ΤΕΙ ΠΕΙΡΑΙΑ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Smart Home (Έξυπνο Σπίτι)

Κωνσταντίνος Ν. Αρκουλής

ΑΜ:39450

Επιβλέπων Καθηγητής

Ιωάννης Ν. Έλληνας, Καθηγητής

(Κενό φύλλο)

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Smart home (Έξυπνο Σπίτι)

Κωνσταντίνος Ν. Αρκουλής
Α.Μ. 39450

Εισηγητής:
Ιωάννης Ν Έλληνας, Καθηγητής

Εξεταστική Επιτροπή:

_____

_____

_____

Ημερομηνία εξέτασης

_____

(Κενό φύλλο)

ΕΥΧΑΡΙΣΤΙΕΣ

Στο σημείο αυτό θα ήθελα να ευχαριστήσω τον υπεύθυνο καθηγητή κ. Ιωάννη Έλληνα για την πολύτιμη βοήθεια και πρόθυμη υποστήριξή του καθ' όλη τη διάρκεια εκπόνησης της εργασίας. Εγκάρδιες ευχαριστίες αξίζουν και στον κ. Νίκο Αδάμ, τον ηλεκτρολόγο του σπιτιού προς ανακαίνιση, ο οποίος μου έδωσε την ευκαιρία για την υλοποίηση της εγκατάσταση της εφαρμογής

(Κενό φύλλο)

ΠΕΡΙΛΗΨΗ

Η παρούσα πτυχιακή εργασία ασχολείται με την ανάπτυξη και υλοποίηση ενός «Έξυπνου σπιτιού». Ο κύριος σκοπός της είναι η περιγραφή της εγκατάστασης και του τρόπου λειτουργίας ενός συστήματος για την εξ αποστάσεως διαχείριση των ηλεκτρικών συσκευών ενός σπιτιού με στόχο τον εύκολο και άμεσο χειρισμό τους.

Η εργασία χωρίζεται σε δύο μέρη, με το πρώτο να είναι το θεωρητικό. Ειδικότερα, στο πρώτο μέρος παρουσιάζονται οι τεχνολογίες που χρησιμοποιήθηκαν για την ανάπτυξη των εφαρμογών, καθώς και η συγκεκριμένη εφαρμογή που αναπτύχθηκε για την εγκατάσταση του συστήματος. Η κατασκευή της μακέτας ενός σπιτιού για την επίδειξη της εφαρμογής, σε συνδυασμό με την εγκατάσταση του συστήματος σε ένα πραγματικό σπίτι, το οποίο ήταν προς ανακαίνιση, αποτελούν το πρακτικό μέρος της εργασίας.

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: Αυτοματισμός, Arduino, Smart Home, Remote Control, Home Automation, Home Control

Abstract

The present thesis examines the development and the materialization of a “Smart House”. Its main goal is to describe the installation process and the function of a remote management system of the electronic appliances of a house, which aims to achieve their easy and direct management. This thesis consists of two parts, the firsts is the theoretical part. Specifically, in the first part we present the technologies used in the software development, as well as the very specific application developed for the installation of the remote electronic appliances management system. The construction of a scale model of a house, for the purpose of displaying the software, in combination with the installation of the system in a real house, which was being renovated, consist the practical part of the thesis.

Περιεχόμενα

Κατάλογος εικόνων

Κατάλογος Πινάκων

ΣΥΝΤΟΜΟΓΡΑΦΙΕΣ

PDA: Personal Digital Assistant
DVD: Digital Versatile Disc
IQ: Intelligence Quotient
HTTP: HyperText Transfer Protocol
CSS: Cascading Style Sheets
AJAX: Asynchronous JavaScript And XML
XML: eXtensive Markup Language
RSS: Rich Site Summary
SOAP: Simple Object Access protocol
XHTML: EXtensible HyperText Markup Language
W3C: World Wide Web Consortium
EBNF: Extended Backus–Naur Form
SGML: Standard Generalized Markup Language
AVR: Alf (Egil Bogen) and Vegard (Wollan)'s RISC processor ATMEGA368:
UART: Universal Asynchronous Receiver/Transmitter
PWM: Pulse width Modulation
ICSP: In Circuit Serial Programming
USB: Universal Serial Bus
SRAM: Static Random Access Memory
EEPROM: Electrically Erasable Programmable Read-Only Memory
GND: GrouND
IOREF: Input Output REFerence
FTDI: Future Technology Devices International
TTL: Transistor Transistor Logic
SPI: Serial Peripheral Interface
LED: Light Emitting Diode
I2C: Inter Integrated Circuit
SDA: Serial Data Line
SCL: Serial Clock Line
AREF: Analog reference
IDE: Integrated Development Environment
ADC: Analog to Digital Converter
DAC: Digital to Analog Converter
ADMUX: Analog to Digital Converter Multiplexer Selection Register
ADCSRA: Analog to Digital Converter Control and Status Register A
LSB: Least Significant Bit
MSB: Most Significant Bit
TCP: Transmission Control Protocol
UDP: User Datagram Protocol
IP: Internet Protocol
SD: Secure Digital
MAC: Media Access Control
DNS: Domain Name System
DHCP: Dynamic Host Configuration Protocol

INT: INTeger
DEC: ECimal
OCT: OCTal numeral
BIN: BINary
HEX: HEXadecimal
GSM: Global System for Mobile communications
LDR: Light Dependent Resistor
DIV: DIVision
EMF: LlectroMagnetic field

Κεφάλαιο 1: Εισαγωγή

Τη σημερινή εποχή οι γοργοί ρυθμοί εξέλιξης της τεχνολογίας έχουν μεγάλο αντίκτυπο στην καθημερινή ζωή των ανθρώπων σε επίπεδο δυνατοτήτων και ευκολιών. Η ευχέρεια στην πρόσβαση πληροφοριών παντός τύπου και μορφής είναι κάτι που χαρακτηρίζει τις σημερινές μέρες, λόγω της ευρείας διάδοσης του διαδικτύου. Ως αποτέλεσμα των ανωτέρω είναι και η συνεχής ανάπτυξη νέων τάσεων και εφαρμογών.

Συγκεκριμένα τα τελευταία χρόνια παρατηρείται η ανάπτυξη ενός κλάδου που αναφέρεται βιβλιογραφικά με τον όρο διαδίκτυο των πραγμάτων (Internet of Things). Αυτός ο κλάδος περιγράφει την διασύνδεση υλικών οντοτήτων και αντικειμένων σε μια μορφή διαδικτύου έτσι ώστε να παρέχουν στον χρήστη πληροφορίες είτε για την κατάσταση της λειτουργία τους, είτε για διάφορα γεγονότα που έχουν αξία, είτε για να παρέχεται η δυνατότητα ενεργειών πάνω σε αυτά ανεξάρτητα από την φυσική τοποθεσία του χρήστη. Σημαντικό κομμάτι αυτού του κλάδου βασίζεται στις τεχνολογίες δικτύων ασύρματων κόμβων αισθητήρων. Τα δίκτυα αυτά αποτελούνται από μικρούς σε διαστάσεις κόμβους, σχεδιασμένους με έμφαση στην χαμηλή κατανάλωση για μεγάλη αυτονομία λειτουργίας, οι οποίοι παρέχουν πληροφορίες σχετικά με διάφορα φυσικά φαινόμενα τα οποία δειγματοληπτούν μέσω μιας μεγάλης ποικιλίας αισθητήρων που έχουν αναπτυχθεί για το σκοπό αυτό.

Ο συνδυασμός αυτών των τεχνολογιών μαζί με την ραγδαία ανάπτυξη των έξυπνων κινητών τηλεφώνων (smartphones) και κυρίως των δυνατοτήτων τους όπως οι πολλαπλές διεπαφές συνδεσιμότητας ανά κινητό τερματικό και η αύξηση της επεξεργαστικής ισχύος αυτών, δίνει στον τελικό χρήστη την ευελιξία ελέγχου και καθορισμού των παραμέτρων απομακρυσμένων αντικείμενων με βάση τις προσωπικές επιθυμίες και προτιμήσεις.

1.1. Τι είναι το «Έξυπνο σπίτι»;

Ως «Έξυπνο σπίτι» μπορούμε να θεωρήσουμε οποιοδήποτε προσωπικό ή εργασιακό περιβάλλον που περικλείει ένα σύνολο τεχνολογικών εφαρμογών με κύριο χαρακτηριστικό την αυτοματοποίηση και τον έλεγχο των επιμέρους τμημάτων του.

Ο βαθμός αυτοματοποίησης, καθώς επίσης και ο βαθμός και ο τρόπος ελέγχου ποικίλουν, αφού εξαρτώνται από πολλές παραμέτρους. Μερικές από αυτές τις παραμέτρους μπορεί να είναι το κόστος, οι προσωπικές επιθυμίες του χρήστη, το είδος των συσκευών που πρόκειται να ελεγχθούν και ο τύπος του κτιρίου στο οποίο θα εγκατασταθεί η τεχνολογία.

Το «Έξυπνο σπίτι» μάς παρέχει τη δυνατότητα να ζούμε και να εργαζόμαστε σε απλοποιημένο και αναβαθμισμένο περιβάλλον, εξασφαλίζοντας τη μείωση των πάγιων εξόδων. Με την πάροδο του χρόνου, το κόστος των ηλεκτρικών και ηλεκτρονικών συσκευών και συστημάτων μειώνεται, έτσι ώστε οι αυτοματισμοί οικίας ή το «Έξυπνο

σπίτι» να αξιοποιούνται ευρύτερα. Γενικά η εύκολη εγκατάσταση, η ελάχιστη συντήρηση, η αθόρυβη λειτουργία, η εξοικονόμηση ενέργειας, η άνεση και ο προσωπικός έλεγχος επί του οικιακού περιβάλλοντος από απόσταση αποτελούν τα κυριότερα πλεονεκτήματα της αυτοματοποίηση. Η τεχνολογία των «Έξυπνων σπιτιών», που συμβάλλει στην απλοποίηση της καθημερινής ζωής των χρηστών, πρέπει να πληροί ορισμένες προϋποθέσεις, όπως:

- τη διασφάλιση της ανθρώπινης ζωής και περιουσίας,
- τη γνώση για τη διαχείριση της τεχνολογίας και
- την επικοινωνία με το εξωτερικό περιβάλλον.

Ένα βασικό χαρακτηριστικό των «Έξυπνων σπιτιών» είναι ότι οι ίδιες περιφερειακές μονάδες χρησιμοποιούνται για πολλές χρήσεις. Παράδειγμα, οι αισθητήρες παρουσίας που μπορούν να χρησιμοποιηθούν για τον έλεγχο του φωτισμού και του συστήματος θέρμανσης, χρησιμεύουν ταυτόχρονα και για το σύστημα του συναγερμού. Ένα άλλο παράδειγμα αφορά στις οθόνες των τηλεοράσεων, οι οποίες μπορούν να προβάλλουν και την εικόνα της θυροτηλεόρασης.

Τα σύγχρονα συστήματα που εφαρμόζονται στις «Έξυπνες κατοικίες» προσφέρουν στους ενοίκους πάρα πολλές διευκολύνσεις και συμβάλλουν στην εξοικονόμηση πολύτιμου χρόνου. Οι παρεχόμενες διευκολύνσεις πολλαπλασιάζονται καθώς, εκτός από τις βασικές λειτουργίες, δίνεται επίσης η δυνατότητα στον ιδιοκτήτη να προγραμματίσει το σύστημα και να δημιουργήσει δικά του σενάρια, προκειμένου να καλύψει πλήρως τις δικές του εξατομικευμένες ανάγκες. Τα σενάρια που μπορούν να εφαρμοστούν είναι πρακτικά άπειρα. Ορισμένα παραδείγματα, όσον αφορά στις συνήθεις λειτουργίες των έξυπνων σπιτιών, παρουσιάζονται ενδεικτικά παρακάτω.

1. Φωτισμός: Αυτοματοποιημένος φωτισμός που αυξομειώνεται κατά τη διάρκεια της μέρας, προκατασκευασμένα σενάρια φωτισμού, αυτόματη ενεργοποίηση και απενεργοποίηση των φώτων κτλ.

2. Ασφάλεια: Προστασία από βραχυκυκλώματα, πλημμύρες, πυρκαγιά ή οποιαδήποτε βλάβη. Σε αυτήν την περίπτωση, το «Έξυπνο σπίτι» λειτουργεί σαν σύνολο συναγερμών.

3. Έλεγχος θέρμανσης, κλιματισμού, αερισμού: Δυνατότητα να ρυθμιστεί εκ των προτέρων ή εξ αποστάσεως η επιθυμητή θερμοκρασία για την κατοικία. Αντίστοιχες ρυθμίσεις μπορούν να πραγματοποιηθούν και για τον κλιματισμό – αερισμό, ενώ υπάρχει επίσης δυνατότητα αυτόματης ενεργοποίησης του συστήματος εξαερισμού σε περίπτωση υψηλής συγκέντρωσης αερίων ή καπνού στο χώρο. Επιπλέον, η θέρμανση μπορεί να κλείνει αν υπάρξει ανοιχτό παράθυρο ή όποτε θεωρείται περιττή.

4. Έλεγχος ηλεκτρικών περσίδων και τεντών: Θα μπορούσε να εντάσσεται στην προηγούμενη ενότητα, ωστόσο αποτελεί αυτοτελές τμήμα των συστημάτων ελέγχου ενός σπιτιού. Οι τέντες, οι περσίδες και τα παράθυρα μπορούν να ανοιγοκλείνουν ανάλογα με τη θερμοκρασία, το φως, ακόμα και τον αέρα, ρυθμίζοντας απόλυτα τις συνθήκες διαβίωσης.

5. Πολυμέσα: Δυνατότητα διασύνδεσης τηλεοπτικών συσκευών, ηχοσυστημάτων, τηλεφωνικών συσκευών είτε μεταξύ τους, είτε με άλλες συσκευές σε όλο το σπίτι.

Συμπερασματικά, στο αυτοποιημένο περιβάλλον ενός «Έξυπνου σπιτιού» όλα τα επιλεγόμενα συστήματα μπορούν να λειτουργήσουν αρμονικά μεταξύ τους, ρυθμίζοντας αυτόματα τις επιθυμητές συνθήκες διαβίωσης και προσφέροντας ένα άνετο, εύχρηστο και λειτουργικό οικιακό περιβάλλον. Το βασικότερο, ωστόσο, είναι ότι μπορούμε να ελέγξουμε πλήρως την κατανάλωση ενέργειας, εξοικονομώντας το μέγιστο, εφόσον η λειτουργία των συσκευών, της θέρμανσης, του κλιματισμού και του φωτισμού καθορίζεται αποκλειστικά απ΄ το χρόνο που μας είναι αναγκαία.

1.2. Περιγραφή ενός «Έξυπνου σπιτιού»

Ένα «Έξυπνο σπίτι» ελέγχει όλες του τις λειτουργίες, εντός και εκτός σπιτιού, από ένα μόνο σημείο διαχείρισης. Με τη χρήση της τεχνολογίας ο ιδιοκτήτης ενός σπιτιού μπορεί να το μετατρέψει σε ένα «Έξυπνο σπίτι», το οποίο αναγνωρίζει ταχύτατα τις ανάγκες του και προσαρμόζεται αναλόγως αλλά και σε ένα πλήρως ελεγχόμενο σπίτι το οποίο μπορεί να διαχειρίζεται από οποιοδήποτε σημείο του κόσμου.

Ένας κεντρικός υπολογιστής (Server) ενώνει σε ένα δίκτυο συσκευές από όλα τα δωμάτια του σπιτιού, ελέγχει τη λειτουργία τους, δίνει εντολές και διευκολύνει τους ενοίκους να πραγματοποιούν δραστηριότητες κάθε είδους, περισσότερο ή λιγότερο σημαντικές. Αυτό σημαίνει ότι ελέγχει όλα τα συστήματα κλιματισμού, φωτισμού, ασφαλείας κ.ά. Ο ιδιοκτήτης μπορεί να έχει πρόσβαση στον συγκεκριμένο Server μέσω διαδικτύου από το κινητό, την κεντρική μονάδα ή τον προσωπικό ψηφιακό βοηθό (PDA).

Παρακάτω αναφέρονται μερικές από τις λειτουργίες ενός «Έξυπνου σπιτιού»:

- Οι κάμερες των συστημάτων ασφαλείας τίθενται σε λειτουργία από το γραφείο ή το δρόμο.
- Στην κουζίνα υπάρχει μία οθόνη που λειτουργεί με το άγγιγμα, μέσω της οποίας ελέγχονται όλες οι ηλεκτρικές συσκευές.
- Το ψυγείο ενημερώνεται (μέσω bar code των προϊόντων) για τις ελλείψεις.
- Με ένα τηλεχειριστήριο ο ιδιοκτήτης ελέγχει τη λειτουργία της τηλεόρασης, του στερεοφωνικού, του DVD κ.ά.
- Ένα ρολόι στέλνει στην καφετιέρα σήμα να αρχίσει να φτιάχνει καφέ.
- Όταν κάποιος σηκώνεται τη νύχτα από το κρεβάτι, το σύστημα αυτοματισμού θα ανάψει ήπια το φωτισμό του δωματίου. Παράλληλα θα ανάψει το φως του διαδρόμου και του μπάνιου. Όταν επιστρέψει στο δωμάτιο και σβήσει το φως, το σύστημα θα σβήσει αυτόματα.
- Στα δωμάτια υπάρχουν οθόνες αφής που ελέγχουν συσκευές σε άλλα δωμάτια, όπως το πλυντήριο ρούχων ή πιάτων.
- Τα φώτα μέσα στο σπίτι και έξω από αυτό ανάβουν στην επιθυμητή ένταση και σβήνουν από απόσταση.
- Ο ένοικος μπορεί να διακόψει την παροχή ρεύματος σε κάποιες ή όλες τις πρίζες, προκειμένου να προστατεύσει άλλα άτομα (π.χ. μικρά παιδιά) από κίνδυνο ηλεκτροπληξίας.

- Η κατοικία διαθέτει τις βασικές λειτουργίες συναγερμού. Εάν παραβιαστούν τα παράθυρα ή οι πόρτες ή εντοπιστεί ανεπιθύμητη παρουσία ατόμου, ενεργοποιείται η σειρήνα, ενώ παράλληλα παρέχεται τηλεφωνική ειδοποίηση. Μετά την απομάκρυνση του ιδιοκτήτη από την οικία και την ενεργοποίηση του συναγερμού, το σύστημα αυτοματισμού θα σβήσει τα φώτα, θα κλείσει την ηλεκτρική κουζίνα ή το θερμοσίφωνα, εάν έχουν ξεχαστεί αναμμένα, θα μαζέψει όλες τις ηλεκτρικές τέντες κ.λ.π.
- Εάν κάτι έκτακτο συμβεί στην κατοικία κατά την απουσία του ιδιοκτήτη, όπως για παράδειγμα πυρκαγιά ή κάποια βλάβη, παρέχεται άμεση πληροφόρηση σε αυτόν από το σύστημα αυτοματισμού.
- Εάν ο ήλιος «καίει» το πάτωμα, το σύστημα θα κατεβάσει την ηλεκτρική τέντα. Εάν όμως ο άνεμος απειλεί να σχίσει την τέντα, τότε θα τη μαζέψει και θα κατεβάσει το ρολό για να προστατέψει το πάτωμα.
- Σε περίπτωση παρατεταμένης απουσίας του ιδιοκτήτη είναι δυνατόν να ρυθμιστούν συγκεκριμένες ώρες ποτίσματος, αλλά και να ληφθούν υπ' όψη άλλοι παράγοντες, όπως η πιθανότητα βροχής, ώστε να μην εκτελεστεί το πότισμα ή σε περίπτωση υψηλής θερμοκρασίας και καύσωνα, ώστε να παραταθεί ο χρόνος ποτίσματος.
- Δημιουργία διαφόρων σεναρίων, τα οποία μπορούν να ενεργοποιούνται ή να ακυρώνονται κατά βούληση.

Γενικά οι λειτουργίες που μπορεί να ενσωματώνει ένα «Έξυπνο σπίτι» είναι άπειρες, καθώς αφορούν στο σύνολο των ανθρώπινων δραστηριοτήτων και αναγκών (όπως η διασκέδαση, η εργασία, οι καθημερινές ασχολίες) και το μοναδικό ουσιαστικά όριο αποτελεί η ανθρώπινη φαντασία.

1.3. Μετρώντας το IQ ενός σπιτιού (ή του τεχνικού και του ιδιοκτήτη)

Ένα ευρύ φάσμα διαφορετικών τεχνολογιών «Έξυπνων σπιτιών» είναι διαθέσιμο στην αγορά και θα μπορούσε να χρησιμοποιηθεί για να παρέχει νέες δυνατότητες και επιλογές στο σπίτι προς όφελος των χρηστών του. Σε γενικές γραμμές όμως οι τεχνολογίες και τα πρότυπα έχουν αποτύχει να δημιουργήσουν τις κατάλληλες προϋποθέσεις για τη μαζική προώθησή τους στην αγορά. Ένα από τα κύρια προβλήματα είναι η περιορισμένη τεχνολογική έρευνα από την πλευρά των κατασκευαστών, οι οποίοι αδυνατούν να αξιολογήσουν σωστά τις τρέχουσες ανάγκες των καταναλωτών. Οι προσπάθειες να αναπτυχθούν κάποιες έξυπνες εφαρμογές δεν αξιοποιήθηκαν για ευρεία χρήση.

Το Ίδρυμα Έρευνας και Τεχνολογίας Joseph Rowntree παρέχει μια ευκαιρία αξιολόγησης των «Έξυπνων σπιτιών». Έχει κατασκευάσει δύο πρότυπα σπίτια, χρησιμοποιώντας τις διαθέσιμες τεχνολογίες αυτοματοποίησης. Το ένα είναι ένα διαμέρισμα στο Εδιμβούργο και το άλλο μία μονοκατοικία στην Νέα Υόρκη. Ο στόχος του συγκεκριμένου εγχειρήματος είναι η ανάδειξη και η ανάπτυξη των

τεχνολογιών των «Έξυπνων σπιτιών», οι οποίες να καλύπτουν τις ακόλουθες απαιτήσεις:

- Να είναι εύχρηστες με στόχο τις ανεξάρτητες δραστηριότητες μέσα στο σπίτι.
- Να έχουν μικρό κόστος εγκατάστασης και συντήρησης, αλλά και χαμηλό λειτουργικό κόστος.
- Να είναι αξιόπιστες και ευέλικτες για τις μελλοντικές προσθήκες και τις προσαρμογές.
- Να είναι εύκολη η εγκατάσταση και η συντήρησή τους.

Απώτερος στόχος τους Ιδρύματος Έρευνας και Τεχνολογίας Joseph Rowntree ήταν να δημιουργήσει την αφετηρία για έρευνα και ανάπτυξη σχεδίων τα οποία θα βρουν εφαρμογή στην αξιοποίηση του προγράμματος στα πλαίσια της ιδιωτικής κατοικίας.

1.4. «Έξυπνες» υπηρεσίες στην Ελλάδα

Τα τελευταία χρόνια έχουν αρχίσει να εφαρμόζονται και στην Ελλάδα οι σχετικές τεχνολογίες για το «Έξυπνο σπίτι». Υπάρχουν εταιρίες που εργάζονται στον τομέα αυτό και έχουν ως στόχο την παροχή «έξυπνων» υπηρεσιών, δημιουργώντας κατάλληλες υποδομές μέσα στο κτίριο του τελικού χρήστη. Ωστόσο, το υψηλό κόστος του σχεδιασμού, της απόκτησης και εγκατάστασης τέτοιων προγραμμάτων και υπηρεσιών δεν επιτρέπει την ευρεία διάδοσή τους.

Κεφάλαιο 2: Θεωρητικό υπόβαθρο

Στο κεφάλαιο αυτό αναλύονται όλες οι απαραίτητες γνώσεις και τα εργαλεία, τα οποία χρησιμοποιήθηκαν για το σχεδιασμό και την υλοποίηση της παρούσας πτυχιακής εργασίας. Για το σχεδιασμό και την κατασκευή του ιστοχώρου διαχείρισης επιλέχθηκαν οι γλώσσες προγραμματισμού που μπορούν να φέρουν εις πέρας ένα τέτοιο έργο.

2.1. Πρωτόκολλο HTTP

Το Πρωτόκολλο Μεταφοράς Υπερκειμένου **HTTP** (HyperText Transfer Protocol) αποτελεί το κύριο πρωτόκολλο που χρησιμοποιείται στους φυλλομετρητές του Παγκόσμιου Ιστού για να μεταφέρει δεδομένα (αρχεία κειμένου, γραφικών, εικόνας, ήχου, video ή οποιουδήποτε multimedia αρχείου) μεταξύ ενός διακομιστή (Server) και ενός πελάτη (Client). Σήμερα το πρωτόκολλο αυτό είναι το πλέον καθιερωμένο και διαδεδομένο, σε σημείο που σχεδόν όλοι οι φυλλομετρητές να το θεωρούν δεδομένο και να το χρησιμοποιούν σε περίπτωση που ο χρήστης δεν επιλέξει κάποιο άλλο πρωτόκολλο. Αν δηλαδή ο χρήστης δεν γράψει http://my.url αλλά γράψει σκέτο το my.url σχεδόν όλοι οι φυλλομετρητές θεωρούν σαν δεδομένο το πρωτόκολλο http και όχι κάποιο άλλο.

Το HTTP υλοποιείται σε δυο προγράμματα: ένα πρόγραμμα εξυπηρετητή – Server και ένα πρόγραμμα πελάτη (φυλλομετρητής – Browser). Το πρόγραμμα πελάτη και το πρόγραμμα εξυπηρετητή, που εκτελούνται σε διαφορετικά τερματικά συστήματα, συνομιλούν μεταξύ τους ανταλλάσσοντας μηνύματα HTTP. Το HTTP ορίζει τη δομή αυτών των μηνυμάτων, καθώς και τον τρόπο ανταλλαγής τους μεταξύ εξυπηρετητή και πελάτη. Όταν ο χρήστης ζητάει μια ιστοσελίδα, το πρόγραμμα περιήγησης στέλνει στον εξυπηρετητή μηνύματα αίτησης HTTP για τα αντικείμενα της σελίδας. Ο εξυπηρετητής δέχεται αιτήματα και ανταποκρίνεται με μηνύματα απόκρισης HTTP, τα οποία περιέχουν τα αντικείμενα.

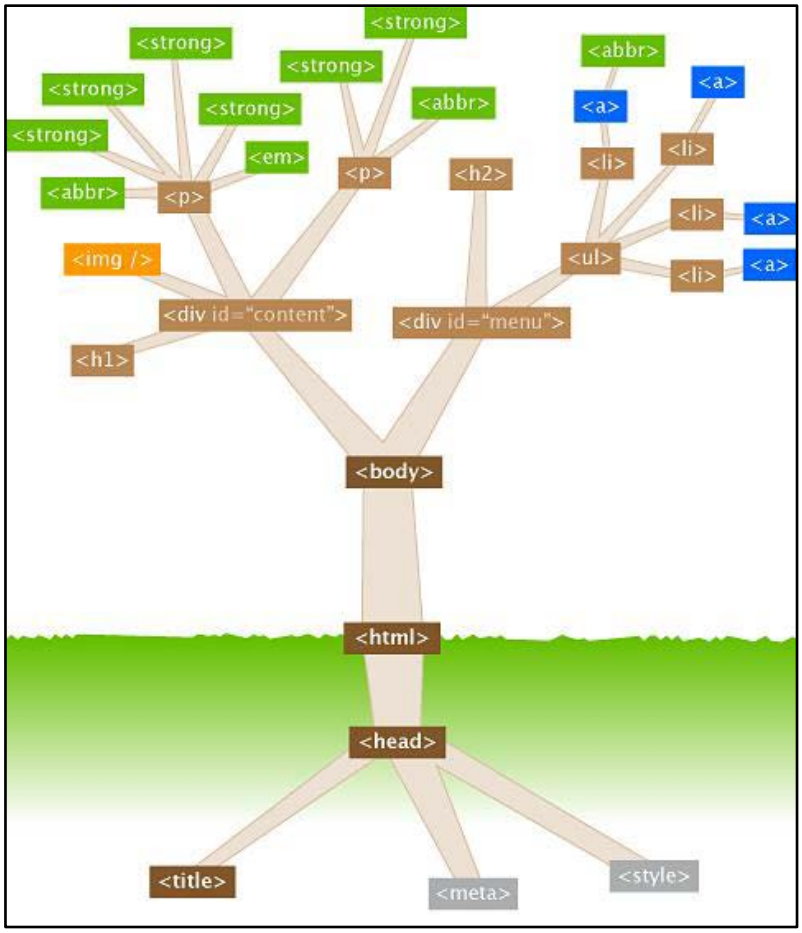
Το HTTP χρησιμοποιεί το TCP ως υποκείμενο πρωτόκολλο μεταφοράς που παρέχει μια αξιόπιστη μεταφορά δεδομένων στο HTTP. Αυτό σημαίνει ότι κάθε μήνυμα αιτήματος HTTP που εκπέμπεται από μια διεργασία ενός πελάτη φτάνει τελικά ανέπαφο στον εξυπηρετητή.

2.2. HTML

Η HTML (HyperText Markup Language ή Γλώσσα Σήμανσης Υπερκειμένου) είναι η κύρια γλώσσα σήμανσης για τις ιστοσελίδες, με τα στοιχεία της να αποτελούν τα βασικά δομικά στοιχεία των ιστοσελίδων.

Η HTML γράφεται υπό μορφή στοιχείων HTML, τα οποία αποτελούνται από ετικέτες, οι οποίες περικλείονται μέσα σε σύμβολα «μεγαλύτερο από» και «μικρότερο από» (για παράδειγμα <html>), μέσα στο περιεχόμενο της ιστοσελίδας. Οι ετικέτες HTML συνήθως λειτουργούν ανά ζεύγη, για παράδειγμα <h1> και </h1>, με την πρώτη να ονομάζεται ετικέτα έναρξης και τη δεύτερη ετικέτα λήξης (ή σε άλλες περιπτώσεις ετικέτα ανοίγματος και ετικέτα κλεισίματος αντίστοιχα). Ανάμεσα στις ετικέτες, οι σχεδιαστές ιστοσελίδων μπορούν να τοποθετήσουν κείμενο, πίνακες, εικόνες κ.λ.π.

Ο σκοπός ενός φυλλομετρητή (Web Browser) είναι να διαβάζει τα έγγραφα HTML και να τα συνθέτει σε σελίδες, τις οποίες μπορεί ο χρήστης να επεξεργαστεί. Ο Web Browser δεν εμφανίζει τις ετικέτες HTML, αλλά τις χρησιμοποιεί για να ερμηνεύσει το περιεχόμενο της σελίδας.



Εικόνα 1.Δένδρο HTML στοιχείων (elements)

Τα στοιχεία της HTML χρησιμοποιούνται για να χτίσουν όλους του ιστότοπους. Η HTML επιτρέπει την ενσωμάτωση ποικίλων μορφών αρχείων (π.χ. εικόνες, κείμενο, video κ.ά.) μέσα στη σελίδα και μπορεί να χρησιμοποιηθεί για να εμφανίσει διαδραστικές φόρμες. Παρέχει τις μεθόδους δημιουργίας δομημένων εγγράφων (δηλαδή

εγγράφων που αποτελούνται από το περιεχόμενο που μεταφέρουν και από τον κώδικα μορφοποίησης του περιεχομένου), καθορίζοντας σημαντικά δομικά στοιχεία για το κείμενο, όπως κεφαλίδες, παραγράφους, λίστες, συνδέσμους, παραθέσεις. Μπορούν επίσης να ενσωματώνονται σενάρια εντολών σε γλώσσες όπως η JavaScript, τα οποία επηρεάζουν τη συμπεριφορά των ιστοσελίδων HTML.

Μερικά από τα στοιχεία HTML που χρησιμοποιήθηκαν φαίνονται στον παρακάτω πίνακα.

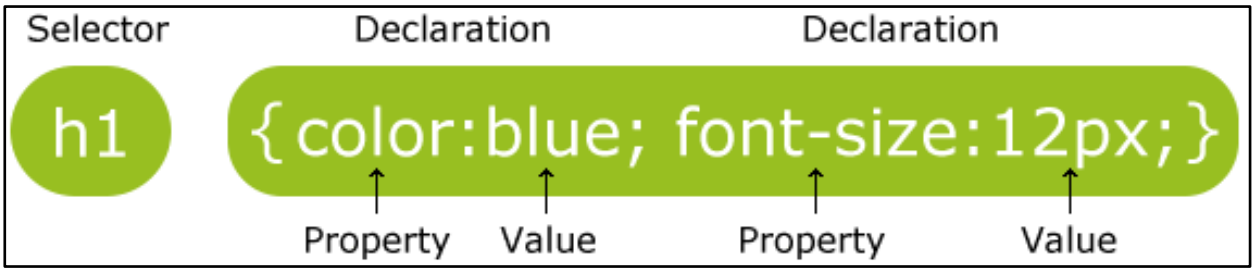
Πίνακας 1. Ετικέτες της HTML

Στοιχείο	**Λειτουργία**
<DOCTYPE>	Η δήλωση του DOCTYPE πρέπει να είναι η πρώτη που γίνεται σε ένα έγγραφο HTML. Είναι μια εντολή στο πρόγραμμα περιήγησης για το ποια είναι η έκδοση της HTML στο έγγραφο.
<table>	Ορίζει έναν πίνακα στην HTML. Ο πίνακας αποτελείται από ένα ή περισσότερα <tr>, <td> ή <th>.
<fieldset>	Η ετικέτα <fieldset> χρησιμοποιείται για την ομαδοποίηση των σχετικών στοιχείων μιας φόρμας.
<button>	Η ετικέτα <button> ορίζει ένα κουμπί.
	Η ετικέτα καθορίζει ένα τμήμα σε ένα έγγραφο HTML. Χρησιμοποιείται για την ομαδοποίηση των στοιχείων, ώστε να δοθούν κοινές μορφοποιήσεις.
<body>	Η ετικέτα <body> ορίζει το σώμα του εγγράφου, δηλαδή όλα τα περιεχόμενα ενός εγγράφου HTML, όπως κείμενο, υπερσύνδεσμοι, εικόνες, πίνακες, λίστες, κλπ. που βρίσκονται μέσα σε αυτό.
<form>	Η ετικέτα <form> χρησιμοποιείται για να δημιουργήσει μια φόρμα HTML. Χρησιμοποιείται επίσης για να μεταφέρει δεδομένα σε ένα διακομιστή.
<h1> έως <h6>	Οι ετικέτες <h1> έως <h6> χρησιμοποιούνται για να καθορίσουν το μέγεθος των γραμμάτων για τις επικεφαλίδες.
<input>	Η ετικέτα <input> χρησιμοποιείται για να εισάγει πληροφορίες ο χρήστης. Περιλαμβάνεται σε μια ετικέτα <form>. Ένα πεδίο εισαγωγής μπορεί να ποικίλλει, ανάλογα με είδος του.
<text area>	Η ετικέτα <text area> ορίζει μία περιοχή εισαγωγής κειμένου σε πολλές γραμμές. Το μέγεθος μιας τέτοιας περιοχής καθορίζεται από τις ιδιότητες των χαρακτηριστικών γραμμής και στήλης.

2.3. CSS

Αναλυτικά CSS σημαίνει Cascading Style Sheets. Αποτελεί μια γλώσσα προορισμένη να αναπτύσσει στυλιστικά μια ιστοσελίδα, δηλαδή να διαμορφώνει περισσότερα χαρακτηριστικά, όπως χρώματα, στοίχιση κ.ά. και να προσφέρει περισσότερες δυνατότητες σε σχέση με την html.

Ένας κανόνας CSS έχει δύο κύρια μέρη: έναν επιλογέα (Selector) και μία ή περισσότερες δηλώσεις (Declaration). Ο επιλογέας είναι συνήθως το στοιχείο που τίθεται προς μορφοποίηση και κάθε δήλωση αποτελείται από μια ιδιότητα και μια τιμή. Ο κανόνας φαίνεται στην παρακάτω εικόνα.



Εικόνα 2.Κανόνας σύνταξης CSS.

Επιπρόσθετα με την παραμετροποίηση της μορφοποίησης ενός στοιχείου HTML, η CSS επιτρέπει και τη χρήση συγκεκριμένων επιλογέων του χρήστη που ονομάζονται ID και Class. Ο επιλογέας ID χρησιμοποιείται για τη μορφοποίηση ενός συγκεκριμένου και μοναδικού στοιχείου HTML. Αυτός ο επιλογέας χρησιμοποιεί την ιδιότητα ID του στοιχείου HTML και ορίζεται με ένα #. Ένας άλλος επιλογέας CSS είναι ο Class. Ο επιλογέας αυτός χρησιμοποιείται για να μορφοποιήσει πολλαπλά στοιχεία της HTML, αντίθετα με τον επιλογέα ID. Αυτό επιτρέπει να γίνει χρήση της ίδιας μορφοποίησης σε πολλά στοιχεία που έχουν την ίδια ιδιότητα Class. Για την επιλογή αυτού του επιλογέα στην CSS χρησιμοποιείται το σύμβολο “.”.

Τέλος, υπάρχουν τρεις τρόποι για την εισαγωγή της CSS σε ένα κείμενο HTML. Ο πρώτος τρόπος επιτυγχάνεται με σύνδεση ενός εξωτερικού αρχείου, το οποίο περιέχει όλους τους κανόνες μορφοποίησης για τα στοιχεία της HTML. Ο δεύτερος τρόπος είναι η ενσωμάτωση των κανόνων μορφοποίησης της CSS στην ετικέτα επικεφαλίδας του κειμένου HTML και ο τρίτος τρόπος είναι η εισαγωγή των κανόνων CSS σαν ιδιότητες στην ετικέτα του στοιχείου HTML, που πρόκειται να μορφοποιηθεί.

2.4. Javascript

Η JavaScript είναι μία γλώσσα προγραμματισμού, η οποία έχει σαν σκοπό την παραγωγή δυναμικού περιεχομένου και την εκτέλεση κώδικα στην πλευρά του πελάτη

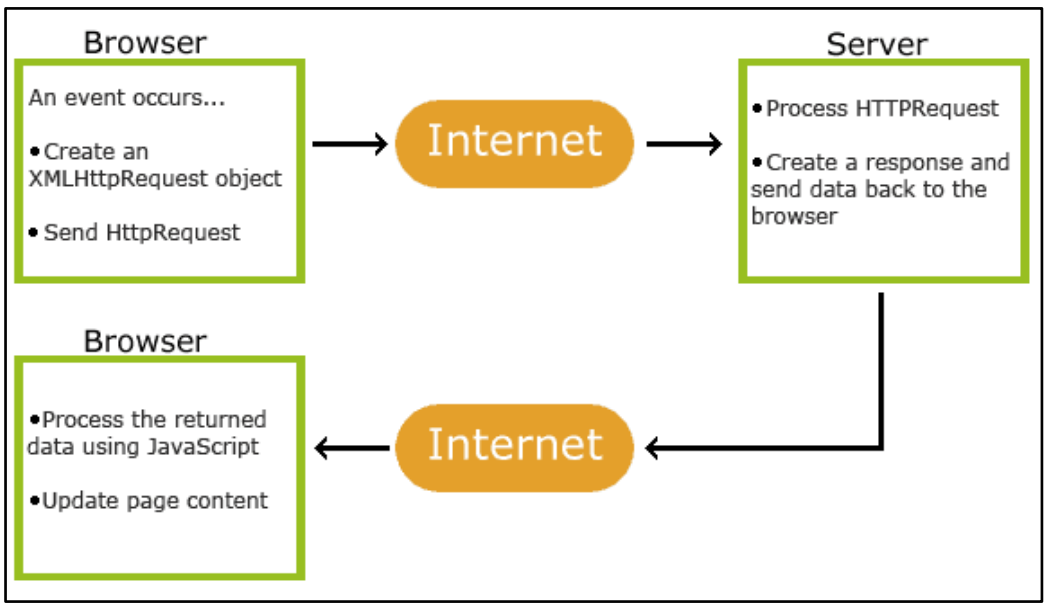
(Client – side) σε ιστοσελίδες. Είναι η πιο δημοφιλής Scripting γλώσσα στον κόσμο. Είναι η τυπική γλώσσα που χρησιμοποιείται σε ιστοσελίδες, αλλά επίσης χρησιμοποιείται ευρέως από Desktop εφαρμογές, εφαρμογές κινητής τηλεφωνίας, καθώς και διακομιστές διαδικτύου.

Μια γλώσσα Scripting είναι μια απλή γλώσσα προγραμματισμού, που υποστηρίζει τη δημιουργία σεναρίων. Σενάρια είναι γραμμές κώδικα που μπορούν να ερμηνεύονται και να εκτελούνται χωρίς μεταγλώττιση.

Η JavaScript μπορεί να διαβάσει και να αλλάξει το περιεχόμενο των στοιχείων της HTML. Μπορεί επίσης να χειριστεί τη CSS. Μπορεί ακόμα να χρησιμοποιηθεί για την επικύρωση δεδομένων, όπως η επικύρωση μιας φόρμας εισόδου. Μπορεί να χρησιμοποιηθεί για την αποθήκευση και ανάκτηση πληροφοριών στον υπολογιστή του χρήστη. Τέλος, μπορεί να ρυθμιστεί, ώστε να εκτελείται όταν συμβαίνει κάτι, όπως όταν ένας χρήστης κάνει κλικ σε ένα στοιχείο HTML. Η εισαγωγή JavaScript σε μια σελίδα HTML πραγματοποιείται από την ετικέτα <script>.

2.5. AJAX

Το AJAX (Asynchronous JavaScript And XML) δεν είναι μια νέα γλώσσα προγραμματισμού, αλλά ένα νέος τρόπος χρησιμοποίησης των προτύπων που ήδη υπάρχουν. Επιτρέπει την ανταλλαγή δεδομένων με έναν Server και την ενημέρωση τμημάτων μιας ιστοσελίδας, χωρίς να φορτώνεται εκ νέου ολόκληρη η σελίδα. Παραδείγματα των εφαρμογών που χρησιμοποιούν AJAX είναι το Google Maps, Gmail, YouTube, Facebook. Η επόμενη εικόνα παρουσιάζει υπό μορφή σχήματος το πώς λειτουργεί το AJAX.



Εικόνα 3. Διάγραμμα λειτουργίας του AJAX

2.6. XML (eXtensive Markup Language)

2.6.1 Γενικά

Η XML είναι μια Markup γλώσσα για έγγραφα τα οποία περιέχουν δομημένη πληροφορία και αποτελεί σημαντικό στοιχείο του Semantic Web. Η δομημένη πληροφορία περιέχει αντικείμενα (λέξεις, εικόνες κ.τ.λ.) και κάποια στοιχεία που προσδιορίζουν τι αντιπροσωπεύει αυτή και τι ρόλο παίζει. Σχεδόν όλα τα έγγραφα έχουν κάποια δομή.

Μια Markup γλώσσα είναι ένας μηχανισμός, με τον οποίο υποδηλώνεται η δομή ενός εγγράφου. Οι προδιαγραφές της XML ορίζουν έναν τυποποιημένο τρόπο με τον οποίο μπορεί κάποιος να προσθέτει Markup στα έγγραφα. Ένα τυπικό παράδειγμα στο οποίο μπορούμε να δούμε μια πληροφορία και το είδος το οποίο αντιπροσωπεύει είναι το εξής: <author>Berners-Lee</author>.

Το πλεονέκτημα της XML είναι ότι το λογισμικό μπορεί να διαβάσει συγκεκριμένες ετικέτες και να εφαρμόσει διαδικασίες, όπως είναι η εξαγωγή βιβλιογραφικής πληροφορίες αλλά και άλλες χρήσιμες διαδικασίες. Οι σχεδιαστικοί στόχοι της XML δίνουν έμφαση στην απλότητα, τη γενικότητα και τη χρήση της στο Internet. Η γλώσσα αυτή υποστηρίζεται από όλες τις σύγχρονες γλώσσες προγραμματισμού του κόσμου και παρ' όλο που στοχεύει στη δόμηση εγγράφων, υπάρχουν πολλές περιπτώσεις στις οποίες χρησιμοποιείται για την παρουσίαση πολύπλοκων δομών δεδομένων, όπως για παράδειγμα τα XML Web Services. Τέλος, πρέπει να αναφερθεί ότι η XML είναι η βάση για ένα σύνολο άλλων γλωσσών όπως οι RSS, Atom, SOAP, XHTML και άλλες.

2.6.2 Ορισμός της δομής ενός XML εγγράφου

Όπως αναφέρθηκε προηγουμένως, η XML είναι ένα έγγραφο με καθορισμένη δομή. Παρακάτω θα δοθούν πληροφορίες για το XML Schema, το οποίο είναι ένας τρόπος για να περιγραφεί η δομή ενός αρχείου XML. Το XML Schema λοιπόν εκφράζει με ένα κοινό λεξιλόγιο το περιεχόμενο του XML εγγράφου. Πιο συγκεκριμένα, παρέχει τα μέσα, ώστε να δηλωθεί η δομή, το περιεχόμενο και τα Tags ενός XML αρχείου. Αυτή η διαδικασία επικυρώνει τη σύνταξη και τη φόρμα που επιτρέπεται σε κάθε τέτοιο έγγραφο και διευκολύνει το διαμοιρασμό πληροφοριών μέσω διαδικτύου. Το XML Schema επιτρέπει το διάβασμα, την επικύρωση και την επεξεργασία XML εγγράφων από εφαρμογές λογισμικού. Αυτό παρέχει τη βάση για τη σύλληψη, αναπαράσταση, ανταλλαγή και αποθήκευση πληροφοριών, στις οποίες μπορούν εύκολα να έχουν όλοι πρόσβαση.

2.6.3 Σχεδιαστικοί στόχοι της XML

Οι προδιαγραφές της XML ορίζουν κάποιους στόχους που μπορεί να πετύχει η γλώσσα. Αυτοί καθορίστηκαν μετά από πρόταση της W3C. Οι στόχοι αυτοί περιγράφονται παρακάτω:

● Να γίνεται χρήση της XML στο Internet.

Οι χρήστες θα πρέπει να μπορούν να βλέπουν XML έγγραφα τόσο εύκολα, όσο βλέπουν και HTML σελίδες. Πρακτικά αυτό είναι δυνατό μόνο όταν οι XML περιηγητές γίνουν παντού διαθέσιμοι, όπως και οι HTML περιηγητές. Παρά τη δυσκολία υλοποίησης του συγκεκριμένου στόχου, αυτός παραμένει.

● Η XML να υποστηρίζει μια ποικιλία προγραμμάτων.

Η XML πρέπει να χρησιμοποιείται προς όφελος των χρηστών σε εφαρμογές συγγραφής, περιήγησης και ανάλυσης περιεχομένου κτλ. Παρ' όλο που αρχικά δημιουργήθηκε για τον καθορισμό της δομής των εγγράφων στο διαδίκτυο, δε σημαίνει ότι δεν μπορεί να χρησιμοποιηθεί για ένα πλήθος άλλων διεργασιών.

● Να είναι εύκολη η συγγραφή προγραμμάτων τα οποία επεξεργάζονται XML έγγραφα.

Αυτό σημαίνει πρακτικά ότι ένας μαθητής της επιστήμης των υπολογιστών μέσα σε 2 εβδομάδες θα μπορεί να δημιουργήσει ένα πρόγραμμα το οποίο θα επεξεργάζεται XML έγγραφα.

● Ο αριθμός των προαιρετικών χαρακτηριστικών της XML να είναι όσο το δυνατόν μικρότερος, ιδανικά μηδενικός.

Αυτό πρέπει να συμβαίνει γιατί όταν οι χρήστες επιθυμούν να μοιραστούν έγγραφα, τα προαιρετικά χαρακτηριστικά αυξάνουν τα προβλήματα συμβατότητας, προκαλώντας στους χρήστες σύγχυση και απογοήτευση.

● Τα XML έγγραφα να είναι αναγνώσιμα από ανθρώπους και απολύτως σαφή.

Με απλά λόγια, ακόμη και αν δεν υπάρχει διαθέσιμος κάποιος XML περιηγητής, θα πρέπει τα έγγραφα να είναι κατανοητά ως προς την περιγραφή και το περιεχόμενο, με ένα από τα προγράμματα ανάγνωσης κειμένου.

● Ο σχεδιασμός της XML να επιτρέπει την εύκολη και γρήγορη πρόσβαση σε αυτήν.

Η χρήση της γλώσσας αυτής είναι απαραίτητη και θα πρέπει να αναπτύσσεται όσο το δυνατόν πιο γρήγορα.

● Ο σχεδιασμός της XML να είναι επίσημος και συνοπτικός.

Αυτό πρακτικά σημαίνει ότι η XML θα πρέπει να εκφράζεται σε EBNF και θα πρέπει η λογική της να είναι συνυφασμένη με αυτή των μοντέρνων μεταγλωττιστικών εργαλείων και τεχνικών. Μια από τις ανάγκες για τη δημιουργία της XML ήταν και αυτή, καθώς η γραμματική της γλώσσας SGML, λόγω κάποιων τεχνικών προβλημάτων, δε μπορεί να εκφραστεί σε EBNF. Η συγγραφή ενός SGML parser απαιτεί το χειρισμό πολλών δύσκολων και σπάνιων χαρακτηριστικών, ενώ η XML όχι. Η δημιουργία των XML

εγγράφων πρέπει να είναι εύκολη. Παρ' όλο που μπορεί να υπάρχουν περίπλοκες εφαρμογές επεξεργασίας τέτοιων εγγράφων, η δημιουργία και η επεξεργασία XML περιεχομένου θα πρέπει να γίνεται εύκολα και από έναν απλό επεξεργαστή κειμένου, όπως από Perl scripts κτλ.

2.6.4 Γιατί χρησιμοποιούμε XML;

Η XML χρησιμοποιείται σε πολλές φάσεις της ανάπτυξης λογισμικού, κυρίως για να απλοποιήσει την αποθήκευση και το διαμοιρασμό δεδομένων. Οι κυριότεροι λόγοι χρήσης της XML είναι οι εξής:

- Η XML διαχωρίζει τα δεδομένα από την HTML.

Εάν προκύψει η ανάγκη για δυναμική απεικόνιση δεδομένων σε μια HTML σελίδα, συνήθως χρειάζεται πολύς χρόνος για την επεξεργασία της, κάθε φορά που απαιτείται μια αλλαγή. Με την XML τα δεδομένα μπορούν να αποθηκευτούν σε διαφορετικά XML έγγραφα. Με αυτό τον τρόπο, το μόνο που απασχολεί τον προγραμματιστή είναι να δημιουργήσει τη διεπαφή της σελίδας, να απεικονίσει τις πληροφορίες και είναι σίγουρο ότι δεν θα χρειαστεί περαιτέρω αλλαγές στον HTML κώδικα. Με λίγες γραμμές JavaScript κώδικα είναι εύκολο να διαβαστεί ένα εξωτερικό XML αρχείο για την απεικόνιση των πληροφοριών.

- Η XML απλοποιεί το διαμοιρασμό πληροφοριών.

Στον πραγματικό κόσμο τα συστήματα υπολογιστών και βάσεων δεδομένων περιέχουν πληροφορίες σε μορφές που δεν είναι γενικά συμβατές μεταξύ τους. Από την άλλη πλευρά τα XML δεδομένα είναι αποθηκευμένα σε απλή μορφή κειμένου. Αυτό παρέχει ανεξαρτησία από κάθε λογισμικό ή υλικό στην αποθήκευση δεδομένων. Ως αποτέλεσμα η δημιουργία δεδομένων και ο διαμοιρασμός τους μεταξύ εφαρμογών γίνεται πάρα πολύ εύκολα.

- Η XML απλοποιεί τη μεταφορά δεδομένων.

Μια από τις μεγαλύτερες προκλήσεις στον κόσμο των προγραμματιστών είναι η ανταλλαγή πληροφοριών μεταξύ συστημάτων διαφορετικής σύνθεσης, στον παγκόσμιο ιστό. Η ανταλλαγή πληροφοριών σε μορφή XML μειώνει την πολυπλοκότητα αυτής της διαδικασίας, καθώς τα δεδομένα μπορούν να διαβαστούν από συστήματα και λογισμικά διαφορετικής φιλοσοφίας.

- Η XML απλοποιεί τις αλλαγές της πλατφόρμας.

Η αναβάθμιση σε νέα συστήματα (λογισμικού ή υλικού), είναι αρκετά χρονοβόρα διαδικασία. Ένας μεγάλος όγκος πληροφοριών πρέπει να μετατραπεί και συνήθως ασύμβατα δεδομένα με την πλατφόρμα χάνονται τελείως. Τα δεδομένα που είναι αποθηκευμένα σε XML είναι πολύ πιο εύκολο να επεκταθούν ή να αναβαθμιστούν από νέα λειτουργικά συστήματα, νέες εφαρμογές ή νέους περιηγητές, χωρίς να χαθούν πληροφορίες λόγω κάποιας ασυμβατότητας, εξαιτίας του το ότι η πληροφορία βρίσκεται σε απλά αρχεία κειμένου.

- Η XML κάνει τα δεδομένα διαθέσιμα σε μεγαλύτερο βαθμό.

Διάφορες εφαρμογές μπορούν να έχουν πρόσβαση στα δεδομένα, όχι μόνο σε HTML σελίδες, αλλά επίσης σε XML πηγές δεδομένων. Με XML τα δεδομένα είναι διαθέσιμα σε όλους τους τύπους συστημάτων (υπολογιστές τσέπης, feeds, μηχανές ομιλίας κτλ.) τα οποία απαιτούν πρόσβαση σε δεδομένα, με αποτέλεσμα να κάνουν διαθέσιμες τις πληροφορίες ακόμα και σε άτομα με ειδικές ανάγκες (π.χ. Μηχανές ομιλίας για την εξυπηρέτηση τυφλών ατόμων).

- Η XML αποτέλεσε τη βάση νέων γλωσσών στο Internet.

Πολλές νέες γλώσσες στο Internet δημιουργήθηκαν με την XML.

Το μέλλον της γλώσσας διαγράφεται λαμπρό, καθώς η τεχνολογία μπορεί να μας δώσει επεξεργαστές κειμένου, εφαρμογές λογιστικών φύλλων και βάσεις δεδομένων των οποίων οι πληροφορίες μπορούν να ανταλλάσσονται, χωρίς καμία προηγούμενη μετατροπή.

2.6.5 XML έννοιες

- (Unicode) Character:

Από τον ορισμό της γλώσσας γίνεται κατανοητό ότι το έγγραφο XML αποτελείται από αλφαριθμητικούς χαρακτήρες. Σχεδόν κάθε Unicode χαρακτήρας μπορεί να εμφανιστεί σε ένα XML έγγραφο.

- Processor and Application:

Ο επεξεργαστής αναλύει το Markup κομμάτι του εγγράφου και παραχωρεί σε μια εφαρμογή τη δομημένη πληροφορία που λαμβάνει από αυτό. Οι προδιαγραφές της γλώσσας ορίζουν τι πρέπει να κάνει ένας επεξεργαστής και τι όχι. Από τη σκοπιά του προγράμματος, δεν υπάρχουν προδιαγραφές για το πώς θα γίνει η διαχείριση των δεδομένων. Ο επεξεργαστής συχνά αναφέρεται και ως XML Parser.

- Markup and Content:

Οι χαρακτήρες οι οποία αποτελούν ένα XML έγγραφο είναι διαμοιρασμένοι σε Markup και περιεχόμενο. Αυτά τα δύο πρέπει να διαχωρίζονται από την εφαρμογή με κάποιους απλούς συντακτικούς κανόνες. Παραδείγματος χάριν όλα τα αλφαριθμητικά, τα οποία ανήκουν στην κατηγορία Markup, ξεκινούν ή τελειώνουν με τα σύμβολα “<” και “>” αντίστοιχα. Επίσης είναι δυνατό να ξεκινούν με το χαρακτήρα “&” και να τελειώνουν με το χαρακτήρα “;”. Τα αλφαριθμητικά τα οποία δεν αποτελούν Markup είναι το περιεχόμενο.

- Ετικέτες (Tags):

Ετικέτα ονομάζεται το Markup αλφαριθμητικό, το οποίο ξεκινάει με το χαρακτήρα “<” και τελειώνει με το χαρακτήρα “>”. Τα Tags αποτελούνται από τρεις κατηγορίες. Αρχικά υπάρχουν τα start-tags (ετικέτες αρχής), πχ. <section>, στη συνέχεια έχουμε τα end-

tags (ετικέτες τέλους), πχ. </section> και τέλος έχουμε τα empty – element tags, πχ. <line-break/>.

- Στοιχείο (Element):

Το στοιχείο είναι κάθε λογικό συστατικό του εγγράφου, το οποίο είτε ξεκινάει με ένα start-tag και τελειώνει με ένα end-tag, είτε αποτελείται από ένα empty – element tag. Οι χαρακτήρες ανάμεσα στα start-tags και τα end-tags, αν υπάρχουν, είναι το περιεχόμενο του στοιχείου, και πιθανώς να περιέχουν Markup στοιχεία, καθώς και άλλα στοιχεία τα οποία καλούνται Elements. Για παράδειγμα το <Greeting>Hello, world.</Greeting>.s

- Ιδιότητες (Attribute):

Ιδιότητα ονομάζεται μια Markup δομή, η οποία αποτελείται από ένα ζευγάρι ονόματος-τιμής, το οποίο βρίσκεται μέσα σε ένα start-tag ή empty – element tag. Για παράδειγμα, παρακάτω το στοιχείο img έχει δύο ιδιότητες, src και alt: . Άλλο ένα παράδειγμα θα ήταν το <step number=”3”>Connect A to B.</step> όπου το όνομα της ιδιότητας είναι το “number” και ο αριθμός είναι το “3”.

- XML δήλωση (XML Declaration):

Στα XML έγγραφα πρέπει να δηλώνονται στην αρχή κάποιες πληροφορίες γύρω από αυτά. Για παράδειγμα ποια έκδοση από το πρότυπο XML χρησιμοποιούν αυτά τα έγγραφα και σε τι κωδικοποίηση είναι γραμμένα.

2.6.6 Σύνοψη

Η XML είναι μια Markup γλώσσα η οποία έχει δημιουργηθεί για την ανταλλαγή πληροφοριών στο διαδίκτυο εύκολα και αποδοτικά. Το μεγάλο πλεονέκτημά της είναι η δυνατότητα προσπέλασης των XML εγγράφων από οποιαδήποτε πλατφόρμα λογισμικού ή υλικού. Παράλληλα είναι μια γλώσσα εύκολη στην κατανόηση και μπορεί να βοηθήσει άτομα με ειδικές ανάγκες να αποκτήσουν πρόσβαση σε ποικιλία δεδομένων. Τέλος, η XML αποτελεί τη βάση δημιουργίας μιας πληθώρας γλωσσών που χρησιμοποιούνται στο Internet και χρησιμοποιείται ευρέως στα Web Services.

Κεφάλαιο 3: Arduino

3.1. Τι είναι το Arduino

Το Arduino θα λέγαμε ότι είναι ένα εργαλείο για να κατασκευάσουμε ένα υπολογιστικό σύστημα μέσω του οποίου θα ελέγχονται διάφορες συσκευές, σε αντίθεση με έναν κοινό Ηλεκτρονικό Υπολογιστή. Είναι ανοιχτού υλικού και λογισμικού και βασίζεται σε μια αναπτυξιακή πλακέτα που ενσωματώνει επάνω ένα μικροελεγκτή και συνδέεται με τον Η/Υ για να προγραμματιστεί μέσα από ένα απλό περιβάλλον ανάπτυξης.

Ένα Arduino μπορεί να χρησιμοποιηθεί για να αναπτύξει διαδραστικά αντικείμενα, να δεχτεί εισόδους από πληθώρα αισθητηρίων οργάνων και διακόπτες, αλλά και να ελέγξει διάφορα φώτα, κινητήρες και άλλες συσκευές εξόδου. Τα Projects στον εν λόγω μικροελεγκτή μπορούν να είναι αυτόνομα (σε επίπεδο Hardware) ή να επικοινωνούν με κάποιο Software στον Η/Υ του προγραμματιστή (προγράμματα όπως τα Flash, Processing, MaxMSP).

3.2. Επιλογή του Arduino

Μερικά από τα πλεονεκτήματα που προσφέρει το Arduino σε σχέση με άλλους μικροελεγκτές για χρήση από δασκάλους, μαθητές και άλλους hobbistες είναι τα παρακάτω:

- Είναι φθηνό. Οι πλακέτες του Arduino είναι εξαιρετικά φθηνές σε σχέση με άλλες πλατφόρμες μικροελεγκτών. Ειδικότερα μπορεί με τα σχηματικά που κυκλοφορούν στο Internet να κατασκευάσει κάποιος τη φθηνότερη εκδοχή ενός Arduino. Ωστόσο ακόμα και αν προμηθευτεί την έτοιμη (μονταρισμένη πλακέτα) αυτή θα κοστίσει το πολύ 50€.

- «Τρέχει» σε διάφορα λειτουργικά συστήματα. Οι μηχανικοί λογισμικού, ανέπτυξαν το περιβάλλον προγραμματισμού του Arduino για Windows, Machinstoh OSX και για λειτουργικά συστήματα Linux. Τα περισσότερα συστήματα ανάπτυξης μικροελεγκτών περιορίζονται στα Windows.

- Προσφέρει ένα απλό προγραμματιστικό περιβάλλον. Το περιβάλλον προγραμματισμού ενός Arduino ενδείκνυται για αρχάριους, αλλά είναι ταυτόχρονα ευέλικτο και για πιο προχωρημένους χρήστες.

- Είναι ανοιχτό λογισμικό που επεκτείνεται και παραμετροποιείται. Το Software του Arduino διανέμεται με τη μορφή εργαλείων ανοιχτού λογισμικού και είναι διαθέσιμο προς επέκταση από έμπειρους προγραμματιστές. Η γλώσσα προγραμματισμού του Arduino μπορεί να επεκταθεί διαμέσου των βιβλιοθηκών της C++ και όσοι θέλουν να ασχοληθούν περισσότερο με τους μικροελεγκτές μπορούν να μεταβούν από το Arduino στην AVR-C, που αφορά τον

προγραμματισμό των Atmel μικροελεγκτών και αποτελεί τη γλώσσα στην οποία βασίστηκε το λογισμικό του Arduino. Ομοίως μπορεί κάποιος να προσθέσει κώδικα της AVR-C στο πρόγραμμα που έχει γράψει για το Arduino του.

- Αποτελεί ανοιχτό υλικού το οποίο μπορεί να επεκταθεί. Το Arduino βασίζεται στους μικροελεγκτές της Atmel ATMEGA368 και ATMEGA2560. Τα σχηματικά διαγράμματα για την ανάπτυξη Arduino βρίσκονται κάτω από την άδεια Creative Commons. Επιτρέπεται λοιπόν σε έμπειρους σχεδιαστές να κατασκευάσουν το δικό τους αναπτυξιακό περιβάλλον, εξελίσσοντας το ήδη υπάρχον, χωρίς να προκύπτουν νομικά προβλήματα. Επίσης λιγότερο έμπειροι χρήστες μπορούν να επιδιώξουν την αντιγραφή και κατασκευή της πλακέτας σε ράστερ για να καταλάβουν τη λειτουργία ενός Arduino.

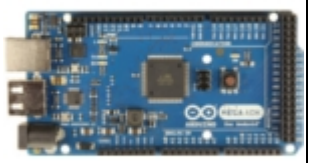
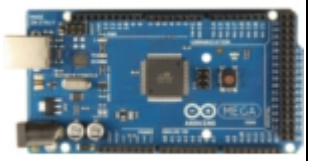
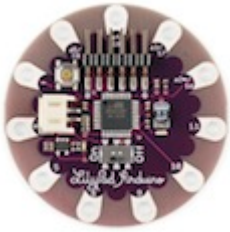
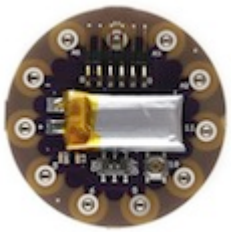
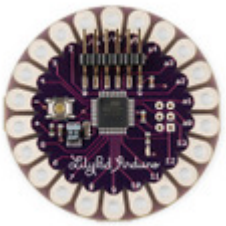
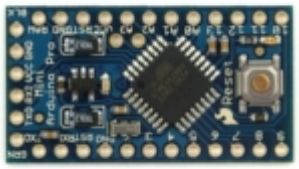
- Η πλακέτα Arduino Uno μπορεί να τροφοδοτηθεί από τη θύρα USB του υπολογιστή, από το εξωτερικό τροφοδοτικό ή ακόμα και από μπαταρία των 9V.

3.3. Μοντέλα Arduino

Τα μοντέλα Arduino που κυκλοφορούν στην αγορά, παρουσιάζονται στον παρακάτω πίνακα.

Πίνακας 2. Μοντέλα Arduino.

			
Arduino uno	Arduino Leonardo	Arduino due	Arduino Yun
			
Arduino tre	Arduino micro	Arduino robot	Arduino esplora

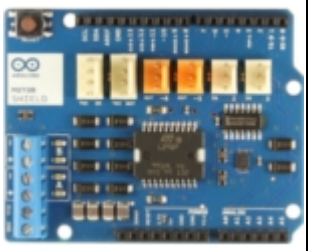
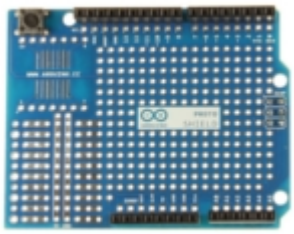
			
Arduino mega adk	Arduino ethernet	Arduino mega	Arduino mini
			
Arduino lilipad usb	Arduino lilipad simple	Arduino lilipad simplesnap	Arduino lilipad
			
Arduino nano	Arduino pro mini	Arduino pro	Arduino fio

3.4. Arduino Shields

Τα Arduino Shields είναι πλακέτες που μπορούν να συνδεθούν στην κορυφή της πλακέτας Arduino και να επεκτείνουν τις δυνατότητές του. Τα Arduino Shields ακολουθούν την ίδια φιλοσοφία με το αρχικό πακέτο εργαλείων: είναι εύκολο να τοποθετηθούν και έχουν μικρό κόστος παραγωγής. Επίσης μπορούν να συνδυαστούν περισσότερα του ενός Arduino Shield πάνω σε μία πλακέτα Arduino.

Τα διαθέσιμα Shields φαίνονται στον παρακάτω πίνακα.

Πίνακας 3. Arduino Shields

			
Arduino gsm shield	Arduino Ethernet shield	Arduino wifi shield	Arduino wireless sd shield
			
Arduino usb host shield	Arduino motor shield	Arduino wireless proto shield	Arduino proto shield

3.5. Arduino Mega 2560

Στην ενότητα αυτή παρουσιάζεται ο μικροελεγκτής Arduino που χρησιμοποιήθηκε στην παρούσα εργασια, τα χαρακτηριστικά του και μερικά από τα διαθέσιμα εξαρτήματα που μπορούν να συνδεθούν απ' ευθείας με αυτό.

3.5.1 Χαρακτηριστικά

Το Arduino που χρησιμοποιήθηκε για την εφαρμογή είναι ένα Arduino Mega 2560 με τα εξής χαρακτηριστικά:

Εικόνα 4. Εμπρόσθια και οπίσθια όψη της πλακέτας Arduino

Η πλακέτα Arduino Mega είναι μια πλακέτα μικροελεγκτή βασισμένη στον Atmega2560. Έχει 54 ψηφιακές εισόδους – εξόδους (από τις οποίες οι 15 μπορούν να χρησιμοποιηθούν σαν PWM έξοδοι), 16 αναλογικές εισόδους, 4 UARTs (Hardware Serial Ports), έναν ταλαντωτή (Crystal Oscillator) στα 16 MHz, σύνδεση USB, μια υποδοχή για τροφοδοσία, υποδοχές (Header) ICSP και ένα κουμπί Reset. Επίσης περιέχει όλα όσα χρειάζονται για την υποστήριξη του μικροελεγκτή. Συνδέεται στον υπολογιστή μέσω USB και είναι συμβατό με τα περισσότερα Shields (πλακέτες επέκτασης) σχεδιασμένα για το Arduino.

Πίνακας 4. Συνοπτικά χαρακτηριστικά Arduino Mega 2560

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

3.5.2 Τροφοδοσία

Το Arduino Mega μπορεί να τροφοδοτηθεί με ρεύμα είτε από τον υπολογιστή μέσω της σύνδεσης USB, είτε από εξωτερική τροφοδοσία που παρέχεται μέσω μιας υποδοχής φις των 2.1mm που βρίσκεται στην κάτω αριστερή γωνία. Η επιλογή πηγής γίνεται αυτόματα. Για την αποφυγή προβλημάτων η εξωτερική τροφοδοσία θα πρέπει να είναι από 7 ως 12V.

Οι ακροδέκτες τροφοδοσίας είναι οι ακόλουθοι:

- **Vin:** Είναι ο ακροδέκτης μέσω του οποίου γίνεται η τροφοδοσία όταν η πλακέτα χρησιμοποιεί εξωτερική πηγή ενέργειας. Η τροφοδοσία τάσης γίνεται μέσω αυτού του ακροδέκτη.
- **5V:** Είναι η σταθεροποιημένη τάση που χρησιμοποιείται για την τροφοδοσία του μικροελεγκτή και των άλλων στοιχείων της πλακέτας.
- **3,3V**: Παρέχει τάση 3,3V που δημιουργείται από το FTDI Chip με μέγιστη παροχή ρεύματος 50mA.
- **GND:** Είσοδοι γείωσης.
- **IOREF:** Αυτός ο ακροδέκτης χρησιμοποιείται για την τάση αναφοράς της πλακέτας.

3.5.3 Μνήμη

Ο μικροεπεξεργαστής ATmega2560 έχει τρεις ομάδες μνήμης.

- SRAM
- EEPROM
- Flash memory

SRAM μνήμης 8 KB: Είναι η ωφέλιμη μνήμη που μπορούν να χρησιμοποιήσουν τα προγράμματα για να αποθηκεύουν μεταβλητές, πίνακες κ.λ.π. Η μνήμη χάνει τα δεδομένα της όταν η παροχή ρεύματος στο Arduino σταματήσει ή πατηθεί το κουμπί επανεκκίνησης.

EEPROM μνήμης 4 KB: Μπορεί να χρησιμοποιηθεί για εγγραφή ή ανάγνωση δεδομένων από τα προγράμματα. Σε αντίθεση με την SRAM δε χάνει τα περιεχόμενά της σε περίπτωση απώλεια της τροφοδοσίας ή επανεκκίνησης.

Flash μνήμης 256 KB: 8 KB χρησιμοποιούνται από το Firmware του Arduino που έχει εγκαταστήσει ήδη ο κατασκευαστής του. Το Firmware είναι αναγκαίο για την εγκατάσταση προγραμμάτων στο μικροελεγκτή μέσω της θύρας USB. Τα υπόλοιπα 248KB της μνήμης Flash χρησιμοποιούνται για την αποθήκευση αυτών ακριβώς των προγραμμάτων, αφού πρώτα μεταγλωττιστούν στον υπολογιστή. Η μνήμη Flash δε χάνει τα περιεχόμενά της με την απώλεια της τροφοδοσίας ή με την επανεκκίνηση.

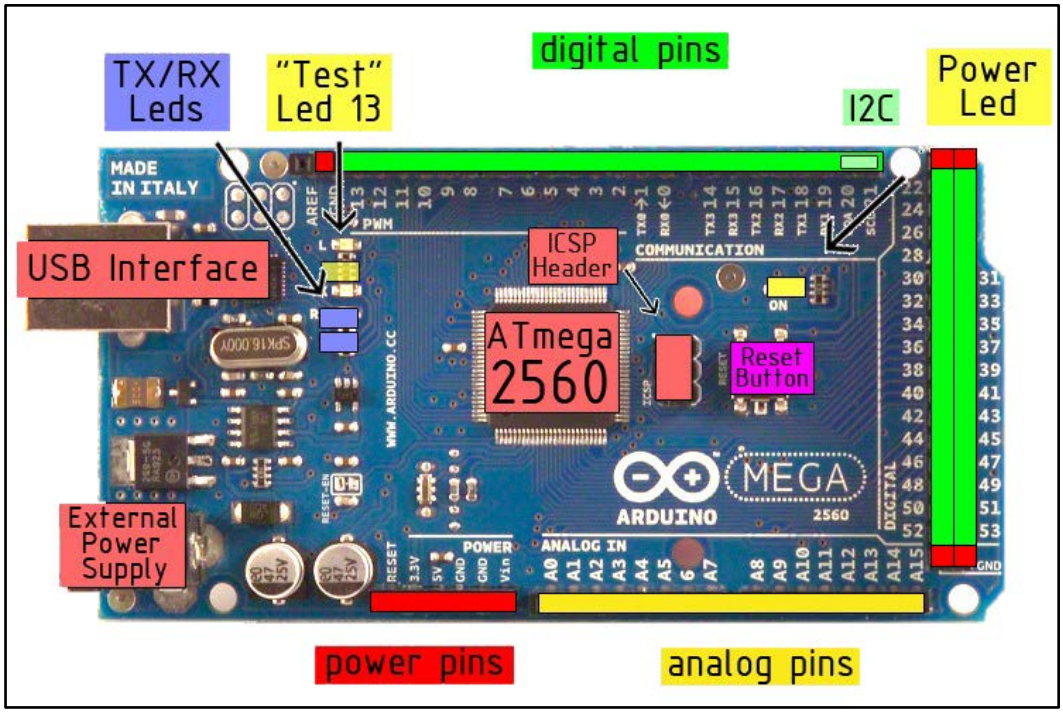
3.5.4 Ακροδέκτες

Η κάθε μια από τις 54 ψηφιακές εισόδους – εξόδους μπορεί να χρησιμοποιηθεί σε συνδυασμό με τις εντολές pinMode(), digitalWrite(), digitalRead(). Η κάθε μία από αυτές λειτουργεί στα 5V και κάθε pin μπορεί να παρέχει ή να λάβει μέχρι και 40mA και έχει εσωτερική αντίσταση από 20 – 50 ΚΩ.

Συγκεκριμένα, υπάρχουν κάποια pins που έχουν ξεχωριστές ιδιότητες:

- **Τα pin Serial:** 0 (RX) και 1 (TX); Serial 1: 19 (RX) και 18 (TX); Serial 2: 17 (RX) και 16 (TX); Serial 3: 15 (RX) και 14 (TX) χρησιμοποιούνται για να λαμβάνουν (RX) ή να στέλνουν (TX) σειριακή πληροφορία (TTL 5V).
- **Εξωτερικά Interrupts:** Τα Interrupts είναι διακοπές της ομαλής ροής του προγράμματος: 2 (Interrupt 0), 3 (Interrupt 1), 18 (Interrupt 5), 19 (Interrupt 4), 20 (Interrupt 3), και 21 (Interrupt 2). Αυτά τα pin μπορούν να ρυθμιστούν για να ξεκινήσουν κάποιο Interrupt χρησιμοποιώντας την εντολή attachInterrupt().
- **PWM:** Τα pin 2 έως 13 και 44 έως 46. Παρέχουν 8 bit PWM (Pulse width Modulation -είναι μια τεχνική για τον έλεγχο της τάσης) έξοδο χρησιμοποιώντας την εντολή analogWrite().
- **SPI:** 50 (MISO), 51 (MOSI), 52 (SCK), 53 (SS). Αυτά τα pin υποστηρίζουν επικοινωνία τύπου SPI .
- **LED:** Το pin 13 είναι συνδεδεμένο με ένα LED που είναι ενσωματωμένο στην πλακέτα.
- **I2C:** 20 (SDA) και 21 (SCL).

Ο Atmega2560 έχει 16 αναλογικές εισόδους κάθε μια από τις οποίες έχει ανάλυση 10 bits (δηλαδή 1024 διαφορετικές τιμές). Σαν προεπιλεγμένη κατάσταση διαβάζουν από 0 – 5 V, ωστόσο υπάρχει η δυνατότητα αλλαγής του εύρους, χρησιμοποιώντας το pin AREF και την εντολή analogReference().



Εικόνα 5. Arduino Mega pin layout

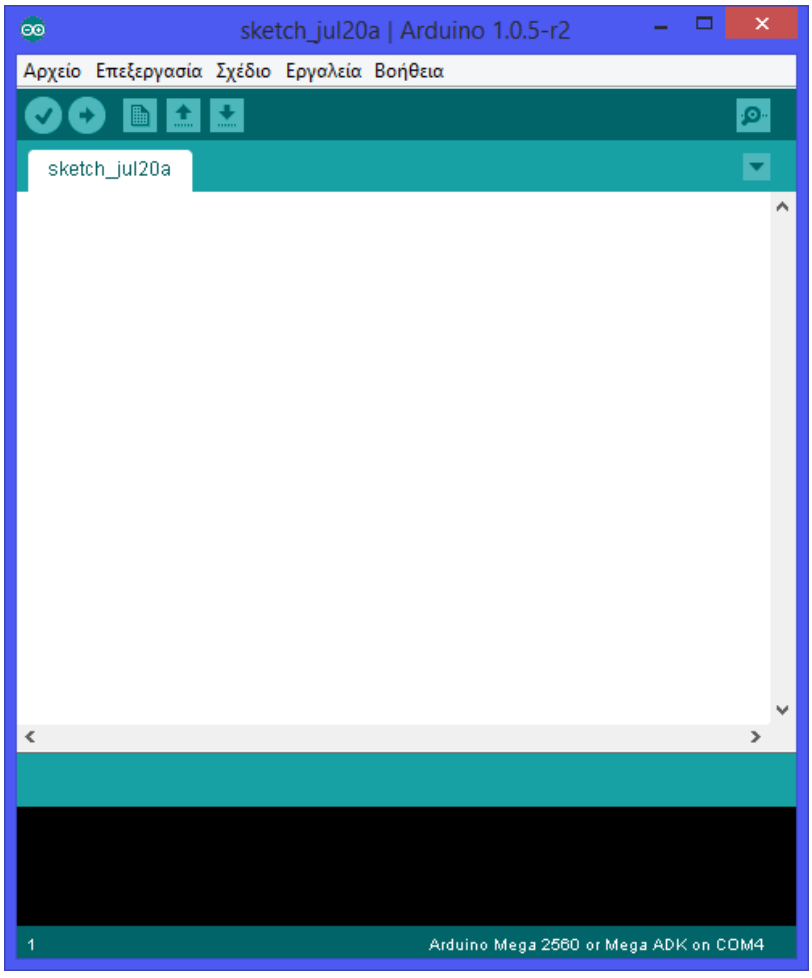
3.5.5 Λογισμικό

Το **IDE** του Arduino είναι γραμμένο σε Java και μπορεί να τρέξει σε πολλαπλές πλατφόρμες. Περιλαμβάνει επεξεργαστή κώδικα (επεξεργαστή κειμένου με διάφορα

εύχρηστα εργαλεία) και μεταγλωττιστή και έχει την ικανότητα να φορτώνει εύκολα το πρόγραμμα μέσω σειριακής θύρας από τον υπολογιστή στην πλακέτα.

Το περιβάλλον ανάπτυξης είναι βασισμένο στην Processing, ένα περιβάλλον ανάπτυξης σχεδιασμένο να εισάγει στον προγραμματισμό μη εξοικειωμένους χρήστες με την ανάπτυξη λογισμικού. Η συγκεκριμένη γλώσσα προγραμματισμού προέρχεται από την Wiring, μια γλώσσα που μοιάζει με την C, η οποία παρέχει παρόμοια λειτουργικότητα για μια πιο περιορισμένης σχεδίασης πλακέτα, της οποίας το περιβάλλον ανάπτυξης βασίζεται επίσης στην Processing. Για Compiler χρησιμοποιείται ο AVR gcc και ως βασική βιβλιοθήκη C χρησιμοποιείται η AVR libc.

Λόγω της καταγωγής της από την C, στην γλώσσα του Arduino μπορούν να χρησιμοποιηθούν ουσιαστικά οι ίδιες βασικές εντολές και συναρτήσεις, με την ίδια σύνταξη, τους ίδιους τύπων δεδομένων και τους ίδιους τελεστές, όπως και στην C. Πέρα από αυτές όμως υπάρχουν κάποιες ειδικές εντολές, συναρτήσεις και σταθερές, που βοηθούν στη διαχείριση του ειδικού Hardware του Arduino.



Εικόνα 6. Arduino IDE

3.5.6 Σύνολο Εντολών

Ο κώδικας των προγραμμάτων του Arduino μπορεί να χωριστεί σε 3 κύρια μέρη: Δομή (Structure), Τιμές (Μεταβλητές και Σταθερές), και Εντολές (Functions).

Πίνακας 5. Σύνολο Εντολών

Structure
• setup() • loop()
Control Structures
• if • if...else • for • switch case • while • do... while • break • continue • return • goto
Further Syntax
• ; (semicolon) • {} (curly braces) • // (single line comment) • /* */ (multi-line comment) • #define • #include
Arithmetic Operators
• = (assignment operator)

Variables Constants
• HIGH | LOW • INPUT | OUTPUT | INPUT_PULLUP • LED_BUILTIN • true | false • integer constants • floating point constants
Data Types
• void • boolean • char • unsigned char • byte • int • unsigned int • word • long • unsigned long • short • float • double • string - char array • String - object array
Conversion
• char()

Functions Digital I/O
• pinMode() • digitalWrite() • digitalRead()
Analog I/O
• analogReference() • analogRead() • analogWrite() - *PWM*
Advanced I/O
• tone() • noTone() • shiftOut() • shiftIn() • pulseIn()
Time
• millis() • micros() • delay() • delayMicroseconds()
Math
• min() • max() • abs() • constrain() • map()

- + (addition)
- - (subtraction)
- * (multiplication)
- / (division)
- % (modulo)

Comparison Operators

- == (equal to)
- != (not equal to)
- < (less than)
- > (greater than)
- <= (less than or equal to)
- >= (greater than or equal to)

Boolean Operators

- && (and)
- || (or)
- ! (not)

Pointer Access Operators

- * dereference operator
- & reference operator

Bitwise Operators

- & (bitwise and)
- | (bitwise or)
- ^ (bitwise xor)

- byte()
- int()
- word()
- long()
- float()

Variable Scope & Qualifiers

- variable scope
- static
- volatile
- const

Utilities

- sizeof()

- pow()
- sqrt()

Trigonometry

- sin()
- cos()
- tan()

Random Numbers

- randomSeed()
- random()

Bits and Bytes

- lowByte()
- highByte()
- bitRead()
- bitWrite()
- bitSet()
- bitClear()
- bit()

External Interrupts

- attachInterrupt()
- detachInterrupt()

Interrupts

- interrupts()
- noInterrupts()

Communication

- Serial Stream

- ~ (bitwise not)
- << (bitshift left)
- >> (bitshift right)

Compound Operators

- ++ (increment)
- -- (decrement)
- += (compound addition)
- -= (compound subtraction)
- *= (compound multiplication)
- /= (compound division)
- &= (compound bitwise and)
- |= (compound bitwise or)

3.6. Ανάλυση ADC

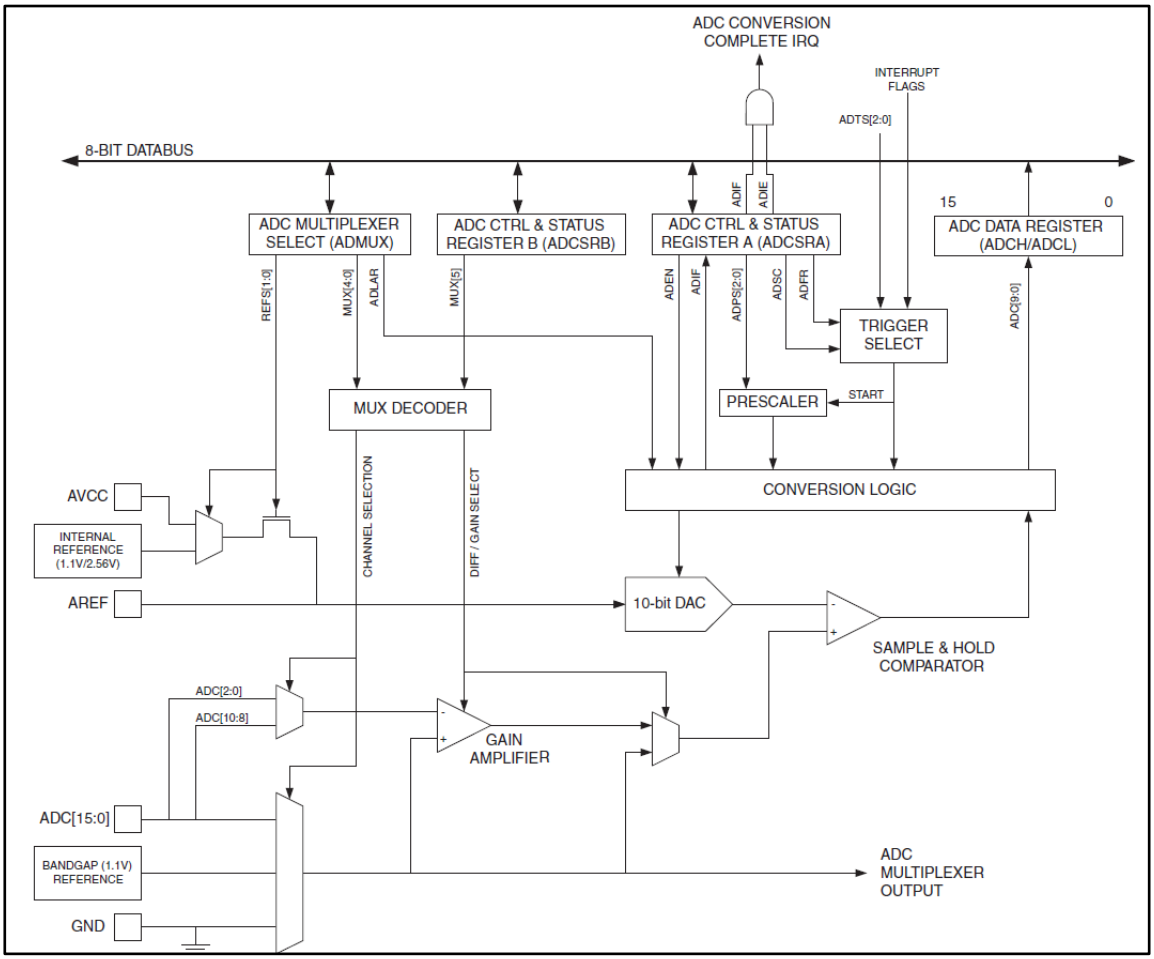
Ένας μετατροπέας αναλογικού σήματος σε ψηφιακό **(ADC)** είναι μια συσκευή η οποία μετατρέπει μια συνεχή ποσότητα ρεύματος σε μία τιμή με ψηφιακή αναπαράσταση σε διακριτό χρόνο. Ένας ADC μπορεί επίσης να παρέχει μια μεμονωμένη μέτρηση. Η αντίστροφη λειτουργία εκτελείται από ψηφιακό σε αναλογικό μετατροπέα (**DAC).**

3.6.1 Γενικά

Τυπικά ένας ADC είναι μια ηλεκτρονική συσκευή που μετατρέπει ένα αναλογικό σήμα τάσης ή ρεύματος σε έναν ψηφιακό αριθμό ανάλογο με το μέγεθος της τάσης ή του ρεύματος. Η ψηφιακή έξοδος μπορεί να χρησιμοποιήσει διαφορετικά συστήματα κωδικοποίησης. Η ψηφιακή έξοδος θα είναι ένας αριθμός δυαδικός (συμπλήρωμα ως προς 2) που είναι ανάλογος με την είσοδο. Με αυτόν τον τρόπο παίρνουμε τις αναλογικές τιμές εισόδου του αισθητήρα και τις μετατρέπουμε σε ψηφιακές, έτσι ώστε να μπορούν να αποτυπωθούν στην οθόνη lcd ή στην οθόνη του υπολογιστή μας.

Η διάρκεια ανάγνωσης αναλογικής εισόδου είναι περίπου 100 microseconds (0.0001 s) και η μέγιστη συχνότητα ανάγνωσης είναι 10.000 φορές το δευτερόλεπτο.

Εδώ βλέπουμε το Schematic Diagram του ADC του Atmega2560.



Εικόνα 7. Σχηματικό διάγραμμα λειτουργιάς του ADC

Ο Atmega2560 έχει δύο Registers, τους οποίους πρέπει να προγραμματίσουμε, έτσι ώστε να κάνουμε τη μετατροπή του αναλογικού σήματος σε ψηφιακό με τον ADC. Αυτοί οι δύο οι καταχωρητές είναι οι: ADMUX και ADCSRA, οι οποίοι είναι 8bit και το κάθε bit τους πρέπει να προγραμματιστεί ανάλογα με τη λειτουργία που θέλουμε να επιτελέσουμε. Παρακάτω βλέπουμε αυτούς τους καταχωρητές και εξηγούμε επιγραμματικά τη λειτουργία του κάθε bit τους.

ADMUX – ADC Multiplexer Selection Register

Bit	7	6	5	4	3	2	1	0	
(0x7C)	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0	ADMUX
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Εικόνα 8.Καταχωρητής ADMUX

Bit 0-4: Επιλέγουμε σε ποια αναλογική είσοδο θα πάρουμε το σήμα.

Bit 5: Δείχνει αν τα bit θα μπουν με το πιο σημαντικό δεξιά ή αριστερά (LSB-MSB).

Bit 6-7: Ορίζουν την τάση αναφοράς που θα χρησιμοποιηθεί στη μετατροπή.

ADCSRA – ADC Control and Status Register A

Bit	7	6	5	4	3	2	1	0	
(0x7A)	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	ADCSRA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Εικόνα 9. Καταχωρητής ADCSRA

Bit 0 – 2: Επιλέγουμε με τι ανάλυση θέλουμε να πραγματοποιηθεί η μετατροπή.

Bit 3: Bit ενεργοποίησης Interrupts κατά την ολοκλήρωση της μετατροπής.

Bit 4: Γίνεται 1 κατά την ολοκλήρωση της μετατροπής.

Bit 5: Καθορίζει την αυτόματη εκκίνηση της μετατροπής κατά την ανίχνευση Rising Edge.

Bit 6: Ορίζει την εκκίνηση της μετατροπής.

Bit 7: Ενεργοποίηση και απενεργοποίηση του ADC.

3.6.2 Ευκρίνεια – Επίπεδο Κβάντισης

Η αντιστοίχιση των αναλογικών τιμών σε ψηφιακές γίνεται ως εξής:

Ο μετατροπέας είναι των 10 bit. Στα ψηφιακά κυκλώματα αυτό σημαίνει ότι η ελάχιστη τιμή σε bit είναι η μηδενική και σε δυαδική μορφή θα είναι το 0000000000, δηλαδή θα παριστάνεται από 10 μηδενικά. Αντίστοιχα η μέγιστη τιμή θα είναι η δυαδική ψηφιακή λέξη 1111111111, δηλαδή θα παριστάνεται με 10 άσσους.

Αρχικά η μέγιστη τιμή των 5 V θα διαιρεθεί με τη μέγιστη δυαδική τιμή του μετατροπέα. Έτσι, θα φανεί η διακριτή και όχι η συνεχής, όπως θα περίμενε κάποιος, μεταβολή που εισάγει ένας μετατροπέας από αναλογικό σε ψηφιακό σήμα. Κατά αυτόν τον τρόπο βρίσκεται το ελάχιστο βήμα μετατροπής των αναλογικών τιμών σε ψηφιακές τιμές.

Η ψηφιακή δυαδική λέξη 1111111111 είναι σε δεκαδική τιμή ίση με 1024. Οπότε διαιρώντας τα 5 V με το 1024 λαμβάνουμε την τιμή του επιπέδου κβάντισης του μετατροπέα, δηλαδή τη διακριτική ικανότητα του ADC, η οποία είναι:

5 /1024 = 4,8828125 x10^{-3} ~= 4,9 mV.

3.7 Κάρτα επέκτασης δικτύου (Ethernet Shield)

Η πλακέτα Arduino Ethernet Shield συνδέει το Arduino με το Internet. Αυτή συνδέεται στο Arduino μέσω της θύρας SPI. Η τάση λειτουργίας της είναι 5V, τα οποία προσφέρει η πλακέτα του Arduino. Η πλακέτα Ethernet διαθέτει έναν ελεγκτή Ethernet W5100 με ενσωματωμένη 16kB μνήμη. Η ταχύτητα επικοινωνίας είναι της μορφής 10/100 Mb. Το ολοκληρωμένο W5100 της Wiznet παρέχει τη στοίβα δικτύου (IP) που είναι ικανή για υλοποίηση προτύπων TCP και UDP.

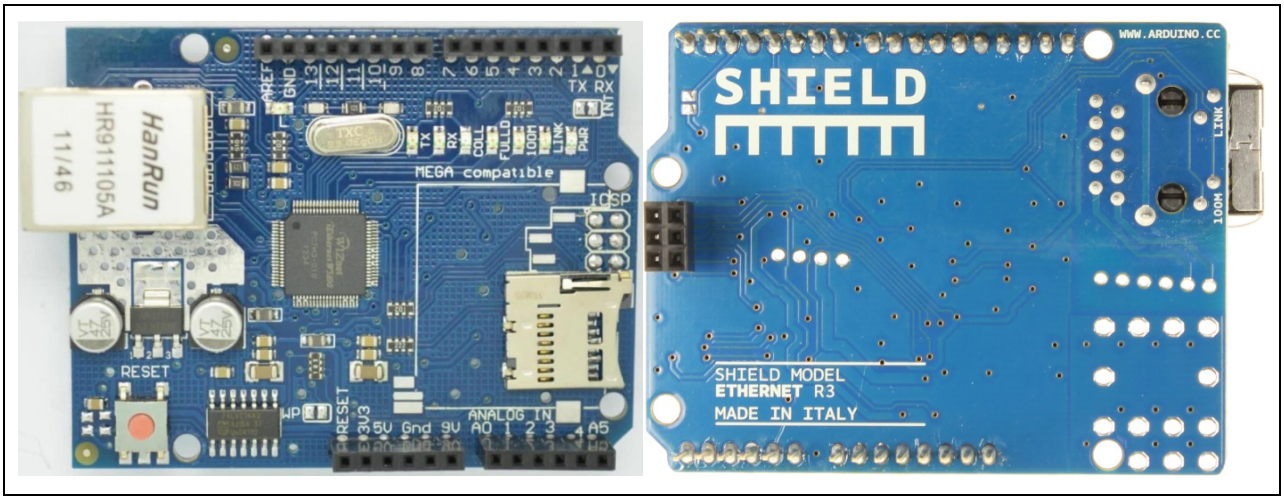
Η πλακέτα Ethernet υποστηρίζει μέχρι και 4 ταυτόχρονες συνδέσεις τύπου Socket. Με τη χρήση της βιβλιοθήκης Ethernet μπορούν να γραφτούν προγράμματα που θα έχουν πρόσβαση στο Internet. Διαθέτει βύσμα τύπου RJ45 για καλώδιο Ethernet και κουμπί για επανεκκίνηση της λειτουργίας της. Επίσης διαθέτει ενσωματωμένη υποδοχή για κάρτα μνήμης τύπου micro-SD η οποία μπορεί να αξιοποιηθεί για αποθήκευση αρχείων που θα χρησιμοποιούνται ή θα διακινούνται στο δίκτυο.

Η υποδοχή της κάρτας μνήμης micro-SD μπορεί επίσης να διαβάσει την κάρτα μνήμης με χρήση της βιβλιοθήκης SD. Το Arduino επικοινωνεί με την πλακέτα Ethernet μέσω των θυρών SPI. Αυτές οι θύρες είναι οι 11, 12 και 13.

Η πλακέτα διαθέτει και μια σειρά από Led, ρόλος των οποίων είναι να παράγουν βοηθητικές ενδείξεις.

- ✓ **PWR**: Το Led αυτό δείχνει ότι η πλακέτα Ethernet έχει τροφοδοσία.
- ✓ **LINK**: Το Led αυτό δείχνει αν υπάρχει συνδεδεμένο καλώδιο δικτύου στην πλακέτα και επίσης αναβοσβήνει όταν η πλακέτα δέχεται ή στέλνει δεδομένα.
- ✓ **FULLD**: Αυτό το Led δείχνει αν η σύνδεση είναι πλήρως αμφίδρομη.

- ✓ **100Mb**: Αυτό το Led δείχνει την ταχύτητα της σύνδεσης στο δίκτυο. Αν είναι αναμμένο, αυτό σημαίνει ότι υποστηρίζεται ταχύτητα 100Mb.
- ✓ **RX**: Αυτό το Led αναβοσβήνει όταν η πλακέτα λαμβάνει δεδομένα.
- ✓ **TX**: Αυτό το Led αναβοσβήνει όταν η πλακέτα στέλνει δεδομένα.
- ✓ **COLL**: Το Led αυτό αναβοσβήνει όταν εντοπιστεί κάποια σύγκρουση στο δίκτυο.



Εικόνα 10. Εμπρόσθια και οπίσθια όψη του Ethernet Shield

Στην παρούσα εργασία το κύκλωμα έχει τοποθετηθεί σε ενσύρματο δίκτυο τύπου Ethernet (οικιακό δίκτυο), μέσω του οποίου γίνεται η πρόσβαση από όλες τις συσκευές ελέγχου του τοπικού δικτύου. Συγκεκριμένα χρησιμοποιείται το πρωτόκολλο TCP/IP.

Κατά την εκκίνηση του συστήματος γίνεται αρχικοποίηση των ρυθμίσεων της κάρτας δικτύου. Αρχικά εισάγουμε τη βιβλιοθήκη Ethernet.h, η οποία βρίσκεται προεγκατεστημένη στο IDE του Arduino και περιέχει όλες τις συναρτήσεις για τη διασύνδεση της κάρτας με το Arduino. Στη συνέχεια ορίζουμε τη διεύθυνση IP του δικτύου, το υποδίκτυο στο οποίο ανήκει το σύστημα και τη διεύθυνση MAC της κάρτας. Κατόπιν αρχικοποιούμε (instantiate) το αντικείμενο και ορίζουμε την πόρτα στην οποία θα ανταποκρίνεται ο εξυπηρετητής. Εξ' ορισμού η πόρτα στην οποία ανταποκρίνονται όλες τα αιτήματα του HTTP πρωτοκόλλου είναι η 80. Στη συνάρτηση setup() γίνεται η αρχικοποίηση και η έναρξη λειτουργίας της κάρτας. Οι συναρτήσεις της βιβλιοθήκης Ethernet.h είναι οι εξής:

- ❖ Εντολές για την αρχικοποίηση της βιβλιοθήκης

- ➢ **Ethernet.begin():** Αρχικοποιεί τη βιβλιοθήκη Ethernet και τις ρυθμίσεις του δικτύου. Με την έκδοση 1.0 η βιβλιοθήκη υποστηρίζει DHCP. Χρησιμοποιώντας Ethernet.begin(mac) με τη σωστή εγκατάσταση του δικτύου, το Ethernet shield αποκτά αυτόματα μια διεύθυνση IP. Αυτό όμως αυξάνει σημαντικά το μέγεθος του προγράμματος.
  - **Σύνταξη:**

    Ethernet.begin (mac);

Ethernet.begin (mac, ip);

Ethernet.begin (mac, ip, dns);

Ethernet.begin (mac, ip, dns, gateway);

Ethernet.begin (mac, ip, dns, gateway, subnet);

- **Παράμετροι:**

  MAC: Η MAC (Media Access Control) διεύθυνση της συσκευής (πίνακας 6 bytes) αποτελεί την Ethernet διεύθυνση υλικού του Shield. Τα νεώτερα Arduino Shields Ethernet περιλαμβάνουν ένα αυτοκόλλητο με τη διεύθυνση MAC της συσκευής.

  IP: Η διεύθυνση IP της συσκευής (πίνακας 4 bytes).

  DNS: Η διεύθυνση για ένα διακομιστή DNS.

  Gateway: Η διεύθυνση IP της πύλης του δικτύου (πίνακας 4 bytes).

  Subnet: Η μάσκα υποδικτύου του δικτύου (πίνακας 4 bytes).

- **Ethernet.localIP():** Αποκτά την IP διεύθυνση του Ethernet Shield. Είναι χρήσιμο όταν η διεύθυνση αποδίδεται αυτόματα μέσω του DHCP.
  - **Σύνταξη:** Ethernet.localIP();

- **IPAddress():** Ορίζει μια διεύθυνση IP. Μπορεί να χρησιμοποιηθεί για να δηλώσει τόσο τις τοπικές όσο και τις απομακρυσμένες διευθύνσεις.
  - **Σύνταξη:** IPAddress (address);
  - **Παράμετροι:**

    address: μια λίστα οριοθετημένη με κόμματα που αντιπροσωπεύει τη διεύθυνση (4 bytes, πχ 192, 168, 1, 1).

❖ Εντολές για τη δημιουργία διακομιστών

- **Server():** Δημιουργεί ένα διακομιστή που ακούει για εισερχόμενες συνδέσεις στη συγκεκριμένη θύρα.
  - **Σύνταξη:** Server(port);
  - **Παράμετροι:**

    Port: Η θύρα στην οποία ακούει (int).

- **begin():** Λέει στο διακομιστή να αρχίσει να ακούει για εισερχόμενες συνδέσεις.
  - **Σύνταξη:** server.begin();

- **available():** Παίρνει δεδομένα που είναι διαθέσιμα για ανάγνωση από έναν πελάτη που συνδέεται με το διακομιστή. Η σύνδεση παραμένει μέχρι τα επιστρεφόμενα δεδομένα από τον πελάτη τελειώσουν.
  - **Σύνταξη:** server.available();

- **write():** Γράφει δεδομένα για όλους τους πελάτες που συνδέονται σε ένα διακομιστή.
  - **Σύνταξη:** server.write(data);
  - **Παράμετροι:**

    data: Η τιμή που γράφει (Byte ή Char).

- **print() και println():** Εκτυπώνει δεδομένα σε όλους τους πελάτες που συνδέονται σε ένα διακομιστή. Η println() προσθέτει και μια αλλαγή γραμμής.
  - **Σύνταξη:**
    server.print(data);

    server.print (data, BASE);
  - **Παράμετροι:**

    data: Τα δεδομένα προς εκτύπωση (char, byte, int, long, ή string).

    BASE (Προαιρετικό): Η βάση στην οποία θα εκτυπωθούν οι αριθμοί: BIN για δυαδικό (βάση 2), DEC για δεκαδικό (βάση 10), OCT για οκταδικό (βάση 8), HEX για δεκαεξαδικό (βάση 16).

❖ Εντολές για τη δημιουργία πελατών

- **Client():** Δημιουργεί έναν πελάτη ο οποίος μπορεί να συνδεθεί στο Internet μέσω μιας καθορισμένης διεύθυνσης IP θύρας.
  - **Σύνταξη:** Client(IP, port);
  - **Παράμετροι:**

    IP: Η διεύθυνση IP με την οποία θα συνδεθεί ο πελάτης (πίνακας 4 bytes).

    port: Η θύρα με την οποία θα συνδεθεί ο πελάτης (int).

- **connected():** Το εάν ή όχι ο πελάτης έχει συνδεθεί.
  - **Σύνταξη:** client.connected();

- **connect():** Συνδέεται με τη διεύθυνση IP και τη θύρα που καθορίστηκε κατά τη δημιουργία. Η τιμή επιστροφής δείχνει την επιτυχία ή την αποτυχία.
  - **Σύνταξη:** client.connect();

- **write():** Γράφει δεδομένα στο διακομιστή στον οποίο ο πελάτης είναι συνδεδεμένος.
  - **Σύνταξη:** client.write(data);
  - **Παράμετροι:**

    data: Η τιμή που πρόκειται να γραφτεί.

- **print() και println():** Εκτυπώνει δεδομένα στο διακομιστή στον οποίο ο πελάτης είναι συνδεδεμένος. Η println() προσθέτει και μια αλλαγή γραμμής.
  - **Σύνταξη:**
    client.print(data);

    client.print (data, BASE);
  - **Παράμετροι:**

    data: Τα δεδομένα προς εκτύπωση (char, byte, int, long, ή string).

    BASE (Προαιρετικό): Η βάση στην οποία θα εκτυπωθούν οι αριθμοί: BIN για δυαδικό (βάση 2), DEC για δεκαδικό (βάση 10), OCT για οκταδικό (βάση 8), HEX για δεκαεξαδικό (βάση 16).

- **available():** Επιστρέφει τον αριθμό των bytes που διατίθενται για ανάγνωση (δηλαδή η ποσότητα των δεδομένων που έχει γραφτεί για τον πελάτη από το διακομιστή που είναι συνδεδεμένος).
  - **Σύνταξη:** client.available();

- **read():** Διαβάζει τα επόμενα bytes που λαμβάνονται από το διακομιστή στον οποίο είναι συνδεδεμένος ο πελάτης.
  - **Σύνταξη:** client.read();

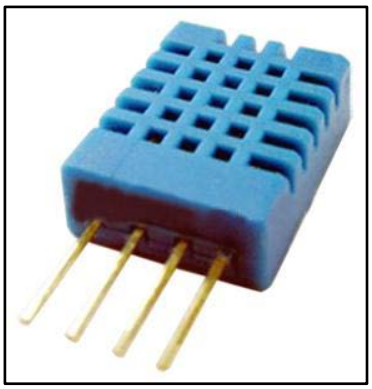
- **flush():** Απορρίπτει τα bytes που έχουν γραφτεί για τον πελάτη, αλλά δεν έχει ακόμη διαβαστεί.
  - **Σύνταξη:** client.flush();

- **stop():** Αποσυνδέεται από το διακομιστή.
  - **Σύνταξη:** client.stop()

Κεφάλαιο 4 :Αισθητήρες

4.1. Αισθητήρας Θερμοκρασίας / Υγρασίας DHT-11

Ο αισθητήρας DHT 11 είναι ένας οικονομικός και αξιόπιστος αισθητήρας. Έχει μικρό μέγεθος, μικρή κατανάλωση ρεύματος και μεγάλη ευαισθησία. Εξαιτίας αυτών των χαρακτηριστικών έχει χρησιμοποιηθεί σε πολλές εφαρμογές και υπάρχει πληθώρα βιβλιοθηκών στο διαδίκτυο.

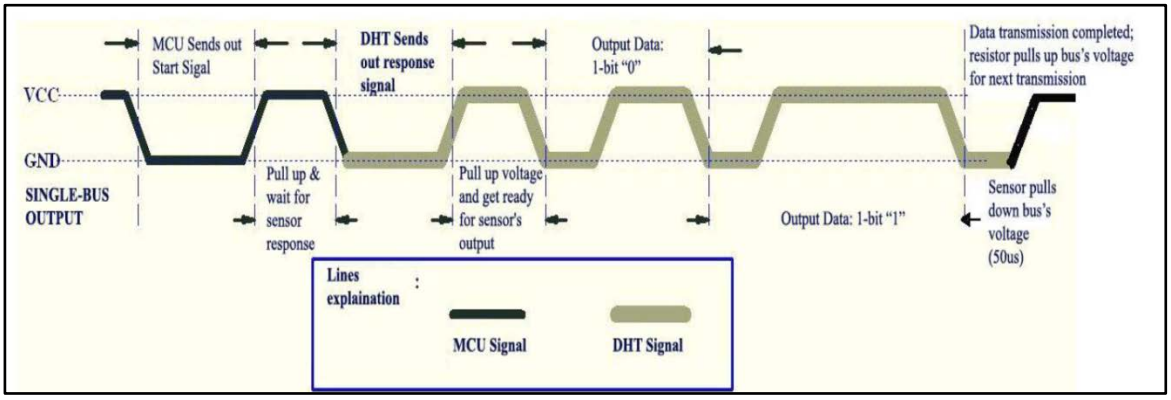


Εικόνα 11. Ο αισθητήρας DHT-11

Σύμφωνα με τα χαρακτηριστικά του, η ευαισθησία του αισθητήρα θερμοκρασίας για την καταγραφή της θερμοκρασία του περιβάλλοντος έχει οριστεί από 0°C έως 50°C και για την υγρασία από 20% έως 90%. Για το λόγο αυτό, η χρήση του εν λόγω αισθητήρα είναι η ιδανική, αν ληφθεί υπόψη και το χαμηλό κόστος του.

4.1.1 Τρόπος επικοινωνίας

Για την επικοινωνία του αισθητήρα με το Arduino χρησιμοποιείται το πρωτόκολλο One Wire. Η διαδικασία επικοινωνίας φαίνεται αναλυτικά στο παρακάτω διάγραμμα:



Εικόνα 12. Διάγραμμα επικοινωνίας του αισθητήρα DHT11

Για την έναρξη της επικοινωνίας, το Arduino στέλνει ένα Low σήμα για τουλάχιστον 18ms και στη συνέχεια ένα High σήμα για 20 – 40us. Μόλις το αισθητήριο λάβει αυτή την ακολουθία στέλνει ένα Low σήμα για 80us και ένα High για 80us. Κατόπιν αποστέλλει τα δεδομένα. Τέλος, μόλις ολοκληρωθεί η μεταφορά, ο αισθητήρας ελευθερώνει το κανάλι, το οποίο επιστέφει σε High κατάσταση λόγο της Pull up αντίστασης.

Ο αισθητήρας εκπέμπει 40 bit πληροφορίας, τα οποία αντιστοιχούν σε:

• 8 bit για την τιμή της υγρασίας,

• 8 bit για την ακέραια τιμή της υγρασίας,

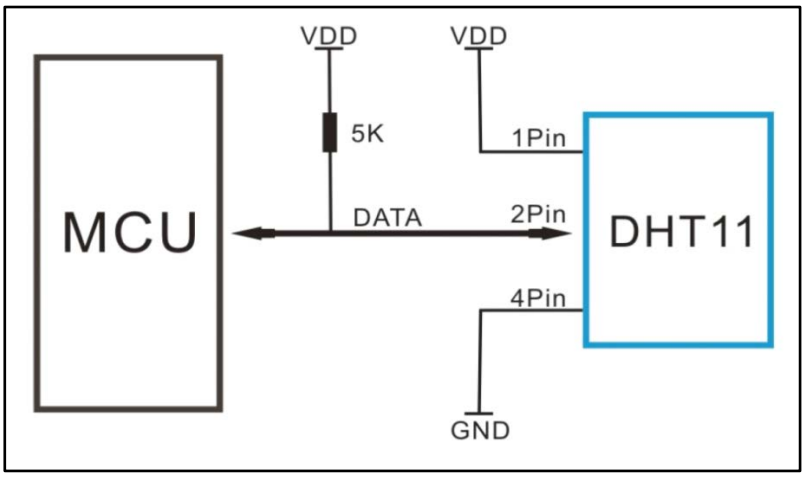
• 8 bit για την τιμή της θερμοκρασίας,

• 8 bit για την ακέραια τιμή της θερμοκρασίας και

• 8 bit για το bit ελέγχου ισοτιμίας των δεδομένων.

4.1.2 Συνδεσμολογία με την πλακέτα Arduino

Η συνδεσμολογία μεταξύ του αισθητήρα και της πλακέτας Arduino είναι αρκετά απλή. Στον ακροδέκτη #1 του αισθητήρα συνδέεται η τάση (+5 V ή 3,3 V), στον ακροδέκτη #4 η γείωση (Ground) και στον ακροδέκτη #2 μία από τις ψηφιακές εισόδους του Arduino. Στην παρακάτω εικόνα βλέπουμε τον τρόπο συνδεσμολογίας του αισθητήρα με την πλακέτα.



Εικόνα 13. Τρόπος σύνδεσης του DHT11 με το μικροεπεξεργαστή

4.1.3Χαρακτηριστικά

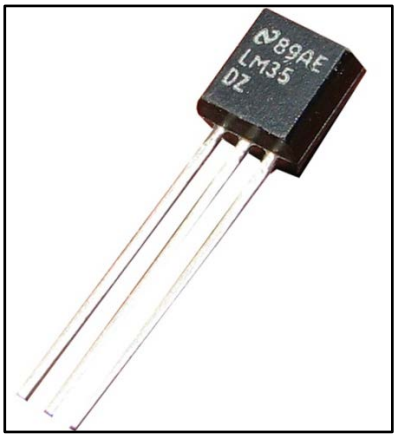
Πίνακας 6. Χαρακτηριστικά αισθητήρα DHT11

Parameters	Conditions	Minimum	Typical	Maximum
Humidity				
Resolution		1%RH	1%RH	1%RH
			8 Bit	
Repeatability			±1%RH	
Accuracy	25°C		±4%RH	
	0-50°C			±5%RH
Interchangeability	Fully Interchangeable			
Measurement Range	0C	30%RH		90%RH
	25°C	20%RH		90%RH
	50°C	20%RH		80%RH
Response Time (Seconds)	1/e(63%)25°C, 1m/s Air	6 S	10 S	15 S
Hysteresis			±1%RH	
Long – Term Stability	Typical		±1%RH/year	
Temperature				
Resolution		1°C	1°C	1°C
		8 Bit	8 Bit	8 Bit
Repeatability			±1°C	
Accuracy		±1°C		±2°C
Measurement Range		0°C		50°C
Response Time (Seconds)	1/e(63%)	6 S		30 S

Για την ανάγνωση των τιμών θερμοκρασίας και υγρασίας από τον αισθητήρα DHT-11 χρησιμοποιήθηκε η βιβλιοθήκη DHT11 v0.3.2. Αρχικά με την εντολή Read ζητούνται από τον αισθητήρα οι τιμές θερμοκρασίας και υγρασίας και στη συνέχεια με τις εντολές Humidity και Temperature επιστρέφονται οι τιμές της υγρασίας και της θερμοκρασίας αντίστοιχα.

4.2. Το ολοκληρωμένο LM35

Το ολοκληρωμένο κύκλωμα LM35 κατασκευάζεται από την εταιρεία National Semiconductor και αποτελεί έναν από τους πιο διαδεδομένους αισθητήρες θερμοκρασίας. Η λειτουργία του στηρίζεται στις ιδιότητες της επαφής *pnp*. Όπως είναι γνωστό, όταν η επαφή *pn* είναι πολωμένη ανάστροφα, τότε το ανάστροφο ρεύμα κόρου είναι συνάρτηση της θερμοκρασίας της επαφής. Τα ολοκληρωμένα αισθητήρια γενικά εμφανίζουν άριστη γραμμικότητα.

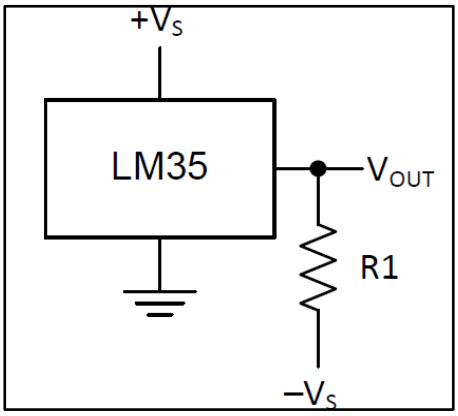


Εικόνα 14. Αισθητήρας θερμοκρασίας LM35

Το κύκλωμα LM35 μετρά θερμοκρασίες μεταξύ – 55 και +150°C και αποδίδει έξοδο περίπου 10,0 mV ανά βαθμό Κελσίου. Το ρεύμα που το διαρρέει είναι μόλις 60 μA και έτσι το κύκλωμα εμφανίζει σε εξαιρετικά χαμηλό βαθμό το φαινόμενο της αυτοθέρμανσης, δηλαδή περίπου 0,1°C όταν βρίσκεται σε ακίνητο αέρα. Επίσης μπορεί να τροφοδοτηθεί από ευρεία περιοχή τάσεων, μεταξύ + 4 και + 30V. Το κύκλωμα βαθμονομείται από την κατασκευή του και δεν απαιτεί επιπλέον ρυθμίσεις από το χρήστη, με αποτέλεσμα να εμφανίζει ακρίβεια 0,25°C σε θερμοκρασία δωματίου και 0,75°C στο θερμοκρασιακό εύρος – 55 έως +150°C. Έχει χαμηλή αντίσταση εξόδου (0,1 Ω για ρεύμα εξόδου 1mA) και μπορεί να συνδεθεί εύκολα με κυκλώματα διασύνδεσης ή ελέγχου.

4.2.1 Συνδεσμολογία του αισθητήρα LM35

Η συνδεσμολογία αυτού του αισθητήρα είναι πολύ απλή, καθώς το μόνο που χρειάζεται είναι τροφοδοσία στον ακροδέκτη #1 (+5V), γείωση στον ακροδέκτη #3 (Ground) και στον ακροδέκτη #2 μία από τις ψηφιακές εισόδους του Arduino, η οποία θα είναι συνδεδεμένη με μία αντίσταση και με τη γείωση. Στην παρακάτω εικόνα βλέπουμε τον τρόπο συνδεσμολογίας του αισθητήρα με την πλακέτα.



Εικόνα 15. Διάγραμμα συνδεσμολογίας του αισθητήρα θερμοκρασίας LM35

4.2.2 Τα Χαρακτηριστικά του LM35

• Γραμμική τάση εξόδου σε συνάρτηση με τη θερμοκρασία 10,0mV/°C.

• Βαθμονομημένος εκ των προτέρων σε βαθμούς Κελσίου. (Δε χρειάζεται μετατροπή από
βαθμούς Κέλβιν σε Κελσίου.)

• Κατανάλωση ρεύματος μικρότερη από 60μA.

• Τάση τροφοδοσίας 4 – 30 Volt.

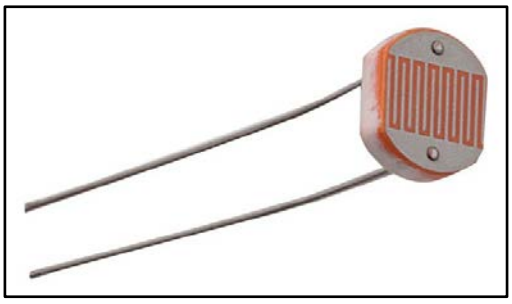
• Μεγάλη ακρίβεια με 0,5°C απόκλιση κατά τη μέτρηση της θερμοκρασίας ενός δωματίου.

• Κλίμακα μέτρησης από – 55 έως + 150°C.

• Χαμηλό κόστος.

4.3. Φωτοαντίσταση (Photoresistor) GL5549

Για τη μέτρηση του φωτός χρησιμοποιήθηκε ένας αισθητήρας φωτός τύπου φωτοαντίστασης.



Εικόνα 16. Φωτοαντίσταση

Όταν προσπίπτει φως σε ένα φωτοαγώγιμο υλικό, αυξάνει ο αριθμός των ελεύθερων ηλεκτρονίων του. Αυτό συμβαίνει επειδή τα προσπίπτοντα φωτόνια διεγείρουν ηλεκτρόνια που είναι δεσμευμένα και τα καθιστούν ελεύθερα. Έτσι η αντίσταση του φωτοαγώγιμου υλικού μειώνεται. Η σχέση μεταξύ της προσπίπτουσας φωτεινής ισχύος P και της αντίστασης R δεν είναι γραμμική αλλά λογαριθμική και έχει τη μορφή: $R = \frac{a}{P^b}$

Από αυτή τη σχέση καταλήγουμε στην εμπειρική σχέση logR = a – b log P

Οι ποσότητες a και b είναι χαρακτηριστικές του φωτοαγώγιμου υλικού. Με βάση αυτά τα υλικά κατασκευάζονται τα εμπορικά προϊόντα των φωτοαντιστάσεων (Photoresistors) ή αντιστάσεων εξαρτώμενων από φωτεινή ένταση (Light – Dependent Resistors – LDRs).

Η λειτουργία της φωτοαντίστασης βασίζεται στη μεταβολή της τιμής της, ανάλογα με την ένταση του φωτός που πέφτει πάνω της. Σε απόλυτο σκοτάδι η αντίσταση είναι η

μέγιστη. Εφαρμόζοντας όμως τάση στα άκρα της φωτοαντίστασης, εμφανίζεται ένα ρεύμα που οφείλεται κυρίως σε θερμική λειτουργία (Ρεύμα σκότους). Με την αύξηση της έντασης του φωτός, η αντίσταση του υλικού μικραίνει και επιτρέπει τη δημιουργία μεγάλων ρευμάτων.

4.3.1 Χαρακτηριστικά

Στον παρακάτω πίνακα βλέπουμε τα χαρακτηριστικά της φωτοαντίστασης.

Πίνακας 6. Πίνακας χαρακτηριστικών της φωτοαντίστασης

Model	Vmax (VDC)	Pmax (mW)	Ambient Temp (C)	Spectral Peak (nm)	Photo Resistance (10Lx) (KΩ)	Dark Resistance (MΩ)	γ min	Response Time (ms)	
								Rise	Decay
PGM5549	150	100	-30~+70	540	45-140	10.0	0.8	20	30

Κεφάλαιο 5: Κατασκευή

5.1. Μία σύντομη περιγραφή του κώδικα

Το σύστημα, μέσω του Ethernet Shield, ελέγχει τον ιστοχώρο, ο οποίος χρησιμοποιείται τόσο για την εμφάνιση ενδείξεων, όπως η θερμοκρασία, όσο και για τον έλεγχο διαφόρων συσκευών, όπως τα φώτα. Οι τιμές και οι καταστάσεις των μεταβλητών δεν αποθηκεύονται με αποτέλεσμα, όταν γίνεται επανεκκίνηση του Arduino, το σύστημα να επανέρχεται στις αρχικές του ρυθμίσεις.

Ο Client επικοινωνεί με τον ανά τακτά χρονικά διαστήματα, τα οποία ορίζονται στη συνάρτηση get_ajax_inputs. Κατά την πρώτη επικοινωνία αποστέλλεται η σελίδα με το γραφικό περιβάλλον που εμφανίζεται στο χρήστη, ενώ στη συνέχεια αποστέλλεται μόνο η σελίδα XLM, η οποία περιέχει τα δεδομένα τα οποία ανανεώνονται από τον Server. Αυτό γίνεται γιατί η αποστολή όλης της σελίδας θα είχε τεράστιες απαιτήσεις σε Bandwidth, κάτι το οποίο θα έκανε τη χρήση του συστήματος μέσω Internet ασύμφορη, λόγω της αυξημένης μεταφοράς δεδομένων.

Για τον έλεγχο των διαφόρων ενεργειών από το χρήστη, χρησιμοποιείται μια Ajax συνάρτηση σε συνδυασμό με την σελίδα HTML. Η συνάρτηση «συντάσσει» ένα αίτημα βασισμένο στην ενέργεια που επέλεξε ο χρήστης και το αποστέλλει στον Server μαζί με το αίτημα για τη σελίδα XLM.

Στην παρούσα εργασία η επικοινωνία γίνεται, όπως αναφέρθηκε προηγουμένως, μέσω του Ethernet Shield. Αυτό συνεπάγεται την ύπαρξη ενσύρματου τοπικού δικτύου για την πραγματοποίησή της. Εναλλακτικά θα μπορούσε να χρησιμοποιηθεί μια GSM Shield για τον έλεγχο του σπιτιού.

Μέσω των κατάλληλων ρυθμίσεων στο Router, στο οποίο είναι συνδεδεμένο το Arduino, είναι η δυνατή η χρήση του συστήματος μέσω Internet και όχι μόνο μέσω τοπικού δικτύου.

5.2. Εξήγηση του κώδικα

Ο κώδικας χωρίζεται στα εξής στάδια: κλήση βιβλιοθηκών, δήλωση μεταβλητών, ορισμός των βασικών συναρτήσεων setup() και loop() και ορισμός των υπολοίπων συναρτήσεων.

5.2.1 Κλήση βιβλιοθηκών

Αρχικά έχουμε την εισαγωγή των βιβλιοθηκών, όπως βλέπουμε στην παρακάτω εικόνα, ξεκινώντας με τις βιβλιοθήκες SPI και Ethernet. Η σειριακή περιφερειακή διεπαφή (SPI) είναι ένα σύγχρονο σειριακό πρωτόκολλο δεδομένων που χρησιμοποιείται από μικροελεγκτές για την επικοινωνία με μία ή περισσότερες συσκευές. Η εισαγωγή της είναι απαραίτητη καθώς η Ethernet Shield επικοινωνεί με το Arduino με αυτό το πρωτόκολλο. Στη συνέχεια εισάγουμε τη βιβλιοθήκη Ethernet, η οποία περιέχει όλες τις συναρτήσεις, οι οποίες θα χρειαστούν για την εκτέλεση του Web Server.

Η βιβλιοθήκη DHT χρησιμοποιείται για την εύκολη ανάγνωση δεδομένων από τον αισθητήρα DHT – 11.

```
#include <SPI.h>                                    //εισαγωγη βιβλιωθηκης spi
#include <Ethernet.h>                               //εισαγωγη βιβλιωθηκης ethernet
#include <dht.h>                                    //εισαγωγη βιβλιβθηκης για τον αισθητηρα dht

#define dht_dpin 30                                 //οριζουμε το πινακι στο οποιο ειναι συνδεδεμενος ο αισθητηρας
```

Εικόνα 17. Εικόνα προγράμματος Arduino

5.2.2 Δήλωση μεταβλητών

Στο επόμενο στάδιο δηλώνουμε τις μεταβλητές που θα χρησιμοποιήσουμε. Στην αρχή ορίζουμε την IP, MAC και PORT στην οποία θα εκτελείται ο Web Server. Συνεχίζουμε με τη δήλωση των μεταβλητών, οι οποίες θα χρησιμοποιηθούν αργότερα στο πρόγραμμα.

```
dht DHT;                                            //δημιουργουμε εναν αισθητηρα με ονομα DHT

byte mac[] = {                                      //η mac address που θα εχει η ethernet shield
  0xDE, 0xAD, 0xBE, 0xEE, 0xFE, 0xED };
IPAddress ip(192,168,2,100);                        //η ip address που θα εχει η ethernet shield
EthernetServer server(80);                          //η πορτα που θα εχει η ethernet shield

String readString;                                  //μεταβλητη για την αποθηκευση των αιτηματων απο web client
String extracted_data="";                           //μεταβλητη για την αποθηκευση της πρωτης εντολης που ληφθηκε απο τον client
String second_comand = "";                          //μεταβλητη για την αποθηκευση της δευτερης εντολης που ληφθηκε απο τον client

boolean lights_status_room1=false;                  //μεταβλητες για την δηλωση καταστασης φωτισμου
boolean lights_status_room2=false;
boolean lights_status_room3=false;
boolean boiler=false;                               //μεταβλητες για την δηλωση καταστασης θερμοσιφωνα
boolean garage_door_open=false;                     //μεταβλητες για την δηλωση καταστασης γκαραζοπορτας
boolean garage_door_close=false;
boolean room1_temp_control=false;                   //μεταβλητες για την δηλωση καταστασης αυτοματου κλιματισμου
boolean room2_temp_control=false;
boolean room3_temp_control=false;
```

Εικόνα 18. Εικόνα προγράμματος Arduino

```
float room1_temp;                              //μεταβητες για την αποθηκευση τιμων απο τους αισθητηρες
float room2_temp;
float room3_temp;
float room1_desired_temp;
float room2_desired_temp;
float room3_desired_temp;
int room1_light;
int room2_light;
int room3_light;
int out_temp;
int out_hum;

int room1_temp_pin = A7;                      //μεταβλητες για την αποθηκευση του ακροδεκτη στον οποιο ειναι συνδεδεμενος καθε αισθητηρας
int room2_temp_pin = A8;
int room3_temp_pin = A6;
int room1_lights_inten_pin = A10;
int room2_lights_inten_pin = A11;
int room3_lights_inten_pin = A9;
int garage_door_open_pin = 22;
int garage_door_close_pin = 23;
int room1_lights_pin = 26;
int room2_lights_pin = 24;
int room3_lights_pin = 25;
int boiler_pin = 27;
//ακροδεκτης 30 για τον αισθητηρα dht
int garage_door_left_stop_pin = 33;
int garage_door_right_stop_pin = 32;
int room1_heater_pin = 35;
int room1_cooling_pin =34;
int room2_heater_pin = 37;
int room2_cooling_pin =36;
int room3_heater_pin = 39;
int room3_cooling_pin =38;

long interval = 300;                           // μεταβλητη για την αποθηκευση του χρονου ανανεωσης των αισθητηρων
long previousMillis = 0;                       // μεταβλητη που αποθηκευεται ο χρονος που εγινε τελευταια φορα ανανεωση αισθητηρων
```

Εικόνα 19. Εικόνα κώδικα του προγράμματος

5.2.3 Η συνάρτηση setup()

Η συνάρτηση setup() εκτελείται όταν ξεκινά το πρόγραμμα. Χρησιμοποιείται για να προετοιμάσει τις μεταβλητές, τους τρόπους με τους οποίους συνδέονται τα pin κ.λ.π. Εκτελείται μόνο μία φορά, μετά από κάθε ενεργοποίηση ή επανεκκίνηση του Arduino.

```
void setup(void){

  pinMode(room1_lights_pin, OUTPUT);            //ορισμος των ακροδεκτων σαν εισοδο η εξοδο
  pinMode(room2_lights_pin, OUTPUT);
  pinMode(room3_lights_pin, OUTPUT);
  pinMode(boiler_pin, OUTPUT);
  pinMode(garage_door_open_pin, OUTPUT);
  pinMode(garage_door_close_pin, OUTPUT);
  pinMode(room1_heater_pin, OUTPUT);
  pinMode(room1_cooling_pin, OUTPUT);
  pinMode(room2_heater_pin, OUTPUT);
  pinMode(room2_cooling_pin, OUTPUT);
  pinMode(room3_heater_pin, OUTPUT);
  pinMode(room3_cooling_pin, OUTPUT);
  pinMode(garage_door_left_stop_pin, INPUT);
  pinMode(garage_door_right_stop_pin, INPUT);
  pinMode(room1_temp_pin, INPUT);
  pinMode(room2_temp_pin, INPUT);
  pinMode(room3_temp_pin, INPUT);
  pinMode(room1_lights_inten_pin, INPUT);
  pinMode(room2_lights_inten_pin, INPUT);
  pinMode(room3_lights_inten_pin, INPUT);

  Ethernet.begin(mac, ip);                      //καταχωρηση των διευθυνσεων στην ethernet καρτα
  server.begin();                               //εκινηση του web server
  delay(300);                                   //ενα μικρο Delay για να σταθεροποιηθει το arduino
}
```

Εικόνα 20. Εικόνα κώδικα του προγράμματος

Με την εντολή pinMode ορίζουμε αν ένας ακροδέκτης θα είναι είσοδος ή έξοδος. Με τις εντολές Ethernet και Server εκκινούμε τον Web Server και ακολουθεί εντολή με μια μικρή καθυστέρηση, ώστε ο Serverva ξεκινήσει επιτυχώς.

5.2.4 Η συνάρτηση loop()

Μετά τη συνάρτηση setup() ακολουθεί η συνάρτηση loop(), η οποία κάνει ακριβώς αυτό που υποδηλώνει το όνομά της, είναι ένας διαδοχικός βρόγχος, που επιτρέπει στο πρόγραμμα να επαναλαμβάνεται. Σε αυτή τη συνάρτηση βρίσκεται όλη η λειτουργία του Arduino.

```
void loop(){                                                    //κυριο loop

    EthernetClient client = server.available();                   //αν ο server ειναι διαθεσιμος κ
  if (client) {                                                 //αν υπαρχει client που εχει συνδε
    readString="";                                              //αρχικοποιησε την μεταβλητη αποθ
    while (client.connected()) {                                //οσο ο client ειναι συνδεδεμενος
      unsigned long currentMillis = millis();                   //εκχωρηση του του χρονου στην μετ
      if(currentMillis - previousMillis > interval) {           //αν εχει περασει χρονος μεγαλητερ
        previousMillis = currentMillis;                         //εκχωρησε τον τορινο χρονο σαν το
        update_sensors();                                       //καλεσε την συναρτηση update_sens
      }
      if (client.available()) {                                 //αν υπαρχει συνδεδεμενος client

        char c = client.read();                                 //διαβασε τον χαρακτηρτηρα τον οπο
        if (readString.length() < 100) {                        //αν το μηκος της μεταβλητης reads
          readString += c;                                      //προσθεσε τον χαρακτηρα που διαβ
        }

        if (c == '\n') {                                      //αν εχει ληφθει ολο το http reques
          client.println("HTTP/1.1 200 OK");                   //απαντηση του server με κωδικο 200

          if (readString.indexOf("ajax_inputs") > -1) {       //αν το request ζητησει την σελιδα
            client.println("Content-Type: text/xml");          //απαντηση ως προς τον τυπο της σελ
            client.println("Connection: keep-alive");
            client.println();
            print_xml_data(client);                           //κληση της συναρτησης για εμφανιση
            extract_commands(readString);                     //κληση της συναρτησης για την διαχ
          }
          else{                                               //οποιοδηποτε αλλο αιτημα και να λα

            client.println("Content-Type: text/html");        //απαντηση ως προς τον τυπο της σελ
            client.println("Connection: keep-alive");
            client.println();
            print_web_page(client);                           //κληση της συναρτησης για την εμφα

          }

          delay(1);                                           //μια μικρη καθηστερηση για να παρε
          client.stop();
        }
      }//if client available
    }//while client availble
  }//if client
}//main loop
```

Εικόνα 21. Εικόνα κώδικα του προγράμματος

Ξεκινάμε ορίζοντας έναν πελάτη (Client), εφόσον ο Server είναι διαθέσιμος. Στη συνέχεια, αν υπάρχει αίτημα από κάποιον Client, αρχικοποιούμε τη μεταβλητή readString και για όσο υπάρχει κάποιος Client συνδεδεμένος, εκχωρούμε την τιμή του χρόνου από την έναρξη του Arduino στη μεταβλητή currentMillis. Με τον έλεγχο που ακολουθεί ελέγχουμε αν έχει περάσει ένα συγκεκριμένο χρονικό διάστημα από την τελευταία φορά που εκτελέστηκε ο κώδικας για την ενημέρωση των αισθητήρων. Συνεχίζουμε ελέγχοντας αν ο Client είναι διαθέσιμος και διαβάζουμε χαρακτήρα χαρακτήρα το αίτημά του στο τέλος του οποίου περιλαμβάνεται ο χαρακτήρας αλλαγής γραμμής (\n).

Όταν ληφθεί αυτός ο χαρακτήρας, ο Server απαντάει «HTTP/1.1 200 OK» που σημαίνει ότι έλαβε σωστά το αίτημα και ξεκινάει την αποστολή της υπόλοιπης απάντησης. Στη μεταβλητή readString έχουμε το αίτημα του Client, το οποίο μπορεί να είναι της μορφής GET http://192.168.1.100/?xxxx HTTP/1.1 (όπου xxxx είναι η εντολή που ζήτησε ο χρήστης) ή της μορφής http://192.168.1.100/?ajax_inputs. Εάν το αίτημα έχει την πρώτη μορφή απαντάμε στον Client με τις απαραίτητες γραμμές που ορίζει το πρωτόκολλο http και εκτελούμε τις συναρτήσεις print_xlm_data και extract_commands. Σε περίπτωση που το αίτημα είναι της μορφής GET 192.168.1.100 HTTP/1.1 αποστέλλουμε και πάλι την καθορισμένη απάντηση και στη συνέχεια εκτελούμε τη συνάρτηση print_web_page. Τέλος, χρησιμοποιούμε ένα μικρό delay, ώστε να προλάβει ο Browser του Client να λάβει όλη τη σελίδα και κλείνουμε τη σύνδεση με τον Client.

5.2.5 Οι υπόλοιπες συναρτήσεις

print_web_page:

```
void print_web_page(EthernetClient cli){                    //συνατηση για την εκτυπωση της html σελιδας

  cli.println(F("<!DOCTYPE html>"));
  cli.println(F("<html>"));
  cli.println(F("<head>"));
  cli.println(F("<meta http-equiv=\"Content-Type\" content=\"text/html; charset=UTF-8\"/>"));
  cli.println(F("<title>Έλεγχος Έξυπνου Σπιτιού</title>"));
```

Εικόνα 22. Εικόνα κώδικα του προγράμματος

Αυτή η συνάρτηση τυπώνει στον Browser τη σελίδα για τον έλεγχο του σπιτιού. Η χρήση της συνάρτησης F() μέσα στην εντολή Print δηλώνει ότι το περιεχόμενο θα αποθηκευτεί στη Flash και όχι στη sram μνήμη. Εάν δεν χρησιμοποιηθεί αυτή η συνάρτηση η μνήμη Sram τελειώνει με αποτέλεσμα το Arduino να μην είναι σταθερό και να παρουσιάζει περίεργη συμπεριφορά.

Η εξήγηση του κώδικα θα γίνει με εικόνες από τον κώδικα του Browser για ευκολότερη ανάγνωση του.

Όπως βλέπουμε από τις πρώτες γραμμές, ο κώδικας που βρίσκεται μέσα στις εντολές cli.println(F()) στέλνεται απευθείας στον Browser. Εδώ πρέπει να αναφερθεί ότι ο ειδικός χαρακτήρας “ δεν μπορεί να αποσταλεί, καθώς αναγνωρίζεται από το Arduino ide και γι' αυτό όταν θέλουμε αυτός ο χαρακτήρας να αποσταλεί βάζουμε μπροστά το χαρακτήρα διαφυγής \.

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
5 <title>Έλεγχος Έξυπνου Σπιτιού</title>
6 <script>
```

Εικόνα 23. Εικόνα κώδικα browser

Οι πρώτες εντολές χρησιμοποιούνται για την ορθή εκκίνηση του κώδικα της σελίδας και τον ορισμό των ιδιοτήτων της. Κατόπιν χρησιμοποιείται το Tag για την εισαγωγή κώδικα JavaScript.

```
var cmd1 = ""; var cmd2 = "";
function get_ajax(){
  var request = new XMLHttpRequest();
  request.onreadystatechange = function(){
  if (this.readyState == 4) {
  if (this.status == 200) {
  if (this.responseXML != null) {
    document.getElementById('room1_temp').value=this.responseXML.getElementsByTagName('room1_temp')[0].childNodes[0].nodeValue;
    document.getElementById('room2_temp').value=this.responseXML.getElementsByTagName('room2_temp')[0].childNodes[0].nodeValue;
    document.getElementById('room3_temp').value=this.responseXML.getElementsByTagName('room3_temp')[0].childNodes[0].nodeValue;
    document.getElementById('room1_desired_temp').value=this.responseXML.getElementsByTagName('room1_desired_temp')[0].childNodes[0].nodeValue;
    document.getElementById('room2_desired_temp').value=this.responseXML.getElementsByTagName('room2_desired_temp')[0].childNodes[0].nodeValue;
    document.getElementById('room3_desired_temp').value=this.responseXML.getElementsByTagName('room3_desired_temp')[0].childNodes[0].nodeValue;
    document.getElementById('room1_light').value=this.responseXML.getElementsByTagName('room1_light')[0].childNodes[0].nodeValue;
    document.getElementById('room2_light').value=this.responseXML.getElementsByTagName('room2_light')[0].childNodes[0].nodeValue;
    document.getElementById('room3_light').value=this.responseXML.getElementsByTagName('room3_light')[0].childNodes[0].nodeValue;
    document.getElementById('out_temp').value=this.responseXML.getElementsByTagName('out_temp')[0].childNodes[0].nodeValue;
    document.getElementById('out_hum').value=this.responseXML.getElementsByTagName('out_hum')[0].childNodes[0].nodeValue;
    if (this.responseXML.getElementsByTagName('lights_status_room1')[0].childNodes[0].nodeValue == 'ON'){
      document.getElementById('lights_room1_status').innerHTML = 'ΑΝΟΙΧΤΑ';
      document.getElementById('lights_room1_status').style.color='green';
    }
```

Εικόνα 24. Εικόνα κώδικα browser

Αυτό το κομμάτι του κώδικα είναι γραμμένο σε Ajax. Ξεκινάμε ορίζοντας δύο μεταβλητές cmd1 και cmd2, οι οποίες είναι αλφαριθμητικές και θα χρησιμοποιηθούν για την αποστολή τον εντολών στο Arduino. Συνεχίζουμε ορίζοντας μια συνάρτηση, η οποία είναι υπεύθυνη τόσο για την ανάγνωση των δεδομένων από την XML σελίδα, όσο και για την αποστολή των εντολών στο Arduino. Η συνάρτηση αυτή επιλέχτηκε να είναι μόνο μία, ώστε οι αιτήσεις να λαμβάνονται ανά τακτά χρονικά διαστήματα και όχι σε τυχαίο χρόνο, με σκοπό να μειωθεί η υπερβολική χρήση του Arduino.

Οι πρώτοι έλεγχοι χρησιμοποιούνται για να επιβεβαιώσουμε ότι λάβαμε σωστή απάντηση από τον Server και ότι το έγγραφο XML περιέχει δεδομένα.

Χρησιμοποιώντας τις μεθόδους getElementById και getElementByTagName μας δίνεται η δυνατότητα να διαβάζουμε ένα συγκεκριμένο Tag από την XML σελίδα και να αλλάζουμε τις ιδιότητες ενός συγκεκριμένου HTML στοιχείου.

```
      document.getElementById('room2_clima_status').style.color='red';
      document.getElementById('room2_temp_control').style.display = 'none';
    }
    if (this.responseXML.getElementsByTagName('room3_temp_control')[0].childNodes[0].nodeValue == 'ON'){
      document.getElementById('room3_clima_status').innerHTML = 'ΕΝΕΡΓΟΠΟΙΗΜΕΝΟΣ';
      document.getElementById('room3_clima_status').style.color='green';
      document.getElementById('room3_temp_control').style.display = 'initial';
    }
    else{
      document.getElementById('room3_clima_status').innerHTML = 'ΑΠΕΝΕΡΓΟΠΟΙΗΜΕΝΟΣ';
      document.getElementById('room3_clima_status').style.color='red';
      document.getElementById('room3_temp_control').style.display = 'none';
    }
  }
  }
  }
  }
  request.open("GET", "ajax_inputs" + cmd1 + cmd2 + "&", true);
  request.send(null);
  setTimeout('get_ajax()', 250);
  cmd1 = ""; cmd2 = "";
}
```

Εικόνα 25. Εικόνα κώδικα Browser

Στο τέλος της συνάρτησης δημιουργούμε ένα get http request, το οποίο θα στέλνεται στον Server κάθε 250 ms, όπως ορίζεται στη συνάρτηση setTimeout. Η σύνταξη του αιτήματος ξεκινάει με τον τύπο του (get), συνεχίζεται με το κείμενο “ajax_inputs”, ώστε να ξαναζητήσουμε την σελίδα http://192.168.1.100/?ajax_inputs. Οι μεταβλητές cmd1 και cmd2 περιέχουν τις εντολές που τυχόν ζήτησε ο χρήστης μέσω της σελίδας. Σε περίπτωση που δεν έχει υπάρξει κάποιο αίτημα δεν θα αποσταλεί τίποτα, καθώς στο τέλος της συνάρτησης «αδειάζουμε» τις μεταβλητές. Ο χαρακτήρας & χρησιμοποιείται για τον ευκολότερο διαχωρισμό των εντολών, ενώ η μεταβλητή True είναι απαραίτητη για να αποσταλεί το αίτημα στον Server.

```
function increase_room1_temp_button() {cmd1 = "&increase_room1_temp"; }
function decrease_room1_temp_button() {cmd1 = "&decrease_room1_temp"; }
function increase_room2_temp_button() {cmd2 = "&increase_room2_temp"; }
function decrease_room2_temp_button() {cmd2 = "&decrease_room2_temp"; }
function increase_room3_temp_button() {cmd1 = "&increase_room3_temp"; }
function decrease_room3_temp_button() {cmd1 = "&decrease_room3_temp"; }
function room1_temp_control_on() {cmd2 = "&room1_temp_control_on"; }
function room1_temp_control_off() {cmd2 = "&room1_temp_control_off"; }
function room2_temp_control_on() {cmd1 = "&room2_temp_control_on"; }
function room2_temp_control_off() {cmd1 = "&room2_temp_control_off"; }
function room3_temp_control_on() {cmd2 = "&room3_temp_control_on"; }
function room3_temp_control_off() {cmd2 = "&room3_temp_control_off"; }
function open_lights_room1() {cmd1 = "&open_lights_room1"; }
function close_lights_room1() {cmd1 = "&close_lights_room1"; }
function open_lights_room2() {cmd2 = "&open_lights_room2"; }
function close_lights_room2() {cmd2 = "&close_lights_room2"; }
function open_lights_room3() {cmd1 = "&open_lights_room3"; }
function close_lights_room3() {cmd1 = "&close_lights_room3"; }
function open_garage_door() {cmd2 = "&open_garage_door"; }
function close_garage_door() {cmd2 = "&close_garage_door"; }
function boiler_on() {cmd1 = "&boiler_on"; }
function boiler_off() {cmd1 = "&boiler_off"; }
```

Εικόνα 26. Εικόνα κώδικα browser

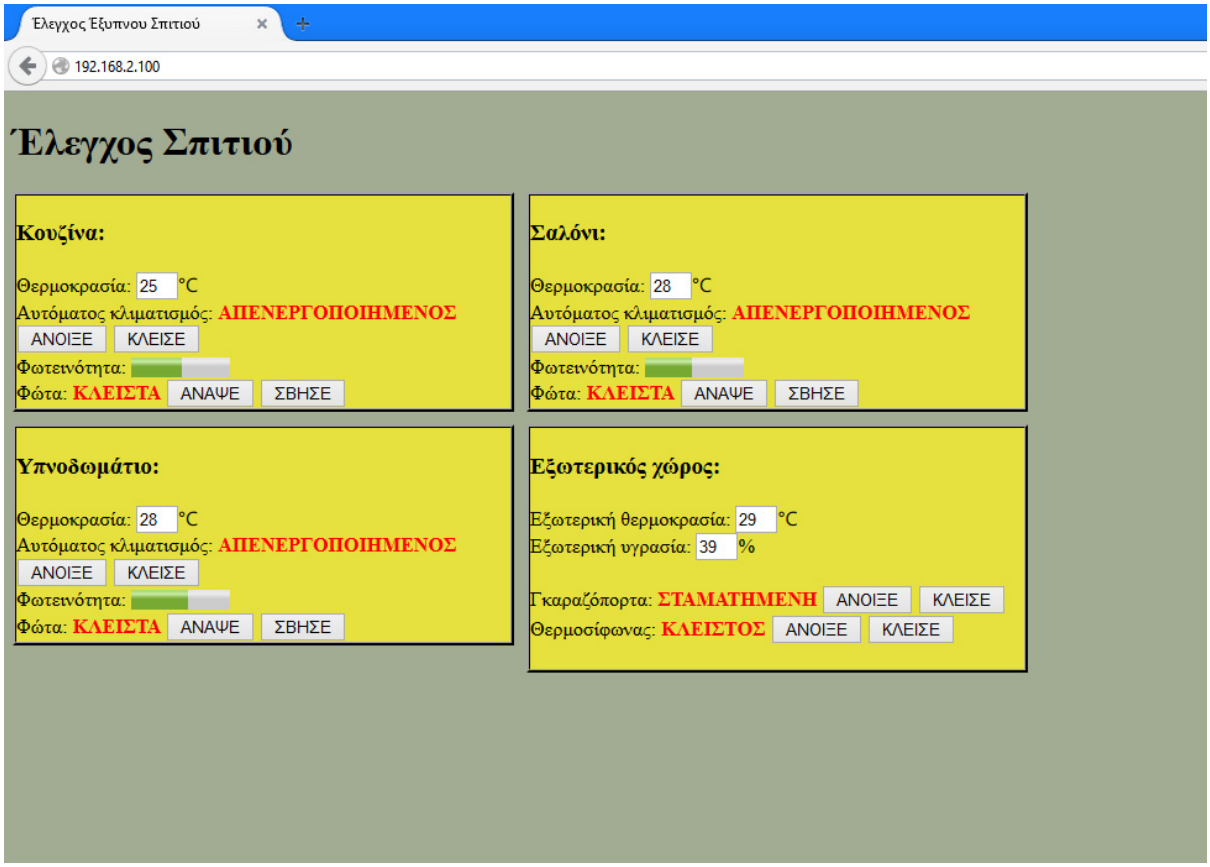
Στη συνέχεια έχουμε τις συναρτήσεις, οι οποίες καθορίζουν ποια εντολή θα αποσταλεί, μόλις ζητηθεί κάποια ενέργεια από το χρήστη. Χρησιμοποιούνται δύο μεταβλητές σε τυχαίες εντολές, με σκοπό τη δυνατότητα εξυπηρέτησης δύο ενεργειών σε μόνο μία ανανέωση της σελίδας.

```
</script>
</head>
<body bgcolor=#A2AC93 onload="get_ajax()">
<h1>Ελεγχος Σπιτιού</h1>
<div style="float:left; width:400px; border:ridge;   margin-right:10px ; margin-bottom:10px;  background-color:#E6E13E       ">
<h3>Κουζίνα:</h3>
  Θερμοκρασία:
    <input type="text" id="room1_temp" size="2" readonly="true">°C
    <br>
  Αυτόματος κλιματισμός:
    <b id="room1_clima_status">wait to update</b>
    <br>
    <button  type="button" onclick="room1_temp_control_on()">ΑΝΟΙΞΕ</button>
    <button  type="button" onclick="room1_temp_control_off()">ΚΛΕΙΣΕ</button>
    <br>
<p id="room1_temp_control">
  Επιθυμητή θερμοκρασία:
    <input type="text" id="room1_desired_temp" size="2" readonly="true">°C
    <button  type="button" onclick="increase_room1_temp_button()">+1 °C</button>
    <button  type="button" onclick="decrease_room1_temp_button()">-1 °C</button>
    <br>
</p>
  Φωτεινότητα:
    <meter id= "room1_light" value="wait to update" min="0" max="100">light meter</meter>
    <br>
  Φώτα:
    <b id="lights_room1_status">wait to update</b>
    <button  type="button" onclick="open_lights_room1()">ΑΝΑΨΕ</button>
    <button  type="button" onclick="close_lights_room1()">ΣΒΗΣΕ</button>
    <br>
</div>
<div style="float:left; width:400px; border:ridge;  margin-right:10px;  margin-bottom:10px;  background-color:#E6E13E            ">
<h3>Σαλόνι:</h3>
  Θερμοκρασία:
    <input type="text" id="room2_temp" size="2" readonly="true">°C
  <br>
  Αυτόματος κλιματισμός:
    <b id="room2_clima_status">wait to update</b>
    <br>
    <button  type="button" onclick="room2_temp_control_on()">ΑΝΟΙΞΕ</button>
    <button  type="button" onclick="room2_temp_control_off()">ΚΛΕΙΣΕ</button>
    <br>
```

Εικόνα 27. Εικόνα κώδικα Browser

Έπειτα κλείνουμε το Tag τόσο για το Script, όσο και το Tag του Head και ξεκινάμε το Body Tag, το οποίο περιέχει το μέρος της σελίδας, το οποίο θα εμφανίζει ο Browser στο χρήστη. Συνεπώς, οι επόμενες εντολές καθορίζουν κυρίως το πώς θα εμφανίζεται το περιεχόμενο στο χρήστη. Γίνεται η χρήση CSS Styles και Division (DIV) για τον κάθε χώρο, ο οποίος ελέγχεται. Χρησιμοποιούνται επίσης πεδία εισόδου τύπου Text για την εμφάνιση των μεταβλητών θερμοκρασίας και πεδία τύπου Meter για την ένδειξη φωτεινότητας. Για τη διεπαφή του χρήστη με το σύστημα χρησιμοποιούνται κουμπιά (Buttons), τα οποία κατά το πάτημα τους εκτελούν μια συνάρτηση ανάλογη του αιτήματος που επέλεξε ο χρήστης. Η σελίδα τελειώνει, κλείνοντας τόσο το Body Tag, όσο και το HTML Tag.

Στην επόμενη εικόνα βλέπουμε το τελικό αποτέλεσμα, όπως εμφανίζεται στον Browser.



Εικόνα 28. Σελίδα ελέγχου σπιτιού, όπως εμφανίζετε στον Browser.

print_xml_data:

Σε αυτή τη συνάρτηση τυπώνουμε στη σελίδα του Browser μια σελίδα μορφής XLM, η οποία περιέχει όλες της μεταβλητές πληροφορίες που χρειάζεται η ιστοσελίδα, ώστε να εμφανίσει σωστά την κατάσταση των διακοπτών και τον άλλων μεταβλητών.

```
void print_xml_data(EthernetClient cl){                  //συνατηση για την\

  cl.println("<?xml version = \"1.0\" ?>");
  cl.println("<inputs>");

  cl.print(" <rooml_temp>");
  cl.print(rooml_temp,0);                                  //εκτυπωση της μ
  cl.println("</rooml_temp>");

  cl.print(" <room2_temp>");
  cl.print(room2_temp,0);
  cl.println("</room2_temp>");

  cl.print(" <room3_temp>");
  cl.print(room3_temp,0);
  cl.println("</room3_temp>");

  cl.print(" <rooml_temp_control>");
  if (rooml_temp_control)    cl.print("ON");            //αν ο ελενχος θερμ
  else{                                                //αλλιως τυπωσε OFF
    cl.print("OFF");
    rooml_desired_temp=rooml_temp;
  }
  cl.println("</rooml_temp_control>");

  cl.print(" <room2_temp_control>");
  if (room2_temp_control)    cl.print("ON");
  else{
    cl.print("OFF");
    room2_desired_temp=room2_temp;
  }
  cl.println("</room2_temp_control>");

  cl.print(" <room3_temp_control>");
  if (room3_temp_control)    cl.print("ON");
  else{
    cl.print("OFF");
    room3_desired_temp=room3_temp;
  }
  cl.println("</room3_temp_control>");
```

Εικόνα 29. Εικόνα προγράμματος Arduino

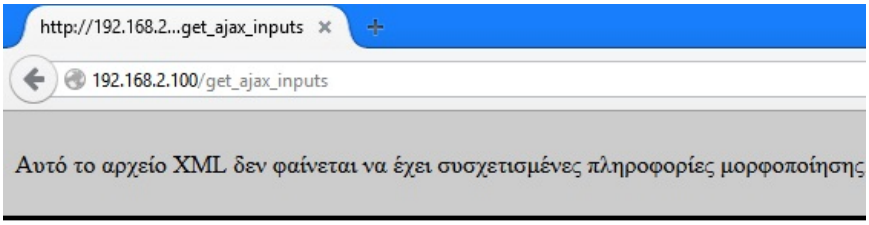
```
cl.print(" <room1_desired_temp>");
cl.print(room1_desired_temp,0);
cl.println("</room1_desired_temp>");

cl.print(" <room2_desired_temp>");
cl.print(room2_desired_temp,0);
cl.println("</room2_desired_temp>");

cl.print(" <room3_desired_temp>");
cl.print(room3_desired_temp,0);
cl.println("</room3_desired_temp>");

cl.print(" <room1_light>");
cl.print(room1_light);
cl.println("</room1_light>");

cl.print(" <room2_light>");
cl.print(room2_light);
cl.println("</room2_light>");

cl.print(" <room3_light>");
cl.print(room3_light);
cl.println("</room3_light>");

cl.print(" <out_temp>");
cl.print(out_temp);
cl.println("</out_temp>");

cl.print(" <out_hum>");
cl.print(out_hum);
cl.println("</out_hum>");

cl.print(" <lights_status_room1>");
if (lights_status_room1) cl.print("ON");
else cl.print("OFF");
cl.println("</lights_status_room1>");

cl.print(" <lights_status_room2>");
if (lights_status_room2) cl.print("ON");
else cl.print("OFF");
cl.println("</lights_status_room2>");

cl.print(" <lights_status_room3>");
if (lights_status_room3) cl.print("ON");
else cl.print("OFF");
cl.println("</lights_status_room3>");
```

Εικόνα 30. Εικόνα προγράμματος Arduino

```
  cl.print(" <garage_door>");
  if (garage_door_open) cl.print("ΑΝΟΙΓΕΙ");
  else if (garage_door_close) cl.print("ΚΛΕΙΝΕΙ");
  else cl.print("ΣΤΑΜΑΤΗΜΕΝΗ");
  cl.println("</garage_door>");

  cl.print(" <boiler_status>");
  if (boiler) cl.print("ON");
  else cl.print("OFF");
  cl.println("</boiler_status>");

  cl.println("</inputs>");

}
```

Εικόνα 31. Εικόνα προγράμματος Arduino

Όταν κληθεί αυτή η συνάρτηση εμφανίζεται το εξής αποτέλεσμα στη σελίδα http://192.168.1.100/?get_ajax




```
- <inputs>
    <room1_temp>31</room1_temp>
    <room2_temp>30</room2_temp>
    <room3_temp>29</room3_temp>
    <room1_temp_control>OFF</room1_temp_control>
    <room2_temp_control>OFF</room2_temp_control>
    <room3_temp_control>OFF</room3_temp_control>
    <room1_desired_temp>31</room1_desired_temp>
    <room2_desired_temp>30</room2_desired_temp>
    <room3_desired_temp>29</room3_desired_temp>
    <room1_light>37</room1_light>
    <room2_light>36</room2_light>
    <room3_light>51</room3_light>
    <out_temp>30</out_temp>
    <out_hum>37</out_hum>
    <lights_status_room1>OFF</lights_status_room1>
    <lights_status_room2>OFF</lights_status_room2>
    <lights_status_room3>OFF</lights_status_room3>
    <garage_door>ΣΤΑΜΑΤΗΜΕΝΗ</garage_door>
    <boiler_status>OFF</boiler_status>
  </inputs>
```

Εικόνα 32. Σελίδα XML, όπως εμφανίζεται στον Browser.

extract_commands:

Σε αυτή τη συνάρτηση γίνεται η αποστολή των εντολών, οι οποίες βρίσκονται στο get http request, σε ενέργειες τις οποίες θα πραγματοποιήσει το Arduino, ώστε να γίνουν οι απαραίτητες αλλαγές στις μεταβλητές ή να εκτελεστεί κάποια ενέργεια.

Η λειτουργία έχει ως εξής: Η συνάρτηση καλείται με όρισμα την εντολή που έστειλε ο Browser, χρησιμοποιώντας ως σημείο αναφοράς το χαρακτήρα &, ώστε να μπορέσει να αποσπάσει την εντολή, η οποία στάλθηκε μέσω του Client. Η εντολή που αποσπάστηκε τοποθετείται στη μεταβλητή extracted_data. Μετά από έναν έλεγχο για τυχόν κενή εντολή, καλείται η συνάρτηση commands. Με τη υλοποίηση της συγκεκριμένης συνάρτησης είναι δυνατή η εκτέλεση μέχρι και δύο εντολών, χωρίς δεύτερη αποστολή από τον Client. Φυσικά ο αριθμός των εντολών μπορεί να αυξηθεί ανάλογα με τις ανάγκες της εκάστοτε εφαρμογής.

```
void extract_commands(String eisodos){
  extracted_data = "";
  second_comand = "";
  extracted_data =eisodos.substring((eisodos.indexOf('&')),eisodos.lastIndexOf('&'));
  if (extracted_data != ""){
    commands(extracted_data);
    second_comand=extracted_data.substring(extracted_data.lastIndexOf('&'));
  }
  if (second_comand != extracted_data){
    commands(second_comand);
  }
}
```

Εικόνα 33. Εικόνα προγράμματος Arduino

commands

Ο σκοπός αυτής της συνάρτησης θα μπορούσαμε να πούμε ότι είναι η καρδιά του προγράμματός μας, καθώς είναι το κομμάτι του κώδικα, το οποίο μετατρέπει τις εντολές που ελήφθησαν σε ενέργειες. Για παράδειγμα, η εντολή «&increase_room1_temp» θα αυξήσει τη μεταβλητή «room1_desired_temp», ενώ η εντολή «boiler_on» θα αλλάξει κατάσταση στη μεταβλητή Boiler, η οποία ελέγχει το αν ο θερμοσίφωνας θα είναι ανοιχτός ή κλειστός.

```
void commands(String entoli){

  if (entoli != ""){
    if (entoli == "&increase_room1_temp") room1_desired_temp+=1;
    if (entoli == "&decrease_room1_temp") room1_desired_temp-=1;
    if (entoli == "&increase_room2_temp") room2_desired_temp+=1;
    if (entoli == "&decrease_room2_temp") room2_desired_temp-=1;
    if (entoli == "&increase_room3_temp") room3_desired_temp+=1;
    if (entoli == "&decrease_room3_temp") room3_desired_temp-=1;
    if (entoli == "&open_lights_room1") lights_status_room1=true;
    if (entoli == "&close_lights_room1") lights_status_room1=false;
    if (entoli == "&open_lights_room2") lights_status_room2=true;
    if (entoli == "&close_lights_room2") lights_status_room2=false;
    if (entoli == "&open_lights_room3") lights_status_room3=true;
    if (entoli == "&close_lights_room3") lights_status_room3=false;
    if (entoli == "&boiler_on") boiler=true;
    if (entoli == "&boiler_off") boiler=false;
    if (entoli == "&room1_temp_control_on") room1_temp_control=true;
    if (entoli == "&room1_temp_control_off") room1_temp_control=false;
    if (entoli == "&room2_temp_control_on") room2_temp_control=true;
    if (entoli == "&room2_temp_control_off") room2_temp_control=false;
    if (entoli == "&room3_temp_control_on") room3_temp_control=true;
    if (entoli == "&room3_temp_control_off") room3_temp_control=false;

    if (entoli == "&open_garage_door"){
      garage_door_open=true;
      garage_door_close=false;
    }
    if (entoli == "&close_garage_door"){
      garage_door_open=false;
      garage_door_close=true;
    }
  }
}
```

Εικόνα 34. Εικόνα προγράμματος Arduino

update_sensors:

Μία ακόμα συνάρτηση που χρησιμοποιείται είναι η συνάρτηση update_sensors, στην οποία πραγματοποιείται η ανανέωση των τιμών των αισθητήρων και η αλλαγή της κατάστασης των εξόδων του Arduino.

Οι πρώτες εντολές διαβάζουν τις τιμές των αισθητηρίων LM35, τις μετατρέπουν σε βαθμούς Κελσίου και τις καταχωρούν στις αντίστοιχες μεταβλητές. Οι επόμενες εντολές διαβάζουν τις τιμές των αισθητήρων φωτεινότητας και με τη χρήση της συνάρτησης map τις μετατρέπουν σε κλίμακα 0 – 100 και τις εκχωρούν και πάλι στις αντίστοιχες μεταβλητές.

Οι εντολές που ακολουθούν διαβάζουν τις τιμές του αισθητήρα DHT11 με κλήση της συνάρτησης που περιέχεται στην βιβλιοθήκη DHT.h και τις καταχωρούν στις μεταβλητές θερμοκρασίας και υγρασίας.

Ανάλογα με την κατάσταση των μεταβλητών γίνεται η αλλαγή της κατάστασης στους ακροδέκτες εξόδου του Arduino. Αυτές οι εντολές είναι υπεύθυνες για το άνοιγμα ή κλείσιμο των φώτων, καθώς και για τον έλεγχο της γκαραζόπορτας.

Τέλος εκτελείται η συνάρτηση temp_control 3 φορές με ορίσματα για το κάθε δωμάτιο.

```
void update_sensors(){                                                    //συναρτηση για ενημερωση

  room1_temp = ((5.0 * analogRead(room1_temp_pin)* 100.0)/1024.0);
  room2_temp = ((5.0 * analogRead(room2_temp_pin)* 100.0)/1024.0);
  room3_temp = ((5.0 * analogRead(room3_temp_pin)* 100.0)/1024.0);
  room1_light = map(analogRead(room1_lights_inten_pin),0,700,0,100);
  room2_light = map(analogRead(room2_lights_inten_pin),0,700,0,100);
  room3_light = map(analogRead(room3_lights_inten_pin),0,700,0,100);

  DHT.readl1(dht_dpin);                                                   //αιτηση για ανανεωση των τ

  out_hum = DHT.humidity;
  out_temp = DHT.temperature;

  if (lights_status_room1)  digitalWrite(room1_lights_pin , HIGH);
  else                      digitalWrite(room1_lights_pin , LOW);
  if (lights_status_room2)  digitalWrite(room2_lights_pin , HIGH);
  else                      digitalWrite(room2_lights_pin , LOW);
  if (lights_status_room3)  digitalWrite(room3_lights_pin , HIGH);
  else                      digitalWrite(room3_lights_pin , LOW);
  if (boiler)               digitalWrite(boiler_pin , HIGH);
  else                      digitalWrite(boiler_pin , LOW);

  if (garage_door_open && (digitalRead(garage_door_left_stop_pin)==LOW)){
    digitalWrite(garage_door_open_pin , HIGH);
    digitalWrite(garage_door_close_pin , LOW);
    garage_door_open=true;
    garage_door_close=false;
  }
  else if (garage_door_close && (digitalRead(garage_door_right_stop_pin)==LOW)){
    digitalWrite(garage_door_open_pin , LOW);
    digitalWrite(garage_door_close_pin , HIGH);
    garage_door_open=false;
    garage_door_close=true;
  }
  else{
    digitalWrite(garage_door_open_pin , LOW);
    digitalWrite(garage_door_close_pin , LOW);
    garage_door_open=false;
    garage_door_close=false;
  }

  temp_control(room1_temp_control,room1_desired_temp,room1_temp,room1_heater_pin,room1_cooling_pin);
  temp_control(room2_temp_control,room2_desired_temp,room2_temp,room2_heater_pin,room2_cooling_pin);
  temp_control(room3_temp_control,room3_desired_temp,room3_temp,room3_heater_pin,room3_cooling_pin);

}
```

Εικόνα 35. Εικόνα προγράμματος Arduino

temp_control:

Είναι η τελευταία συνάρτηση, η οποία υλοποιήθηκε στο πρόγραμμα και είναι επιφορτισμένη με τον έλεγχο του αυτόματου κλιματισμού. Ελέγχει τόσο την κατάσταση του αυτόματου κλιματισμού, όσο και τη διαφορά μεταξύ των μεταβλητών επιθυμητής θερμοκρασίας και τωρινής θερμοκρασίας του δωματίου, ώστε να ανοίξει τη θέρμανση ή τον κλιματισμό ανάλογα με την ανάγκη.

```
void temp_control(boolean room_temp_control, int  room_desired_temp,int room_temp, int room_heater_pin, int room_cooling_pin){

  if (room_temp_control){                  //αν εχει ενεργοποιηθει ο ελενχος σε καποιο δωματιο
    if (room_desired_temp > (room_temp+1)){          //αν η επιθυμητη θερμοκρασια ειναι μεγαλητερη απο αυτη του δωματιου ανοιξε
      digitalWrite(room_heater_pin, HIGH);
      digitalWrite(room_cooling_pin, LOW);
    }
    else if (room_desired_temp < (room_temp-1)){       //αν η επιθυμητη θερμοκρασια ειναι μικροτερη απο αυτη του δωματιου ανοιξ
      digitalWrite(room_heater_pin, LOW);
      digitalWrite(room_cooling_pin, HIGH);
    }
    else{                                             //αλλιως κλειστα ολα
      digitalWrite(room_heater_pin, LOW);
      digitalWrite(room_cooling_pin, LOW);
    }
  }
  else{                                        //αν ειναι κλειστος ο ελενχος θερμοκρασιας κλειστα ολα
    digitalWrite(room_heater_pin, LOW);
    digitalWrite(room_cooling_pin, LOW);
  }
}
```

Εικόνα 36. Εικόνα προγράμματος Arduino

5.3. Κατασκευή πλακέτας δοκιμών

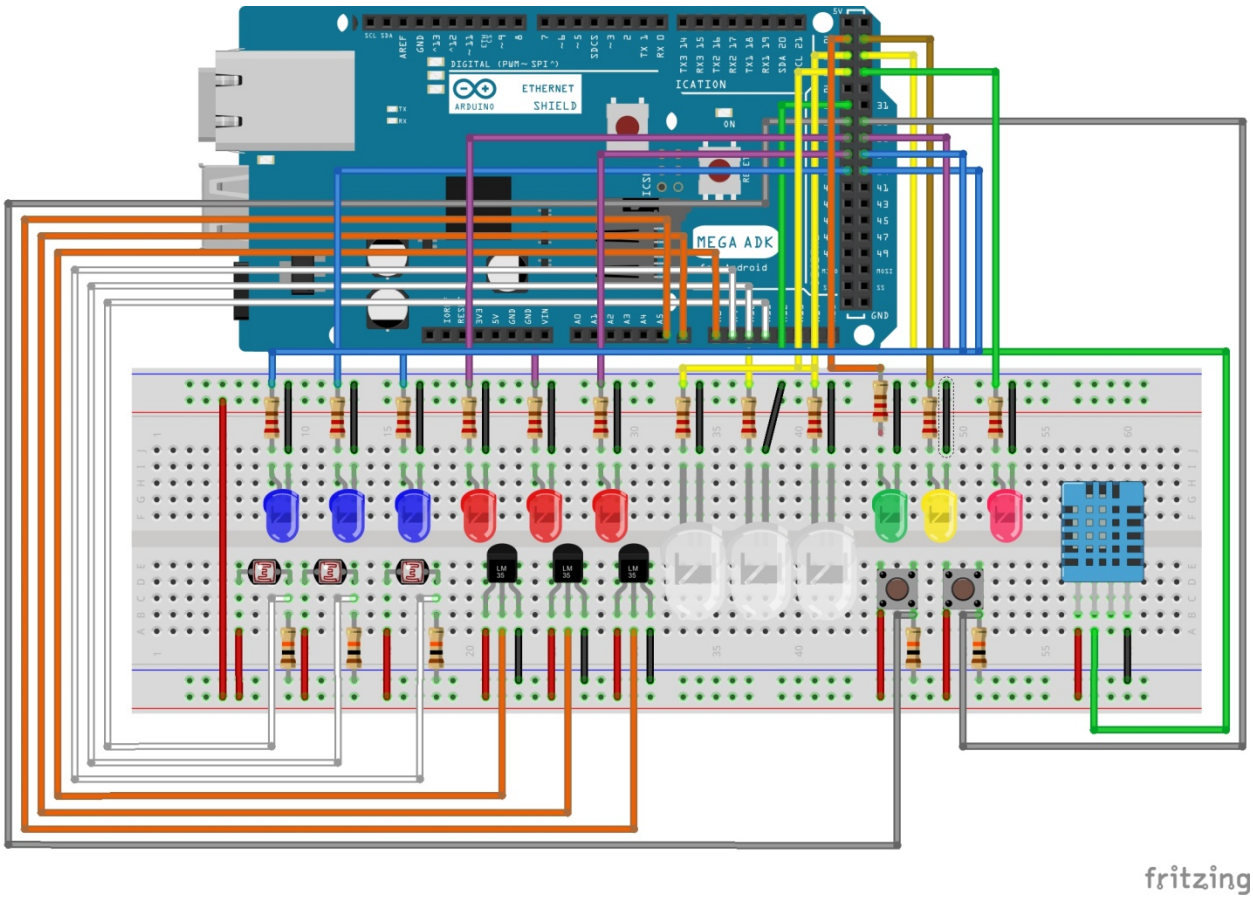
Το πρώτο βήμα, πριν από τη δημιουργία της τελικής κατασκευής, ήταν η υλοποίηση ενός δοκιμαστικού κυκλώματος σε Breadboard, το οποίο χρησιμοποιήθηκε για όλες τις δοκιμές.

Η συνδεσμολογία των αισθητήρων που χρησιμοποιήθηκαν ήταν πολύ απλή, καθώς είναι από τους πιο κοινούς και εύχρηστους αισθητήρες.

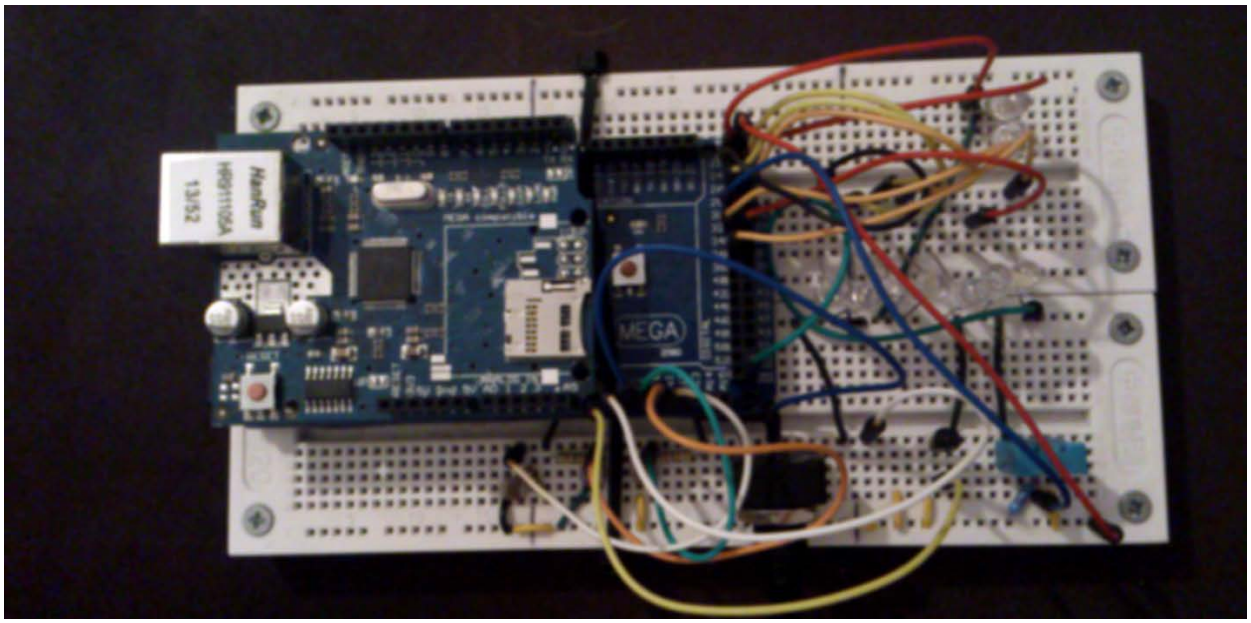
Για τη σύνδεση των φωτοαντιστάσεων χρησιμοποιήθηκαν αντιστάσεις 10 ΚΩ ως Pull down. Από τον ίδιο ακροδέκτη έγινε και η λήψη του σήματος, το οποίο πηγαίνει στο Arduino. Ο άλλος ακροδέκτης τροφοδοτήθηκε με 5V. Αντίστοιχη συνδεσμολογία χρησιμοποιήθηκε και στους διακόπτες. Οι αισθητήρες LM35 τροφοδοτήθηκαν με 5 V στον ακροδέκτη 1 και γείωση στον ακροδέκτη 3. Η λήψη του σήματος έγινε από τον ακροδέκτης 2. Ο αισθητήρας DHT11 τροφοδοτήθηκε στους ακροδέκτες 1 – 4, ενώ η λήψη του σήματος έγινε στον ακροδέκτη 2.

Για τα Led χρησιμοποιήθηκαν αντιστάσεις των 220 Ω.

Ακολουθεί η εικόνα του σχηματικού δοκιμών και η εικόνα του πραγματικού κυκλώματος δοκιμών.



Εικόνα 37. Συνδεσμολογία κυκλώματος στο πρόγραμμα Fritzing



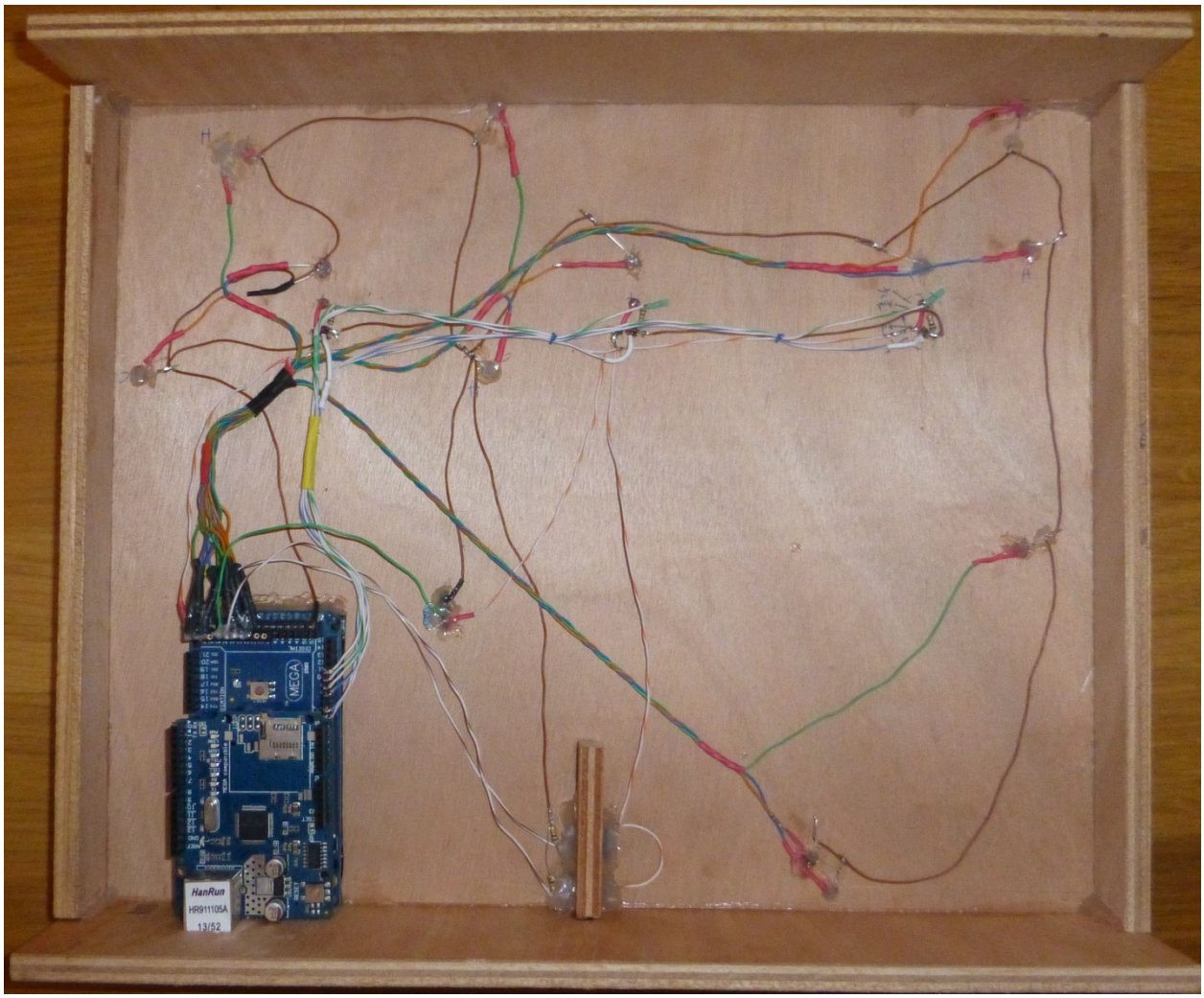
Εικόνα 38. Συνδεσμολογία κυκλώματος δοκιμών

5.4. Κατασκευή

Για την παρουσίαση της πτυχιακής δεν κατασκευάστηκε πλήρως μακέτα σπιτιού, καθώς η κατασκευή της θα ήταν χρονοβόρα και με μεγάλο κόστος. Αντί αυτού, κατασκευάστηκε μια πολύ πιο απλή, η οποία βέβαια καλύπτει όλες τις ανάγκες για την παρουσίαση αυτής της πτυχιακής εργασίας.

Εικόνα 39. Εικόνα της μακέτας

Το κύκλωμα που ελέγχει το «Έξυπνο σπίτι» φαίνεται στην παρακάτω εικόνα.



Εικόνα 40. Το κύκλωμα της μακέτας

Κεφάλαιο 6: Πρακτική εφαρμογή

Κατά την διάρκεια εκπόνησης της πτυχιακής εργασίας, συζητώντας με ένα φίλο ηλεκτρολόγο για τα πλεονεκτήματα ενός «Έξυπνου σπιτιού», όπως για το πόσο εύκολο είναι να τοποθετηθεί, τι δυσκολίες παρουσιάζονται τόσο κατά την ανάπτυξη, όσο και κατά την τοποθέτηση, μου προτάθηκε να αναπτύξω ένα σύστημα «Έξυπνου σπιτιού» για έναν πελάτη του.

Το σπίτι βρισκόταν σε κατάσταση ανακαίνισης και τα ηλεκτρολογικά δεν είχαν εγκατασταθεί ακόμα, κάτι το οποίο έκανε τις συνθήκες ανάπτυξης ιδανικές, καθώς το μεγαλύτερο πρόβλημα τοποθέτησης ενός συστήματος διαχείρισης σπιτιού είναι ότι οι υπάρχουσες καλωδιώσεις δεν είναι ποτέ αρκετές για να καλύψουν τις ανάγκες ενός τέτοιου συστήματος.

Σε αυτό το κεφάλαιο δεν θα αναλυθεί ο κώδικας, ο οποίος είναι εφάμιλλος αυτού που αναπτύχθηκε για την μακέτα και εξηγήθηκε στο προηγούμενο κεφάλαιο, αλλά θα αναλυθούν τα προβλήματα που παρουσιάστηκαν κατά την εγκατάσταση του κυκλώματος στο σπίτι και ο τρόπος σύνδεσης των διαφόρων εξαρτημάτων.

6.1. Παρουσίαση και αντιμετώπιση των προβλημάτων

Το πρώτο πρόβλημα που παρουσιάστηκε ήταν ο τρόπος διεπαφής του Arduino με το υπόλοιπο σπίτι. Η λύση που δόθηκε ήταν η επιλογή ρελέ καστάνιας στον ηλεκτρολογικό πίνακα, μαζί με τη χρήση Push Buttons αντί των συμβατικών διακοπτών για τα φώτα. Το ρελέ καστάνιας λειτουργεί σαν ένας διακόπτης με τη διαφορά ότι δεν χρειάζεται να έχουμε συνεχώς συνδεδεμένη μία επαφή (όπως ένας διακόπτης τοίχου για να έχουμε ανοιχτό το φως). Το μόνο που χρειάζεται είναι ένας παλμός για να οπλιστεί και ένας παλμός για να αφοπλιστεί ο διακόπτης. Αυτό επιτρέπει τη συνύπαρξη του Arduino με τους διακόπτες του σπιτιού, καθώς είναι δυνατή η χρήση και των δυο συστημάτων για τον έλεγχο του σπιτιού. Για παράδειγμα, μπορεί ένα φως να ανοίξει με το Arduino και να κλείσει με το διακόπτη από τον τοίχο. Η χρήση των συγκεκριμένων ρελέ μας βοήθησε επίσης στην απομόνωση του Arduino από τα 220 V, όχι λόγω τάσης, αλλά λόγω χρόνου ενεργοποίησης. Για να διατηρηθεί ένα φως αναμμένο πρέπει να περνάει ρεύμα από το ρελέ καστάνιας και όχι από το ρελέ που ελέγχει το Arduino. Αυτό μειώνει τον κίνδυνο ηλεκτροπληξίας, καθώς ρεύμα υψηλής τάσης περνάει από τα ρελέ του Arduino μόνο για 70 ms, όσος είναι και ο χρόνος που απαιτείται για να αλλάξει την κατάσταση του ρελέ καστάνιας.

Εικόνα 41. Ρελέ καστάνιας τοποθετημένα στον ηλεκτρολογικό πίνακα

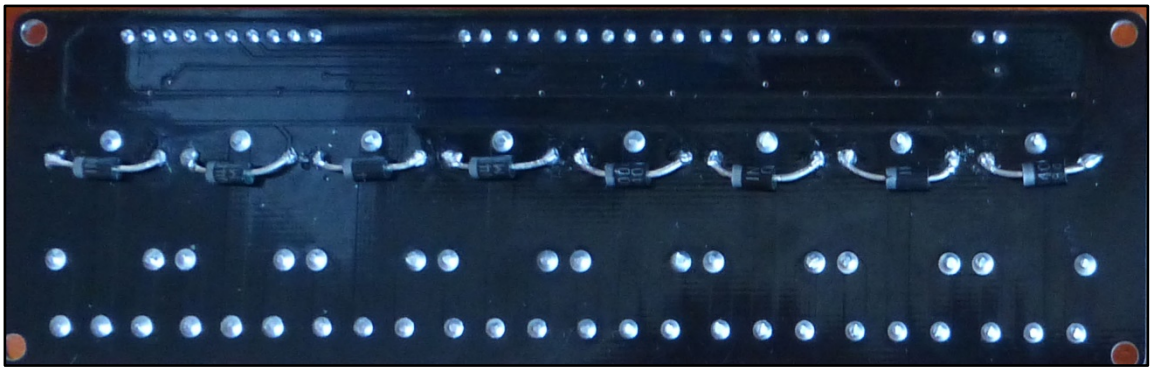
Τα ρελέ καστάνιας αποτελούν ταυτόχρονα τον τρόπο σύνδεσης των διακοπτών του σπιτιού και την υποδοχή για τη σύνδεση του Arduino. Για τη σύνδεση του Arduino με τα ρελέ καστάνιας αρχικά επιλέχτηκε ένα Relay Board, το οποίο λειτουργεί με τα 5V, τα οποία παρέχει το Arduino και μας επιτρέπει να συνδέσουμε μέχρι και 250Vac 10A σε κάθε ένα από τα ρελέ τα οποία διαθέτει.

Εικόνα 42. Συνδεσμολογία Arduino με το Relay Board

Με την πρώτη όμως δοκιμή παρουσιάστηκε πρόβλημα, καθώς εμφανίστηκαν επιστρεφόμενα ρεύματα από το πηνίο του ρελέ καστάνιας. Τα επιστρεφόμενα αυτά ρεύματα ή Counter Electromotive Force ή Back EMF είναι ένα φαινόμενο που

παρουσιάζεται σε όλα τα πηνία. Εν ολίγοις, το πηνίο αντιστέκεται σε οποιαδήποτε μεταβολή του ρεύματος που περνάει μέσα του. Αυτό συμβαίνει επειδή αναπτύσσεται στα άκρα του τάση αντίστροφης πολικότητας ως προς την τάση της πηγής που το τροφοδοτεί. Όσο πιο απότομη είναι η μεταβολή του ρεύματος, τόσο πιο μεγάλη είναι και η τάση που θα αναπτυχθεί.

Για την επίλυση αυτού του προβλήματος αρχικά τοποθετήθηκαν δίοδοι στις επαφές των ρελέ, οι οποίες όμως, όπως αποδείχτηκε, δεν ήταν αρκετές, καθώς εκτός από τα επιστρεφόμενα ρεύματα, τα ρελέ είναι ευπαθή και σε ηλεκτρομαγνητικές παρεμβολές από τα καλώδια υψηλής τάσης. Και τα δυο αυτά προβλήματα οδηγούσαν το Arduino σε επανεκκινήσεις κάθε φορά που άνοιγε ή έκλεινε ένας διακόπτης.

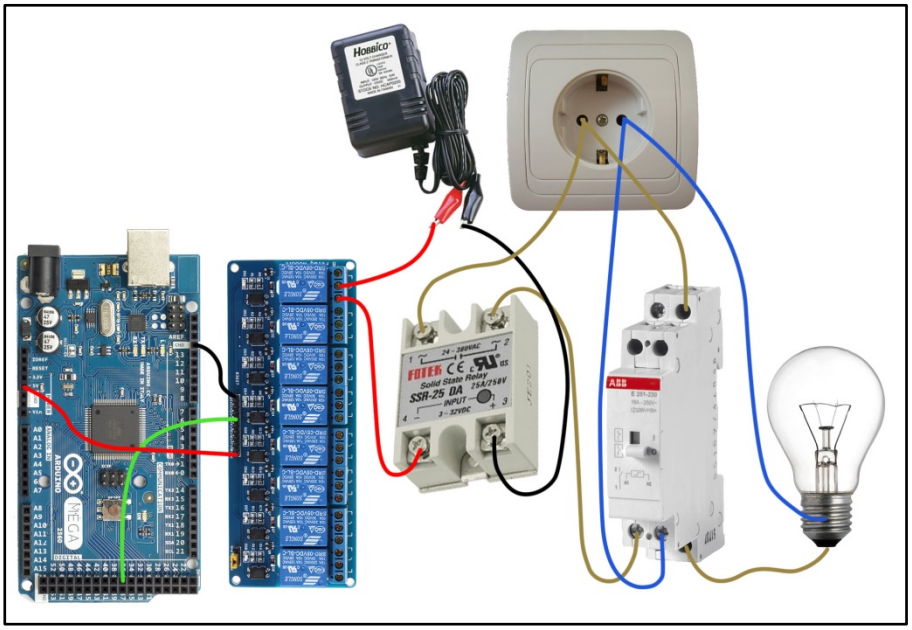


Εικόνα 43. Πίσω όψη Relay Board με τις διόδους

Η λύση που δόθηκε για την αποφυγή των επανεκκινήσεων του Arduino είναι η χρήση Solid State Relay ανάμεσα στο relay board και στα ρελέ καστάνιας του ηλεκτρολογικού πίνακα. Πρόκειται για ρελέ το οποίο δεν χρησιμοποιεί πηνίο και επαφές για να επιτύχει την απομόνωση των 2 σταδίων, αλλά χρησιμοποιεί θυρίστορ και τρανζίστορ για να το επιτύχει. Παρότι μπορεί να δεχτεί σήμα εισόδου από 3 V, κάτι που σημαίνει ότι θα μπορούσαν να χρησιμοποιηθούν τα 5 V από το Arduino, επιλέχτηκε ξεχωριστό τροφοδοτικό 12 V, ώστε να μην φορτωθεί υπερβολικά το κύκλωμα τροφοδοσίας του Arduino.

Εικόνα 44. Solid State Relay

Η τελική συνδεσμολογία φαίνεται στην παρακάτω εικόνα:

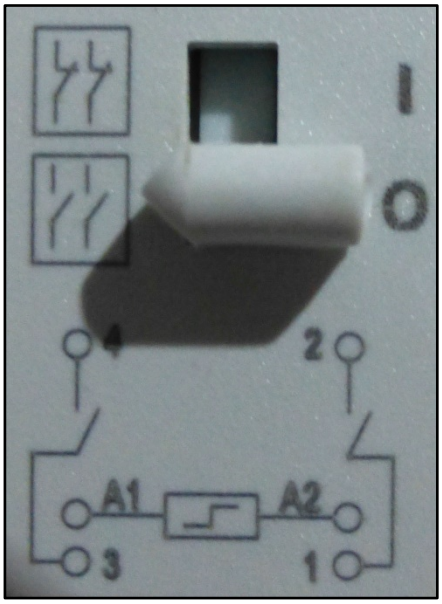


Εικόνα 45. Τελική συνδεσμολογία

Με αυτόν τον τρόπο λοιπόν ολοκληρώθηκε η διεπαφή του Arduino με την ηλεκτρολογική εγκατάσταση του σπιτιού.

6.2. Ανάγνωση κατάστασης συσκευών

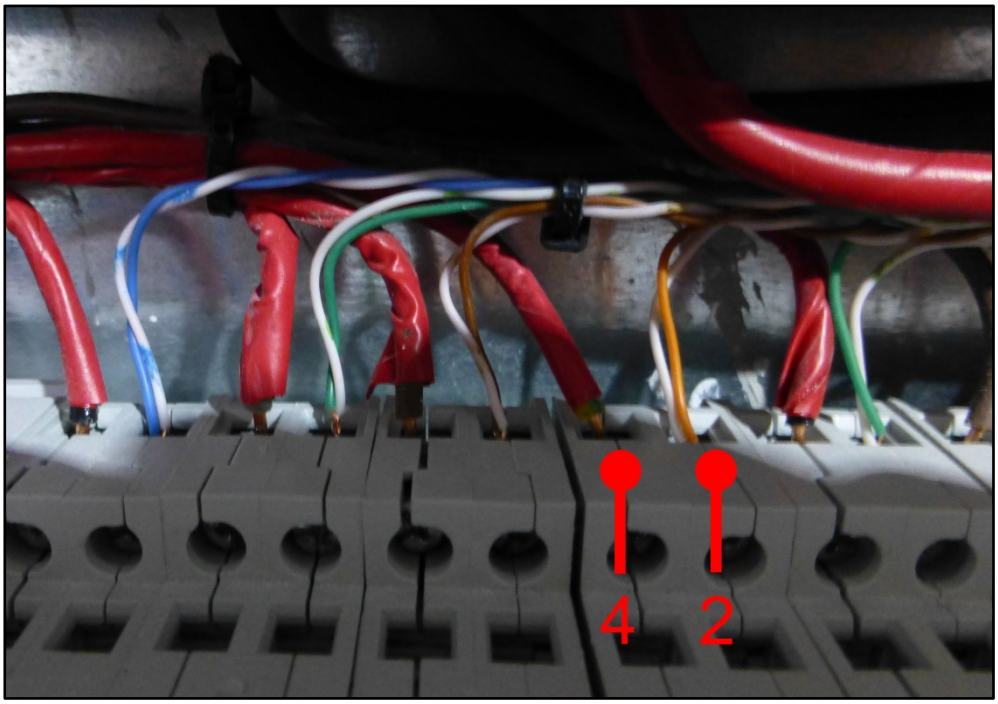
Για την ορθότερη λειτουργία του «Έξυπνου σπιτιού» ήταν απαραίτητο να υπάρχει ένδειξη της κατάστασης του κάθε διακόπτη στο σύστημα χειρισμού. Εάν τον έλεγχο τον είχε αποκλειστικά το Arduino αυτό θα ήταν πολύ εύκολο να καθοριστεί, καθώς γνωρίζουμε ανά πάσα στιγμή την κατάσταση της κάθε εξόδου. Επειδή όμως μπορεί ένας διακόπτης σπιτιού να αλλάξει την κατάσταση της συσκευής που ελέγχουμε, πρέπει να μπορούμε να διαβάσουμε την κατάσταση του ρελέ καστάνιας ανεξάρτητα από το σήμα που στέλνει το Arduino. Για αυτό το λόγο τα ρελέ καστάνιας που επιλέχθησαν είχαν 2 ανεξάρτητες επαφές, όπως φαίνεται και στο παρακάτω διάγραμμα που υπάρχει πάνω στο ρελέ.



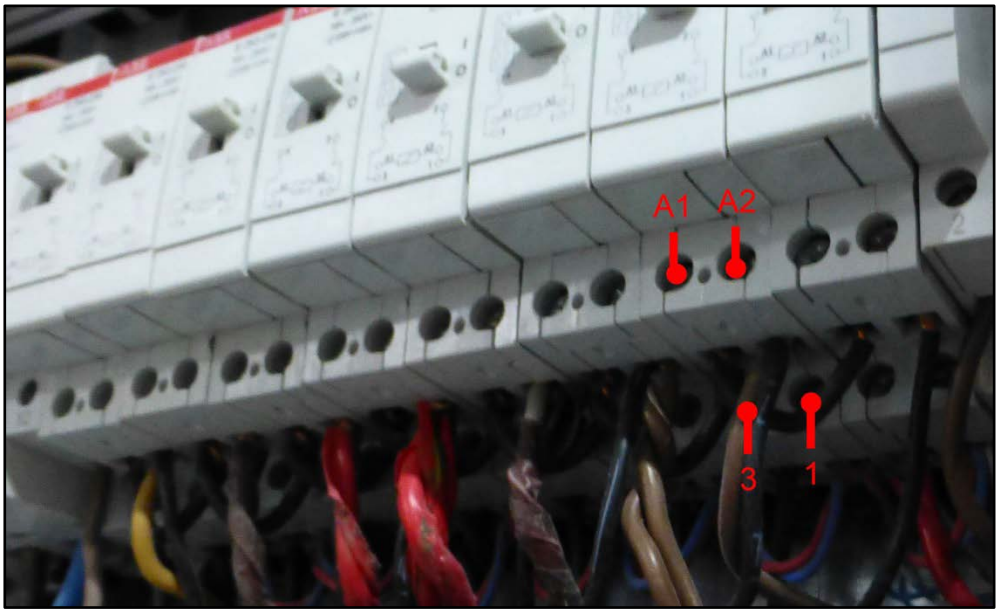
Εικόνα 46. Διάγραμμα εσωτερικής συνδεσμολογίας ρελέ

Για να οπλίσουμε και να αφοπλίσουμε το ρελέ, στέλνουμε ρεύμα στις επαφές A1 – A2 και αυτό έχει ως συνέπεια το κλείσιμο ή το άνοιγμα 2 διακοπτών μεταξύ των επαφών 4 – 3 και 2 – 1 αντίστοιχα.

Όπως φαίνεται στην επόμενη εικόνα, στις επαφές 4 – 3 συνδέθηκαν οι συσκευές έλεγχου, δηλαδή τα φώτα και η αντλία θερμότητας, ενώ οι επαφές 2 – 1 χρησιμοποιήθηκαν σαν διακόπτης των 5 V, ο οποίος τροφοδοτείται από το Arduino και συνδέεται σε έναν ακροδέκτη του Arduino, από τον οποίο και διαβάζουμε την κατάσταση του ρελέ καστάνιας.

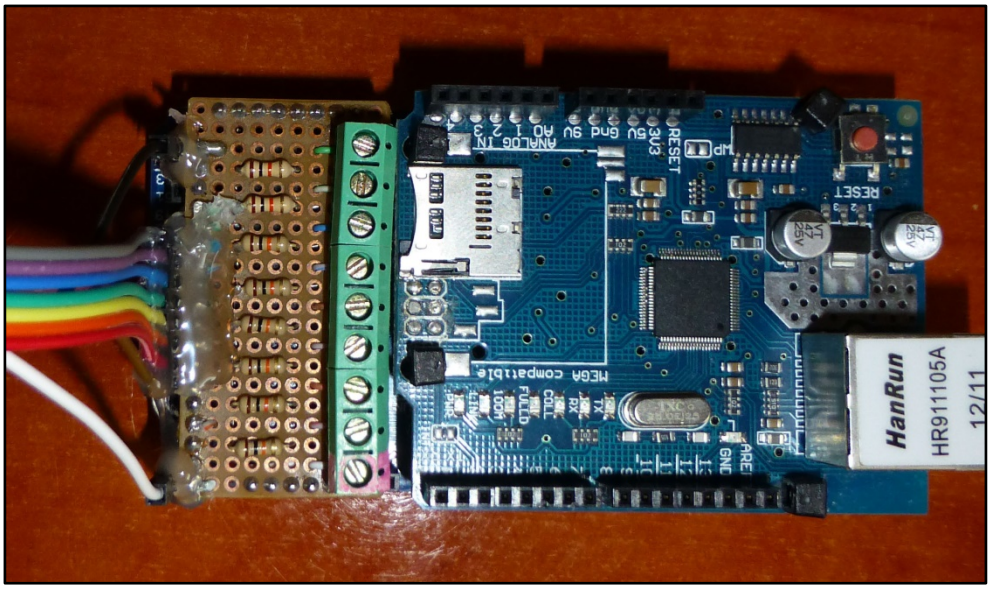


Εικόνα 47. Ακροδέκτες ρελέ καστάνιας



Εικόνα 48. Ακροδέκτες ρελέ καστάνιας

Η πλακέτα που σχεδιάστηκε φαίνεται παρακάτω συνδεδεμένη πάνω στο Arduino. Πρόκειται για ένα πολύ απλό κύκλωμα ανάγνωσης εισόδου, στο οποίο ο πρώτος ακροδέκτης χρησιμοποιείται για την τροφοδοσία 5 V στα ρελέ καστάνιας και οι υπόλοιποι για τις επιστροφές. Οι αντιστάσεις χρησιμεύουν, ώστε να μην παίρνουμε λανθασμένες μετρήσεις, όταν είναι ανοιχτή η επαφή (δεν περνάει ρεύμα 5 V).

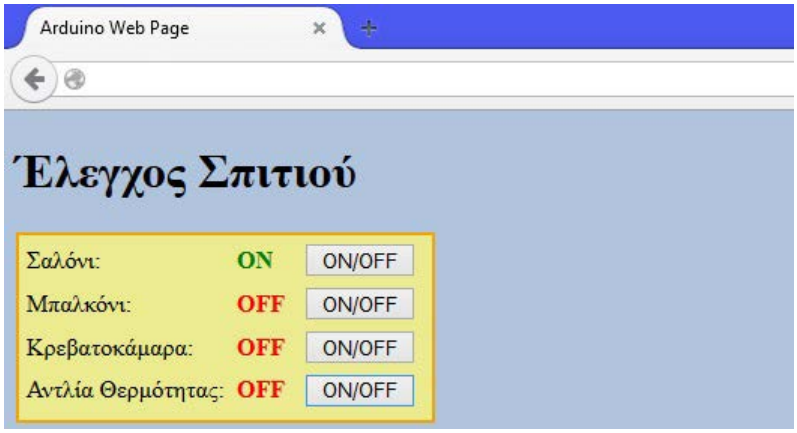


Εικόνα 49. Το Arduino της εγκατάστασης

Τέλος, ένα πρόβλημα που αντιμετωπίστηκε μετά από αρκετές δοκιμές, ήταν ότι λόγω της κατασκευής της Ethernet Shield κάποιες φορές, ιδιαίτερα όταν το Arduino χρησιμοποιούσε εξωτερική τροφοδοσία, η ασπίδα δικτύου δεν ξεκινούσε σωστά, με αποτέλεσμα να μην μπορεί να πάρει IP από το Ρούτερ. Για να λυθεί αυτό το πρόβλημα, συνδέθηκε ένας πυκνωτής στους ακροδέκτες GND – reset του Arduino, πράγμα το οποίο έχει σαν αποτέλεσμα να καθυστερεί η έναρξη της λειτουργίας του Arduino, μέχρι να φορτίσει ο πυκνωτής. Αυτό δίνει αρκετό χρόνο στην Ethernet Shield να ξεκινήσει χωρίς κανένα πρόβλημα.

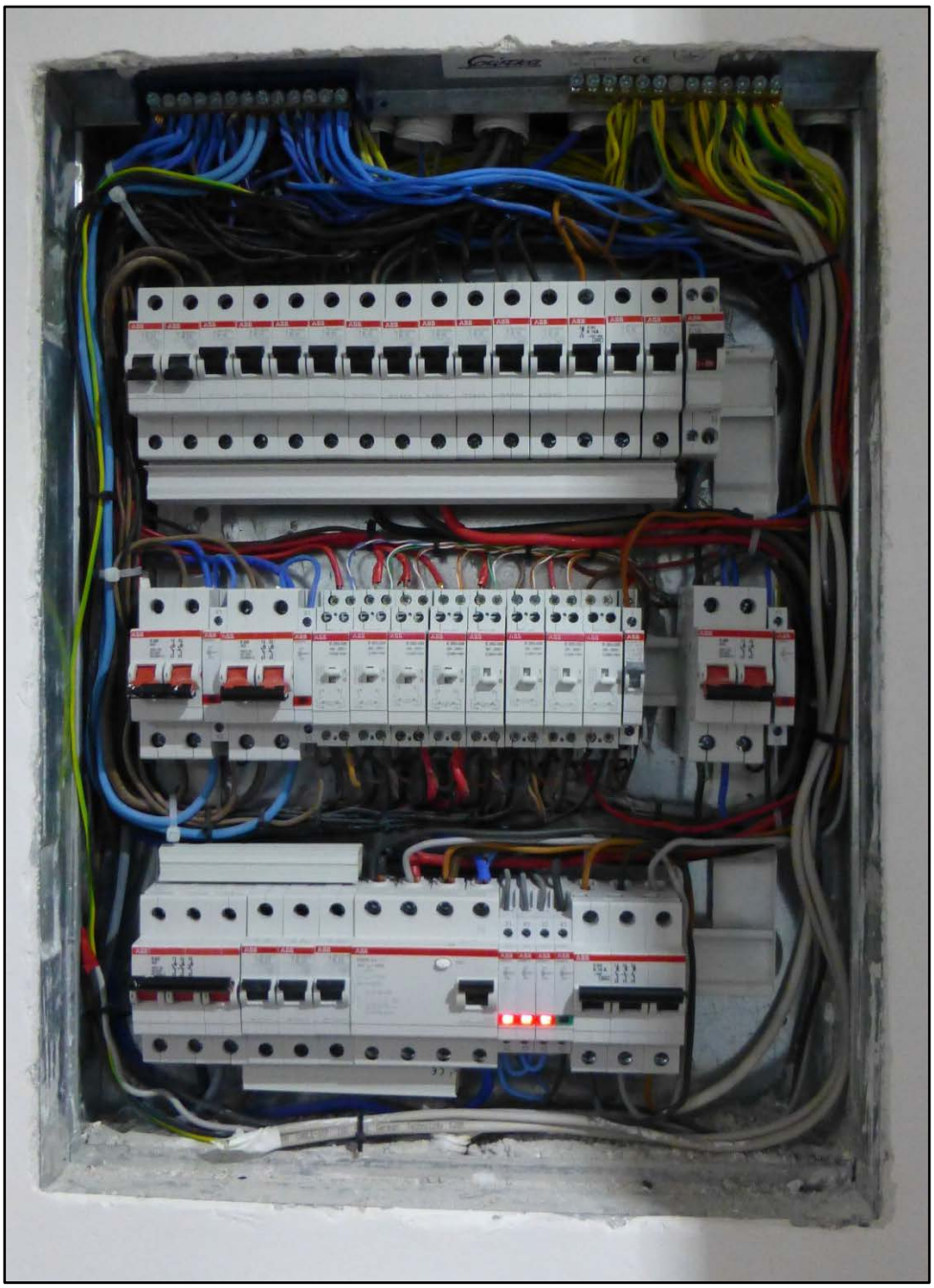
Για τον έλεγχο του σπιτιού επιλέχτηκε να μην αναπτυχτεί Android εφαρμογή, καθώς η ανάπτυξη της είναι χρονοβόρα και τα πλεονεκτήματά της ελάχιστα σε σχέση με μια ιστοσελίδα. Για τη διευκόλυνση της πρόσβασης, η ιστοσελίδα προστέθηκε σαν σελιδοδείκτης στις εφαρμογές που χρησιμοποιεί ο ιδιοκτήτης, ώστε η πρόσβαση στον έλεγχο του σπιτιού να είναι ακόμα πιο εύκολη.

Όσον αφορά στην ασφάλεια του συστήματος, δεν κρίθηκε απαραίτητη η δημιουργία κωδικού πρόσβασης, καθώς η επιλογή μιας συγκεκριμένης «πόρτας» σε συνδυασμό με ένα DNS όνομα παρέχει αρκετή ασφάλεια για ένα τέτοιο σύστημα.

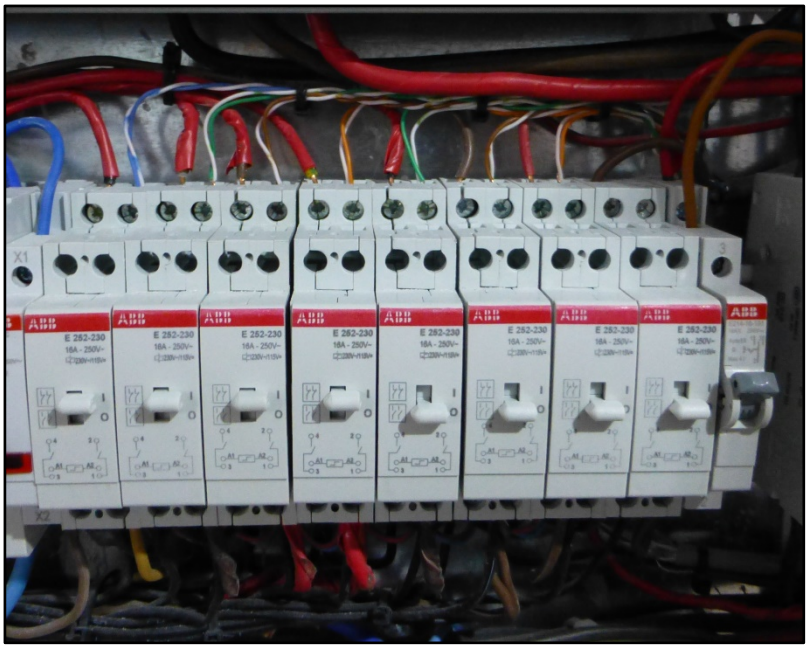


Εικόνα 50. Σελίδα ελέγχου του σπιτιού

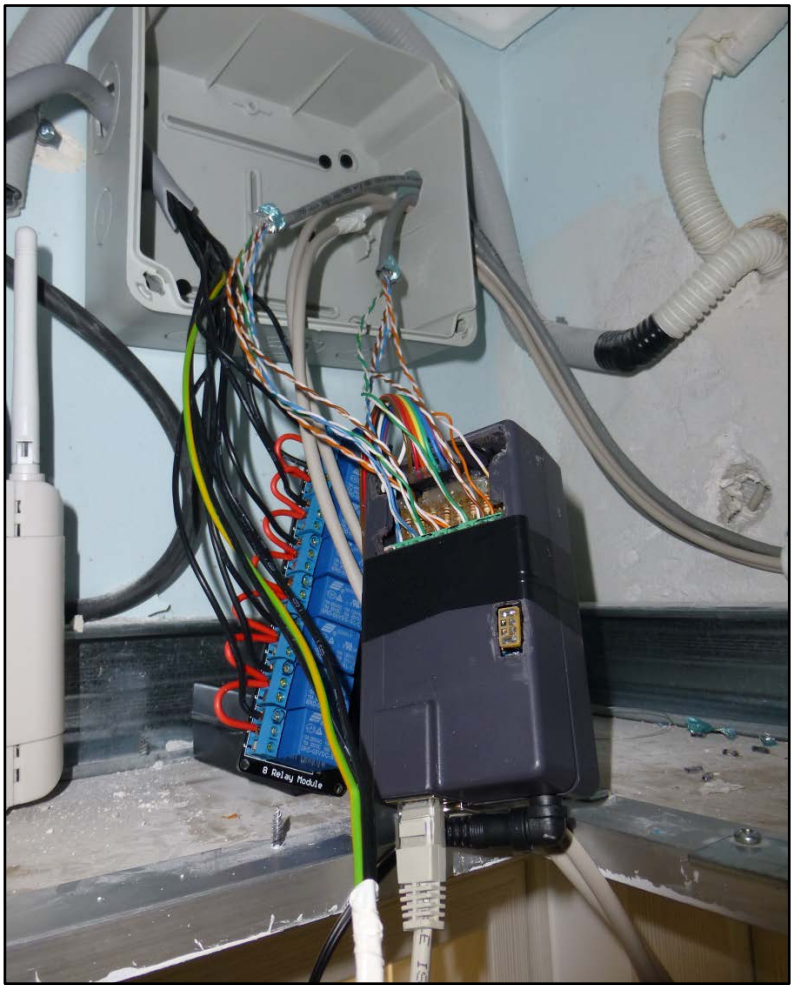
6.3. Εικόνες από την τελική εγκατάσταση



Εικόνα 51. Ο ηλεκτρολογικός πίνακας του σπιτιού



Εικόνα 52. Τα ρελέ καστάνιας που χρησιμοποιήθηκαν



Εικόνα 53. Το κουτί τοποθέτησης του Arduino και του Relay Board

Εικόνα 54. Η τελική εγκατάσταση με τα Solid State Relay

Εικόνα 55. Δοκιμάζοντας το αρχικό κύκλωμα

Συμπεράσματα

Στην παρούσα πτυχιακή εργασία κατασκευάστηκε σε μακέτα ένα πρωτότυπο σύστημα έξυπνου σπιτιού με χαμηλό κόστος το οποίο έχει την δυνατότητα απομακρυσμένου ελέγχου διαφόρων οικιακών συσκευών. Ο έλεγχος μπορεί να γίνει μέσω διαδικτύου ή τοπικού δικτύου. Εκτός από την μακέτα το σύστημα υλοποιήθηκε και σε πραγματικό σπίτι, κάτι το οποίο υποδεικνύει ότι είναι δυνατή η πρακτική εφαρμογή του συστήματος. Οι δυνατότητες που μπορεί να μας προσφέρει ένα τέτοιο σύστημα είναι πάρα πολλές και εξαρτάται από την κάθε υλοποίηση και τις ανάγκες του κάθε πελάτη το τι θα περιλαμβάνει κάθε «έξυπνο σπίτι».

ΒΙΒΛΙΟΓΡΑΦΙΑ

- http://arduino.cc/
- http://forum.arduino.cc/
- http://www.atmel.com/Images/Atmel-2549-8-bit-AVR-Microcontroller-ATmega640-1280-1281-2560-2561_datasheet.pdf
- http://en.wikipedia.org/wiki/Main_Page
- http://playground.arduino.cc/main/DHT11Lib
- Dht 11 datasheet
- Lm35 datasheet
- Gl5546 datasheet
- http://fubar.gr
- The market potential for Smart Homes,Mark Pragnell, Lorna Spence and Roger Moore, 15 November 2000