντας την ιδέα της προηγούμενης παραγράφου, θα ήταν πολύ σημαντική και η δυνατότητα να αναγράφονται στην ιστοσελίδα ελέγχου και επιτήρησης, τα στοιχεία αποτελεσμάτων της βάσης δεδομένων. Με βάση αυτήν την δυνατότητα ο χρήστης θα αποκτά πρόσβαση και ενημέρωση σε πραγματικό χρόνο της όλης κατανάλωσης είτε μεμονωμένα ανά συσκευή είτε κατά σύνολο της οικίας ή του χώρου σε κάποιο σύνολο συσκευών που θα ορίζει ο χρήστης.

Όμως και σε γενικότερο επίπεδο η “έξυπνη παρακολούθηση” θα μπορούσε να βρει άμεση εφαρμογή και σε άλλους τομείς με αντίστοιχους αισθητήρες λειτουργίας όπως π.χ. η παροχή ειδοποιήσεων σχετικά με την κατάσταση τροφοδοσίας ρεύματος του Arduino ή αν δουλεύει με μπαταρίες να υπάρχει σχετική ενημέρωση, όπως ακόμη και πληροφόρηση με την ποσότητα του νερού που βρίσκεται μέσα στο καζάνι του θερμοσίφωνου ή την σωστή ροή του νερού μέσα στις σωληνώσεις κτλ.

Όσον αναφορά τη ροή του νερού, μία ιδέα που βρήκε άμεση εφαρμογή σε συνεργασία με το τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Πανεπιστημίου Πατρών και παρουσιάστηκε στην έκθεση καινοτομίας “Patras IQ 2016” ήταν η δημιουργία μίας έξυπνης βρύσης η οποία είχε τον έλεγχο της ροής σωστής ποσότητας νερού και η παρούσα εργασία είχε τον έλεγχο της θέρμανσης του νερού. Στην σύνθεση της όλης εργασίας υπήρχε επικοινωνία μεταξύ δύο διαφορετικών μηχανών (Machine to Machine Communication δύο διαφορετικών συσκευών Arduino), που συνεργάστηκαν για την υλοποίηση ενός μεγαλύτερου έργου.

Σχετικά με τις πληροφορίες που αποθηκεύονται στη βάση δεδομένων, οι ιδέες που μπορούν να υλοποιηθούν για τη διάθεση και την επεξεργασία τους είναι αμέτρητες από τη στιγμή που υπάρχει η δυνατότητα χειρισμού τους. Μπορούν να λειτουργήσουν σαν ένα υποσύστημα ανατροφοδότησης για να κάνουν την οικονομική λειτουργία μίας οικίας ή γενικότερα ενός χώρου με ακόμη πιο έξυπνο και αποτελεσματικό τρόπο.

Μεγάλη πρόκληση θα αποτελούσε η ιδέα να μπορεί το σύστημα να μαθαίνει τις συνήθειες θέρμανσης του νερού του χρήστη και να προσπαθεί αυτόματα να βελτιώσει το ενεργειακό προφίλ προς οικονομικό του όφελος,

Όλες αυτές οι παραπάνω ιδέες για επέκταση θα συνδυάζονται με τις υπάρχουσες δυνατότητες της παρούσας εργασίας και θα αποτελούν μία πρόταση για ένα ολοκληρωμένο έξυπνο σπίτι.

Οι ιδέες υλοποίησης απαιτούν συνεργασία μηχανικών, ενεργειακού ηλεκτρολόγου μηχανικού με μηχανικό λογισμικού εξειδικευμένο στα διαδικτυακά εργαλεία. Υπόψιν πρέπει να ληφθεί το γεγονός ότι προς μία επαγγελματική χρήση της εργασίας πρέπει να προηγηθεί δομημένη καλωδίωση του χώρου και σωστή εγκατάσταση σε πίνακα ελέγχου.

7. Βιβλιογραφία

- Jacob Morgan, A Simple Explanation Of 'The Internet Of Things', Forbes Magazine, http://www.forbes.com/sites/jacobmorgan/2014/05/13/simple-explanation-internet-things-that-anyone-can-understand/#52a5f5a56828
- Internet of Things, Wikipedia, <https://en.wikipedia.org/wiki/Internet_of_things>
- Γιώργος Πολύζος, Η Εποχή του Internet of Things, Ναυτεμπορική, <http://www.naftemporiki.gr/story/1022645/i-epoxi-tou-internet-of-things>
- Jack Gold, Gaining an “edge” in IoT, Computerworld Online Magazine, <http://www.computerworld.com/article/3131422/internet-of-things/gaining-an-edge-in-iot.html>
- What is the Smart Grid, U.S. Department of Energy, <https://www.smartgrid.gov/the_smart_grid/smart_grid.html>?
- Smart Grid, Wikipedia, <https://en.wikipedia.org/wiki/Smart_grid>
- Smart Grid, U.S. Office of Electricity Delivery and Energy Reliability, <http://energy.gov/oe/services/technology-development/smart-grid>
- Joberto S.B. Martins, Smart Grid and (Tele)Communication Needs – Issues and Approaches, Salvador University – IEEE Smart Grid Workshop, <http://sites.ieee.org/bahia/files/2013/10/Palestra-Prof-Joberto-Martins.pdf>
- Murat Kuzlu, Manisa Pipattanasomporn, Saifur Rahman, Communication network requirements for major smart grid applications in HAN, NAN and WAN, Survey Paper – Science Direct, <http://www.sciencedirect.com/science/article/pii/S1389128614001431>
- Q. D. Ho, Y. Gao, G Rajalingham, T. Le Ngoc, Wireless Communications Networks for the Smart Grid, Springer Books, ISBN: 978-3-319-10346-4
- Margaret Rouse, Machine to Machine (m2m), Internet of Things Agenda site, <http://internetofthingsagenda.techtarget.com/definition/machine-to-machine-M2M>
- Tim Crosby, How Machine to Machine Communication Works, How Stuff Works site, <http://computer.howstuffworks.com/m2m-communication.htm>

- Smart meter, Wikipedia, <https://en.wikipedia.org/wiki/Smart_meter>
- Smart meters, British Gas on line leaflet, <https://www.britishgas.co.uk/smarter-living/control-energy/smart-meters.html>
- Smart meters, Smart Grid Consumer Collaborative non-profit organization, <http://www.whatissmartgrid.org/smart-grid-101/smart-meters>
- Arduino, Wikipedia, <https://en.wikipedia.org/wiki/Arduino>
- Arduino Mega 2560, arduino.org, <http://www.arduino.org/products/boards/arduino-mega-2560>
- Arduino Ethernet, arduino.org, <http://www.arduino.org/products/boards/arduino-ethernet>
- Brian W. Evans, Arduino programming notebook, www.arduino.cc, <http://playground.arduino.cc/uploads/Main/arduino_notebook_v1-1.pdf>
- Microsoft SQL Server, Wikipedia, <https://en.wikipedia.org/wiki/Microsoft_SQL_Server>
- SQL Tutorial, w3schools.com, <http://www.w3schools.com/sql/>
- Ιωάννης Μανωλόπουλος, Απόστολος Ν. Παπαδόπουλος, Συστήματα Βάσεων Δεδομένων – Θεωρία και Πρακτική Εφαρμογή, www.studentguru.gr, <studentguru.gr/m/books/134627/download>

8. Παράρτημα

8.1. Τιμολόγιο ΔΕΗ

Παρατίθεται το τιμολόγιο της ΔΕΗ που χρησιμοποιήθηκε στην εργασία (Εικόνα 83 και Εικόνα 84). Χάριν ευκολίας λήφθηκαν υπόψιν οι χρεώσεις με μονάδα €/kWh

Χρέωση Προμήθειας (με ισχύ από 25.7.2014)

Περιλαμβάνει το κόστος και τις λοιπές δαπάνες της ΔΕΗ για την παραγωγή και την προμήθεια της ηλεκτρικής ενέργειας στους πελάτες.

Κλιμάκια στο σύνολο της κατανάλωσης (kWh)	Χρεώσεις Κατανάλωσης "Ημέρας"			Χρεώσεις Κατανάλωσης "Νύχτας"	
	Ενέργεια (€/kWh)	Πάγιο (€/τετράμηνο)		Ενέργεια (€/kWh)	Πάγιο (€/τετράμηνο)
		1Φ παροχή	3Φ παροχή		
0-2000 >2000	0,09460 0,10252	1,52	4,80	0,06610	2,00

Ελάχιστη Χρέωση: Μονοφασική (1Φ) παροχή: 5,30€/τετράμηνο & Τριφασική (3Φ) παροχή: 8,58€/τετράμηνο.

Εικόνα 83: 1ος πίνακας με χρεώσεις προμήθειας

Οι ρυθμιζόμενες χρεώσεις που παρατίθενται στην Εικόνα 84 είναι χρεώσεις που εφαρμόζονται σε οποιοδήποτε πελάτη κάνει χρέωση του Εθνικού Ηλεκτρικού Συστήματος, ανεξαρτήτως προμηθευτή ενέργειας.

Κλιμάκια στο σύνολο της κατανάλωσης "Ημέρας" (kWh)	Σύστημα Μεταφοράς		Δίκτυο Διανομής		Λοιπές Χρεώσεις (€/kWh)	ΕΤΜΕΑΡ (€/kWh)	ΥΚΩ (€/kWh)
	Ισχύς (ΜΠΧ) €/kVA*ΣΙ/έτος	Ενέργεια (ΜΜΧ) €/kWh	Ισχύς (ΜΠΧ) €/kVA*ΣΙ/έτος	Ενέργεια (ΜΜΧ) €/kWh			
0-1600	0,13	0,00527	0,54	0,0213	0,00007	0,02477	0,00699
1601-2000							0,01570
2001-3000							0,03987
>3000							0,04488
Κατανάλωση "Νύχτας" (kWh)	0,00	0,0000	0,00	0,0000	0,00007	0,02477	0,00889

Εικόνα 84: 2ος πίνακας με ρυθμιζόμενες χρεώσεις

Στη συνέχεια παρουσιάζεται το Τμηματικό ωράριο που ισχύει το Ημερήσιο και το Νυχτερινό τιμολόγιο.

Τμηματικό ωράριο

Χειμερινή περίοδος: 1 Νοεμβρίου έως 30 Απριλίου

• 02:00-08:00 και 15:00-17:00, για τους πελάτες που είναι συνδεδεμένοι στο Δίκτυο της Ηπειρωτικής Χώρας και των διασυνδεδεμένων με αυτήν νησιών

Θερινή περίοδος: 1 Μαΐου έως 31 Οκτωβρίου

23:00-07:00

Εικόνα 85:Το τμηματικό ωράριο χρέωσης

8.2. Κώδικας SQL Server

```
USE [master]
GO
/****** Object:  Database [Thesis]    Script Date: 27/11/2016 7:28:22 μμ ******/
CREATE DATABASE [Thesis]
 CONTAINMENT = NONE
 ON  PRIMARY
( NAME = N'Thesis', FILENAME = N'C:\My\Projects\Thesis\Data\Thesis.mdf' , SIZE = 12288KB , MAXSIZE = UNLIMITED, FILEGROWTH = 1024KB )
 LOG ON
( NAME = N'Thesis_log', FILENAME = N'C:\My\Projects\Thesis\Data\Thesis_log.ldf' , SIZE = 1536KB , MAXSIZE = 2048GB , FILEGROWTH = 10%)
GO
ALTER DATABASE [Thesis] SET COMPATIBILITY_LEVEL = 120
GO
IF (1 = FULLTEXTSERVICEPROPERTY('IsFullTextInstalled'))
begin
EXEC [Thesis].[dbo].[sp_fulltext_database] @action = 'enable'
end
GO
ALTER DATABASE [Thesis] SET ANSI_NULL_DEFAULT OFF
GO
ALTER DATABASE [Thesis] SET ANSI_NULLS OFF
GO
```

```
ALTER DATABASE [Thesis] SET ANSI_PADDING OFF
GO
ALTER DATABASE [Thesis] SET ANSI_WARNINGS OFF
GO
ALTER DATABASE [Thesis] SET ARITHABORT OFF
GO
ALTER DATABASE [Thesis] SET AUTO_CLOSE OFF
GO
ALTER DATABASE [Thesis] SET AUTO_SHRINK OFF
GO
ALTER DATABASE [Thesis] SET AUTO_UPDATE_STATISTICS ON
GO
ALTER DATABASE [Thesis] SET CURSOR_CLOSE_ON_COMMIT OFF
GO
ALTER DATABASE [Thesis] SET CURSOR_DEFAULT  GLOBAL
GO
ALTER DATABASE [Thesis] SET CONCAT_NULL_YIELDS_NULL OFF
GO
ALTER DATABASE [Thesis] SET NUMERIC_ROUNDABORT OFF
GO
ALTER DATABASE [Thesis] SET QUOTED_IDENTIFIER OFF
GO
ALTER DATABASE [Thesis] SET RECURSIVE_TRIGGERS OFF
GO
ALTER DATABASE [Thesis] SET  DISABLE_BROKER
GO
```

```
ALTER DATABASE [Thesis] SET AUTO_UPDATE_STATISTICS_ASYNC OFF

GO

ALTER DATABASE [Thesis] SET DATE_CORRELATION_OPTIMIZATION OFF

GO

ALTER DATABASE [Thesis] SET TRUSTWORTHY OFF

GO

ALTER DATABASE [Thesis] SET ALLOW_SNAPSHOT_ISOLATION OFF

GO

ALTER DATABASE [Thesis] SET PARAMETERIZATION SIMPLE

GO

ALTER DATABASE [Thesis] SET READ_COMMITTED_SNAPSHOT OFF

GO

ALTER DATABASE [Thesis] SET HONOR_BROKER_PRIORITY OFF

GO

ALTER DATABASE [Thesis] SET RECOVERY FULL

GO

ALTER DATABASE [Thesis] SET  MULTI_USER

GO

ALTER DATABASE [Thesis] SET PAGE_VERIFY CHECKSUM

GO

ALTER DATABASE [Thesis] SET DB_CHAINING OFF

GO

ALTER DATABASE [Thesis] SET FILESTREAM( NON_TRANSACTED_ACCESS = OFF )

GO
```

```
ALTER DATABASE [Thesis] SET TARGET_RECOVERY_TIME = 0 SECONDS

GO

ALTER DATABASE [Thesis] SET DELAYED_DURABILITY = DISABLED

GO

EXEC sys.sp_db_vardecimal_storage_format N'Thesis', N'ON'

GO

USE [Thesis]

GO

/****** Object:  UserDefinedFunction [dbo].[getCost]    Script Date: 27/11/2016 7:28:23 μμ ******/

SET ANSI_NULLS ON

GO

SET QUOTED_IDENTIFIER ON

GO

-- =============================================

-- Author:              <IOANNIS TSAKALAKIS>

-- Create date: <27/11/2016>

-- Description:         <THESIS DATABASE>

-- =============================================

CREATE FUNCTION [dbo].[getCost]

(

        -- Add the parameters for the function here

         @Opened dateTime, @Closed datetime, @Power decimal(18, 2)

)

RETURNS

@Result TABLE (
```

```
TimeNight bigint, TimeDay bigint, CostNight decimal(18, 10), CostDay decimal(18, 10)

)

AS

BEGIN

	-- Fill the table variable with the rows for your result set

	Declare @TimeNight bigint = 0

	Declare @TimeDay bigint = 0

	Declare @CostNight decimal(18, 10) = 0

	Declare @CostDay decimal(18, 10) = 0


	declare @month int

	Select @month = Month(@Opened)

	declare @hour int

	declare @period bigint

	Select @period = datediff(SECOND, @Opened, @Closed)

	if (@month >= 5 and @month <= 10)

	Begin

		if (@hour >= 7 and @hour < 23) -- Hmerisio

		begin

			Select @TimeDay = @period, @CostDay = @period * @Power * 0.15373 / 3600 -- 1 --> Cost

		end

		else

		Begin

			Select @TimeNight = @period, @CostNight = @period * @Power * 0.12523 / 3600 -- 1 --> Cost
```

```
		end
	End
	Else
	Begin
		if ( (@hour >= 2 and @hour < 8) OR (@hour >= 15 and @hour
< 17) ) -- Nixterino
		begin
			Select @TimeNight = @period, @CostNight = @period *
@Power * 0.12523 / 3600 -- 1 --> Cost
		end
		else
		Begin
			Select @TimeDay = @period, @CostDay = @period *
@Power * 0.15373 / 3600 -- 1 --> Cost
		end

	End

	insert into @Result
	(TimeNight, TimeDay, CostNight, CostDay)
	Values (@TimeNight, @TimeDay, @CostNight, @CostDay)
	RETURN
END

GO

/****** Object:  Table [dbo].[LogInfo]    Script Date: 27/11/2016 7:28:23 μμ
******/
```

```
SET ANSI_NULLS ON

GO

SET QUOTED_IDENTIFIER ON

GO

SET ANSI_PADDING ON

GO

CREATE TABLE [dbo].[LogInfo](

	[ID] [bigint] IDENTITY(1,1) NOT NULL,

	[Info] [varchar](2048) NOT NULL,

	[LogDate] [datetime] NOT NULL CONSTRAINT [DF_LogInfo_LogDate] DEFAULT (getdate()),

	[Processed] [bit] NOT NULL CONSTRAINT [DF_LogInfo_Processed] DEFAULT ((0)),

	[Temperature] [decimal](18, 2) NULL,

	[RelayIsOn] [bit] NOT NULL CONSTRAINT [DF_LogInfo_openClose] DEFAULT ((0)),

	[RelaySwitch] [bit] NOT NULL CONSTRAINT [DF_LogInfo_switch] DEFAULT ((0)),

	[ButtonPressed] [bit] NOT NULL CONSTRAINT [DF_LogInfo_buttonOn]  DEFAULT ((0)),

	[EventSrc] [int] NOT NULL CONSTRAINT [DF_LogInfo_ButtonOff] DEFAULT ((0)),

	[SDRead] [bit] NOT NULL CONSTRAINT [DF_LogInfo_SDRead] DEFAULT ((0)),

 CONSTRAINT [PK_LogInfo] PRIMARY KEY CLUSTERED

(

	[ID] ASC
```

```
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]


GO

SET ANSI_PADDING OFF

GO

/****** Object:  Table [dbo].[loginfoBackup]    Script Date: 27/11/2016 7:28:23 μμ ******/

SET ANSI_NULLS ON

GO

SET QUOTED_IDENTIFIER ON

GO

SET ANSI_PADDING ON

GO

CREATE TABLE [dbo].[loginfoBackup](

	[ID] [bigint] IDENTITY(1,1) NOT NULL,

	[Info] [varchar](2048) NOT NULL,

	[LogDate] [datetime] NOT NULL,

	[Processed] [bit] NOT NULL,

	[Temperature] [decimal](18, 2) NULL

) ON [PRIMARY]


GO

SET ANSI_PADDING OFF

GO
```

```
/****** Object:  Table [dbo].[OpenClose]    Script Date: 27/11/2016 7:28:23 µµ ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[OpenClose](
	[id] [bigint] IDENTITY(1,1) NOT NULL,
	[Opened] [datetime2](7) NULL,
	[Closed] [datetime2](7) NULL,
	[TimeNight] [bigint] NULL,
	[TimeDay] [bigint] NULL,
	[CostNight] [decimal](18, 10) NULL,
	[CostDay] [decimal](18, 10) NULL,
	[Processed] [bit] NULL
) ON [PRIMARY]

GO
/****** Object:  Table [dbo].[Switch]    Script Date: 27/11/2016 7:28:23 µµ ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Switch](
	[Pressed] [datetime2](7) NOT NULL CONSTRAINT [DF_Switch_Pressed]  DEFAULT (getdate()),
```

[OnOff] [bit] NOT NULL CONSTRAINT [DF_Switch_OnnOff] DEFAULT ((1)),

[Switched] [bit] NOT NULL CONSTRAINT [DF_Switch_Switeched] DEFAULT ((0)),

CONSTRAINT [PK_Switch] PRIMARY KEY CLUSTERED

(

[Pressed] ASC

)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

/****** Object: StoredProcedure [dbo].[AddLogInfo] Script Date: 27/11/2016 7:28:23 μμ ******/

SET ANSI_NULLS ON

GO

SET QUOTED_IDENTIFIER ON

GO

-- ===

-- Author: Ioannis Tsakalakis

-- Create date: 2016.10.01

-- Description: Add log info from Arduino

-- ===

CREATE PROCEDURE [dbo].[AddLogInfo]

@LogInfo varchar(2048),

@dtLog datetime2,

@RelayIsOn bit,

```
@RelaySwitch bit,
@ButtonPressed bit,
@EventSrc int,
@SDRead bit,
@Temperature decimal(18,2)
AS
BEGIN
	-- SET NOCOUNT ON added to prevent extra result sets from
	-- interfering with SELECT statements.
	SET NOCOUNT ON;

	declare @lastOpen bigint

	begin try
		begin transaction
		  --print 'insert 1'
			insert into LogInfo
				([Info], [LogDate], [Temperature], RelayIsOn, RelaySwitch, ButtonPressed, EventSrc, SDRead)
			values
				(@LogInfo, @dtLog, @Temperature, @RelayIsOn, @RelaySwitch, @ButtonPressed, @EventSrc, @SDRead)

				--print 'check'
			if @RelaySwitch = 1  -- anapse
			begin
			  if @RelayIsOn = 1
```

```
                begin
                        --print 'insert 2'
                        insert into [dbo].[OpenClose]
                                ([Opened])
                        values
                                (@dtlog)
                end
        else
                begin
                        --print 'insert 3 - update'
                        --print 'insert 3 - get top 1'
                        select top 1 @lastOpen = ID from [dbo].[OpenClose]
                        order by id desc

                        --print 'insert 3 = check'
                        if (@lastOpen is not null) and exists (select top 1 1 from [dbo].[OpenClose] where @lastOpen = ID and [Closed] is null)
                        Begin
                                --print 'insert 3 - update'
                          update [dbo].[OpenClose]
                          Set
                                [Closed] = @dtlog
                          where ID = @lastOpen
                        End
                end
        end
```

```
            commit transaction
        end try
        begin catch
        -- error handling
            print 'error handling'
            rollback transaction

        end catch

END

GO
/****** Object:  StoredProcedure [dbo].[Process]    Script Date: 27/11/2016 7:28:23 μμ ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =============================================
-- Author:              <IOANNIS TSAKALAKIS>
-- Create date: <27/11/2016>
-- Description:         <THESIS DATABASE>
-- =============================================
CREATE PROCEDURE [dbo].[Process]
        @Power decimal(18 , 2)
AS
```

```
BEGIN

	SET NOCOUNT ON;



	update oc
	set
	oc.TimeNight = gc.TimeNight,
	oc.TimeDay = gc.TimeDay ,
	oc.CostNight = gc.CostNight,
	oc.CostDay =  gc.CostDay,
	 Processed = 1
	 From [Thesis].[dbo].[OpenClose] oc
	outer apply dbo.getCost(oc.Opened, oc.Closed, @Power ) gc
	where processed = 0 or processed is NULL

END

GO
USE [master]
GO
ALTER DATABASE [Thesis] SET  READ_WRITE
GO
```

8.3. Κώδικας Visual C#

Το πρόγραμμα SerialTest2.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.IO.Ports;
using System.Data.SqlClient;
using System.Diagnostics;

namespace SerialTest2 {
    public partial class Form1 : Form
    {
        StringBuilder sb;
        Queue<string> Info;
        SqlConnection cnn = new SqlConnection("Integrated Security=SSPI;Persist Security Info=False;User ID=\"\";Initial Catalog=Thesis;Data Source=DESKTOP-07RBA1T;Initial File Name=\"\";");
        SqlCommand cmd;
```

```
        int counter = 0;
        public Form1()
        {
            InitializeComponent();
            getAvailablePorts();
            Info = new Queue<string>();

            cnn.Open();
            cmd = cnn.CreateCommand();

//Παραμετροποίηση και αποθήκευση στη βάση δεδομένων
            cmd.CommandType = CommandType.StoredProcedure;
            cmd.CommandText = "AddLogInfo";

//Ακολουθεί η δήλωση των παραμέτρων που υπάρχουν σε μορφή πίνακα στην βάση δεδομένων
            cmd.Parameters.Add("@LogInfo", SqlDbType.VarChar, 2048);
            cmd.Parameters.Add("@dtLog", SqlDbType.DateTime);
            cmd.Parameters.Add("@RelayIsOn", SqlDbType.Bit);
            cmd.Parameters.Add("@RelaySwitch", SqlDbType.Bit);

cmd.Parameters.Add("@ButtonPressed", SqlDbType.Bit);
cmd.Parameters.Add("@EventSrc", SqlDbType.Int);
            cmd.Parameters.Add("@SDRead", SqlDbType.Bit);
            cmd.Parameters.Add("@Temperature", SqlDbType.Float);
        }
```

```
//Διαθέσιμες θύρες
void getAvailablePorts()
        {
            String[] ports = SerialPort.GetPortNames();
            comboBox1.Items.AddRange(ports);
        }
//Επιλογή θύρας από το αναδυόμενο μενού comboBox1
        private void button1_Click(object sender, EventArgs e)
        {
            try{
                if (comboBox1.Text == ""){
                    MessageBox.Show("Please Select COM Port");
                }
                else
                {
                    serialPort1.PortName = comboBox1.Text;
                    serialPort1.Open();
                    richTextBox1.Enabled = true;
                    buttonOpen.Enabled = false;
                    buttonClose.Enabled = true;
                }
            }
            catch (UnauthorizedAccessException) //Σε περίπτωση σφάλματος
            {
                MessageBox.Show("Unauthorized Access!!");
            }
```

```
        }
//Κουμπί για εκκίνηση σειριακής θύρας
        private void button2_Click(object sender, EventArgs e)
        {
            serialPort1.Close();
            buttonOpen.Enabled = true;
            buttonClose.Enabled = false;
            richTextBox1.Enabled = false;
        }
//Κουμπί για κλείσιμο σειριακής επικοινωνίας
        private void button1_Click_1(object sender, EventArgs e)
        {
            this.Close();
        }
//Έλεγχος της σειριακής θύρας
        private void serialPort1_DataReceived(object sender,
SerialDataReceivedEventArgs e)
        {
            this.Invoke(new EventHandler(textshow));
        }
//Μορφή εμφάνισης της σειριακής θύρας
        private void textshow(object sender, EventArgs e)
        {
//Ανάγνωση της σειριακής θύρας
            string str = serialPort1.ReadLine();
            string formattedStr = "";
```

```
            if (str.Trim().Length > 0 )
            {
                writeLog(str, out formattedStr);
            } else
            {
                counter += 1;
            }
           if (Info.Count > 5)
            {
                Info.Dequeue();
            }
            Info.Enqueue(formattedStr);
            richTextBox1.Text = QueueToStr(); //εμφάνιση της σειριακής //θύρας
        }
//Προσάρτηση κειμένου από τη σειριακή θύρα στην ουρά
        private string QueueToStr()
        {
            StringBuilder sb = new StringBuilder();
            foreach (var str in Info)
            {
                sb.Append(str);
            }
            return sb.ToString();
        }
//"Σπλιτάρισμα" του κειμένου της σειριακής θύρας ανά γραμμή
        private void writeLog(String log, out String formatted)
```

```
        {
            //Αναζητείται το σύμβολο (‘,’) για να γίνει το κόψιμο
            string[] parts = log.Split(',');
            if ((parts.Length >= 8) && (parts[5].Trim() == "1"))
            {
//Αποθηκεύω σε κάθε στήλη παραμέτρων της SQL τα αντίστοιχα δεδομένα
                cmd.Parameters[0].Value = log;  //0: @LogInfo
                cmd.Parameters[1].Value = DateTime.Now; //1: @dtLog
                cmd.Parameters[2].Value = (parts[4] == "1"); //2: //@RelayIsOn
                cmd.Parameters[3].Value = (parts[5] == "1"); //3: //@RelaySwitch
                cmd.Parameters[4].Value = (parts[3] == "1"); //4: //@ButtonPressed
                cmd.Parameters[5].Value = parts[6];          //5: //@EventSrc
                cmd.Parameters[6].Value = (parts[2] == "1"); //6: @SDRead
//Δήλωση θερμοκρασίας
                float temperature;
                float.TryParse(parts[7],out temperature);
                cmd.Parameters[7].Value = temperature;
                cmd.ExecuteNonQuery(); //Εκτελεί και επιστρέφει τις στήλες από τη
                                       // βάση δεδομένων
// Η ενημέρωση των στηλών της βάσης δεδομένων
                sb.AppendFormat("(dateTime: {0} - {1} ), ", parts[0], parts[1]);
                sb.AppendFormat("(SDpresent: {0} ), ", parts[2]);
                sb.AppendFormat("(ButtonPressed: {0} ), ", parts[3]);
                sb.AppendFormat("(RelayIsOn: {0} ), ", parts[4]);
                sb.AppendFormat("(RelaySwitch: {0} ), ", parts[5]);
                sb.AppendFormat("(SwitchSrc: {0} ), ", parts[6]);
```

```
            sb.AppendFormat("(temperature: {0} )", parts[7]);
            sb.AppendLine(); //τερματίζεται η γραμμή

            formatted = sb.ToString();
            DBLogInfo.AppendText(formatted);
            sb.Clear();
        }
        else
        {
            formatted = log;
        }
//Στο παρακάτω path αποθηκεύεται εφεδρικό αρχείο .txt με όλα τα δεδομένα
//της σειριακής θύρας

System.IO.File.AppendAllText(@"C:\Users\werty\Desktop\projectDB\WriteLines.txt", log + Environment.NewLine);
    }
// Όταν πατηθεί το κουμπί Select Day εκτελείται το πρόγραμμα επιλογής
//ημέρας που είναι αποθηκευμένο στο παρακάτω μονοπάτι
    private void button1_Click_2(object sender, EventArgs e)
    {
        Process.Start(@"C:\Users\werty\Documents\Visual Studio 2015\Projects\giannis\giveDay\selectDay\selectDay\bin\Debug\selectDay.exe");
    }
    private void Form1_Load(object sender, EventArgs e)
    {
        sb = new StringBuilder();
```

```
        }
    }
}
```

Το πρόγραμμα selectDay

```
namespace selectDay
{
    public partial class Form1 : Form
    {
//Αναπαριστά μία ανοιχτή σύνδεση σε μία βάση δεδομένων στον SQL
        SqlConnection cnn = new SqlConnection("Integrated Security=SSPI;Persist Security Info=False;User ID=\"\";Initial Catalog=Thesis;Data Source=DESKTOP-07RBA1T;Initial File Name=\"\";");
        SqlCommand cmd;
        public Form1()
        {
            InitializeComponent();
        }

//
        private void btnGetData_Click(object sender, EventArgs e)
        {
            GetData(!cbGrouped.Checked);
        }
//        Εξαγωγή δεδομένων από την βάση δεδομένων
private void GetData(bool detail = false)
```

```
    {
        try
        {
            cnn.Open();
            cmd = cnn.CreateCommand();
            cmd.CommandType = CommandType.Text;
            cmd.CommandText = "select " +
                              (detail ? " * " : " Count(*) as Counter,
Convert(varchar, dateAdd(second, Sum(TimeDay), 0), 114) as TimeDay,
Convert(varchar, dateAdd(second, Sum(TimeNight), 0), 114) as TimeNight,
Sum(CostNight) as CostNight, Sum(CostDay) as CostDay ") + " from
[dbo].OpenClose where datePart(weekday, opened) = " +
(cbDay.SelectedIndex + 1).ToString();
            SqlDataAdapter da = new SqlDataAdapter(cmd);
            DataSet ds = new DataSet("myData");
            da.Fill(ds);
            gvData.DataSource = ds.Tables[0];
        }
// Σε περίπτωση σφάλματος, εμφάνιση αντίστοιχου μηνύματος
        catch (Exception ex)
        {
            MessageBox.Show("Error: " + ex.Message, "Error retreiving data",
MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
        finally
        {
            if (cnn != null)
            {
```

```
                if (cnn.State == ConnectionState.Open)
                {
                    cnn.Close();
                }
            }
        }
    }
//      Κουμπί για κλείσιμο προγράμματος
    private void button1_Click(object sender, EventArgs e)
    {
        this.Close();
    }
// Προγραμματισμός για ζήτηση εύρους ημερών
    private void btnGiveData_Click(object sender, EventArgs e)
    {
        getData4Range(!cbGrouped.Checked);
    }
    private void getData4Range(bool detail = false)
    {
        try
        {
            cnn.Open();
// Εξαγωγή αποτελεσμάτων από τη βάση δεδομένων
            cmd = cnn.CreateCommand();
            cmd.CommandType = CommandType.Text;
```

```
            cmd.CommandText = "select " + (detail ? " * " : " Count(*) as Counter, Convert(varchar, dateAdd(second, Sum(TimeDay), 0), 114) as TimeDay, Convert(varchar, dateAdd(second, Sum(TimeNight), 0), 114) as TimeNight, Sum(CostNight) as CostNight, Sum(CostDay) as CostDay ") + " from [dbo].OpenClose where opened >= @dateFrom and opened < @dateTo";


//Ημέρα ΑΠΟ

            cmd.Parameters.Add("@dateFrom", SqlDbType.Date).Value = dtpFrom.Value;


//Ημέρα Έως

            cmd.Parameters.Add("@dateTo", SqlDbType.Date).Value = dtpTo.Value;


            SqlDataAdapter da = new SqlDataAdapter(cmd);

            DataSet ds = new DataSet("myData");

            da.Fill(ds);

            gvData.DataSource = ds.Tables[0];


        }

        catch

        {

        }

        finally

        {

            if (cnn != null)

            {

                if (cnn.State == ConnectionState.Open)
```

```
                {
                    cnn.Close();
                }
            }
        }
    }
}
//Υπολογισμός κόστους
private void Cost()
    {
        try
        {
//Εξαγωγή δεδομένων από τη βάση δεδομένων
            cnn.Open();
            cmd = cnn.CreateCommand();
            cmd.CommandType = CommandType.StoredProcedure;
            cmd.CommandText = "Process";
            cmd.Parameters.Add("@Power", SqlDbType.Decimal).Value =
Convert.ToDecimal(txtPower.Text);
            cmd.ExecuteNonQuery();
            MessageBox.Show("Ο υπολογισμός ολοκληρώθηκε",
"Ύπολογισμός κόστους", MessageBoxButtons.OK,
MessageBoxIcon.Information);
        }
//Περίπτωση σφάλματος
        catch (Exception ex)
        {
```

```
            MessageBox.Show("Ο υπολογισμός απέτυχε: " + ex.Message,
"Υπολογισμός κόστους", MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
        finally
        {
            if (cnn != null)
            {
                if (cnn.State == ConnectionState.Open)
                {
                    cnn.Close();
                }
            }
        }
    }
//      Κουμπί καθαρισμού πλαισίου αποτελεσμάτων
    private void btnClearResults_Click(object sender, EventArgs e)
    {
        this.gvData.DataSource = null;
    }
  }
}
```

8.4. Κώδικας Arduino

Το κύριο πρόγραμμα του Arduino, το mac2.ino

```
#include <SPI.h>
#include <DS3231.h>
#include <Ethernet2.h>
#include "RealTimeClock.h"
#include "variables.h"
#include "sensors.h"
#include "webServer.h"
#include "card.h"
#include "result.h"

#define Critical_Temperature 70

//Who triggered relay
#define evntSrcNONE 0
#define evntSrcSD 1
#define evntSrcButton 2

int LastEventSrc = evntSrcNONE;

bool ResultOfCompare = false;
```

```
float currentTemperature;
bool RelayIsOpen = false;
//---------------------

int delayCounter;
int openRele = 0;
int count = 0;
int timer1;
int timer2;
String infoExecute;

DS3231 rtc(SDA, SCL);

Time time;
 paramsInfo *params;

byte mac[] = {
  0x90, 0xA2, 0xDA, 0x10, 0x4F, 0x49
};
IPAddress ip(169, 254, 11, 100);
EthernetServer server(8080);

result* res = new result();
```

```
void setup() {

// Initialize the rtc object
  rtc.begin();

  Serial.begin(9600);

  digitalRead(SD_CARD_STATUS_PIN);

  pinMode(Relay_Switch, INPUT);
  digitalWrite(Relay_Switch, RELAY_OFF);

  pinMode(Relay_Mode, OUTPUT);
  digitalWrite(Relay_Mode, RELAY_OFF);

  delay(SECOND * 4);

  Ethernet.begin(mac, ip);

  server.begin();
}
bool compare2SD(Time curTime, int sdHour, int sdMin, int sdSec, int duration)
{
    bool result;
    long icurTime;
```

```
    long iFrom;
    long iTo;
    result = false;
    icurTime = (( (long)curTime.hour * 60) +  (long)curTime.min) * 60 + (long)curTime.sec;
    iFrom = (((long)sdHour * 60) +  (long)sdMin) * 60 + (long)sdSec;
    iTo = iFrom + duration;

    if ( (icurTime >= iFrom) && (icurTime <= iTo) )
    {
      result = true;
    }
    return result;
}

void RelayOpen(int eventSrc)
{
  LastEventSrc = eventSrc;
  digitalWrite(Relay_Mode, RELAY_ON);
  delay (100);
}
void RelayClose()
{
  digitalWrite(Relay_Mode,RELAY_OFF);
  delay (100);
}
```

```
void loop(){

  res->dtLogDate = rtc.getDateStr();
  res->dtLogTime = rtc.getTimeStr();
  res->dtEpoch = rtc.getUnixTime(time);

  time = rtc.getTime();
  int dow = time.dow;

  timer1 = time.sec;

  currentTemperature = WaterTemperature();
  res->temperature = currentTemperature;

  int relayStatusInput = digitalRead(Relay_Switch);
  int relayStatusOutput = digitalRead(Relay_Mode);

  res->ButtonDown = (relayStatusInput == 1);

  debug("Check if Relay is open");
  RelayIsOpen = (relayStatusOutput != 0);
  debug("RelayIsOpen ");
  debug(RelayIsOpen ? "True " : "False" );

  if ( (RelayIsOpen) && (currentTemperature >= Critical_Temperature) )
```

```
{
  debug("Command: Close Relay (Critical Temperature)");
  RelayClose();
}
else
{
  params = new paramsInfo();

  res->addOptional("");

  readSD(res, params);

  debug("Read SD. IsPresent ");
  debug(res->SDpresent ? "True " : "False ");

  // Sunday=0, Saturday = 6
  Day curParams;
  ResultOfCompare = false;
  for (int curDay=0; curDay < 7; curDay++)
  {
     for (int i = 0; i < params->myDayInfo[curDay].counter; i++)
      {
         curParams = params->myDayInfo[curDay].myDay[i];
         if ( (!ResultOfCompare) && (curDay == dow)) {
          debug("checking for current day");
```

```
          ResultOfCompare = compare2SD(time, curParams.hour, curParams.minute, curParams.seconds, curParams.duration);
        }
      }
    }
  debug("ResultOfCompare ");
  debug(ResultOfCompare ? " True " : "False");

   res->SwitchSrc = evntSrcNONE;
   res->RelayIsOn = RelayIsOpen;

   res->RelaySwitch = false;
   if (!RelayIsOpen)
   {
      debug("Relay is open");

      bool toOpen = false;
      int evntSrc = evntSrcNONE;
      debug("Is SD Present and active schedule?");
      if (ResultOfCompare)
      {
        debug("YES SD IS  Present and active schedule");
        toOpen = true;
        evntSrc = evntSrcSD ;

      }
```

```
      else
      {
        debug("NO: SD IS NOT Present or not active schedule. CHECK Button");

        if (relayStatusInput != 0)
        {
          debug("Button Pressed");
          toOpen = true;
          evntSrc = evntSrcButton;
        }
      }
      if (toOpen)
      {
        debug("Command: Open Relay");
        RelayOpen(evntSrc);
        res->SwitchSrc = evntSrc;
        res->RelayIsOn = true;
        res->RelaySwitch = true;
      }
    }
    else // RelayIsOpen
    {
      debug("RELAY is ON. Should I turn it OFF?. First Check SD");
      bool toClose = false;
      if ( (LastEventSrc == evntSrcButton) && (ResultOfCompare) ){
```

```
        LastEventSrc = evntSrcSD;
      }
      else
      {
        if ( (LastEventSrc == evntSrcSD) && (!ResultOfCompare) )
        {
          debug("SD caused Realy to open and now OUT of active schedule. Turn it off");
          toClose = true;
        }
        else
        {
          debug("Button Pressed. Turn it OFF");
          if ( (relayStatusInput != 0)  && (LastEventSrc == evntSrcButton)  ) toClose = true;
        }
      }
      if (toClose) {
        debug("Command: Close Relay (schedule)");
        RelayClose();
        res->SwitchSrc = LastEventSrc;
        res->RelayIsOn = false;
        res->RelaySwitch = true;
      }
    }
    free(params);
  }
```

```
  Serial.println(res->ToString(0));

  showClientStatusForBrowser(relayStatusInput, relayStatusOutput, rtc, server);

  delay(LOOP_DELAY);
}
```

Ακολουθεί το RealTimeClock.h

```
#ifndef REALTIMECLOCK_H
#define REALTIMECLOCK_H
#include <DS3231.h>

void RealTimeClock(DS3231 rtc);

#endif
```

Ακολουθεί το RealTimeClock.cpp

```
#include "RealTimeClock.h"
void RealTimeClock(DS3231 rtc){
        Serial.print(rtc.getDateStr());
        Serial.print(",");
        Serial.print(rtc.getTimeStr());
        Serial.print(",");
```

```
}
```

Ακολουθεί το variables.h

```
#ifndef VARIABLES_H
#define VARIABLES_H

#define RELAY_ON 1
#define RELAY_OFF 0
#define Relay_Switch 2
#define Relay_Mode 8

#define SECOND 1000
#define LOOP_DELAY (SECOND / 5)
#define SWITCH_OFF_DELAY 10000

#define HTML_RELOAD 1

#endif
```

Ακολουθεί το sensors.h

```
#ifndef SENSORS_H
#define SENSORS_H
#include <SPI.h>
#include "variables.h"
```

```
void digitalTemperatureSensor();
float analogTemperatureSensor();

float WaterTemperature();

#endif
```

Ακολουθεί το sensers.cpp

```
#include "sensors.h"

float WaterTemperature()
{
  float cel = analogTemperatureSensor();
  return cel;
}

float _calculateCelsiusForAnalog(int analogValue)
{
  float milliVolt = (analogValue/1024.0)*5000;
  return milliVolt / 10;
}

float analogTemperatureSensor()
{
  int tempPin = 15;
```

```
  int val = analogRead(tempPin);
  float celsius = _calculateCelsiusForAnalog(val);
  return celsius;
}
```

Ακολουθεί το webserver.h

```
#ifndef WEB_SERVER_H
#define WEB_SERVER_H

#include <SPI.h>
#include <DS3231.h>

#include <Ethernet2.h>
#include "variables.h"
#include "sensors.h"

#define IMG_ON "https://estore-gewater.force.com/resource/1430891853000/GreenCheckButton";
#define IMG_OFF "https://i.ytimg.com/i/ayPTi6JJr7v9AIQAQGu3zw/mq1.jpg?v=51680b9f";

void showClientStatusForBrowser(int in, int out, DS3231 rtc, EthernetServer server);

#endif
```

Ακολουθεί το webserver.cpp

```
#include "webServer.h"

String _showRelayStatus(EthernetClient client, int relayStatus)
{
  String output = "";
  if (relayStatus == RELAY_ON) {
    output += "<div style='color:Green;'><div align='center'>ON<br><img src='";
    output += IMG_ON;
  } else {
    output += "<div style='color:Red;'><div align='center'>OFF<br><img src='";
    output += IMG_OFF;
  }

  output += "' style='width:120px;height:120px;'>";
  output += "</div></div>";

  return output;
}

void _response(EthernetClient client, String html)
{
  client.println("HTTP/1.1 200 OK");
  client.println("Content-Type: text/html");
  client.println("Connection: close");
```

```
  client.print("Refresh: ");
  client.println(HTML_RELOAD);
  client.println();
  client.println(html);
}

void showClientStatusForBrowser(int in, int out, DS3231 rtc, EthernetServer server)
{
  EthernetClient client = server.available();

  String html = "<!DOCTYPE HTML>"
"  <html>"
"    <head><title>Arduino Web Page</title></head>"
"    <body>"
"        <div align=\'center\'>"
"          <h1>Arduino is here</h1>"
"          <h1>";
  html += rtc.getDOWStr();
  html += " ";
  html += rtc.getDateStr();
  html += " ";
  html += rtc.getTimeStr();
  html += " Water Temperature is ";
  html += WaterTemperature();
  html += " ";
```

```
html += "</h1>";

html += "<h4><p>status for input: </p></h4>";
html += _showRelayStatus(client, in);
html += "<br />";
html += "<h4><p>status for output: </p></h4>";
html += _showRelayStatus(client, out);

html += "</div>";
html += "</body>";
html += "</html>";

if (client) {

  boolean currentLineIsBlank = true;

  while (client.connected()) {
    if (client.available()) {
      char htmlRequest = client.read();

      if (htmlRequest != (char) 0) {
        _response(client, html);
        break;
      }
    }
  }
```

```
    client.flush();
    client.stop();
  }
}
```

Ακολουθεί το card.h

```
#ifndef SD_H
#define SD_H
#include <SD.h>
#include "result.h"
#include <SPI.h>
#define SD_CARD_STATUS_PIN 50

#define DAY_MAX_PARAMS 5
#define Days_In_Week 7

struct Day{
        int hour;
        int minute;
        int seconds;
        int duration;
        String getInfo();
};
```

```
struct DayInfo
{
	int counter=0;
	Day myDay[DAY_MAX_PARAMS];
};

struct paramsInfo
{
	DayInfo myDayInfo[Days_In_Week];
};

 void readSD(result* res, paramsInfo *pi);

#endif
```

Ακολουθεί η card.cpp

```
//#include <SD.h>
#include <SPI.h>
#include <DS3231.h>
#include <Ethernet2.h>

#include "RealTimeClock.h"
#include "variables.h"
```

```
#include "sensors.h"
#include "webServer.h"
#include "interrupt.h"
#include "card.h"
//#include "MemoryFree.h"
#include "result.h"

#include "debug.h"



#define Critical_Temperature 70

//Who triggered relay
#define evntSrcNONE 0
#define evntSrcSD 1
#define evntSrcButton 2

int LastEventSrc = evntSrcNONE;

bool ResultOfCompare = false;
float currentTemperature;
bool RelayIsOpen = false;
//---------------------

int delayCounter;
int openRele = 0; // default false
```

```
int count = 0;
int timer1;
int timer2;
//int timeExecute = 0;
String infoExecute;


DS3231 rtc(SDA, SCL);

Time time;
 paramsInfo *params;

byte mac[] = {
  0x90, 0xA2, 0xDA, 0x10, 0x4F, 0x49
};
IPAddress ip(169, 254, 11, 100);
EthernetServer server(8080);

result* res = new result();


void setup() {
  // Initialize the rtc object
  rtc.begin();

  Serial.begin(9600);
```

```
  digitalRead(SD_CARD_STATUS_PIN);

  pinMode(Relay_Switch, INPUT);
  digitalWrite(Relay_Switch, RELAY_OFF);

  pinMode(Relay_Mode, OUTPUT);
  digitalWrite(Relay_Mode, RELAY_OFF);

  delay(SECOND * 4);

  Ethernet.begin(mac, ip);
  server.begin();
}

bool compare2SD(Time curTime, int sdHour, int sdMin, int sdSec, int duration)
{
    bool result;
    long icurTime;
    long iFrom;
    long iTo;
    result = false;
    icurTime = (( (long)curTime.hour * 60) +  (long)curTime.min) * 60 + (long)curTime.sec;
    iFrom = (((long)sdHour * 60) +  (long)sdMin) * 60 + (long)sdSec;
    iTo = iFrom + duration;
```

```
    if ( (icurTime >= iFrom) && (icurTime <= iTo) )
    {
      result = true;
    }
    return result;
}

void RelayOpen(int eventSrc)
{
  LastEventSrc = eventSrc;
  digitalWrite(Relay_Mode, RELAY_ON);
  delay (100);
}
void RelayClose()
{
  digitalWrite(Relay_Mode,RELAY_OFF);
  delay (100);
}



void loop(){

  res->dtLogDate = rtc.getDateStr();
  res->dtLogTime = rtc.getTimeStr();
  res->dtEpoch = rtc.getUnixTime(time);
```

```
time = rtc.getTime();
int dow = time.dow;

timer1 = time.sec;

currentTemperature = WaterTemperature();
res->temperature = currentTemperature;

int relayStatusInput = digitalRead(Relay_Switch);
int relayStatusOutput = digitalRead(Relay_Mode);

res->ButtonDown = (relayStatusInput == 1);

debug("Check if Relay is open");
RelayIsOpen = (relayStatusOutput != 0);
debug("RelayIsOpen ");
debug(RelayIsOpen ? "True " : "False" );

if ( (RelayIsOpen) && (currentTemperature >= Critical_Temperature) )
{
  debug("Command: Close Relay (Critical Temperature)");
  RelayClose();
}
else
{
    params = new paramsInfo();
```

```
    res->addOptional("");
  readSD(res, params);
  debug("Read SD. IsPresent ");
  debug(res->SDpresent ? "True " : "False ");

  // Sunday=0, Saturday = 6
  Day curParams;
  ResultOfCompare = false;
  for (int curDay=0; curDay < 7; curDay++)
  {
    for (int i = 0; i < params->myDayInfo[curDay].counter; i++)
     {
       curParams = params->myDayInfo[curDay].myDay[i];

       if ( (!ResultOfCompare) && (curDay == dow)) {
        debug("checking for current day");
         ResultOfCompare = compare2SD(time, curParams.hour,
curParams.minute, curParams.seconds, curParams.duration);
       }
     }
  }

 debug("ResultOfCompare ");
 debug(ResultOfCompare ? " True " : "False");
```

```
res->SwitchSrc = evntSrcNONE;
res->RelayIsOn = RelayIsOpen;

res->RelaySwitch = false;
if (!RelayIsOpen)
{
    debug("Relay is open");

    bool toOpen = false;
    int evntSrc = evntSrcNONE;
    debug("Is SD Present and active schedule?");
    if (ResultOfCompare)
    {
      debug("YES SD IS  Present and active schedule");
      toOpen = true;
      evntSrc = evntSrcSD ;

    }
    else
    {
      debug("NO: SD IS NOT Present or not active schedule. CHECK Button");

      if (relayStatusInput != 0)
      {
        debug("Button Pressed");
```

```
        toOpen = true;
        evntSrc = evntSrcButton;
      }
    }
    if (toOpen)
    {
      debug("Command: Open Relay");
      RelayOpen(evntSrc);
      res->SwitchSrc = evntSrc;
      res->RelayIsOn = true;
      res->RelaySwitch = true;
    }
  }
  else // RelayIsOpen
  {
    debug("RELAY is ON. Should I turn it OFF?. First Check SD");
    bool toClose = false;
    if ( (LastEventSrc == evntSrcButton) && (ResultOfCompare) ){
      LastEventSrc = evntSrcSD;
    }
    else
    {
      if ( (LastEventSrc == evntSrcSD) && (!ResultOfCompare) )
      {
        debug("SD caused Realy to open and now OUT of active schedule.
Turn it off");
```

```
          toClose = true;
        }
        else
        {
          debug("Button Pressed. Turn it OFF");
          if ( (relayStatusInput != 0)  && (LastEventSrc == evntSrcButton)  ) toClose = true;
        }
      }
      if (toClose) {
        debug("Command: Close Relay (schedule)");
        RelayClose();
        res->SwitchSrc = LastEventSrc;
        res->RelayIsOn = false;
        res->RelaySwitch = true;
      }
    }

    free(params);
  }
  Serial.println(res->ToString(0));

  showClientStatusForBrowser(relayStatusInput, relayStatusOutput, rtc, server);

  delay(LOOP_DELAY);
}
```

Ακολουθεί το result.h

```
#ifndef RESULT_H

#define RESULT_H

#include <DS3231.h>
#include "variables.h"

#define debugInfoOFF 0
#define debugInfoON 1

class result {

  public:
    String dtLogDate;
    String dtLogTime;
    long dtEpoch;
    bool SDpresent;
    bool ButtonDown;
    bool RelayIsOn;
    bool RelaySwitch;
    int SwitchSrc;
    float temperature;
    String extraInfo;
        result(){
```

```
    SDpresent = false;
    ButtonDown = false;
    RelayIsOn = false;
    RelaySwitch = false;
    SwitchSrc = 0;
    temperature = 0;
  }

  String ToString(int debugInfo = debugInfoOFF) {
    return
       dtLogDate + "," + dtLogTime + "," +
        (debugInfo ? "(SDpresent :" : "") + (SDpresent ? "1" : "0") + (debugInfo ? ")" : "") + "," +
        (debugInfo ? "(ButtonDown :" : "") + (ButtonDown ? "1" : "0") + (debugInfo ? ")" : "") + "," +
        (debugInfo ? "(RelayIsOn :" : "")  + (RelayIsOn ? "1" : "0") + (debugInfo ? ")" : "") + "," +
        (debugInfo ? "(RelaySwitch :" : "") + (RelaySwitch ? "1" : "0") + (debugInfo ? ")" : "") + "," +

        (debugInfo ? "(SwitchSrc :" : "") + SwitchSrc  + (debugInfo ? ")" : "") + "," +

        (debugInfo ? "(temperature :" : "") + temperature + (debugInfo ? ")" : "") +
        ", [" + extraInfo + "]";
  }
```

```
  void addOptional(String message)
  {
    extraInfo = message;
  }
};

#endif
```

Ακολουθεί το debug.h

```
#ifndef _DEBUG_H_
#define _DEBUG_H_

#ifndef DEBUG
//#define DEBUG
#endif

void debug(String message) {
#ifdef DEBUG
    Serial.println("DEBUG: " + message);
#endif
}

#endif
```