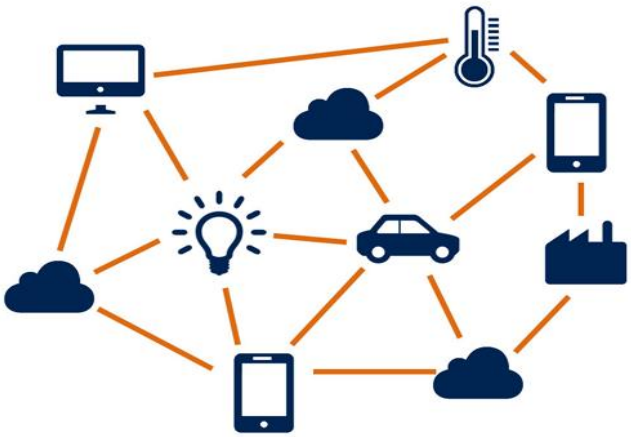
ΤΕΙ ΗΠΕΙΡΟΥ
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ Τ.Ε.

ΣΥΣΤΗΜΑ ΔΙΑΧΕΙΡΙΣΗΣ ΚΑΙ ΑΠΟΜΑΚΡΥΣΜΕΝΟΥ ΕΛΕΓΧΟΥ ΑΙΣΘΗΤΗΡΩΝ



Πτυχιακή Εργασία

Του **ΙΩΑΝΝΗ ΜΑΣΚΛΑΒΑΝΟΥ**
ΑΜ:11133

Υπό την εποπτεία του
Dr. **ΓΡΗΓΟΡΙΟΥ ΔΟΥΜΕΝΗ**

Άρτα
Ιούνιος 2016

ΣΥΣΤΗΜΑ ΔΙΑΧΕΙΡΙΣΗΣ ΚΑΙ ΑΠΟΜΑΚΡΙΣΜΕΝΟΥ ΕΛΕΓΧΟΥ ΑΙΣΘΗΤΗΡΩΝ

"*When wireless is perfectly applied the whole earth will be converted into a huge brain, which in fact it is, all things being particles of a real and rhythmic whole.....and the instruments through which we shall be able to do this will be amazingly simple compared with our present telephone. A man will be able to carry one in his vest pocket.*"

Nikola Tesla

Part of an interview to Colliers Magazine 1926

Περιεχόμενα

ΔΗΛΩΣΗ ΠΝΕΥΜΑΤΙΚΗΣ ΙΔΙΟΚΤΗΣΙΑΣ

Η παρούσα εργασία αποτελεί προϊόν αποκλειστικά δικής μου προσπάθειας. Όλες οι πηγές που χρησιμοποιήθηκαν περιλαμβάνονται στη βιβλιογραφία και γίνεται ρητή αναφορά σε αυτές μέσα στο κείμενο όπου έχουν χρησιμοποιηθεί.

…….

ΥΠΟΓΡΑΦΗ

ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα να ευχαριστήσω όλα τα μέλη της οικογένειας μου για την ακατάπαυστη υποστήριξη τους καθ' όλη τη διάρκεια των σπουδών μου και όχι μόνο. Επίσης θα ήθελα να ευχαριστήσω του φίλους μου τόσο στην Άρτα όσο και την Αθήνα και κυρίως τους κυρίους Πεταρούδα Θωμά και Παντελή, για υποστήριξη τους, την υπομονή και την βοήθεια στην προσπάθεια μου για την πραγματοποίηση αυτής της εργασίας, διότι μέσα από συζητήσεις μπόρεσα να λύσω αρκετούς προβληματισμούς μου τόσο στο σχεδιασμό όσο και στην υλοποίηση της εργασίας. Ακόμη θα ήθελα να ευχαριστήσω τα μέλη της ομάδας μου κύριους Λισγάρα Βασίλειο και Πανταζή Γρηγόριο για την πολύ καλή προσπάθεια που κάναμε όλοι μαζί σαν ομάδα αλλά και ο καθένας ξεχωριστά κατά τη διάρκεια εκπόνησης της πτυχιακής εργασίας αυτής για να φέρουμε εις πέρας το πολύ μεγάλο αυτό project. Τέλος θα ήθελα να ευχαριστήσω θερμά τον καθηγητή και εισηγητή της εργασίας μου dr. Γρηγόριο Δουμένη για την εισαγωγή στον χώρο των αισθητήρων και ενεργοποιητών μέσω των μαθημάτων του στη σχολή και για την πολύτιμη καθοδήγηση του καθ' όλη τη διάρκεια υλοποίησης της πτυχιακής εργασίας όχι μόνο σε θέματα αυτής αλλά και σε θέματα επαγγελματικού προσανατολισμού.

ΠΕΡΙΛΗΨΗ

Ο σκοπός της παρούσας διπλωματικής είναι η μελέτη και η ανάπτυξη μια εφαρμογής διακομιστή/ message broker η οποία θα μπορεί να δεχθεί μηνύματα από μικροελεγκτές που θα είναι συνδεδεμένοι στο τοπικό ασύρματο δίκτυο καθώς και η επικοινωνία με μια ιστοσελίδα μέσω της οποίας θα μπορεί να γίνει απομακρυσμένα η διαχείριση των μικρολεγκτών αυτών.

Αρχικά γίνεται μια ανάλυση της σχεδίασης του προβλήματος. Στη συνέχεια γίνεται ανάλυση της γλώσσας και των εργαλείων που χρησιμοποιήθηκαν για την υλοποίηση της εφαρμογής αυτής, καθώς ανάλυση των διαδικασιών που τρέχουν εσωτερικά αυτής. Τέλος γίνεται μια παρουσίαση της εγκατάστασης και επίδειξη λειτουργίας αυτής.

ABSTRACT

The aim of this bachelor thesis is study and develop a server/message broker windows form application that can accept messages from microcontroller sensors which will be connected to it via the Wireless Local Area Network. It could also connect to a webpage through the administrator will be able to control all the sensors remotely.

At first, analysis is done on the application plan. Next we make an introduction to the programming language and the tools we used during the development. Lastly, there is an presentation on the installation of the application and a demonstration on how it works.

Κεφάλαιο I: Internet of Things

Στο πρώτο κεφάλαιο, γίνεται εισαγωγή στον όρο Internet of Things, σε εφαρμογές αυτού και τεχνολογίες που αναπτύχθηκαν βασισμένες σε αυτό.

Κεφάλαιο II: Σχεδίαση

Στο δεύτερο κεφάλαιο, γίνεται ανάλυση της σχεδίασης της εφαρμογής τόσο σε εσωτερικό όσο και εξωτερικό επίπεδο.

Κεφάλαιο III: Υλοποίηση

Στο τρίτο κεφάλαιο, γίνεται εισαγωγή στη γλώσσα προγραμματισμού και το προγραμματιστικό περιβάλλον και ανάλυση όλων των συναρτήσεων που τρέχουν εσωτερικά της συνάρτησης.

Κεφάλαιο IV: Εγκατάσταση

Στο τέταρτο κεφάλαιο, δίνονται οδηγίες για την εγκατάσταση της εφαρμογής και επίδειξη της λειτουργίας της.

ΚΕΦΑΛΑΙΟ Ι

Internet of Things

1. Εισαγωγή

Internet of Things ή Internet of Everything ή Network of Everything είναι ο όρος που περιγράφει ένα παγκόσμιο δίκτυο φυσικών αντικειμένων τα οποία περιέχουν ενσωματωμένα συστήματα (ηλεκτρονικά, αισθητήρες ή λογισμικό) το οποίο επιτρέπει την αποστολή δεδομένων στον διαχειριστή ή στις συνδεδεμένες συσκευές. Με αυτό τον τρόπο επιτρέπεται ο απομακρυσμένος έλεγχος των αισθητήρων ή ενεργοποιήτων από το χρήστη ή διαχειριστή του συστήματος. Έτσι αυξάνεται συνολικά η ακρίβεια και η απόδοση του συστήματος τόσο ενεργειακά όσο και οικονομικά.

Εικόνα 1.1(Internet of Things)

2. Ιστορία

Το Internet of Things σαν ιδέα υπάρχει από της αρχές του 20^{ου} αιώνα όταν ο Nikola Tesla ανέφερε σε συνέντευξη του στο περιοδικό Colliers ότι " Όταν θα καλυφθεί ο πλανήτης από ασύρματα δίκτυα, τότε θα λειτουργεί ως ενας τεράστιος εγκέφαλος όπως είναι και στην πραγματικότητα. Τα εργαλεία που θα χρησιμοποιούμε για να το κάνουμε αυτό θα είναι πολύ πιο απλά σε σχέση με το σημερινό τηλέφωνο". Αργότερα άλλοι μεγάλοι επιστήμονες της εποχής εκείνης ανέφεραν την ανάγκη προσθήκης νοημοσύνης στις μηχανές, ο Alan Turing το 1950 στο άρθρο του "Computing Machinery and Intelligence" στο Oxford Mind Journal αναφέρει ότι " Είναι καλύτερο να παρέχουμε στη μηχανή τα καλύτερα αισθητήρια όργανα που υπάρχουν και μετά να της μάθουμε να καταλαβαίνει και να μιλάει Αγγλικά. Δηλαδή την ίδια εκπαιδευτική διαδικασία που ακολουθεί και ένα παιδί". Η μεγάλη επανάσταση ήρθε το 1969 όταν δημιουργήθηκε το Arpanet. Το Arpanet ή Advanced Research Projects Agency Network ήταν η πρώτη μορφή του σημερινού Internet. Το Arpanet ήταν ένα δίκτυο που βασιζόταν στην ανταλλαγή πακέτων. Ήταν το πρώτο δίκτυο που υποστήριζε το πρωτόκολλο TCP/IP.

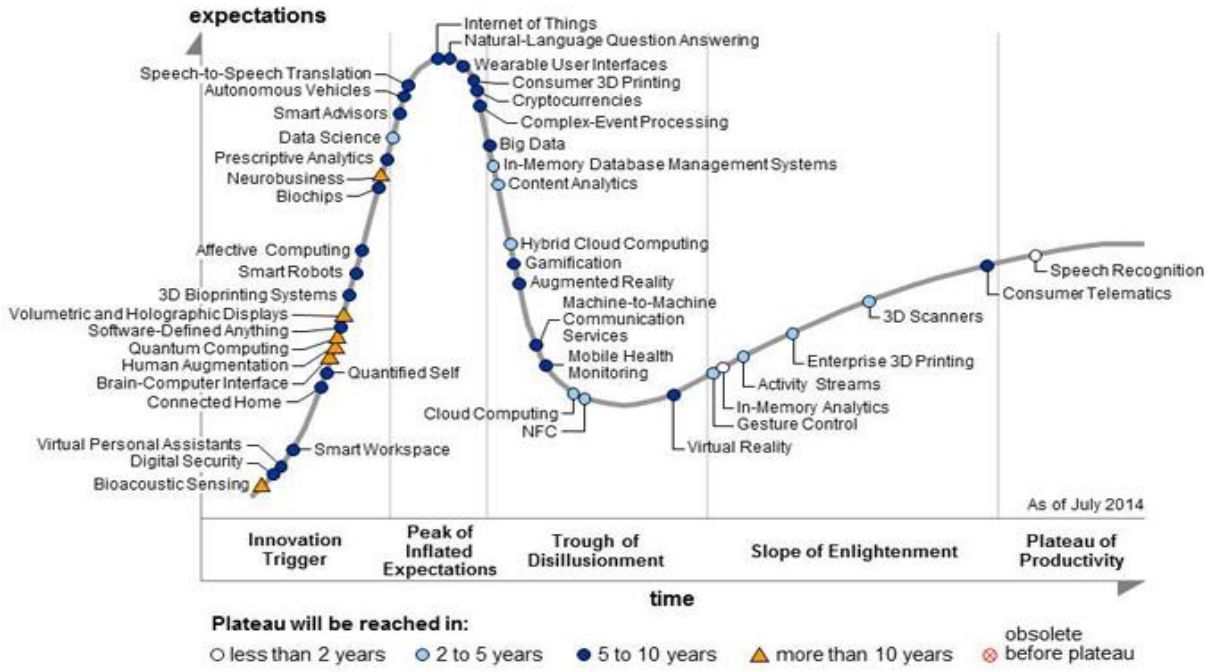
Η πρώτη "έξυπνη" συσκευή εμφανίστηκε το 1982 στο Carnegie Mellon University. Ήταν μια μηχανή της εταιρίας Coca Cola η οποία ήταν η πρώτη μηχανή που είχε πρόσβαση στο Internet μέσω του οποίου ενημέρωνε το περιεχόμενο της και αν τα αναψυκτικά που είχαν εισαχθεί πρόσφατα ήταν κρύα. Μεταξύ 1993 και 1996 εταιρίες πρότειναν λύσεις για αυτοματισμούς για μικρά σπίτια μέχρι ολόκληρα

εργοστάσια. Δύο από αυτά ήταν το Microsoft at Work και το Novell's Nest. Παρ' όλα αυτά δεν υπήρχε μεγάλη ανταπόκριση από το κοινό, μέχρι το 1999 όπου ο Bill Joy οραματίστηκε την επικοινωνία συσκευής με συσκευή (Device to Device), το οποίο και παρουσίασε στο Παγκόσμιο Οικονομικό Φόρουμ στο Davos το 1999 σαν μέρος του "Six Webs framework" του.

Ο όρος Internet of Things επινοήθηκε από τον Kevin Ashton, Διευθύνων σύμβουλο και ιδρυτή του Auto-ID Center, ο οποίος αναφέρει σε συνέντευξη του ότι τα Radio Frequency Identification (RFID) θα είναι απαραίτητα για την ύπαρξη του Internet of Things, καθώς αναφέρει ότι αν όλοι οι άνθρωποι και τα αντικείμενα ήταν εξοπλισμένα με αναγνωριστικούς αισθητήρες τότε οι υπολογιστές θα μπορούσαν να τα διαχειριστούν και να κάνουν την απογραφή τους πολύ πιο εύκολα.

Ένα μεγάλο βήμα για την παγκόσμια ανάπτυξη του Internet of Things έγινε το 2011 όταν απελευθερώθηκε η χρήση των διευθύνσεων IPv6. Πλέον δεν υπάρχει όριο στον αριθμό των διευθύνσεων, καθώς το νέο πρωτόκολλο επιτρέπει τη χρήση 2^128 ή 340 undecillion διευθύνσεων ή όπως αναφέρει ο Steven Leibson "Μπορούμε να αποδώσουμε μια Ipv6 διεύθυνση σε κάθε άτομο στην επιφάνεια της γης και πάλι να έχουμε αρκετές διευθύνσεις για της 100+ πλανήτες".

Το Internet of Things συνέχισε να αναπτύσσετε ώσπου το 2015 ονομάστηκε η "Η χρονιά του Internet of Things" καθώς σύμφωνα με την ιστοσελίδα Gartner ήταν η πιο αναπτυσσόμενη τεχνολογία σκαρφαλώνοντας από την 12^{η} θέση που κατείχε το 2011. Σύμφωνα με την ίδια ιστοσελίδα αναφέρεται ότι περίπου 4,9 δισεκατομμύρια συσκευές ήταν συνδεδεμένες το 2015 και αναμένεται ότι ο αριθμός της θα αυξηθεί περίπου 30% για το 2016 φτάνοντας τα 6,4 δισεκατομμύρια και τα 20.8 δισεκατομμύρια μέχρι το 2020.



Εικόνα 1.2 (Expectation of Internet of Things-Gartner,August2014)

3. Εφαρμογές

Το Internet of Things σήμερα βρίσκεται σχεδόν παντού. Εφαρμογές διασύνδεσης βασισμένες στην τεχνολογία αυτή βρίσκονται παντού, από μέσα μαζικής μεταφοράς, μέχρι ιατρικές συσκευές και οικιακές συσκευές. Ωστόσο οι εφαρμογές πλέον δεν βασίζονται μόνο στην παρακολούθηση και την καταγραφή δεδομένων μέσω των αισθητήρων αλλά και την λήψη αποφάσεων και τη χρήση ενεργοποιητών.

Αξίζει να αναφερθούν υλοποιήσεις και εφαρμογές σε διάφορους τομείς:

- **Παρακολούθηση του φυσικού περιβάλλοντος**

Εφαρμογές του IoT που έχουν υλοποιηθεί για περιβαλλοντολογικούς σκοπούς συνήθως είναι εφαρμογές παρακολούθησης. Αυτές χρησιμοποιούν αισθητήρες για την παρακολούθηση περιβαλλοντολογικών συνθηκών όπως η ταχύτητα και κατεύθυνση του αέρα ή η ποιότητα του νερού ή ακόμα και η οξύτητα του εδάφους σε μια καλλιεργημένη έκταση. Υπάρχουν εφαρμογές οι οποίες ασχολούνται με την πανίδα, τον τρόπο με τον οποίο κινούνται τα κοπάδια ή ακόμη και τις συνθήκες διαβίωσης της. Τέλος, αξιοσημείωτο είναι να αναφέρουμε της εφαρμογές αυτές οι οποίες έχουν δημιουργηθεί για την αποφυγή φυσικών καταστροφών της πυρκαγιών ή τσουνάμι.

Εκτός από της εφαρμογές στον τομέα της παρακολούθησης του φυσικού περιβάλλοντος, υπάρχουν εφαρμογές οι οποίες χρησιμοποιούνται και για την παρακολούθηση και διαχείριση του ανθρώπινου περιβάλλοντός.

- **Διαχείριση υποδομών**

Πολύ σημαντικές είναι οι εφαρμογές αυτές οι οποίες χρησιμοποιούνται για την παρακολούθηση και τον έλεγχο αστικών και μη αστικών υποδομών όπως γέφυρες, σιδηροδρομικές γραμμές, αιολικές γεννήτριες κτλ. Με τη χρήση των εφαρμογών αυτών έχουμε συνεχή έλεγχο κατάστασης και άμεση ενημέρωση για τυχόν αλλαγές στη δομή κάτι το οποίο μας εξασφαλίζει καλύτερο έλεγχο σε θέματα ασφαλείας και αποφυγής ζημιών. Έτσι με τον τρόπο αυτό γίνεται καλύτερος προγραμματισμός για επιδιορθώσεις και οι συντηρήσεις είναι πιο αποδοτικές. Τέλος, με τον απομακρυσμένο έλεγχο υποδομών κερδίζεται χρόνος και γίνεται πιο εύκολη η πρόσβαση στα μέσα μεταφοράς όπως στην χρήση εφαρμογών IoT που χρησιμοποιούνται της γέφυρες για να γίνεται πιο εύκολη η πρόσβαση των πλοίων.

Με τη χρήση των IoT εφαρμογών στις υποδομές επιτυγχάνεται η μείωση εκτάκτων περιστατικών και η καλύτερη διαχείριση αυτών. Έτσι γίνεται καλύτερη και πιο αποδοτική συντήρηση των υποδομών και αυξάνεται το όριο ζωής αυτών. Τέλος, γίνεται μεγιστοποίηση του χρόνου λειτουργίας αυτών και αλλά και μείωση του κόστους συντήρησης.

- **Διαχείριση ενέργειας**

Ένας από τους πιο σημαντικούς τομείς στους οποίους έχει αναπτυχθεί το Internet of Things είναι η διαχείριση της ενέργειας των συστημάτων που το ενσωματώνουν. Με την ενσωμάτωση συστημάτων ανίχνευσης και ενεργοποίησης

συνδεδεμένα στο internet, επιτυγχάνεται η βελτίωση της ενεργειακής κατανάλωσης του συνολικού συστήματος.

Αναμένεται στο μέλλον IoT συσκευές να αντικαταστήσουν της σημερινές ενεργοβόρες συσκευές όλων των τύπων που χρησιμοποιούμε σήμερα (όπως λάμπες, διακόπτες, τηλεοράσεις κτλ). Οι συσκευές αυτές θα επικοινωνούν με τον server της εταιρίας παροχής ενέργειας και θα προσπαθούν να βελτιστοποιήσουν τόσο την ενεργειακή παραγωγή όσο και την ενεργειακή κατανάλωση. Έτσι θα παρέχεται η δυνατότητα παρακολούθησης και ενεργοποίησης αυτών μέσω του συστήματος διεπαφής που θα βασίζεται στην τεχνολογία "cloud". Έτσι ο χρήστης θα έχει τη δυνατότητα να διαχειριστεί εξ αποστάσεως τον οικιακό φωτισμό, το σύστημα θέρμανσης, το θερμοσίφωνα κτλ.

- **Συστήματα υγείας και ιατροφαρμακευτικής περίθαλψης**

Ένας επίσης από τους σημαντικότερους τομείς στους οποίους αναπτύσσεται το IoT είναι αυτός της υγειάς. Αναμένεται στο μέλλον IoT συσκευές να χρησιμοποιούνται για την απομακρυσμένη παρακολούθηση ασθενών και την άμεση ενημέρωση του ιατρού σε περίπτωση κάποιου έκτακτου συμβάντος. Οι συσκευές αυτές θα ξεκινούν από μετρητές πίεσης και σφυγμών μέχρι ειδικές συσκευές για την παρακολούθηση μοσχευμάτων όπως βηματοδοτών ή ακουστικά ενίσχυσης ακοής. Με τη χρήση των συσκευών αυτών κάποιες ειδικές ομάδες ανθρώπων όπως και οι ηλικιωμένοι άνθρωποι κυρίως θα έχουν καλύτερη ιατρική φροντίδα και περίθαλψη. Έτσι δεν θα είναι απαραίτητη η συνεχής μετακίνηση τους για ιατρικές εξετάσεις. Επίσης, σε περίπτωση κάποιου επείγοντος περιστατικού οι γιατροί θα έχουν πιο άμεση επαφή με τον ασθενή.

Τα τελευταία χρόνια έχει εμφανιστεί μια νέα μόδα στον τομέα της τεχνολογίας ο οποίος αναπτύσσεται συνεχώς. Ο λόγος για τις συσκευές οι οποίες "φοριούνται" στο σώμα του χρήστη. Ξεκινώντας από περικάρπια μέχρι γυαλιά, η τεχνολογία αυτή τείνει να καταλαμβάνει το καταναλωτικό κοινό και σύμφωνα με το περιοδικό Forbes το 2014 ποσοστό 71% των ερωτηθέντων ηλικίας από 16 έως 24 θέλουν να αποκτήσουν μια τέτοια συσκευή. Οι περισσότερες από τις συσκευές αυτές ενθαρρύνουν τους χρήστες για έναν πιο υγιεινό τρόπο ζωής. Αυτό διότι ενσωματώνουν συστήματα παρακολούθησης καρδιακών παλμών ή μετρητή βημάτων οι οποίοι σε συνδυασμό με μια εφαρμογή στο "smartphone", τους επιτρέπει να αθληθούν πιο σωστά και έχουν έναν πιο υγιεινό τρόπο ζωής. Ένα από τα σημαντικά πλεονεκτήματα που προσφέρουν οι συσκευές αυτές είναι το χαμηλό κόστος τους και η μεγάλη ποικιλία αλλά και συμβατότητα με σχεδόν όλα τα "έξυπνα" κινητά σήμερα.

4. Άλλες τεχνολογίες που αναπτύχθηκαν μαζί με το Internet of Things

Κατά τη διάρκεια της διαχρονικής εξέλιξης του Internet of Things και στην διαρκή προσπάθεια για ανάπτυξη νέων εφαρμογών, εφευρέθηκαν κάποιες τεχνολογίες οι οποίες χρησιμοποιούνται σήμερα για την πιο εύκολη υλοποίηση διαφόρων εφαρμογών.

- **Machine to machine (M2M)**

Ο όρος "Machine to machine" αναφέρεται στην απευθείας επικοινωνία που έχουν μεταξύ τους οι συσκευές χρησιμοποιώντας ένα κανάλι επικοινωνίας είτε ενσύρματο είτε ασύρματο. Για να υπάρχει επικοινωνία μεταξύ των συσκευών είναι απαραίτητη η χρήση του πρωτοκόλλου IP και ειδικότερα η έκδοση Ipv6 διότι επιτρέπει έναν πάρα πολύ μεγάλο αριθμό διαθέσιμων διευθύνσεων.

Με τη χρήση της τεχνολογίας αυτής γίνεται εφικτή η αποστολή μηνυμάτων (με τα δεδομένα που διαβάζονται) από έναν αισθητήρα στον ενεργοποιητή απευθείας, ο οποίος φέρει λογισμικό το οποίο μπορεί να διαχειριστεί σωστά τα δεδομένα αυτά. Μια τέτοια επικοινωνία στο παρελθόν ήταν εφικτή με τη χρήση δικτύου υπολογιστών το οποίο δεχόταν τα δεδομένα, τα έστελνε σε έναν κεντρικό server για ανάλυση και μετά ξανά πίσω στον υπολογιστή.

Στις μέρες μας, με τη χρήση του πρωτοκόλλου IP παγκοσμίως, η επικοινωνία "μηχανής με μηχανή" γίνεται πολύ πιο γρήγορα, πιο εύκολα και πάνω από όλα με τη χρήση πολύ λιγότερης ενέργειας.

- **Personal Area Network (PAN)**

Ένα δίκτυο προσωπικής περιοχής είναι ένα δίκτυο το όποιο χρησιμοποιείται για την αποστολή δεδομένων μεταξύ υπολογιστών, τηλεφώνων ή άλλων ψηφιακών συσκευών. Τα δίκτυα αυτά χρησιμοποιούνται για την επικοινωνία συσκευών μεταξύ τους ή την επικοινωνία με το ευρύτερο δίκτυο (internet).

Στην ασύρματη μορφή της, η τεχνολογία αυτή χρησιμοποιεί ένα από τα παρακάτω πρωτοκολλά:

i. Υπέρυθρες (IrDA)
ii. Bluetooth
iii. ZigBee
iv. INSTEON
v. Wireless USB
vi. Z-Wave
vii. Body Area Network

Με τους παραπάνω τρόπους επιτυγχάνεται η δικτύωση συσκευών σε αποστάσεις από λίγα εκατοστά μέχρι μερικά μέτρα ανάλογα με το πρωτόκολλο και τις συσκευές που χρησιμοποιούμε.

Στην ενσύρματη μορφή της, ο τρόπος δικτύωσης επιτυγχάνεται συνήθως με τη χρήση ενσύρματων μέσων όπως το USB ή το Firewire.

- **Wearable Technology**

Όπως προαναφέρθηκε σε προηγούμενο υποκεφάλαιο, τα τελευταία χρόνια έχει εμφανιστεί μια νέα "μόδα" στον χώρο της τεχνολογίας. Ο λόγος για την τεχνολογία που "φοριέται". Οι συσκευές αυτές εκτός από συσκευές τελευταίας

τεχνολογίας, τις περισσότερες φορές αγοράζονται από το αγοραστικό κοινό και σαν αξεσουάρ μόδας.

Οι συσκευές αυτές συνδυάζουν ρουχισμό ή κάποιο άλλο αξεσουάρ που "φοριέται", με κάποιο μικροαισθητήρα ή ακόμη και ολόκληρο υπολογιστικό σύστημα. Οι συσκευές αυτές περιέχουν μικροηλεκτρονικά υποσυστήματα, αισθητήρες ή λογισμικό. Έτσι η ικανότητα τους να συνδέονται στο δίκτυο και να ανταλλάσουν δεδομένα μεταξύ τους ή με τον διαχειριστή τους, τα καθιστά ένα πολύ καλό παράδειγμα του Internet of Things.

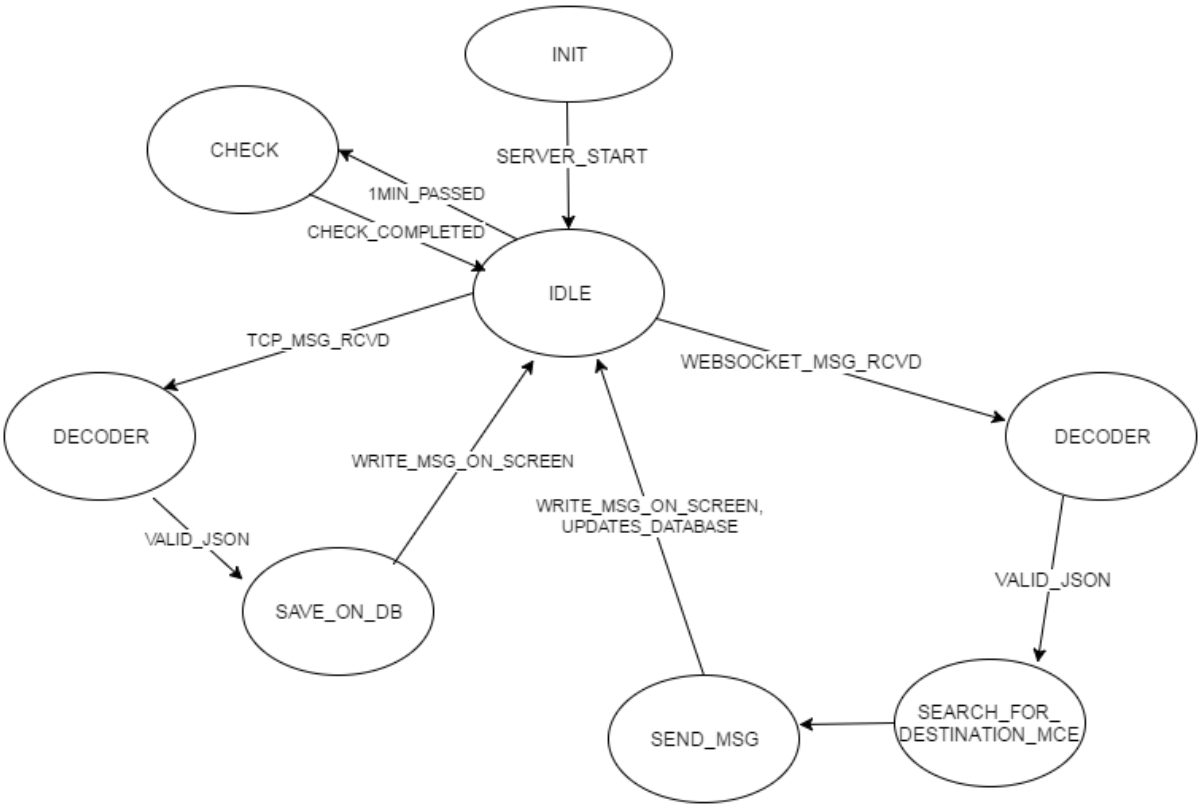
Οι συσκευές αυτές συνήθως χρησιμοποιούνται για κάποιον από τους παρακάτω σκοπούς:

- Αξεσουάρ μόδας
- Ανιχνευτής γυμναστικής
- Συσκευές παρακολούθησης για κάποια θέματα υγείας
- Μετρητής ενέργειας και επιπέδου ετοιμότητας
- Συσκευή πολυμέσων
- Εργαλείο πλοήγησης
- Συσκευή επικοινωνίας

Ένα πολύ αξιόλογο παράδειγμα συσκευής που "φοριέται" είναι τα Google Glasses. Τα Google Glasses είναι ένα εγχείρημα της εταιρίας Google να δημιουργήσει ένα ζευγάρι γυαλιά τα οποία θα έχουν δίνουν στο χρήστη τη δυνατότητα να βλέπει μηνύματα ή άλλες ειδοποιήσεις σε μια οθόνη η οποία θα εμφανίζεται στο πάνω δεξιά μέρος του δεξιού φακού τους. Επίσης, η συσκευή αυτή διαθέτει κάμερα και φωνητικές εντολές οι οποίες δίνουν στο χρήστη τη δυνατότητα να δώσει οποιαδήποτε εντολή στα γυαλιά απλά μιλώντας τους. Δυστυχώς το εγχείρημα αυτό σταμάτησε το 2015 όταν η εταιρία σταμάτησε να πουλάει το πρωτότυπο αυτό προϊόν. Αξίζει να αναφερθεί ότι το συγκεκριμένο προϊόν πωλήθηκε μόνο στην Αμερική χωρίς να προλαβαίνει να εισαχθεί και στην ευρωπαϊκή αγορά. Ένας από τους λόγους αποτυχίας του συγκεκριμένου εγχειρήματος ήταν ίσως το πολύ υψηλό κόστος που απαιτούνταν για την απόκτηση τους.

ΚΕΦΑΛΑΙΟ ΙΙ

Σχεδίαση



Εικόνα 2.1(Διάγραμμα λειτουργίας εφαρμογής)

Στο κεφάλαιο αυτό θα δούμε την σχεδίαση της εφαρμογής που υλοποιήθηκε κατά τη διάρκεια αυτής της διπλωματικής. Η ανάλυση της σχεδίασης θα γίνει τόσο σε εσωτερικό επίπεδο όσο και εξωτερικό.

1. Εσωτερική σχεδίαση

Η εφαρμογή υλοποιεί έναν communication server/message broker που υλοποιεί αμφίδρομη σύνδεση μεταξύ των μικροελεγκτών και της ιστοσελίδας διαχείρισης.

- Communication Server

Αρχικά έπρεπε να αποφασιστεί με τα υπόλοιπα μέλη της ομάδας ο τρόπος με τον οποίο θα γίνεται η δικτύωση με την εκάστοτε πλευρά. Από την πλευρά των μικροελεγκτών η επικοινωνία γίνεται με τη χρήση TCP πακέτων με τη χρήση του TCP πρωτοκόλλου. Για την σωστή λειτουργία του server γίνεται ασύγχρονη χρήση του TCP Listener για την ανταλλαγή πολλών μηνυμάτων από πολλούς χρήστες ταυτόχρονα.

Από την άλλη πλευρά η επικοινωνία με την ιστοσελίδα αποφασίστηκε να γίνεται μέσω WebSocket που τρέχει πάνω από το TCP. Της και το TCP, το WebSocket είναι connection oriented που σημαίνει ότι πρέπει να εγκαθιδρυθεί σύνδεση πριν την ανταλλαγή οποιουδήποτε πακέτου. Εκτός από την επικοινωνία μεταξύ του server και της ιστοσελίδας, η εφαρμογή υλοποιεί και επικοινωνία με τη βάση δεδομένων όπου αποθηκεύονται τα δεδομένα των κόμβων. Η βάση υλοποιείται με τη βοήθεια του Xammp. Με τη χρήση αυτού υλοποιείται της apache server και μια βάση δεδομένων με τη χρήση της MySQL.

Και της δύο περιπτώσεις της είδαμε γίνεται χρήση του πρωτοκόλλου TCP. Ένα από τα κύρια χαρακτηριστικά του TCP είναι η χρήση πόρτων για την ανταλλαγή των μηνυμάτων. Ύστερα από έρευνα στο διαδίκτυο αποφασίστηκε να γίνει χρήση των πορτών 5020 και 5030 για την εκάστοτε πλευρά. Έτσι ζητήθηκε από τους υπεύθυνους της σχολής, η απελευθέρωση των πορτών αυτών από το firewall ώστε να μην εμποδίζεται η ανταλλαγή μηνυμάτων από αυτό.

- Message broker

Η εφαρμογή εκτός από την επικοινωνία των κόμβων με την ιστοσελίδα, υλοποιεί και το “message broker” ο οποίος αποκωδικοποιεί τα μηνύματα πριν αποθηκεύσει της τιμές τους στη βάση ή τα αποστείλει από την ιστοσελίδα στον εκάστοτε κόμβο.

Αρχικά έπρεπε να οριστεί μια αρχιτεκτονική μηνύματος η οποία θα χρησιμοποιούνταν τόσο από τον broker όσο και από τους κόμβους που θα συνδεόταν

σε αυτόν. Ύστερα από αρκετή έρευνα αποφασίστηκε να χρησιμοποιηθεί η αρχιτεκτονική τύπου JSON. Η αρχιτεκτονική αυτή ήταν υλοποιήσιμη από τη γλώσσα προγραμματισμού που χρησιμοποιήθηκε στον broker αλλά και από αυτή που χρησιμοποιήθηκε της κόμβους. Το μήνυμα που θα παραλαμβάνει από τους κόμβους θα είναι της μορφής

{
“NodeID”: “00:11:22:33:44:55”,
“Type”: “ControlSensor/Sensor”,
“Humidity”: <int>,
“Temperature”: <int>,
“CoolantTemp”: <int>,
“OnOff”: <Boolean>,
“HiLow”: <Boolean>,
“msgCount”: <int>
}

Όπου

- NodeID: Ταυτότητα του εκάστοτε κόμβου
- Type: Τύπος του αποστολέα (κόμβος ή ιστοσελίδα)
- Humidity: Τιμή υγρασίας που μετράτε
- Temperature: Τιμή θερμοκρασίας που μετράτε
- CoolantTemp: Τιμή θερμοκρασίας του νερού που έρχεται από το λέβητα
- OnOff: Μεταβλητή που ελέγχει την ενεργοποίηση του ανεμιστήρα
- HiLow: Μεταβλητή που ελέγχει την ταχύτητα του ανεμιστήρα
- msgCount: Αριθμός μηνύματος που στάλθηκε από τον κόμβο

Η εφαρμογή ύστερα θα στέλνει ένα μήνυμα της μορφής

{

\Check\:true,

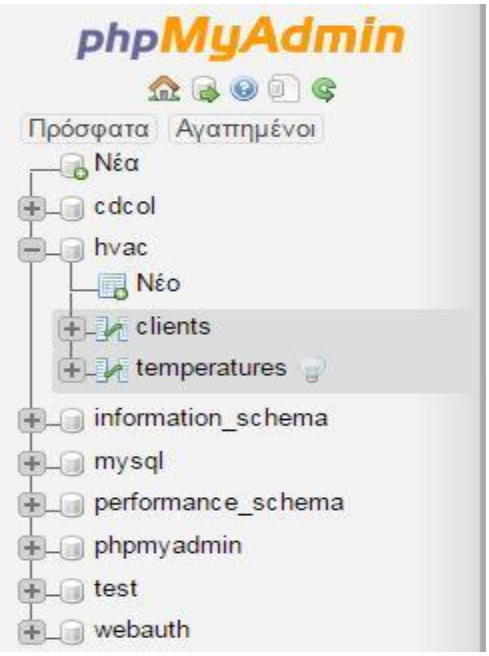
\msgCount\:msgCount

}

Όπου msgCount είναι αριθμός του μηνύματος που παρέλαβε από τον κόμβο ώστε να γίνει η επιβεβαίωση παραλαβής του μηνύματος.

Μετά την αποκωδικοποίηση του μηνύματος οι τιμές αποθηκεύονται στη βάση δεδομένων που είναι συνδεδεμένη η εφαρμογή. Για την κατασκευή της βάσης επιλέχθηκε η γλώσσα MySQL. Το σύστημα που περιλαμβάνει τον apache server που χρησιμοποιήθηκε για να υποστηριχθεί η βάση δημιουργήθηκε από τον συνάδερφο κ. Πανταζή Γρηγόριο ως μέρος της διπλωματικής του. Παρ’ όλα αυτά θα αναφερθούμε περιληπτικά στην αρχιτεκτονική της βάσης.

Η βάση αποτελείται από δύο πίνακες, τον πίνακα clients και τον πίνακα temperatures.



Εικόνα 2.1.3(Απόκομμα βάσης)

Στον πίνακα clients εισάγεται κάθε κόμβος που συνδέεται στην εφαρμογή μόνο μία φορά ενημερώνοντας τις τιμές του κάθε φορά που στέλνει μήνυμα. Οι εγγραφές του πίνακα αυτού χρησιμοποιούνται για την καλύτερη διαχείριση του συστήματος από τον διαχειριστή.

Ο πίνακας clients περιέχει τα εξής πεδία:

- ID : Ταυτότητα κόμβου
- IP : IP από την οποία παραλήφθηκε το τελευταίο μήνυμα
- Port : Πόρτα από την οποία παραλήφθηκε το τελευταίο μήνυμα
- Nickname : Όνομα κόμβου (δίνεται μέσω ιστοσελίδας)
- Temperature : Τιμή θερμοκρασίας
- Humidity : Τιμή υγρασίας
- Time : Ώρα παραλαβής τελευταίου μηνύματος
- State : Κατάσταση κόμβου
- Userstate : Κατάσταση που έχει ζητηθεί από το χρήστη
- Fan : On/Off
- Season : Winter/Summer
- Usertemp : Θερμοκρασία που έχει ζητηθεί από το χρήστη
- Fluidtemp : Θερμοκρασία νερού λέβητα
- Nodestate : Online/Offline/Obsolete

Στον πίνακα temperatures εισάγονται οι τιμές όλων των μηνυμάτων που έρχονται στην εφαρμογή. Οι εγγραφές που περιέχονται στον πίνακα αυτό θα χρησιμοποιούνται για στατιστικές αναλύσεις των κόμβων.

Ο πίνακας temperatures περιέχει τα εξής πεδία:

- Serial : Αριθμός εγγραφής (Αύξων αριθμός)
- ID : Ταυτότητα κόμβου
- Temp : Τιμή θερμοκρασίας

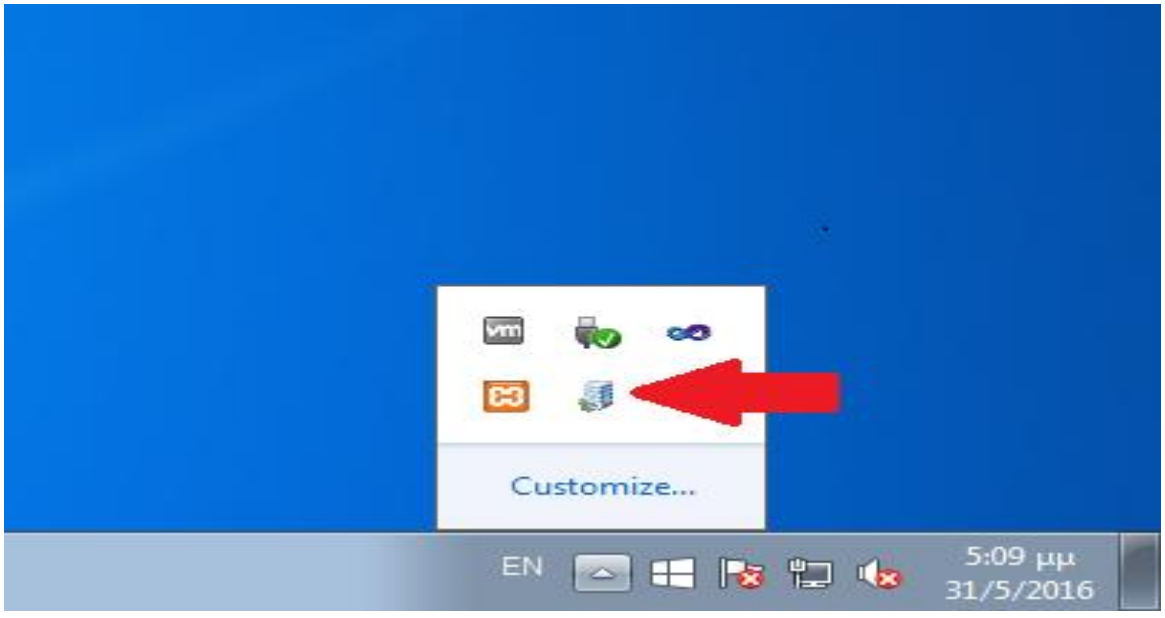
- Humidity : Τιμή υγρασίας
- Time : Ώρα παραλαβής τελευταίου μηνύματος
- FluidTemp : Θερμοκρασία νερού λέβητα
- State : Κατάσταση κόμβου
- Fan : Ταχύτητα ανεμιστήρα

2. Εξωτερική σχεδίαση

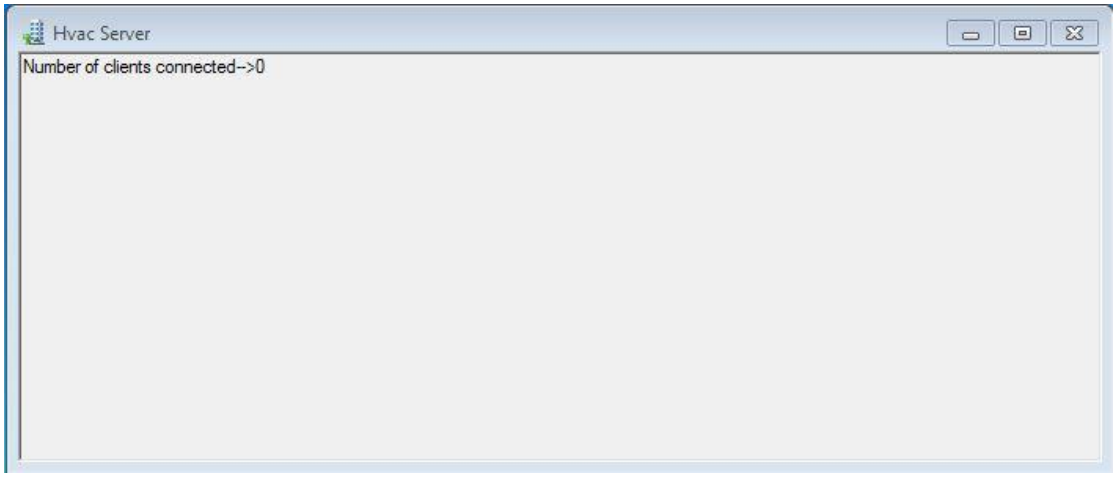
Η εξωτερική σχεδίαση της εφαρμογής περιλαμβάνει τα χαρακτηριστικά αυτά που αντιλαμβάνεται ο χρήστης.

Η εφαρμογή προγραμματίστηκε σε γλώσσα προγραμματισμού C#. Για την υλοποίηση χρησιμοποιήθηκε το περιβάλλον ανάπτυξης Microsoft Visual Studio 2010. Η γλώσσα αυτή επιτρέπει να χρησιμοποιήσουμε όλα τα εργαλεία που αποφασίσαμε ότι θα χρειαστούμε στην εφαρμογή αυτή. Στο επόμενο κεφάλαιο θα γίνει πιο εκτεταμένη ανάλυση τόσο στην γλώσσα προγραμματισμού όσο και το περιβάλλον ανάπτυξης.

Με την εκκίνηση της εφαρμογής ο διαχειριστής έχει τη δυνατότητα να δει ένα παράθυρο στο όποιο αναγράφονται όλα τα μηνύματα που έρχονται σε αυτήν. Τα μηνύματα τυπώνονται μετά την αποθήκευση τους στη βάση και με τη μορφή που είδαμε στην αρχή του κεφαλαίου. Η εφαρμογή αποφασίστηκε με την εκκίνηση της να τρέχει σε ελαχιστοποιημένη μορφή και έπειτα από επιλογή του χρήστη να επαναφέρεται σε μεγιστοποιημένο παράθυρο.



Εικόνα 2.2.1(Εικονίδιο εφαρμογής στη γραμμή εργαλείων)



Εικόνα 2.2.2(Interface εφαρμογής)

ΚΕΦΑΛΑΙΟ ΙΙΙ

Υλοποίηση

1. Γλώσσα προγραμματισμού C# και προγραμματιστικό περιβάλλον Microsoft Visual Studio 2010

Για την υλοποίηση της εφαρμογής χρησιμοποιήθηκε η γλώσσα προγραμματισμού C# με τη χρήση του προγραμματιστικού περιβάλλοντος Microsoft Visual Studio 2010.

▪ Γλώσσα προγραμματισμού C#

Η γλώσσα προγραμματισμού C# εμφανίστηκε στις αρχές του 2000. Είναι μια ολοκληρωμένη αντικειμενοστραφής γλώσσα που βασίζεται στη δημιουργία λογισμικού με τη χρήση ".Net Framework".

Το όνομα "C Sharp" είναι εμπνευσμένο από τη μουσική σημείωση όπου η δίεση συμβολίζει ότι η νότα που είναι γραμμένη πρέπει να παιχτεί ένα ημιτόνιο πιο ψηλά. Κάτι αντίστοιχο συμβαίνει και με την C++, όπου το "++" συμβολίζει ότι η μεταβλητή πρέπει να είναι επαυξημένη κατά 1. Το σύμβολο της δίεσης μοιάζει επίσης με την ένωση τεσσάρων "+" , κάτι το οποίο υπονοεί ότι η γλώσσα αυτή είναι επαύξηση της C++.

Η C# έχει εφτά εκδόσεις, και κάθε μια υποστηρίζει επιπλέον λειτουργίες από την προηγούμενη της. Η τελευταία έκδοση (C# 6.0) κυκλοφόρησε τον Ιούλιο του 2015 και υποστηρίζεται από τις βιβλιοθήκες .Net Framework 4.6 και την έκδοση Visual Studio 2015. Η εφαρμογή έχει προγραμματιστεί στην έκδοση C# 4.0 με την υποστήριξη του .Net Framework 4.

Η C# αναδείχθηκε από τον οργανισμό Popularity of Programming Language ως η δημοφιλέστερη γλώσσα προγραμματισμού 2012 ενώ αυτή τη στιγμή κατέχει την 5η θέση στις πιο δημοφιλής γλώσσες παγκοσμίως, σύμφωνα με τον οργανισμό IEEE. Μπορεί να μην είναι τόσο δημοφιλής αυτή τη στιγμή όπως η Java ή η PHP αλλά είναι συνεχώς αναπτυσσόμενη με διαρκώς αυξανόμενο κοινό και αναμένεται να κυριαρχήσει τα επόμενα χρόνια.

Μερικά από τα κύρια πλεονεκτήματα της είναι ότι είναι :

- Παρέχει άμεση πρόσβαση σε τεράστιες βιβλιοθήκες κλάσεων του .Net Framework
- Τα πάντα στη C# είναι αντικείμενα
- Εύκολη στη μάθηση
- Built-in functional programming capabilities
- Μεγάλη κοινότητα που μπορεί να σε βοηθήσει σε οποιαδήποτε απορία ή πρόβλημα

▪ Προγραμματιστικό περιβάλλον Microsoft Visual Studio

Το Microsoft Visual Studio είναι ένα ενσωματωμένο προγραμματιστικό περιβάλλον της Microsoft. Χρησιμοποιείται για την ανάπτυξη εφαρμογών για τα Windows αλλά και για ιστοσελίδες, εφαρμογές διαδικτύου και διαδικτυακές υπηρεσίες. Η πρώτη έκδοση του περιβάλλοντος αυτού κυκλοφόρησε το 1997.

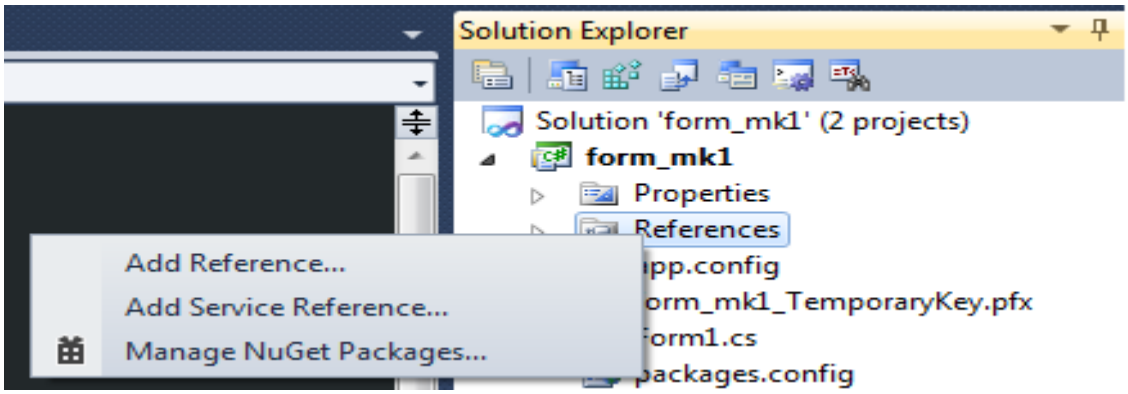
Η σημερινή του Visual Studio περιέχει έναν code editor και ένα debugger που επιτρέπουν στο χρήστη να προγραμματίσει σε σχεδόν οποιαδήποτε γλώσσα αυτός επιθυμεί. Το Visual Studio έχει ήδη ενσωματωμένες τις :

- C
- C++(Visual C++)
- Visual Basic .NET
- C# (Visual C#)
- F# (Έκδοση Visual Studio 2010 ή μεταγενέστερη)

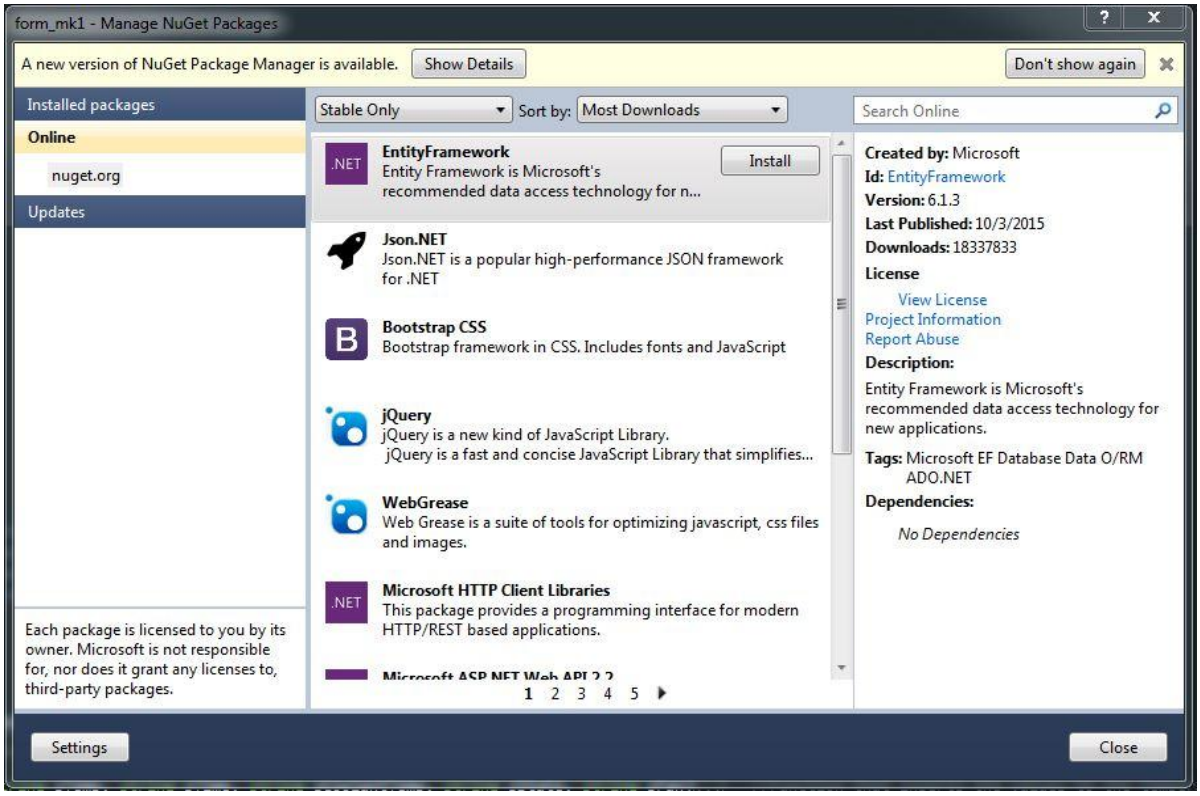
Για την υποστήριξη περισσότερων γλωσσών όπως οι :

- Python
- Ruby
- Node.js
- M
- XML
- HTML
- JavaScript
- CSS

Πρέπει να γίνει εγκατάσταση βιβλιοθηκών χωριστά. Για την εγκατάσταση των επεκτάσεων αυτών η Microsoft δημιούργησε το NuGet. Το NuGet είναι ένα εργαλείο που επιτρέπει στο χρήστη την εύκολη αναζήτηση και εγκατάσταση των επεκτάσεων αυτών.



Εικόνα 3.1.1 (Επιλογή NuGet από Solution Explorer)

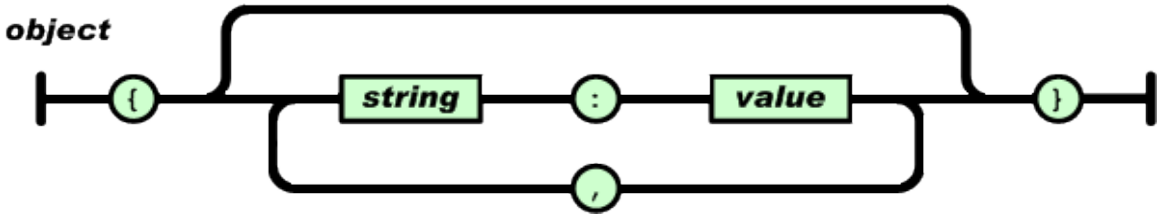


Εικόνα 3.1.2 (NuGet Interface)

2. Επιπλέον βιβλιοθήκες που χρησιμοποιήθηκαν στην εφαρμογή

- **JSON (JavaScript Object Notation)**

Για την αρχιτεκτονική του μηνύματος όπως αναφερθήκαμε και στο προηγούμενο κεφάλαιο, χρησιμοποιήθηκε η βιβλιοθήκη JSON. Το JSON είναι αρχιτεκτονική που έχει ελαφριά μορφή και επιτρέπει την ανταλλαγή δεδομένων. Βασίζεται στην γλώσσα προγραμματισμού JavaScript. Είναι εύκολο για τον άνθρωπο να το διαβάσει και να το γράψει και για τη μηχανή να το αναλύσει και να το παράγει.



Εικόνα 3.2.1 (JSON Object)

- **MySQL**

Για την υλοποίηση της επικοινωνίας μεταξύ εφαρμογής και βάσης δεδομένων χρησιμοποιήθηκε η βιβλιοθήκη MySQL. Η MySQL πρωτοεμφανίστηκε στα μέσα στης δεκαετίας του 90’ από την Oracle Corporation. Ακόμη και σήμερα αποτελεί ένα από τα πιο πολυχρησιμοποιημένα συστήματα διαχείρισης σχεσιακών βάσεων δεδομένων, μετρώντας περισσότερες από 11 εκατομμύρια εγκαταστάσεις.

- **Fleck**

Η πιο δύσκολη επιλογή ήταν αυτή της βιβλιοθήκης που θα υλοποιεί τον WebSocket Server για την επικοινωνία με την ιστοσελίδα. Για την υλοποίηση ενός WebSocket Server υπάρχουν αρκετές βιβλιοθήκες διαθέσιμες. Μερικές από αυτές είναι:

- Fleck
- SignalR
- AlchemyWebSocket
- XSockets
- Microsoft.WebSocket
- SuperSocket

Επιλέχθηκε η βιβλιοθήκη Fleck διότι ήταν η πιο εύκολη στην χρήση χωρίς να χρειάζεται πολλές απαιτήσεις από το σύστημα. Ένα από τα βασικά μειονεκτήματα της είναι ότι δεν είναι αρκετά παραμετροποιήσιμη. Σε άλλη περίπτωση αυτό θα ήταν αρκετά μεγάλο πρόβλημα διότι παρουσιάζει αρκετά κενά ασφαλείας. Στην περίπτωση μας όμως η εφαρμογή τρέχει σε τερματικό εντός του δικτύου της σχολής αποκλείοντας αυτόν τον κίνδυνο.

3. Ανάλυση συναρτήσεων εφαρμογής

Με την εκκίνηση της εφαρμογής ενεργοποιείται η διαδικασία που δημιουργεί το παράθυρο. Αμέσως δηλώνονται οι global μεταβλητές και καλείται η διαδικασία Server().

- **Server()**

Η διαδικασία αυτή ενεργοποιεί τον TcpListener και τον WebSocket Server.

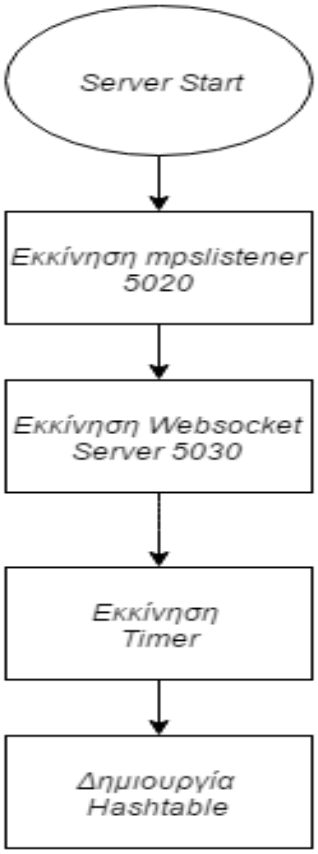
```
private void Server() //Functions that creates both the TcpListener and the
WebSocket Server
        {
            this.msplistener = new TcpListener(IPAddress.Any, 5020); //Creates TCP
Socket Listener for port 5020
            this.listenthread  =  new  Thread(new  ThreadStart(ListenForClients));
//Creates a new Thread for every new client
            this.listenthread.IsBackground = true; //Runs all threads on backround
            this.listenthread.Start(); //Starts the thread
            ListenFromWeb(); //Calls ListenFromWeb function
            Timer(); //Calls Timer Function
        }
```

Η εντολή this.msplistener = new TcpListener(IPAddress.Any, 5020) δημιουργεί έναν TCP Server που ακούει σε όλες τις διαθέσιμες διευθύνσεις IP του υπολογιστή και στην πόρτα 5020 που έχουμε δηλώσει ότι θα περνάνε τα μηνύματα που θα ανταλλάσσονται με τους κόμβους.

Για την καλύτερη διαχείριση της επικοινωνίας της εφαρμογής με τους κόμβους, πρέπει να γίνεται ασύγχρονη χρήση του TcpListener έτσι ώστε αυτή να μπορεί να δεχτεί μηνύματα από οποιονδήποτε κόμβο σε οποιαδήποτε χρονική στιγμή. Γι' αυτό λοιπόν γίνεται η χρήση νημάτων. Με την εντολή this.listenthread = new Thread(new TheadStart(ListenForClients)) δημιουργούμε ένα νήμα για κάθε νέα σύνδεση κόμβου στην εφαρμογή και αυτό διαχειρίζεται τη σύνδεση μέχρι να κλείσει η εφαρμογή.

Έπειτα καλείται η διαδικασία ListenFromWeb η οποία ενεργοποιεί τον WebSocket Server και διαχειρίζεται τα μηνύματα που ανταλλάσσονται με την ιστοσελίδα.

Τέλος, καλείται η διαδικασία Timer που ενεργοποιεί το εσωτερικό ρολόι για τον έλεγχο των κόμβων.



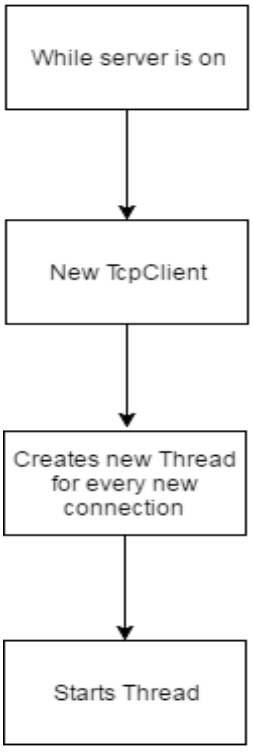
Εικόνα 3.3.1(Διάγραμμα ροής με την εκκίνηση της εφαρμογής)

- **ListenForClients**

Αμέσως μετά την κλήση της διαδικασίας server, καλείται η διαδικασία ListenForClients. Η διαδικασία αυτή είναι υπεύθυνη να διαχειριστεί κάθε νέα TCP σύνδεση.

```
private void ListenForClients() //Function that handles every new connection on
port 5020
        {
            this.msplistener.Start(); //Starts the mpslistener
            while (true) //Never ends until the Server is closed
            {
                TcpClient client = this.msplistener.AcceptTcpClient(); //Waits for
a client to connect
                Thread clientThread = new Thread(new
ParameterizedThreadStart(HandleClientComm)); //Creates a new thread to handle
connection with the connected client
                clientThread.Start(client); // Start very new Thread
            }
```

Για την σωστή λειτουργία της εφαρμογής, πρέπει ο server να μπορεί να δεχτεί μηνύματα ασύγχρονα. Το πρόβλημα αυτό παρουσιάστηκε αρχικά στην υλοποίηση της εφαρμογής όταν δοκιμαζόταν η ανταλλαγή μηνυμάτων με τους κόμβους. Η ανταλλαγή όταν ήταν μόνο ένας κόμβος συνδεδεμένος στο server λειτουργούσε σωστά. Όταν όμως συνδεόταν και δεύτερος κόμβος θα έπρεπε πρώτα να κλείσει η TCP σύνδεση του πρώτου για να μπορέσει να ανταλλάξει μήνυμα με τον δεύτερο.



Εικόνα 3.3.2 (Διάγραμμα ροής ListenForClients Function)

Με την εντολή `Thread clientThread = new Thread(new ParameterizedThreadStart(HandleClientComm))` δημιουργείται ένα νέο νήμα που διαχειρίζεται χωριστά κάθε νέα σύνδεση με έναν κόμβο. Κάθε φορά που εκτελείται η εντολή αυτή, καλείται εσωτερικά της η διαδικασία HandleClientComm.

- **HandleClientComm**

Η διαδικασία αυτή είναι μία από τις πιο βασικές διαδικασίες την εφαρμογής. Είναι υπεύθυνη για την διαχείριση της επικοινωνίας μεταξύ κόμβων και εφαρμογής. Εκτός από την διαχείριση της επικοινωνίας, εσωτερικά της διαδικασίας αυτής γίνεται αποκωδικοποίηση του μηνύματος που έρχεται στον server και καλούνται η αντίστοιχες διαδικασίες που θα το αποθηκεύσουν στην βάση.

```
            NetworkStream clientStream = tcpClient.GetStream(); //Creates a unique
NetworkStream for each client
```

Με την παραπάνω εντολή δημιουργείται ένα κανάλι επικοινωνίας για κάθε κόμβο που συνδέεται στο server. Στο κομμάτι κώδικα που ακολουθεί φαίνεται η αποκωδικοποίηση του μηνύματος και ο τρόπος με τον οποίο η εφαρμογή παίρνει τις πληροφορίες από αυτό.

```
                    var obj = JToken.Parse(result); //Creates a new JSON object
with the message before parsing
                    JObject o = JObject.Parse(result); //Parses the message and
saves it to Object "o"
                    string NodeID = (string)o["NodeID"]; //Gets the NodeID from
decoded message
```

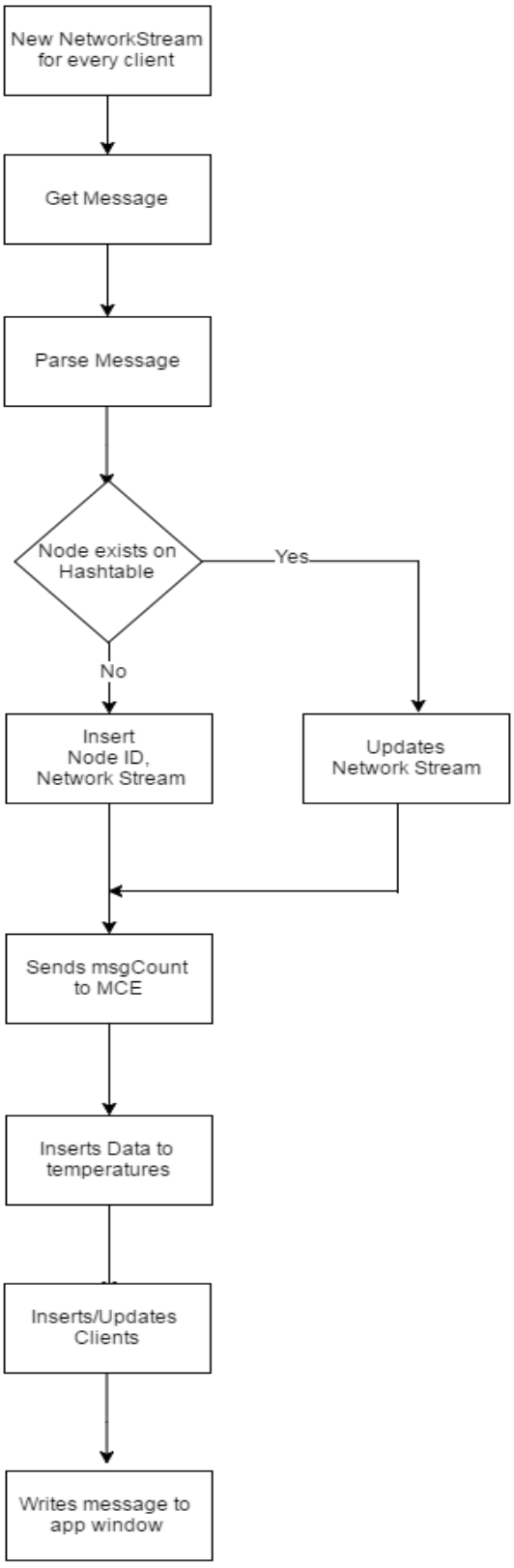
Για να γίνει πιο σωστή η επικοινωνία της εφαρμογής με τους κόμβους αποφασίστηκε να γίνεται αποθήκευση του ζευγαριού NodeID, NetworkStream σε έναν πίνακα όπου θα χρησιμοποιούνται για την αποστολή μηνυμάτων προς τους κόμβους όταν αυτό ζητηθεί. Αρχικά η αποστολή από τον server στον κόμβο γινόταν αποστέλλοντας μήνυμα σε όλους τους κόμβους και ο κόμβος αποφάσιζε αν το μήνυμα προοριζόταν γι' αυτόν ή όχι. Αυτή η λύση ήταν πιο εύκολα υλοποιήσιμη αλλά δεν προοριζόταν για μεγάλα συστήματα. Στο κομμάτι κώδικα που ακολουθεί λοιπόν βλέπουμε τον έλεγχο που γίνεται στον πίνακα εάν υπάρχει η εγγραφή NodeID. Εάν υπάρχει κάνει ενημέρωση στο αντικείμενο NetworkStream. Εάν όχι δημιουργεί μια νέα εγγραφή και αποθηκεύει και τις δύο τιμές.

```
                    if (node.ContainsKey(NodeID)) //Checks if NodeID exists on the
Hashtable
                    {
                        node[NodeID] = clientStream; //If yes, updates the
NetworkStream
                    }
                    else
                    {
                        node.Add(NodeID, clientStream); //If not,add a new line
with NodeID and NetworkStream
                    }
```

Έπειτα γίνεται έλεγχος από της μεταβλητής "type" αν έχει την τιμή "Control Sensor". Εάν ισχύει η προηγούμενη συνθήκη, τότε στέλνει πίσω μήνυμα που έχει μέσα τον αριθμό του μηνύματος που έστειλε έτσι ώστε να γίνει επιβεβαίωση παράδοσης. Εάν δεν γίνει η επιβεβαίωση, ο κόμβος ξαναστέλνει το μήνυμα.

```
                    if (NodeType == "ControlSensor") //Checks if NodeType ==
control sensor
                    {
                        StreamWriter sw = new StreamWriter(clientStream); //Creates
a new stream writer obj
```

```
                    sw.Write("{" + "\"Check\"" + ":" + "true" + "," +
"\"msgCount\"" + ":" + msgCount + "}"); // Replies with the msgCount that received
the message
                    sw.Flush(); //Sends message to node
                }
```



Εικόνα 3.3.3 (Διάγραμμα ροής συνάρτησης HandleClientComm)

- **Εισαγωγή δεδομένων στη βάση (InsertTempSQL , InsertUpdateClientSQL)**

Η εισαγωγή δεδομένων στη βάση γίνεται κάθε φορά που έρχεται κάποιο μήνυμα στο server. Για την αποθήκευση των δεδομένων χρησιμοποιούνται οι συναρτήσεις InsertTempSQL και InsertUpdateClientSQL. Όπως αναφέρεται και στο όνομα τους η πρώτη εισάγει μόνο τιμές στον πίνακα temperatures ενώ η δεύτερη ελέγχει αν υπάρχει ο κόμβος στην βάση πριν εισάγει τις τιμές σε αυτήν. Εάν υπάρχει τότε ενημερώνει τον πίνακα, αν όχι τότε εισάγει εκ νέου τις τιμές.

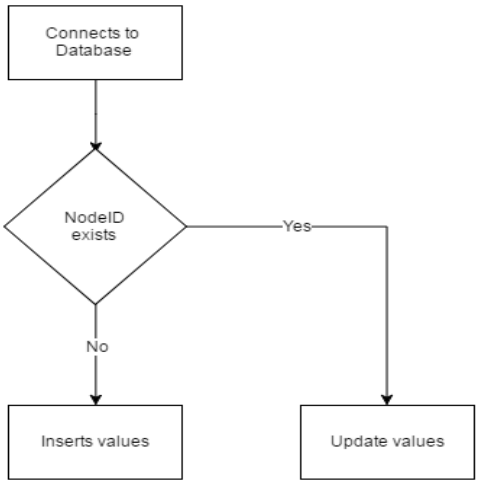
Και στις δύο περιπτώσεις ο τρόπος με τον οποίο γίνεται η επικοινωνία με την βάση είναι ο ίδιος.

Αρχικά ορίζεται η διαδρομή που θα ακολουθηθεί για την επικοινωνία. Με την παρακάτω εντολή ορίζεται σε μια μεταβλητή string το μονοπάτι. Ορίζεται στην αρχή η τοποθεσία, έπειτα το όνομα της βάσης και τέλος το username και password. Στην περίπτωση μας όμως δεν χρησιμοποιείται κάποιος κωδικός οπότε το πεδίο αυτό μένει κενό.

```
String str = @"server=localhost;database=hvac;userid=root;password=;";
```

Έπειτα με τη χρήση της εντολής "`MySqlConnection con = new MySqlConnection(str);`" δημιουργείται η σύνδεση με τη βάση και έπειτα με τις εντολές `MySqlCommand update = new MySqlCommand(updateQuery, con);` ή `MySqlCommand insert = new MySqlCommand(insertQuery, con);` γίνεται η αντίστοιχη ενέργεια στη βάση.

Αξίζει να σημειωθεί ότι η επικοινωνία με την βάση δεν είναι μόνιμα ανοιχτή. Ανοίγει κάθε φορά που καλείται κάποια από τις παραπάνω διαδικασίες και κλείνει αμέσως μετά την εκτέλεση της εντολής εισαγωγής δεδομένων.



Εικόνα 3.3.4(Διάγραμμα ροής InsertUpdateClientSQL function)

- **ListenFromWeb**

Με τη διαδικασία ListemFromWeb υλοποιείται η σύνδεση μεταξύ ιστοσελίδας και εφαρμογής. Όπως προαναφέρθηκε στην αρχή του κεφαλαίου η υλοποίηση γίνεται με τη χρήση της βιβλιοθήκης Fleck.

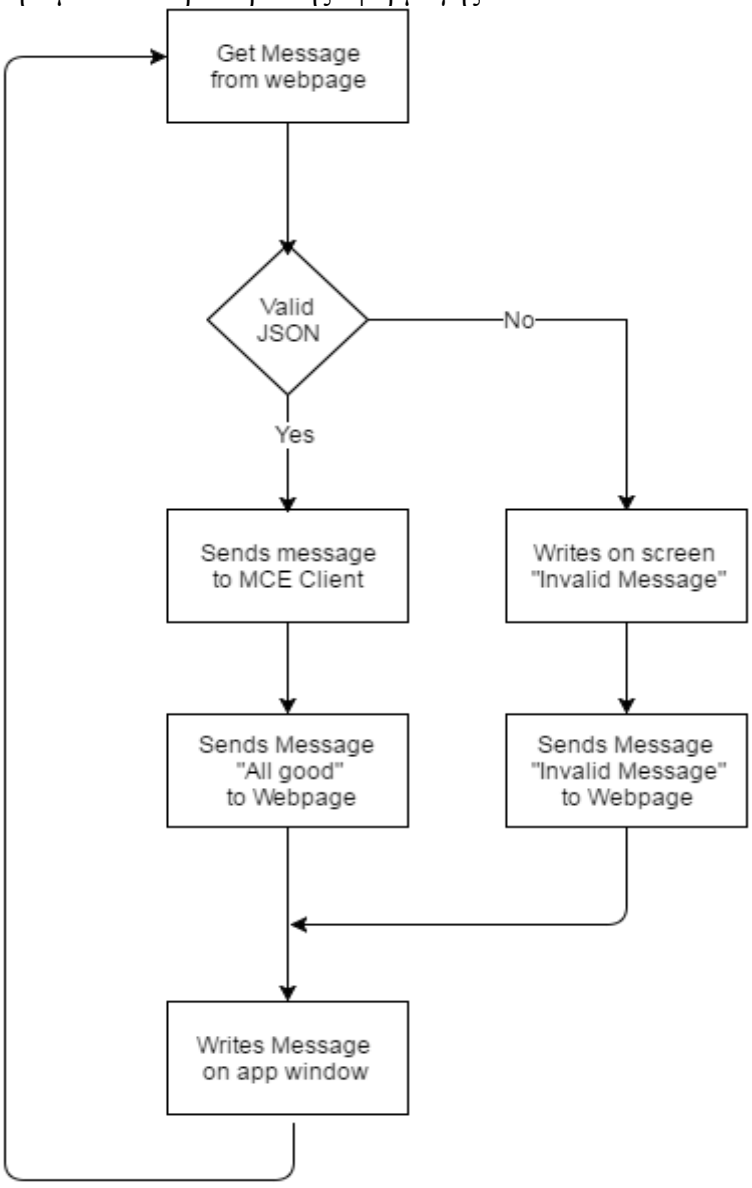
Όπως ο TCPListener έτσι και ο WebSocket server "ακούει" σε όλες τις διαθέσιμες IP του τερματικού. Ο δεύτερος όμως ακούει στην πόρτα 5030.

```
var server = new WebSocketServer("ws://0.0.0.0:5030");
```

Με τη χρήση της παραπάνω εντολής δημιουργείται ο WebSocket server. Έπειτα με τη χρήση της εντολής server.Start εκκινείται η λειτουργία του.

```
socket.OnMessage = (message) =>
```

Με την παραπάνω εντολή λαμβάνεται ένα μήνυμα από την ιστοσελίδα. Ύστερα το μήνυμα ελέγχεται αν είναι "valid JSON". Αν ναι, καλείται η διαδικασία SendToClient και στέλνεται μήνυμα στην ιστοσελίδα ότι το μήνυμα παραλήφθηκε. Αν όχι, στέλνεται μήνυμα στην ιστοσελίδα ότι το μήνυμα δεν είναι έγκυρο. Τέλος τυπώνεται το μήνυμα στο παράθυρο της εφαρμογής.

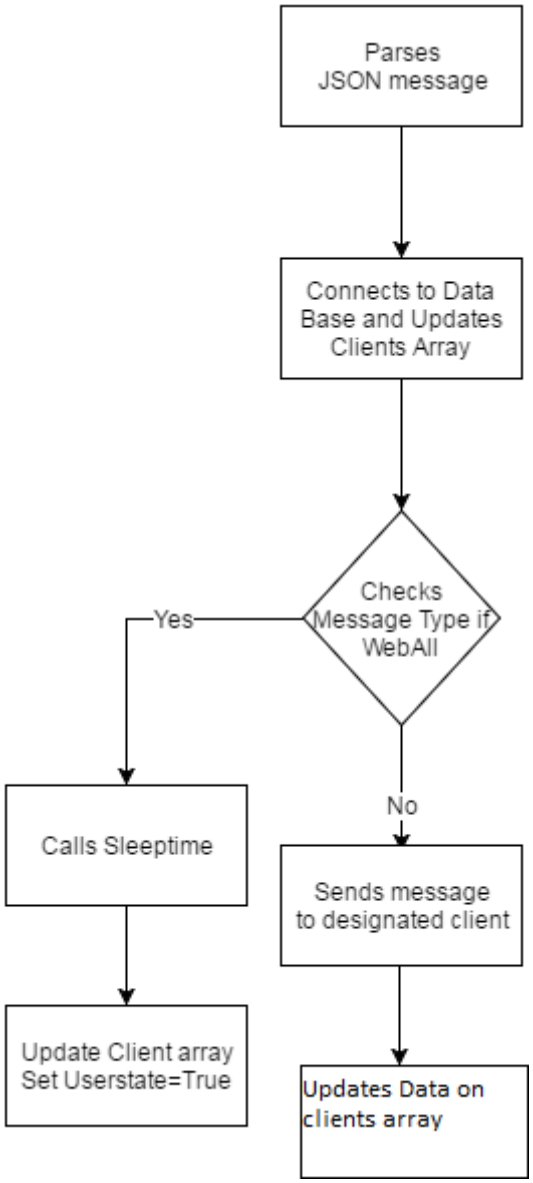


Εικόνα 3.3.5(Διάγραμμα ροής ListenFromWeb function)

- **SendToClient**

Η διαδικασία αυτή αρχικά αποκωδικοποιεί το μήνυμα που λήφθηκε από την ιστοσελίδα. Έπειτα δημιουργεί μια σύνδεση με τη βάση και τον πίνακα clients και

ενημερώνει αυτόν με τις νέες τιμές. Έπειτα υπάρχει ένας έλεγχος στον οποίο ελέγχεται ο τύπος του μηνύματος. Αν ο τύπος του μηνύματος είναι "WebbAll" τότε σημαίνει ότι ο χρήστης θέλει να κλείσει όλους του διαθέσιμους κόμβους. Γι' αυτό καλείται η διαδικασία Sleeptime και ενημερώνεται ο πίνακας clients στον όποιο όλοι οι κόμβοι με κατάσταση online γίνονται αυτόματα offline. Αν ο τύπος του μηνύματος είναι "Web" τότε καλείται η διαδικασία Echo.



Εικόνα 3.3.6(Διάγραμμα ροής διαδικασίας SendToClient)

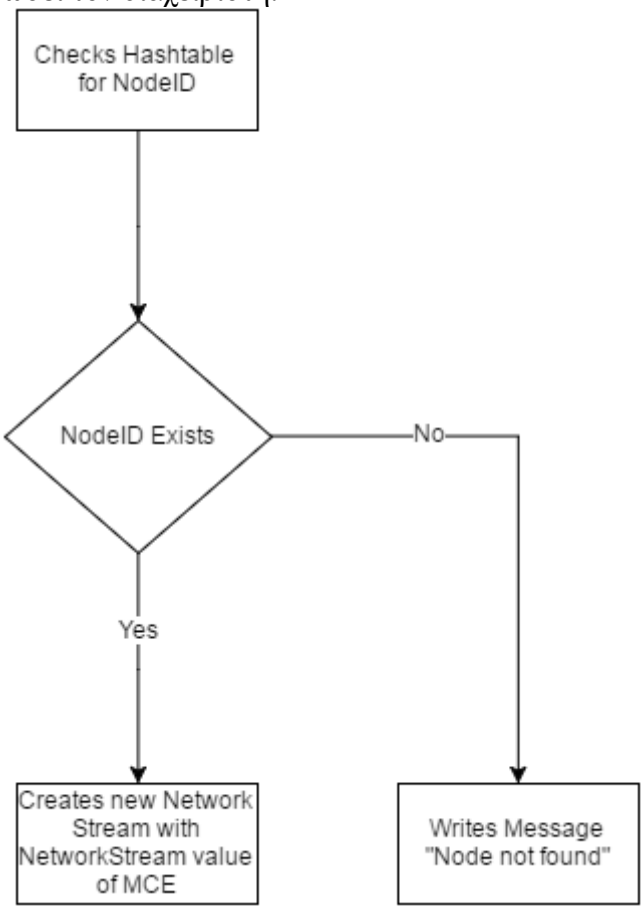
- **SleepTime**

Η διαδικασία αυτή στέλνει έναν συγκεκριμένο τύπο μηνύματος σε όλους του κόμβους που είναι διαθέσιμοι στο Hashtable.

- **Echo**

Η διαδικασία Echo λειτουργεί με παρόμοιο τρόπο όπως η διαδικασία Sleeptime. Μόνο που σε αυτή την περίπτωση γίνεται έλεγχος του NodeID πριν επιλεχθεί το NetworkStream στο όποιο θα σταλεί το μήνυμα. Εάν το NodeID του

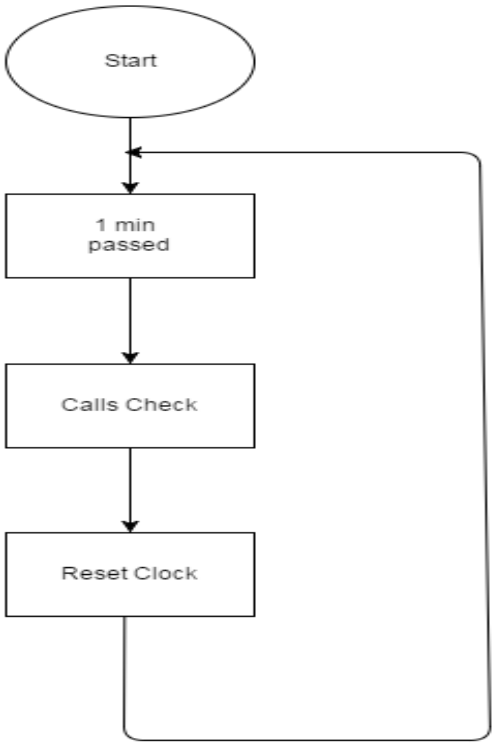
κόμβου που ζητήθηκε δεν υπάρχει στο Hashtable τότε τυπώνεται μήνυμα στην όθονη που για να ενημερώσει τον διαχειριστή.



Εικόνα 3.3.7(Διάγραμμα ροής Echo function)

- **Timer**

Η διαδικασία αυτή χρησιμοποιεί ένα εσωτερικό ρολόι του υπολογιστή για την καταγραφή χρόνου. Το ρολόι αυτό μετράει έχει περίοδο ένα λεπτό. Με το πέρας της περιόδου καλείται η διαδικασία Check. Ύστερα μηδενίζεται το ρολόι και ξεκινάει από την αρχή.

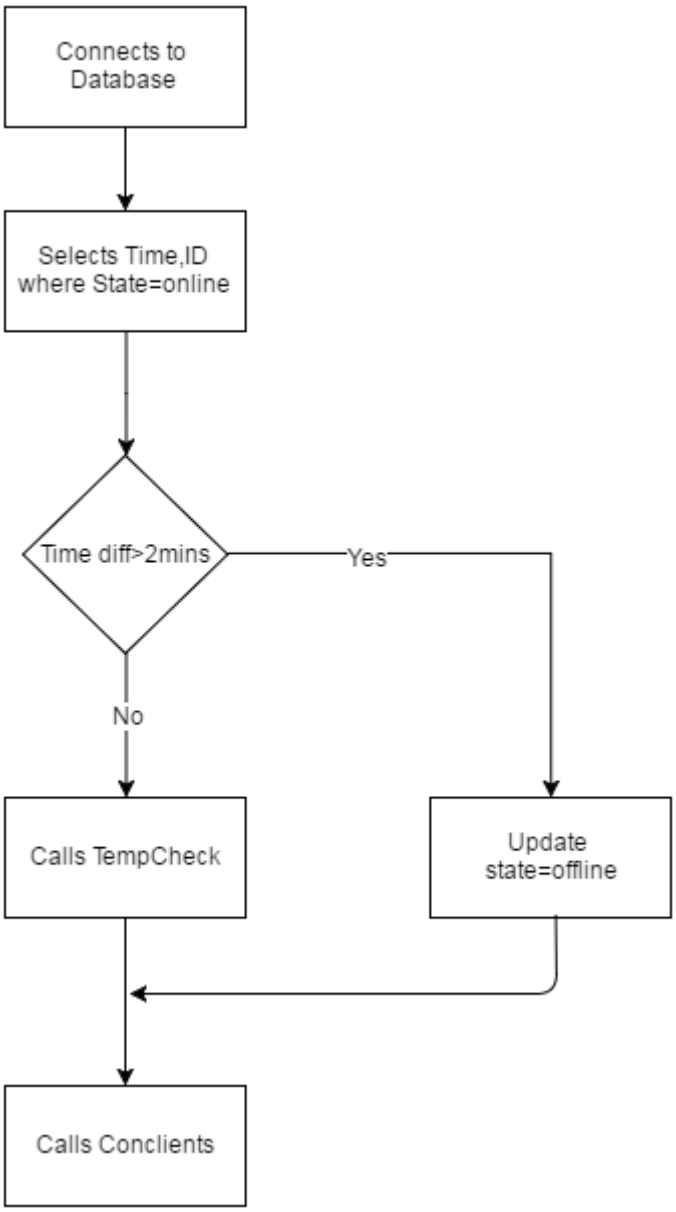


Εικόνα 3.3.8(Διάγραμμα ροής Timer function)

- **Check**

Η διαδικασία αυτή καλείται κάθε φορά που ολοκληρώνεται μια περίοδος του Timer. Η διαδικασία αυτή είναι υπεύθυνη να αποφασίσει ποιος κόμβος είναι online ή offline, συγκρίνοντας το χρόνο που παραλήφθηκε το τελευταίο μήνυμα με την στιγμή που καλέστηκε η διαδικασία.

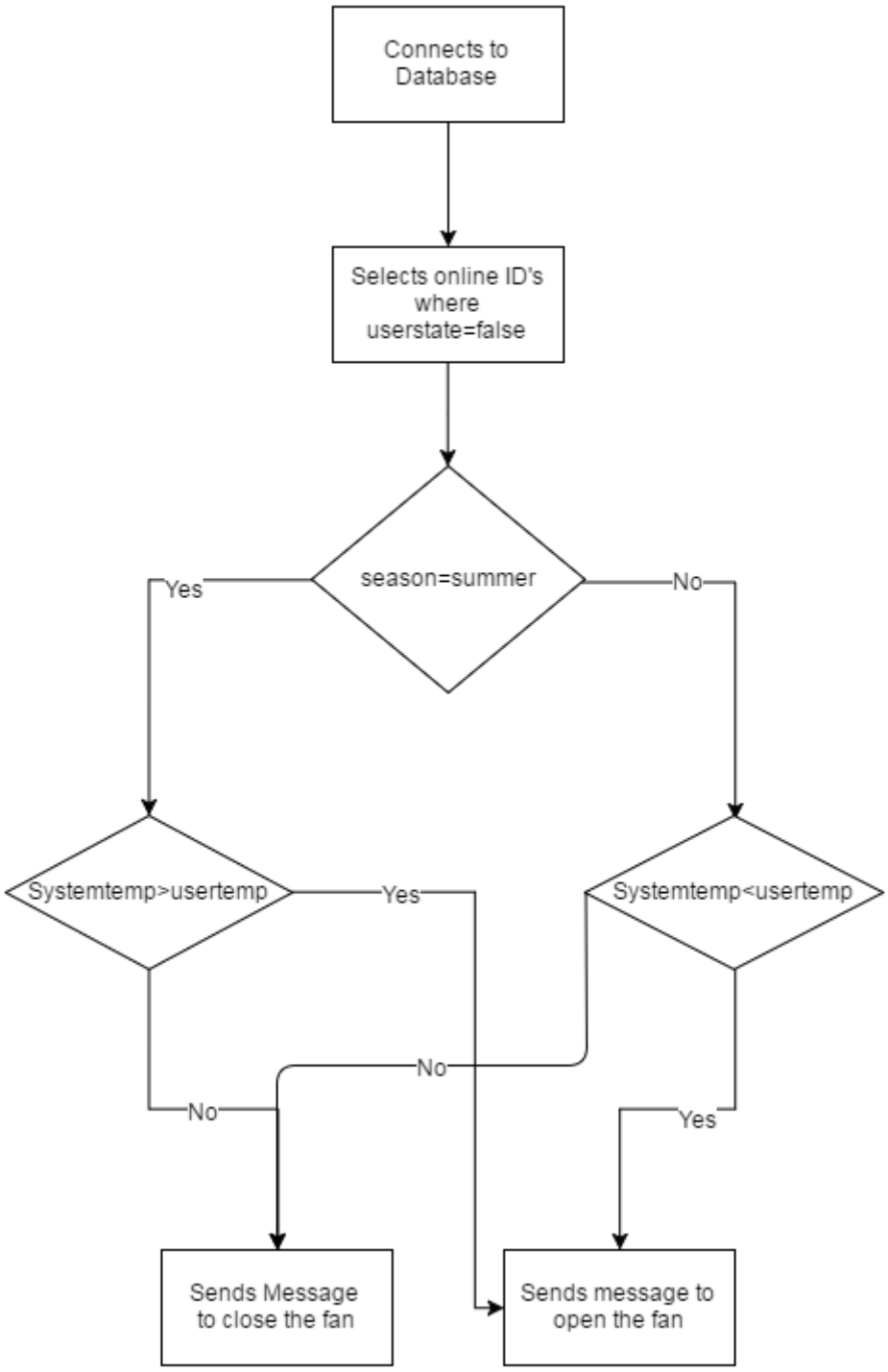
Αρχικά η διαδικασία συνδέεται στη βάση και επιλέγει όσους κόμβους έχουν κατάσταση online. Έπειτα ελέγχει το χρόνο που παραλήφθηκε τελευταία φορά μήνυμα από τον κόμβο αυτόν και αφαιρεί τη χρονική στιγμή από την ώρα που γίνεται ο έλεγχος. Εάν η διαφορά τους είναι μεγαλύτερη των δύο λεπτών τότε γίνεται αυτόματα ενημέρωση στη βάση και ο κόμβος είναι πλέον offline. Εάν η διαφορά τους είναι μικρότερη των δύο λεπτών τότε καλείται η διαδικασία TempCheck που είναι υπεύθυνη για τον έλεγχο λειτουργίας τον κόμβων. Τέλος καλείται η διαδικασία Conclients η οποία είναι υπεύθυνη για την ενημέρωση του διαχειριστή για τον αριθμό των ενεργών κόμβων.



Εικόνα 3.3.9 (Διάγραμμα ροής Check function)

- **TempCheck**

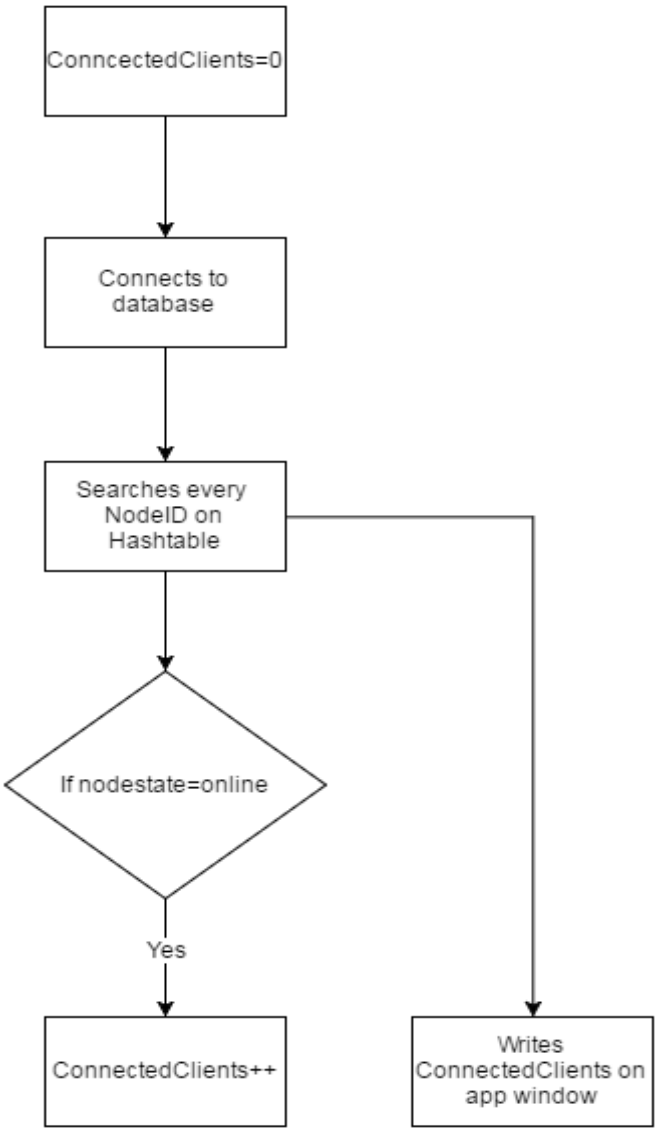
Η διαδικασία αυτή είναι υπεύθυνη για τον έλεγχο λειτουργίας των ενεργών κόμβων. Αρχικά γίνεται μια αναζήτηση στη βάση και επιλέγονται οι κόμβοι που η κατάσταση που έχει ζητηθεί από τον διαχειριστή για αυτές είναι false (να είναι ανοιχτός). Έπειτα γίνεται έλεγχος της τιμής season για τον κόμβο αυτόν και συγκρίνεται η θερμοκρασία του με τη θερμοκρασία που έχει ζητήσει ο χρήστης.



Εικόνα 3.3.10(Διάγραμμα ροής TempCheck function)

- **Conclients**

Η διαδικασία αυτή είναι η υπεύθυνη για την ενημέρωση του χρήστη σχετικά με τον αριθμό των ενεργών κόμβων στο σύστημα. Αρχικά η διαδικασία αυτή δημιουργεί μια σύνδεση με το βάση δεδομένων. Έπειτα ελέγχει ένα-ένα τα NodeID και ελέγχει την κατάσταση τους στη βάση. Αν είναι online αυξάνεται το πλήθος των κόμβων. Τέλος τυπώνει το πλήθος των κόμβων στο παράθυρο της εφαρμογής.



Εικόνα 3.3.11(Διάγραμμα ροής Conclients function)

Η εφαρμογή περιέχει επίσης τις:

- OnFormClosing
- OnResize
- Show_form_Click
- Exit_Click

οι οποίες είναι υπεύθυνες για τις ενέργειες του παραθύρου της εφαρμογής.

Αφού ολοκληρωθεί η αποσφαλμάτωση του κώδικα. Δημιουργήθηκε ένα νέο project μέσα στο ήδη υπάρχων το όποιο θα μας εξάγει ένα αρχείο setup.

ΚΕΦΑΛΑΙΟ IV

Εγκατάσταση

1. Εγκατάσταση

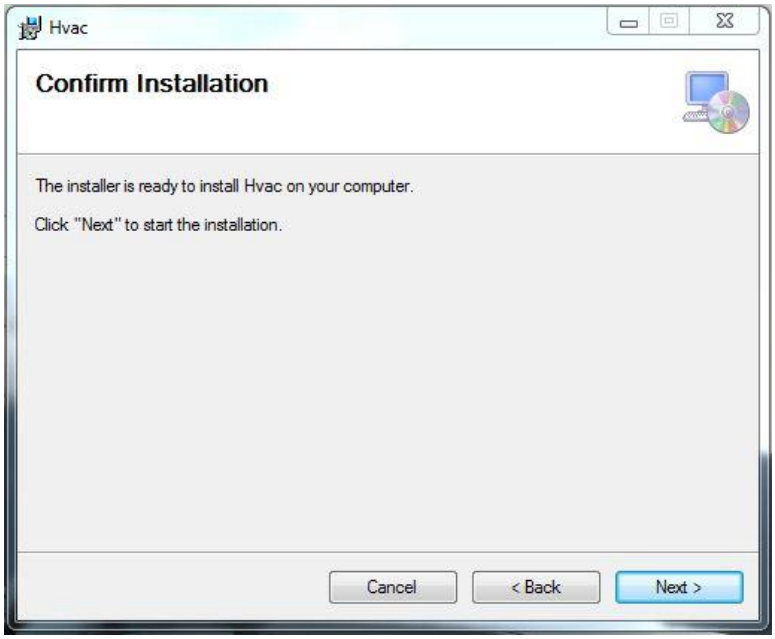
Για την σωστή εγκατάσταση του προγράμματος χρειάζεται το τερματικό να έχει λογισμικό Microsoft Windows Vista ή νεότερο καθώς και να έχει εγκατεστημένο το πακέτο .NetFramework 4.0 ή νεότερο.

Για να εκκινηθεί η εγκατάσταση κάνουμε διπλό κλικ στην το εικονίδιο setup.

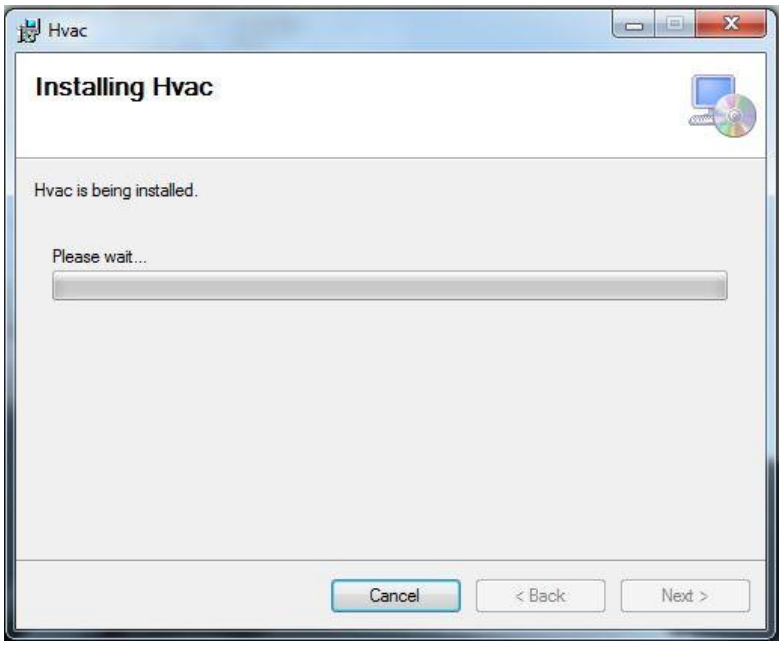
Αμέσως εκκινείται ο windows installer.

Εικόνα 4.1.1

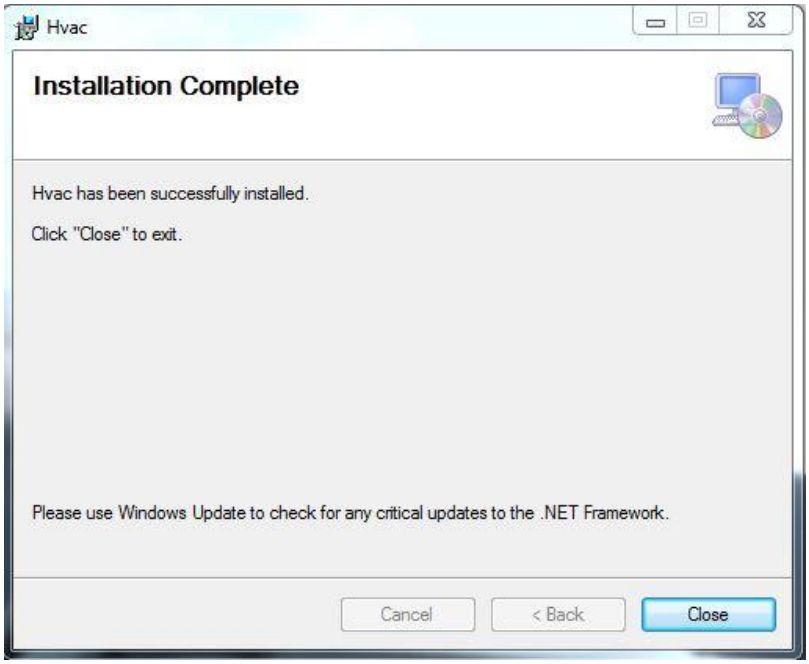
Εικόνα 4.1.2 (Επιλογή διαδρομής αποθήκευσης)



Εικόνα 4.1.3 (Επιβεβαίωση εγκατάστασης)



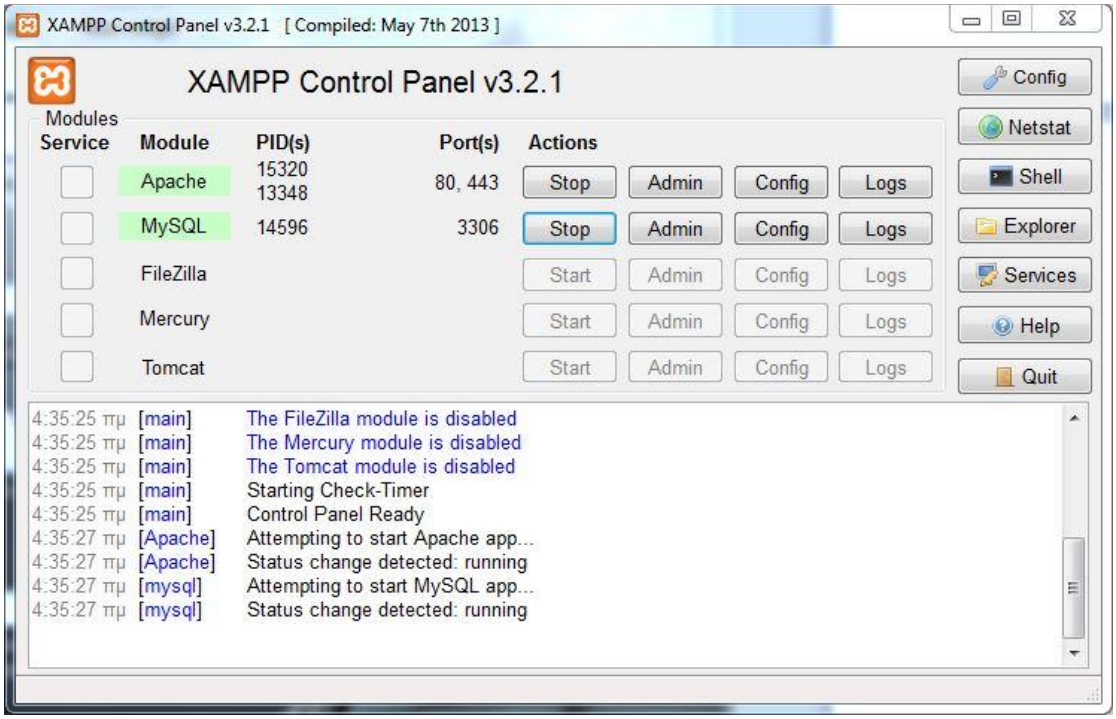
Εικόνα 4.1.4(Πρόοδος εγκατάστασης)



Εικόνα 4.1.5 Τέλος εγκατάστασης

2. Λειτουργία

Πριν γίνει εκκίνηση της εφαρμογής πρέπει να βεβαιωθούμε ότι υπάρχει Apache server ο οποίο υποστηρίζει MySQL βάση.

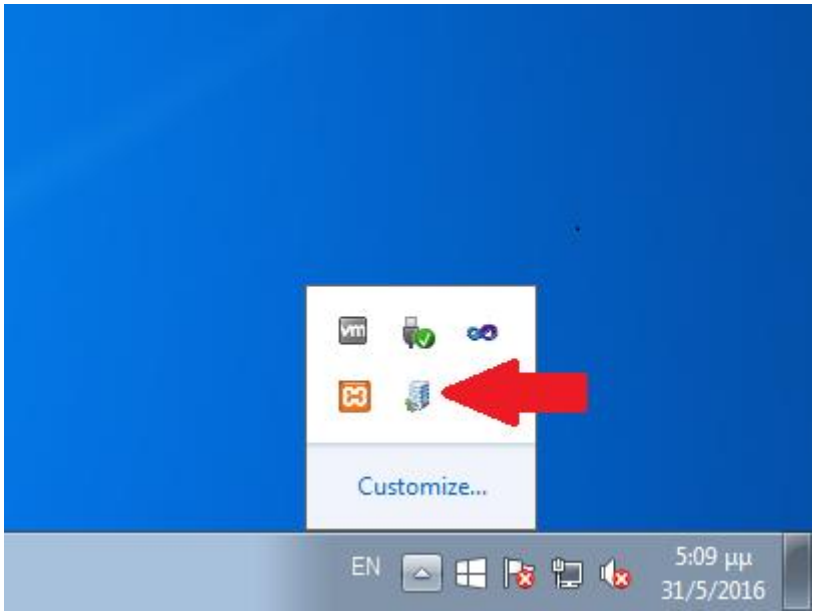


Εικόνα 4.2.1(Xammp interface)

Αφού βεβαιωθούμε ότι υπάρχουν τα προαπαιτούμενα πατάμε διπλά κλικ στο εικονίδιο που δημιουργήθηκε στην επιφάνεια εργασίας.

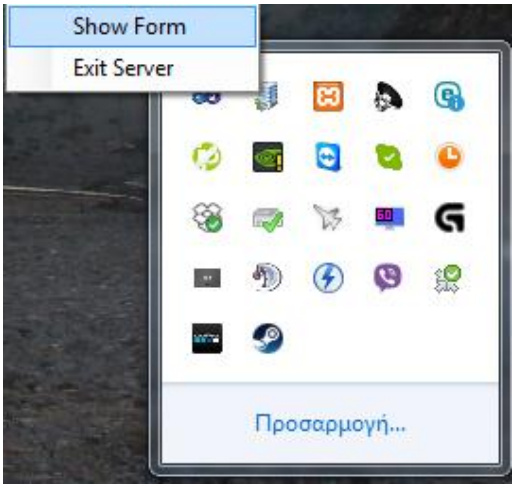
Εικόνα 4.2.2(Εικονίδιο Hvac Server)

Αμέσως ξεκινάει η λειτουργία του server σε tray mode. Πηγαίνοντας στα εικονίδια στην γραμμή εργαλείων θα δούμε το εικονίδιο της εφαρμογής.



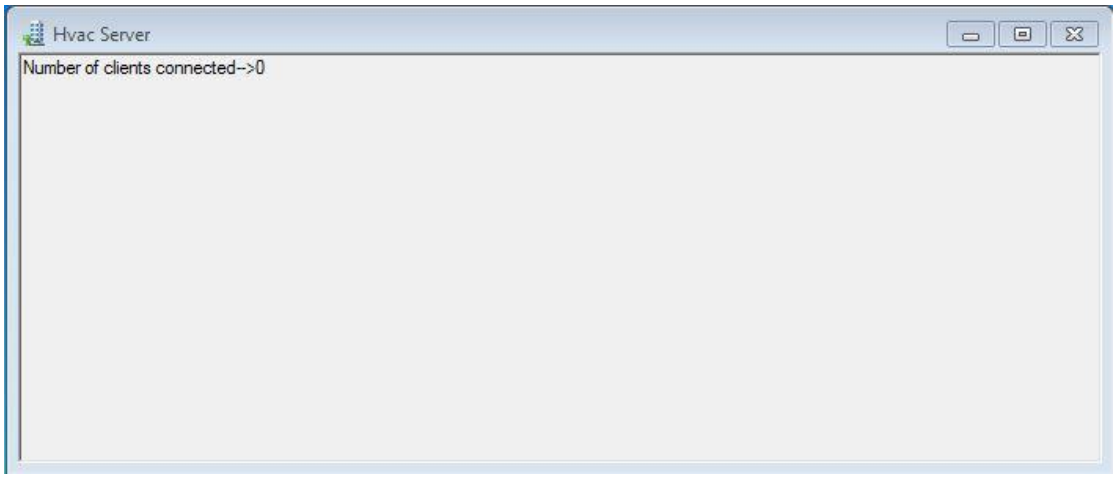
Εικόνα 4.4.3(Εικονίδιο σε tray)

Για να επαναφέρουμε την εφαρμογή σε πλήρες παράθυρο πατάμε δεξί κλικ πάνω στο εικονίδιο και επιλέγουμε την επιλογή “Show Form”



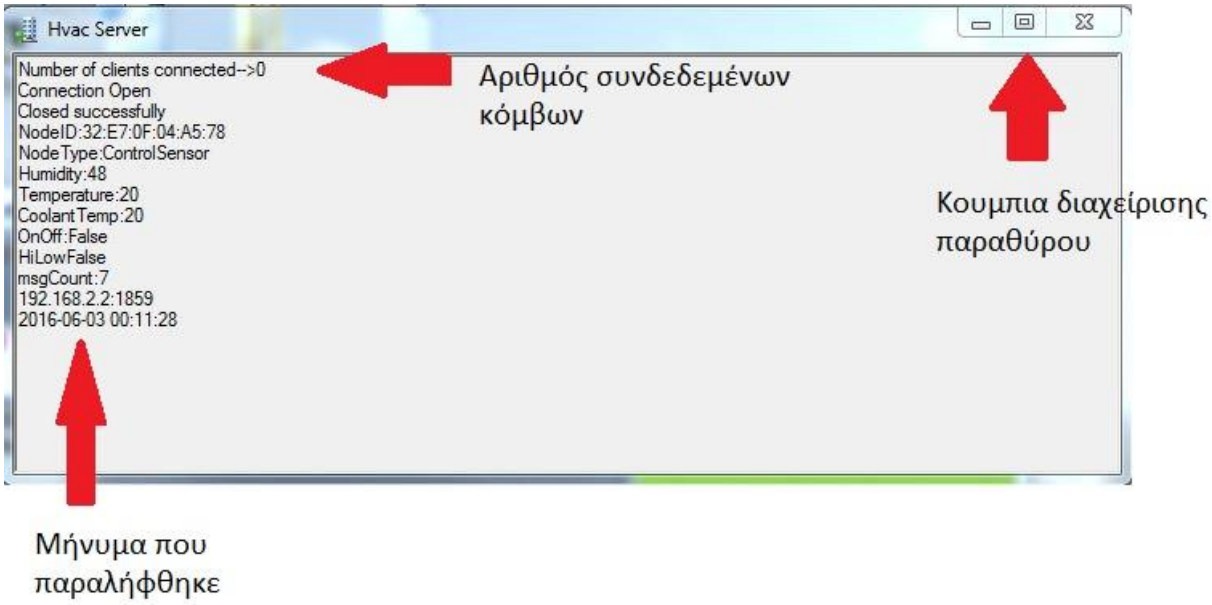
Εικόνα 4.4.4(Right Click Options)

Αμέσως ανοίγει ένα παράθυρο στο οποίο μπορούμε να δούμε τα μηνύματα που έρχονται στην εφαρμογή καθώς και τα μηνύματα που μας τυπώνει αυτή σχετικά με την κατάσταση των κόμβων.



Εικόνα 4.4.5(Application interface)

Τέλος ακολουθεί μια εικόνα του interface της εφαρμογής αφού έχει δεχθεί μήνυμα από έναν κόμβο.



Εικόνα 4.4.6(Application interface)

Αξίζει να σημειωθεί ότι όπως φαίνεται στην εικόνα δεν πρόλαβε να τρέξει η διαδικασία Check και γι αυτό ο αριθμός των κόμβων φαίνεται να είναι “0”.

Βιβλιογραφία

Κεφάλαιο 1ο

Internet of Things

https://en.wikipedia.org/wiki/Internet_of_things

http://postscapes.com/internet-of-things-history

http://www.forbes.com/sites/jacobmorgan/2014/05/13/simple-explanation-internet-things-that-anyone-can-understand/#7e3465246828

M2M

https://en.wikipedia.org/wiki/Machine_to_machine

PAN

https://en.wikipedia.org/wiki/Personal_area_network

Wearable technology

https://en.wikipedia.org/wiki/Wearable_technology

Κεφάλαιο 2ο

C#

https://msdn.microsoft.com/en-us/library/67ef8sbd.aspx

http://spectrum.ieee.org/computing/software/the-2015-top-ten-programming-languages

https://en.wikipedia.org/wiki/C_Sharp_(programming_language)

https://mathemagenesis.com/%CE%88%CE%BE%CE%B9-%CE%BB%CF%8C%CE%B3%CE%BF%CE%B9-%CE%B3%CE%B9%CE%B1-%CF%84%CE%BF%CF%85%CF%82-%CE%BF%CF%80%CE%BF%CE%AF%CE%BF%CF%85%CF%82-%CF%80%CF%81%CE%AD%CF%80%CE%B5%CE%B9-%CE%BD%CE%B1-%CE%BC%CE%AC%CE%B8%CE%B5%CF%84%CE%B5-c

http://studentguru.gr/w/tutorials/01-c

Microsoft Visual Studio

https://en.wikipedia.org/wiki/Microsoft_Visual_Studio

NuGet

https://en.wikipedia.org/wiki/NuGet

JSON

http://www.json.org/

MySQL

https://el.wikipedia.org/wiki/MySQL

https://en.wikipedia.org/wiki/MySQL

Fleck

http://www.codeproject.com/Articles/733297/WebSocket-libraries-comparison

Βιβλιογραφία Εικόνων

Εικόνα εξώφυλλου

http://thehackernews.com/2015/08/hacking-internet-of-things-drone.html

Κεφάλαιο 1ο

Εικόνα 1.1 Internet of Things, available at
http://osarena.net/sites/default/files/files/2015/05/31/iot-graphic.png

Εικόνα 1.2 Expectation of Internet of Things-Gartner,August2014, available at

http://www.forbes.com/sites/gilpress/2014/08/18/its-official-the-internet-of-things-takes-over-big-data-as-the-most-hyped-technology/#cb9e1d01aaa6

Κεφάλαιο 3ο

Εικόνα 3.2.1 JSON structure, available at

http://www.json.org/

Τα διαγράμματα ροής κατασκευάστηκαν με τη χρήση του

https://www.draw.io/

Application Icon, available at

http://www.iconarchive.com/show/vista-artistic-icons-by-awicons/office-building-icon.html

Πηγαίος κώδικας

```
/* HVAC Server/Message Broker
 * Description;
 * Creates a TCP server that connects on any IP of the terminal and
 * on Port 5020 for MCE nodes and a Web Socket server over TCP on same
 * IP and Port 5030 that is used for webpage to app connection.
 *
 *
 * Built from Ioannis Masklavanos,
 * Bachelor Thesis, TEIEP, November 2014, August 2015
 * Built with Microsoft Visual Studio 2010, C# Windows Form Application
 * with .Net Framework 4.0
 * Also used:
 * NuGet,
 * MySQL v.6.9.6.0 ,
 * Newtonsoft JSON v6.0.0.0,
 * Fleck v. 0.13.0.52
 * */
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Net;
using System.Net.Sockets;
using System.Text;
using System.Threading;
using System.Windows.Forms;
using Newtonsoft.Json;
using System.IO;
using Newtonsoft.Json.Linq;
using MySql.Data.MySqlClient;
using System.Collections;
using System.Security.Cryptography;
using System.Text.RegularExpressions;
using Fleck;
using Microsoft.Win32;

namespace form_mk1
{
    public partial class Form1 : Form
    {
        private TcpListener msplistener;  //socket listener for port 5020,
listening from msps
        private Thread listenthread; //Creates a new Thread for every new incomming
connection
        private delegate void writemessagedelegate(string msg); // Writes message
to servers textbox
        public static Hashtable node = new Hashtable();


        public Form1()
        {
            InitializeComponent();
            Server();
        }

        private void InsertUpdateClientSQL(string iID, string iHumidity, string
iTemp, string iTime, string iIP, string iPort, string iCoolantTemp, string iState,
string iHiLow)
        //Function that inserts/updates the values on client table
        {
            String str = @"server=localhost;database=hvac;userid=root;password=;";
//String that contains path for database
```

```
            MySqlConnection con = new MySqlConnection(str); //Creates a new MySQL
connection


            MySqlCommand cmd = new MySqlCommand("SELECT ID FROM clients WHERE
ID=@ID", con); //Creates new MySQL command
            con.Open(); //Opens the MySQL connection
            cmd.Parameters.AddWithValue("@ID", iID); //Command that connects the
@ID char with the ID value
            cmd.ExecuteNonQuery(); //Executes the command
            string insertQuery = "INSERT INTO
clients(ID,Temperature,Humidity,Time,IP,Port,Fluidtemp,State,Fan,Nodestate) VALUES
(@ID,@Temp,@Humidity,@Time,@IP,@Port,@Fluidtemp,@State,@Fan,@Nodestate)"; //String
used for inserting data
            string updateQuery = "UPDATE clients SET Temperature = @Temp , Humidity
= @Humidity , Time = @Time , IP = @IP , Port = @Port , Fluidtemp = @Fluidtemp ,
State = @State , Fan = @Fan , Nodestate = @Nodestate WHERE ID=@ID"; //String used
for updating data
            MySqlDataReader dr = cmd.ExecuteReader(); //Executes new MySQL command
to read from Data Base
            if (dr.Read()) //Checks if ID exists on base
            {
                dr.Close(); //Closes dr reader
                MySqlCommand update = new MySqlCommand(updateQuery, con); //Creates
command to update the data base
                update.Parameters.AddWithValue("@ID", iID); //Commands that connect
parameters with values
                update.Parameters.AddWithValue("@Temp", iTemp);
                update.Parameters.AddWithValue("@Humidity", iHumidity);
                update.Parameters.AddWithValue("@Time", iTime);
                update.Parameters.AddWithValue("@IP", iIP);
                update.Parameters.AddWithValue("@Port", iPort);
                update.Parameters.AddWithValue("@Fluidtemp", iCoolantTemp);
                update.Parameters.AddWithValue("@State", iState);
                update.Parameters.AddWithValue("@Fan", iHiLow);
                string stateofnode = "online"; // Creates string with value online
                update.Parameters.AddWithValue("@Nodestate", stateofnode); //Sets
Nodestate=online
                update.ExecuteNonQuery(); //Executes MySQL command
                con.Close(); //Closes MySQL connection
            }
            else
            {
                dr.Close(); //Closes dr reader
                MySqlCommand insert = new MySqlCommand(insertQuery, con); //Creates
command to insert data to the data base
                insert.Parameters.AddWithValue("@ID", iID); //Commands that connect
parameters with values
                insert.Parameters.AddWithValue("@Temp", iTemp);
                insert.Parameters.AddWithValue("@Humidity", iHumidity);
                insert.Parameters.AddWithValue("@Time", iTime);
                insert.Parameters.AddWithValue("@IP", iIP);
                insert.Parameters.AddWithValue("@Port", iPort);
                insert.Parameters.AddWithValue("@Fluidtemp", iCoolantTemp);
                insert.Parameters.AddWithValue("@State", iState);
                insert.Parameters.AddWithValue("@Fan", iHiLow);
                string stateofnode = "online";  // Creates string with value online
                insert.Parameters.AddWithValue("@Nodestate", stateofnode);  //Sets
Nodestate=online
                insert.ExecuteNonQuery();  //Executes MySQL command
                con.Close();     //Closes MySQL connection
            }
        }

        private void InsertTempSQL(string sID, string sHumidity, string sTemp,
string sTime, string sCoolantTemp, string sState, string sFan)
```

```
        {
            String str = @"server=localhost;database=hvac;userid=root;password=;";
//String that contains path for database
            MySqlConnection conn = new MySqlConnection(str);    //Creates a new
MySQL connection
            try
            {
                conn.Open(); //open the connection

                string insertSQL = "INSERT INTO
temperatures(ID,Temp,Humidity,Time,FluidTemp,State,Fan) VALUES (@ID, @Temp,
@Humidity, @Time, @FluidTemp, @State, @Fan)"; //String used for inserting data
                MySqlCommand comm = new MySqlCommand(insertSQL, conn);  //Creates
command to insert data to the data base
                comm.Parameters.AddWithValue("@ID", sID);    //Commands that
connect parameters with values
                comm.Parameters.AddWithValue("@Temp", sTemp);
                comm.Parameters.AddWithValue("@Humidity", sHumidity);
                comm.Parameters.AddWithValue("@Time", sTime);
                comm.Parameters.AddWithValue("@FluidTemp", sCoolantTemp);
                comm.Parameters.AddWithValue("@State", sState);
                comm.Parameters.AddWithValue("@Fan", sFan);
                comm.ExecuteNonQuery(); //Executes MySQL command
                conn.Close();    //Closes MySQL connection
            }
            catch (MySqlException err) //We will capture and display any MySql
errors that will occur
            {
                Writemessage("Error: " + err.ToString()); //Writes error to app
window
            }
            finally
            {
                if (conn != null)
                {
                    conn.Close(); //safely close the connection
                }
            }
        }   //Function that inserts the values to the temperatures table

        private void Server()
        {
            this.msplistener = new TcpListener(IPAddress.Any, 5020); //Creates TCP
Socket Listener for port 5020
            this.listenthread = new Thread(new ThreadStart(ListenForClients));
//Creates a new Thread for every new client
            this.listenthread.IsBackground = true; //Runs all threads on backround
            this.listenthread.Start();  //Starts the thread
            ListenFromWeb(); //Calls ListenFromWeb function
            Timer(); //Calls Timer Function
        } //Function that creates both the TcpListener and the WebSocket Server

        private void ListenFromWeb()
        {
            var server = new WebSocketServer("ws://0.0.0.0:5030"); //Listens to all
available ips on the pc and port 5030
            server.Start(socket =>
            {
                socket.OnMessage = (message) => //Gets the message from Web
                    {
                        try
                        {
                            var obj = JToken.Parse(message); //Checks if valid JSON
string
                            SendToClient(message); //Uses SentToClient function to
send message to MCE client
```

```
                            socket.Send("All good");  //Replies to Web that all
good
                        }
                        catch
                        {
                            Writemessage("Invalid Message received"); //Writes on
app window that invalid message was received
                            socket.Send("Invalid Message"); //Sends to Webpage that
invalid message was received
                        }
                        Writemessage(message); //Writes message on app window
                    };
            });

        }   //Function that establishes the websocket server and receives messages
from the webpage

        private void SendToClient(string message)
        {
            JObject o = JObject.Parse(message); //Parses message from JSON object
            string NodeID = (string)o["NodeID"]; //Gets values
            string Type = (string)o["Type"];
            string UserTemp = (string)o["Temperature"];
            string Userstate = (string)o["Userstate"];
            String str = @"server=localhost;database=hvac;userid=root;password=;";
//Database path sting
            MySqlConnection con = new MySqlConnection(str); //Creates a new MySQL
connection
            string UpdateStr = "UPDATE clients SET Usertemp=@UserTemp ,
Userstate=@Userstate WHERE ID=@NodeID";  //String used for updating data
            con.Open(); //Opens connection
            MySqlCommand update = new MySqlCommand(UpdateStr, con); //New MySQL
command
            update.Parameters.AddWithValue("@NodeID", NodeID); //Commands that
connect parameters with values
            update.Parameters.AddWithValue("@UserTemp", UserTemp);
            update.Parameters.AddWithValue("@Userstate", Userstate);
            update.ExecuteNonQuery(); //Executes MySQL command
            con.Close(); //Closes MySQL connection
            if (string.Equals(Type, "WebAll")) //Checks if message type= "WebAll"
            {
                Sleeptime(); //Calls Sleeptime
                string updateuserstatestr = "UPDATE clients SET
Userstate=@Userstate"; //String used for updating data
                string userstate = "True"; //String used to Update data on DataBase
                con.Open();//Opens connection
                MySqlCommand updateuserstate = new MySqlCommand(updateuserstatestr,
con);  //New MySQL command
                updateuserstate.Parameters.AddWithValue("@Userstate", userstate);
//Commands that connect parameters with values
                con.Close(); //Closes MySQL connection
            }
            else
            {
                Echo(NodeID, message); //Calls Echo function
            }
        }     //Function that saves the usertemp to the clients table and and roots
the message to specific client

        private void ListenForClients() //Function that handles every new
connection on port 5020
        {
            this.msplistener.Start(); //Starts the mpslistener
            while (true) //Never ends until the Server is closed
            {
```

```
                TcpClient client = this.msplistener.AcceptTcpClient(); //Waits for
a client to connect

                Thread clientThread = new Thread(new
ParameterizedThreadStart(HandleClientComm)); //Creates a new thread to handle
connection with the connected client
                clientThread.Start(client); // Start very new Thread
            }
        }
        private void HandleClientComm(object client)    //Function that controls
the communication with each client
        {
            TcpClient tcpClient = (TcpClient)client;    //Creates a new TCP client
object
            NetworkStream clientStream = tcpClient.GetStream(); //Creates a unique
NetworkStream for each client
            byte[] message = new byte[4096]; //States the max length of the message
            int bytesRead; //States the variable needed to save the message before
parsing
            while (true) //While there is something written in the byte variable
            {
                bytesRead = 0; //Initializes bytesRead variable before writing
                Array.Clear(message, 0, message.Length); //Cleans the message
variable before accepting a message
                try
                {
                    bytesRead = clientStream.Read(message, 0, 4096); //Blocks until
a clients sends a message
                }
                catch
                {
                    break; //Socket error
                }

                if (bytesRead == 0) //Client has disconnected from the server
                {
                    break;
                }
                //Message received
                ASCIIEncoding encoder = new ASCIIEncoding(); //States a new encoder
needed to decode the message
                string result = System.Text.Encoding.UTF8.GetString(message);
//Gets the encoded message
                string ipendpoint = tcpClient.Client.RemoteEndPoint.ToString();
//Gets the IP,Port from client
                try //Checks if message is valid JSON
                {
                    var obj = JToken.Parse(result); //Creates a new JSON object
with the message before parsing
                    JObject o = JObject.Parse(result); //Parses the message and
saves it to Object "o"
                    string NodeID = (string)o["NodeID"]; //Gets the NodeID from
decoded message
                    if (node.ContainsKey(NodeID)) //Checks if NodeID exists on the
Hashtable
                    {
                        node[NodeID] = clientStream; //If yes, updates the
NetworkStream
                    }
                    else
                    {
                        node.Add(NodeID, clientStream); //If not,add a new line
with NodeID and NetworkStream
                    }
                    string NodeType = (string)o["Type"]; //Gets type value and
saves it to the variable
```

```
                    string Humidity = (string)o["Humidity"]; //Gets Humidity value
and saves it to the variable
                    string Temperature = (string)o["Temperature"]; //Gets
Temperature value and saves it to the variable
                    string CoolantTemp = (string)o["CoolantTemp"]; //Gets type
Coolant temp value and saves it to the variable
                    string OnOff = (string)o["OnOff"]; //Gets state value and saves
it to the variable
                    string HiLow = (string)o["HiLow"]; //Gets fan value and saves
it to the variable
                    string msgCount = (string)o["msgCount"]; //Gets message count
value and saves it to the variable
                    if (NodeType == "ControlSensor") //Checks if NodeType ==
control sensor
                    {
                        StreamWriter sw = new StreamWriter(clientStream); //Creates
a new stream writer obj
                        sw.Write("{" + "\"Check\"" + ":" + "true" + "," +
"\"msgCount\"" + ":" + msgCount + "}"); // Replies with the msgCount that received
the message
                        sw.Flush(); //Sends message to node
                    }
                    string[] ipport = ipendpoint.Split(new char[] { ':' });
//Splits the ip/port variable into two parts
                    string ip = ipport[0]; //Gets the IP
                    string port = ipport[1]; //Gets the Port
                    DateTime datetime = System.DateTime.Now; //Gets system time
                    string MySQLFormatDate = datetime.ToString("yyyy-MM-dd
HH:mm:ss"); //Saves it to specific format
                    InsertTempSQL(NodeID, Humidity, Temperature,
MySQLFormatDate,CoolantTemp,OnOff,HiLow); //Calls InsertTempSQL function
                    InsertUpdateClientSQL(NodeID, Humidity, Temperature,
MySQLFormatDate, ip, port, CoolantTemp, OnOff, HiLow); //Calls
InsertUpdateClientSQL function
                    Writemessage("NodeID:" + NodeID); //Writes message to app
windows
                    Writemessage("NodeType:" + NodeType);
                    Writemessage("Humidity:" + Humidity);
                    Writemessage("Temperature:" + Temperature);
                    Writemessage("CoolantTemp:" + CoolantTemp);
                    Writemessage("OnOff:" + OnOff);
                    Writemessage("HiLow" + HiLow);
                    Writemessage("msgCount:" + msgCount);
                    Writemessage(ipendpoint);
                    Writemessage(MySQLFormatDate);
                }
                catch
                {
                    Writemessage("Invalid Message received from" + ipendpoint);
//If message not valid JSON writes it to app window
                }
            }

            tcpClient.Close(); //Closes tcpclient object
        }

        private void Timer()
        {
            var timer = new System.Threading.Timer(
            e => Check(), //Calls Check when 1min passes
            null,
            TimeSpan.Zero,
            TimeSpan.FromMinutes(1));

        }       //Sets the timer for Check method every x minutes
```

```
        private void Check()    //Function that checks the database for connected
clients
        {
            {
                String str =
@"server=localhost;database=hvac;userid=root;password=;"; //Database path sting
                MySqlConnection con = new MySqlConnection(str); //Creates a new
MySQL connection
                MySqlConnection conn = new MySqlConnection(str); //Creates a new
MySQL connection

                MySqlCommand cmd = new MySqlCommand("SELECT ID,Time FROM clients
WHERE Nodestate LIKE 'online'", con); //New MySQL command
                con.Open(); //Opens connection
                cmd.ExecuteNonQuery(); //Executes MySQL command
                MySqlDataReader reader = cmd.ExecuteReader();
                {
                    while (reader.Read())
                    {
                        var ID = reader["ID"]; //Gets ID from DataBase
                        var Time = reader["Time"]; //Gets Time From DataBase
                        DateTime datetime = System.DateTime.Now; //Gets system time
                        DateTime myDate = DateTime.Parse(Time.ToString()); //Parses
Time variable to DateTime variable
                        TimeSpan diff = datetime.Subtract(myDate); //Substracts
Time from datetime
                        double diffmins = diff.TotalMinutes; //Gives the result of
substraction

                        if (diffmins > 2) //Checks if difference more that 2 mins
                        {
                            conn.Open(); //Opens MySQL connection
                            string offupdate = "UPDATE Clients SET Nodestate =
@Nodestate WHERE ID=@ID";//String used for updating data
                            MySqlCommand updateoff = new MySqlCommand(offupdate,
conn); //New MySQL command
                            updateoff.Parameters.AddWithValue("@ID", ID);
//Commands that connect parameters with values
                            updateoff.Parameters.AddWithValue("@Nodestate",
"offline");
                            updateoff.ExecuteNonQuery(); //Executes MySQL command
                            conn.Close(); //Closes MySQL connection
                        }
                        else
                        {
                            TempCheck(ID.ToString());    //Calls tempcheck to check
temp
                        }
                    }
                    reader.Close();
                    con.Close();
                }
            }
            Conclients();   //Call conclients to check online clients
        }

        private void TempCheck(string ID)    //Function that checks usertemp and
systemtemp
        {
            String str = @"server=localhost;database=hvac;userid=root;password=;";
//Database path sting
            MySqlConnection con = new MySqlConnection(str); //Creates a new MySQL
connection
            MySqlCommand cmd = new MySqlCommand("SELECT
Temperature,Usertemp,Season,Userstate FROM clients WHERE ID=@ID", con);  //New
MySQL command
```

```
            con.Open(); //Opens MySQL connection
            cmd.Parameters.AddWithValue("@ID", ID); //Commands that connect
parameters with values
            cmd.ExecuteNonQuery(); //Executes MySQL command
            MySqlDataReader check_reader = cmd.ExecuteReader(); //Executes MySQL
command
            while (check_reader.Read())
            {
                var season = check_reader["Season"];  //Gets values
                var systemtemp = check_reader["Temperature"];
                var usertemp = check_reader["Usertemp"];
                var userstate = check_reader["Userstate"];
                string seasonstr = season.ToString();
                if (userstate.ToString() == "False") //Checks if userstate=false
User->true
                {
                    if (seasonstr == "summer") //Checks if season=summer
                    {
                        if (Int32.Parse(systemtemp.ToString()) >
(Int32.Parse(usertemp.ToString()))) //Compares temperatures
                        {
                            string summer_onclosestr =
"{\"OnOff\":false,\"HiLow\":true}"; //OnOff == State
                            Echo(ID, summer_onclosestr); //Sends message
                        }
                        else
                        {
                            string summer_offclosestr =
"{\"OnOff\":true,\"HiLow\":true}";
                            Echo(ID, summer_offclosestr); //Sends message
                        }
                    }
                    else
                    {
                        if (Int32.Parse(systemtemp.ToString()) <
(Int32.Parse(usertemp.ToString()))) //Compares temperatures
                        {
                            string winter_onclosestr =
"{\"OnOff\":false,\"HiLow\":true}";
                            Echo(ID, winter_onclosestr); //Sends message
                        }
                        else
                        {
                            string winter_offclosestr =
"{\"OnOff\":true,\"HiLow\":true}";
                            Echo(ID, winter_offclosestr); //Sends message
                        }
                    }
                }
            }
            check_reader.Close(); //Closes MySQL command
            con.Close(); //Closes MySQL connection
        }

        private void Writemessage(string msg)       //Function that displays
message to richtextbox
        {
            if (this.rtbServer.InvokeRequired)
            {
                writemessagedelegate d = new writemessagedelegate(Writemessage);
                this.rtbServer.Invoke(d, new object[] { msg });
            }
            else
            {
                this.rtbServer.AppendText(msg + Environment.NewLine);
            }
```

```
        }

        private void Echo(string nodeid, string msg)   //Function that sends the
message to specific client
        {
            if (node.ContainsKey(nodeid)) //Checks Hashtable is NodeID exists
            {
                try
                {
                    NetworkStream clientStream = (NetworkStream)node[nodeid];
//Creates a new NetworkStream
                    ASCIIEncoding encoder = new ASCIIEncoding();

                    StreamWriter sw = new StreamWriter(clientStream); //Prepares
the Streamwriter
                    sw.Write(msg); //Gives the streamWriter the value of the
message
                    sw.Flush(); //Sends the message
                }
                catch (Exception e)
                {
                    Writemessage(e.ToString()); //Writes exception to the app
window
                }
            }
            else
            {
                Writemessage("Node not found"); //Writes on app windows that node
is not found
            }
        }

        private void Conclients() //Function that checks for the number of
connected clients
        {
            int conclients = 0; //Initializes conclients variable
            String str = @"server=localhost;database=hvac;userid=root;password=;";
//Database path sting
            MySqlConnection con = new MySqlConnection(str); //Creates a new MySQL
connection

            foreach (DictionaryEntry nid in node) //Checks every NodeID entry in
Hashtable
            {
                MySqlCommand cmd = new MySqlCommand("SELECT Nodestate FROM clients
WHERE ID = @ID", con);  //New MySQL command
                con.Open(); //Opens MySQL connection
                cmd.Parameters.AddWithValue("@ID", nid.Key); //Commands that
connect parameters with values
                cmd.ExecuteNonQuery(); //Executes MySQL command
                MySqlDataReader reader = cmd.ExecuteReader();
                {
                    while (reader.Read()) //Reads from DataBase
                    {
                        var nodestate = reader["Nodestate"];
                        if (nodestate.ToString() == "online") //Checks if
nodestate=online
                        {
                            conclients++; //Increases conclients number
                        }
                    }
                }
                reader.Close(); //Closes MySQL Reader
                con.Close(); //Closes MySQL Connection
            }
```

```
            Writemessage("Number of clients connected-->" + conclients); //Writes
conclients to app window
        }
        private void Sleeptime()    //Function that closes all connected clients
        {
            string goodnight = "{\"OnOff\":true,\"HiLow\":true}"; //String that
closes all nodes
            foreach (DictionaryEntry nid in node) //Checks all entries on Hashtable
            {
                NetworkStream clStream = (NetworkStream)node[nid.Key]; //Creates
new NetworkStream
                StreamWriter sw = new StreamWriter(clStream); //Creates new
StreamWriter
                sw.Write(goodnight); //Writer get the value of string
                sw.Flush(); //Sends message
            }
        }
        protected override void OnFormClosing(FormClosingEventArgs e)    //Fuction
that control the actions of "x" button
        {
            base.OnFormClosing(e);
            if (e.CloseReason == CloseReason.WindowsShutDown) return;
            // Confirm user wants to close
            switch (MessageBox.Show(this, "Are you sure you want to close?",
"Closing", MessageBoxButtons.YesNo))
            {
                case DialogResult.No:
                    e.Cancel = true;
                    break;
                default:
                    listenthread.IsBackground = false;
                    Environment.Exit(0);
                    break;
            }
        }
        protected override void OnResize(EventArgs e)
        {
            base.OnResize(e);
            bool cursorNotInBar =
Screen.GetWorkingArea(this).Contains(Cursor.Position);
            if (this.WindowState == FormWindowState.Minimized && cursorNotInBar)
            {
                this.ShowInTaskbar = false;
                notifyIcon1.Visible = true;
                this.Hide();
            }
        } //Function that controls the application window after resize button is
clicked
        private void Show_Form_Click(object sender, EventArgs e)
        {
            this.WindowState = FormWindowState.Normal;
            this.ShowInTaskbar = true;
            notifyIcon1.Visible = false;
            this.Visible = true;
        } //Function that controls the application after right clicked on tray mode

        private void Exit_Click(object sender, EventArgs e)
        {
            listenthread.IsBackground = false;
            Environment.Exit(0);
            Close();
        } //Function that closes the application after "x" button is clicked

    }
}
```