ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΠΕΛΟΠΟΝΝΗΣΟΥ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ Τ.Ε.

Δομή και προγραμματισμός του συστήματος Arduino

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

της

Αυγουστάτου Ασπασίας

Επιβλέπων:

Μπάρδης Γεώργιος

Σπάρτη, Οκτώβριος 2016

ΔΗΛΩΣΗ ΜΗ ΛΟΓΟΚΛΟΠΗΣ ΚΑΙ ΑΝΑΛΗΨΗΣ ΠΡΟΣΩΠΙΚΗΣ ΕΥΘΥΝΗΣ

Με πλήρη επίγνωση των συνεπειών του νόμου περί πνευματικών δικαιωμάτων, δηλώνω ενυπογράφως ότι είμαι αποκλειστικός συγγραφέας της παρούσας Πτυχιακής Εργασίας, για την ολοκλήρωση της οποίας κάθε βοήθεια είναι πλήρως αναγνωρισμένη και αναφέρεται λεπτομερώς στην εργασία αυτή. Έχω αναφέρει πλήρως και με σαφείς αναφορές, όλες τις πηγές χρήσης δεδομένων, απόψεων, θέσεων και προτάσεων, ιδεών και λεκτικών αναφορών, είτε κατά κυριολεξία είτε βάση επιστημονικής παράφρασης. Αναλαμβάνω την προσωπική και ατομική ευθύνη ότι σε περίπτωση αποτυχίας στην υλοποίηση των ανωτέρω δηλωθέντων στοιχείων, είμαι υπόλογος έναντι λογοκλοπής, γεγονός που σημαίνει αποτυχία στην Πτυχιακή μου Εργασία και κατά συνέπεια αποτυχία απόκτησης του Τίτλου Σπουδών, πέραν των λοιπών συνεπειών του νόμου περί πνευματικών δικαιωμάτων. Δηλώνω, συνεπώς, ότι αυτή η Πτυχιακή Εργασία προετοιμάστηκε και ολοκληρώθηκε από εμένα προσωπικά και αποκλειστικά και ότι, αναλαμβάνω πλήρως όλες τις συνέπειες του νόμου στην περίπτωση κατά την οποία αποδειχθεί, διαχρονικά, ότι η εργασία αυτή ή τμήμα της δε μου ανήκει διότι είναι προϊόν λογοκλοπής άλλης πνευματικής ιδιοκτησίας.

Όνομα και Επώνυμο Συγγραφέα (Με Κεφαλαία):

……

Υπογραφή (Ολογράφως, χωρίς μονογραφή):

……

Ημερομηνία (Ημέρα – Μήνας – Έτος):

……

Περίληψη

Αντικείμενο της παρούσας εργασίας είναι η υλοποίηση και η παρουσίαση μιας εφαρμογής με τη χρήση της πλατφόρμας Arduino, με απώτερο στόχο να γίνει κατανοητή η δομή και η διαδικασία προγραμματισμού της. Η επικαιρότητα και η σημασία του συγκεκριμένου θέματος προκύπτει από τα πλεονεκτήματα που παρουσιάζει η χρήση των υπολογιστικών πλατφορμών αυτού του είδους, όπως το χαμηλό κόστος, η δυνατότητα εύκολης επέμβασης, η απλότητα στον προγραμματισμό, η χρήση ανοιχτού – ελεύθερου κώδικα, η επεκτασιμότητα και το εύρος των εφαρμογών. Για τις ανάγκες της παρούσας εργασίας επιλέχθηκε η υλοποίηση μιας εφαρμογής ελέγχου θερμοκρασίας με τη χρήση του αισθητήρα θερμοκρασίας DS18B20 Dallas και την αποτύπωση των αποτελεσμάτων μέτρησης στη σειριακή οθόνη της πλατφόρμας. Ως σημείο που χρήζει περαιτέρω διερεύνησης αναφέρεται η επέκταση της συγκεκριμένης εφαρμογής, με δυνατότητα απομακρυσμένης καταγραφής και ελέγχου θερμοκρασίας με τη χρήση του διαδικτύου και του απαραίτητου πρόσθετου εξοπλισμού (π.χ. Ethernet shield).

Λέξεις – Κλειδιά: Arduino, Αισθητήρας Θερμοκρασίας, Μικροελεγκτής, Ενσωματωμένο Σύστημα, Ανοιχτό Λογισμικό

Περιεχόμενα

Λίστα Εικόνων

Λίστα Διαγραμμάτων

Λίστα Πινάκων

ΚΕΦΑΛΑΙΟ 1: ΕΙΣΑΓΩΓΗ

1.1 Πρόλογος

Η αναγκαιότητα της παρούσας εργασίας πηγάζει από το γεγονός ότι πραγματεύεται ένα ζήτημα άμεσα συνδεδεμένο με τις ραγδαίες εξελίξεις στον τομέα των ηλεκτρονικών και του προγραμματισμού, δύο πεδία που αναπτύσσονται παράλληλα με το σύνολο των τεχνολογικών τομέων, σε ένα πλαίσιο εξεύρεσης λύσεων αυξημένης ευκολίας, χαμηλού κόστους και ευρέως φάσματος δυνατοτήτων.

Το αντικείμενο λοιπόν της εργασίας που είναι η περιγραφή της λειτουργίας της ηλεκτρονικής πλατφόρμας Arduino και η εξοικείωση με τη δομή και τον προγραμματισμό του, στοιχειοθετεί τη χρησιμότητά του από τα βασικά πλεονεκτήματα που χαρακτηρίζουν τη συγκεκριμένη πλατφόρμα και έχουν να κάνουν με το χαμηλό κόστος υλοποίησης διάφορων εφαρμογών, με τη δυνατότητα εύκολης επέμβασης στη λειτουργία για την πραγματοποίηση διορθωτικών κινήσεων και τη βελτιστοποίηση της απόδοσης του εκάστοτε συστήματος (ευκολία η οποία προκύπτει τόσο από την απλότητα στον προγραμματισμό όσο και από το γεγονός ότι πρόκειται για πλατφόρμα ανοιχτού – ελεύθερου κώδικα).

Ο στόχος της εργασίας είναι να γίνει γνωστός ο τρόπος χρήσης της συγκεκριμένης ηλεκτρονικής πλατφόρμας σε ευρύτερα πλαίσια εμμένοντας στην περιγραφή των εντολών της γλώσσας προγραμματισμού, στην παρουσίαση των περιφερειακών του, καθώς και στη δημιουργία συγκεκριμένης εφαρμογής που ανάγει το θεωρητικό πλαίσιο σε πρακτικό επίπεδο.

Τα κεφάλαια στα οποία αναπτύσσεται η παρούσα εργασία μπορούν να συνοψιστούν ως εξής: Στο πρώτο κεφάλαιο εντοπίζεται ο στόχος και ταυτόχρονα επισημαίνεται η αναγκαιότητα της παρούσας εργασίας. Επιπλέον, γίνεται μια σύντομη επισκόπηση στην εξέλιξη των τεχνολογιών της πληροφορικής, των επικοινωνιών και της ηλεκτρονικής και την προδιαγραφόμενη τάση για την ολοένα και αυξανόμενη ανάπτυξη ενσωματωμένων υπολογιστικών συστημάτων (embedded systems)

Στο δεύτερο κεφάλαιο περιγράφεται η τεχνολογία της ηλεκτρονικής πλατφόρμας Arduino. Γίνεται μια ιστορική αναδρομή της κατασκευής της, καταγράφονται τα υπάρχοντα μοντέλα και αναλύεται ο τρόπος με τον οποίο υλοποιείται μια εφαρμογή μέσω της συγκεκριμένης πλατφόρμας (τόσο σε επίπεδο υλικού όσο και σε επίπεδο λογισμικού). Επισημαίνονται τέλος τα βασικά της χαρακτηριστικά (πλεονεκτήματα και μειονεκτήματα).

Στο τρίτο κεφάλαιο η παρουσίαση επικεντρώνεται στη βασική πλατφόρμα της σειράς, την Arduino Uno η οποία βασίζεται στο ολοκληρωμένο ATmega 328, έναν 8-bit RISC

μικροελεγκτή που χρονίζει στα 16MHz, παραθέτοντας τα δομικά στοιχεία από τα οποία αποτελείται καθώς και τις χρησιμοποιούμενες πλακέτες επέκτασης (shields) οι οποίες διευρύνουν κατά πολύ το εύρος εφαρμογών της συγκεκριμένης τεχνολογίας.

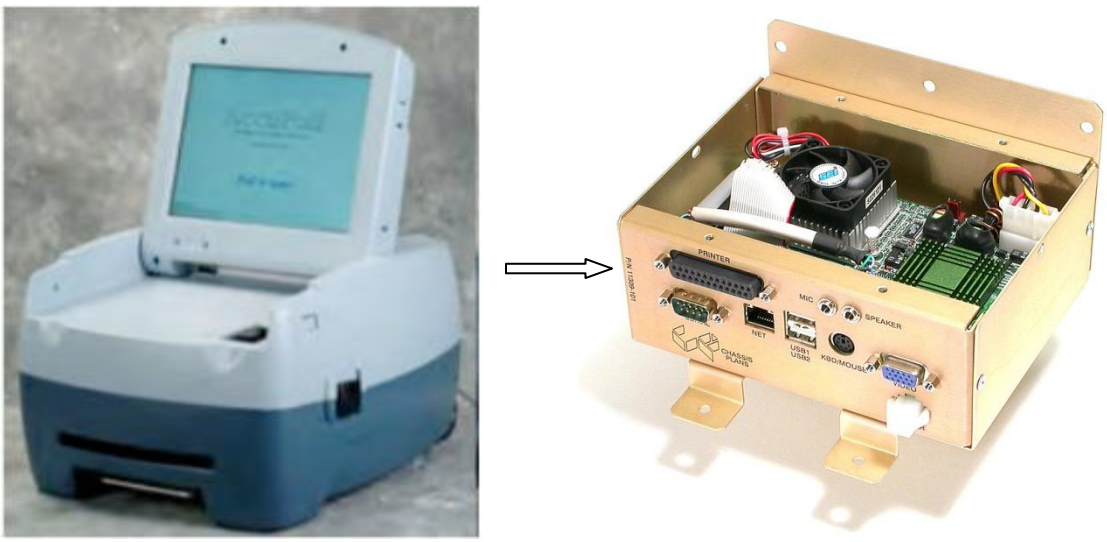
Στο τέταρτο κεφάλαιο συμπεριλαμβάνεται η παρουσίαση της εφαρμογής της παρούσας εργασίας. Παρουσιάζεται η δημιουργία ενός συστήματος καταγραφής της θερμοκρασίας του περιβάλλοντος του συστήματος χώρου, με το σύστημα να χρησιμοποιεί την πλατφόρμα Arduino σε συνεργασία με κατάλληλο αισθητήριο θερμοκρασίας (π.χ. DS18B20 Dallas temp sensor) για την αρχική καταγραφή της, ενώ στη συνέχεια προτείνονται εναλλακτικές επέκτασης του συστήματος για την αποτελεσματικότερη και αποδοτικότερη αξιοποίησή του.

Τέλος, στο πέμπτο κεφάλαιο διατυπώνονται τα εξαγόμενα συμπεράσματα καθώς και συγκεκριμένες προτάσεις βελτίωσης της ευρύτερης διαδικασίας σύμφωνα με κενά – αδύνατα σημεία που έχουν εντοπιστεί στην προαναφερόμενη ανάλυση.

1.2 Ενσωματωμένα υπολογιστικά συστήματα και προσανατολισμός προς την ευρεία εφαρμογή τους

Αρχικά θα πρέπει να περιγραφεί η έννοια των ενσωματωμένων υπολογιστικών συστημάτων. Πρόκειται για υπολογιστές ειδικού σκοπού που σχεδιάζονται και βελτιστοποιούνται για συγκεκριμένες εφαρμογές (πχ, οικιακές, δικτυακές συσκευές, αυτοματισμοί κλπ) και δεν αποτελούν γενικές πλατφόρμες ανάπτυξης οποιονδήποτε εφαρμογών όπως είναι οι Προσωπικοί Υπολογιστές (Διομήδης και Μέγα, 2015). Ουσιαστικά, πρόκειται για διατάξεις που αποτελώντας μέρη ενός ευρύτερου συνόλου (συσκευή - κύκλωμα προς έλεγχο) επιτελούν μια ειδική λειτουργία στο πλαίσιο εξυπηρέτησης του γενικού σκοπού της αντίστοιχης συσκευής – κυκλώματος. Τα βασικά τους μέρη είναι η Κεντρική Μονάδα Επεξεργασίας (διαχείριση πληροφορίας σύμφωνα με ένα πρόγραμμα εντολών), οι μονάδες εισόδου (μετατροπή αναλογικών σημάτων σε ψηφιακά, διακόπτες κτλ.) τις μονάδες εξόδου (οθόνες, beeper, κεραίες, ρελέ κλπ.) για την επικοινωνία του συστήματος με το περιβάλλον, η μνήμη και το σύστημα χρονισμού (ρολόι).

Στην εικόνα που ακολουθεί εμφανίζεται ένα χαρακτηριστικό παράδειγμα ενσωματωμένου υπολογιστικού συστήματος που βρίσκεται στο εσωτερικό της συσκευής ηλεκτρονικής ψηφοφορίας AccuPoll — AVS 1000.



Εικόνα 1: Ενσωματωμένο υπολογιστικό σύστημα που βρίσκεται στο εσωτερικό της συσκευής ηλεκτρονικής ψηφοφορίας AccuPoll — AVS 1000 (Electronic Voting Machine Information Sheet Accupoll AVS 1000; http://en.wikipedia.org/wiki/Embedded_systems)

Περαιτέρω αποσαφήνιση χρειάζεται επίσης για τους όρους μικροελεγκτή και μικροεπεξεργαστή που συχνά συγχέονται. Ένας μικροεπεξεργαστής αποτελείται από ένα ενιαίο ολοκληρωμένο κύκλωμα (IC) στο οποίο υλοποιούνται λειτουργίες της κεντρικής μονάδας επεξεργασίας (ΚΜΕ). Με τον όρο μικροελεγκτής ονομάζεται το λειτουργικό υπολογιστικό σύστημα σε πλακίδιο ("system-on-a-chip"), το οποίο περιέχει έναν επεξεργαστικό πυρήνα, μνήμη και περιφερειακές συσκευές εισόδου/εξόδου (αποτελώντας δηλαδή ένα υπερσύνολο του μικροεπεξεργαστή). Ακόμα μια βασική διαφορά τους σχετίζεται με το σκοπό για τον οποίο προορίζεται κάθε φορά η χρήση τους. Έτσι, για την περίπτωση που ο μικροεπεξεργαστής αφορά σε έναν προσωπικό υπολογιστή δίνεται μεγαλύτερη βαρύτητα στην υπολογιστική του ισχύ ενώ όταν προορίζεται για ένα ενσωματωμένο υπολογιστικό σύστημα δίνεται έμφαση στο μικρό αριθμό ολοκληρωμένων κυκλωμάτων, το χαμηλό κόστος και την εξειδίκευση (Βουρδουρίδης, 2012).

Στη συνέχεια καταγράφονται χρονικά σημεία – σταθμοί στην εξέλιξη των ενσωματωμένων υπολογιστικών συστημάτων

- 1946: ENIAC- Ο πρώτος ηλεκτρονικός υπολογιστής
- 1940-1950: Ο υπολογιστής Whirlwind του MIT σχεδιάστηκε για λειτουργίες πραγματικού χρόνου (με τη βασική ιδέα να αφορά στη χρήση του για τον έλεγχο ενός προσομοιωτή αεροσκαφών)

- 1961: Κατασκευάζεται ο υπολογιστής καθοδήγησης Autonetics D-17 του πυραύλου Minuteman
- 1966: Η μαζική παραγωγή των ολοκληρωμένων κυκλωμάτων συμπίπτει με την επίσης μαζική παραγωγή του παραπάνω πυραύλου
- 1960-1970: Σημειώνεται μεγάλη μείωση του κόστους των ενσωματωμένων συστημάτων, ενώ παράλληλα υπάρχει και μια δραματική αύξηση της επεξεργαστικής ισχύος και της λειτουργικότητας τους. Εμφάνιση του πρώτου μικροεπεξεργαστή (Intel 4004)
- 1972: Η αριθμομηχανή χειρός HP-35 χρησιμοποιούσε αρκετά ολοκληρωμένα κυκλώματα/τσιπ για την υλοποίηση ενός μικροεπεξεργαστή.
- 1970-1980: Εκτεταμένη χρήση των συστημάτων στη βιομηχανία των αυτοκινήτων για τον έλεγχο της μηχανής (έλεγχος μίγματος καυσίμου/αέρα, χρονισμός μηχανής, προθέρμανση, σύστημα αυτόματου πιλότου, ανάβαση λόφου, κ.λ.π.)
- 1978: Η National Engineering Manufacturers Association δημιουργεί ένα "πρότυπο" για προγραμματιζόμενους μικροελεγκτές
- 1980-1990: Η χρήση τους αυξάνεται κατακόρυφα, ιδιαίτερα εξαιτίας της ενσωμάτωσης εξωτερικών παλαιότερα εξαρτημάτων (π.χ. ποτενσιόμετρα και μεταβλητοί πυκνωτές) στο ίδιο chip(http://en.wikipedia.org/wiki/Embedded_systems; Σολωμονίδης, 2011)

Τα χρόνια που ακολουθούν η εξέλιξη είναι ραγδαία, αξιοποιώντας τις εξελίξεις στον τομέα της ηλεκτρονικής, των τηλεπικοινωνιών και των υπολογιστών. Η ευρύτητα των εφαρμογών των ενσωματωμένων συστημάτων είναι ενδεικτική από την εικόνα που ακολουθεί και το πλήθος των πεδίων εφαρμογής.

Anti-lock brakes
Auto-focus cameras
Automatic teller machines
Automatic toll systems
Automatic transmission
Avionic systems
Battery chargers
Camcorders
Cell phones
Cell-phone base stations
Cordless phones
Cruise control
Curbside check-in systems
Digital cameras
Disk drives
Electronic card readers
Electronic instruments
Electronic toys/games
Factory control
Fax machines
Fingerprint identifiers
Home security systems
Life-support systems
Medical testing systems
Modems
MPEG decoders
Network cards
Network switches/routers
On-board navigation
Pagers
Photocopiers
Point-of-sale systems
Portable video games
Printers
Satellite phones
Scanners
Smart ovens/dishwashers
Speech recognizers
Stereo systems
Teleconferencing systems
Televisions
Temperature controllers
Theft tracking systems
TV set-top boxes
VCR's, DVD players
Video game consoles
Video phones
Washers and dryers

Εικόνα 2: Εφαρμογές ενσωματωμένων συστημάτων (http://slideplayer.gr/slide/2953946/)

Η συγκεκριμένη ευρύτητα σχετίζεται άμεσα με το πλήθος των πλεονεκτημάτων τους, τα βασικότερα από τα οποία είναι (Βουρδουρίδης, 2012):

- Αυτονομία: Μπορούν να λειτουργήσουν χωρίς τη χρήση πρόσθετων υπολογιστικών διατάξεων
- Χαμηλή κατανάλωση ισχύος
- Φορητότητα
- Χαμηλό κόστος
- Αυξημένη αξιοπιστία εξαιτίας του μικρού αριθμού διασυνδέσεων και της λειτουργίας ειδικού σκοπού που τα διακρίνει
- Μειωμένες εκπομπές ηλεκτρομαγνητικών παρεμβολών και μειωμένη ευαισθησία σε αντίστοιχες παρεμβολές από άλλες ηλεκτρικές και ηλεκτρονικές συσκευές (εξαιτίας μικρότερου αριθμού και μήκους εξωτερικών διασυνδέσεων καθώς και χαμηλότερων ταχυτήτων λειτουργίας).
- Αυξημένη διαθεσιμότητα όσον αφορά σε ψηφιακές εισόδους-εξόδους (λόγω απουσίας εξωτερικών περιφερειακών).

ΚΕΦΑΛΑΙΟ 2: Τεχνολογία της πλατφόρμας Arduino

2.1 Γενικά στοιχεία

Περιγράφοντας το Arduino θα μπορούσε να ειπωθεί πως πρόκειται για μια πρωτότυπη ηλεκτρονική πλατφόρμα διαμόρφωσης ανοικτού λογισμικού, βασισμένη στο υλικό ενός μικροεπεξεργαστή, καθώς και σε κατάλληλο για τον προγραμματισμό του λογισμικό (Παπαναστασίου, 2009). Σύμφωνα με τον επίσημο ιστότοπο της πλατφόρμας (https://www.arduino.cc/), βασική ιδέα της συγκεκριμένης τεχνολογίας είναι η ανάγνωση εισόδων (από έναν αισθητήρα, από την κίνηση ενός δαχτύλου ή από ένα μήνυμα του Twitter) και η μετατροπή τους σε επιθυμητά σήματα εξόδου (όπως για παράδειγμα η ενεργοποίηση ενός κινητήρα, η ενεργοποίηση ενός LED, η δημοσίευση ενός αρχείου στο διαδίκτυο κτλ). Χρησιμοποιεί τη γλώσσα προγραμματισμού Wiring (σύνταξη και βιβλιοθήκες) η οποία παρουσιάζει πολλές ομοιότητες με τη C++ με απλοποιήσεις και αλλαγές, καθώς και ένα ολοκληρωμένο περιβάλλον ανάπτυξης (Integrated Development Environment-IDE).

2.1.1 Ιστορική Αναδρομή

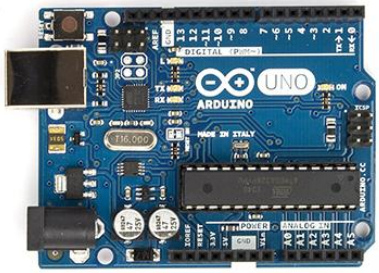
Πρόκειται ουσιαστικά για μια νέα αλλά ραγδαία εξελισσόμενη πλατφόρμα που έκανε την εμφάνισή της για εκπαιδευτικούς σκοπούς από το 2005 και συγκεκριμένα για τον έλεγχο προγραμμάτων διαδραστικών σχεδίων από μαθητές με βασικό ζητούμενο εκτός από τη λειτουργικότητα το χαμηλό κόστος (Lahart, 2009). Η ιδρυτική ομάδα ήταν οι David Cuartielles, Gianluca Martino, Tom Igoe, David Mellis και Massimo Banzi (Kushner, 2011) οι οποίοι ονόμασαν τη δημιουργία τους εμπνευσμένοι από τον Ιταλό ευγενή Arduin της Ivrea που διατέλεσε Μαρκήσιος της Ivrea (990-1015) και Βασιλιάς της Ιταλίας (1002-1014).(https://en.wikipedia.org/wiki/Arduin_of_Ivrea). Από εκείνο το χρονικό σημείο ακολούθησαν γρήγορες εξελίξεις με συνεχείς τεχνολογικές αναβαθμίσεις της πλατφόρμας, με βασικά χρονικά σημεία τα ακόλουθα(https://el.wikipedia.org/wiki/Arduino):

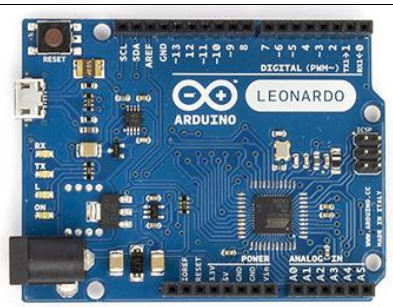
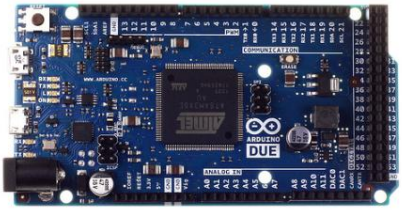
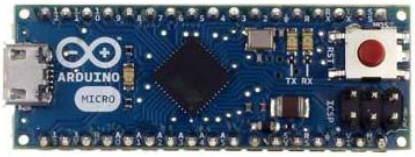
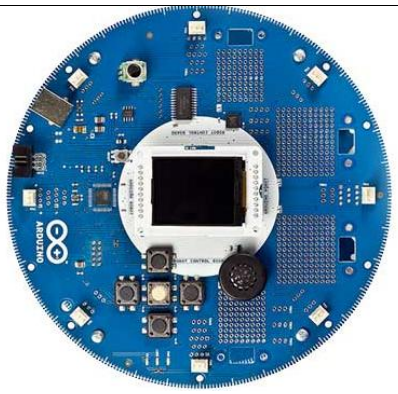
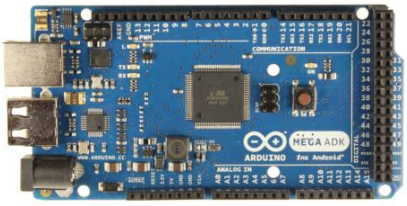
- 2005: Την ίδια χρονιά με την εμφάνιση της πλατφόρμας ξεκινά η μαζική της παραγωγή σε ένα μικρό εργοστάσιο στην Ιβρέα
- 2006: Ανακοινώνεται το Arduino Mini, ενώ το ίδιο έτος το Arduino λαμβάνει τιμητική αναφορά στο ψηφιακό τμήμα Κοινοτήτων των ARS Electronica Prix (Παπαναστασίου, 2009).

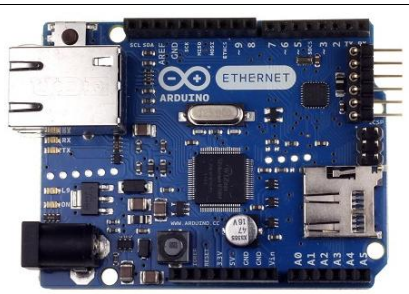
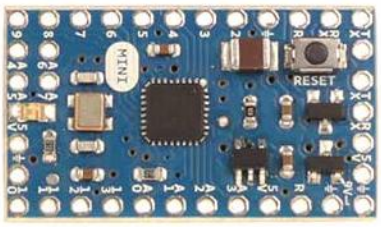
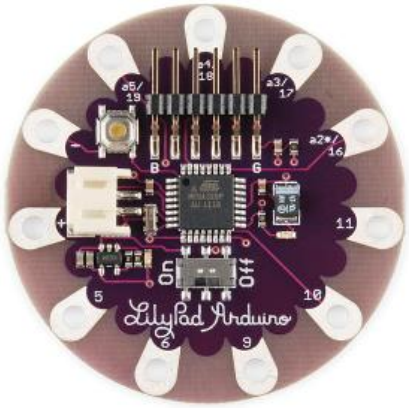
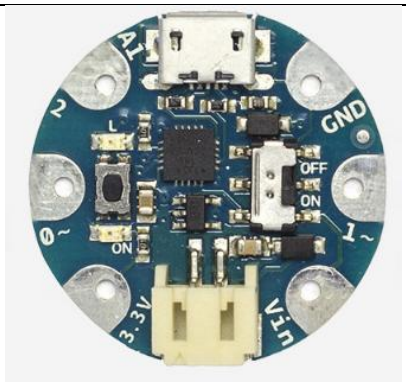
- 2008: Ανακοινώνεται το Arduino Duemilanove
- 2009: Ανακοινώνεται το Arduino Mega.
- 2011: Βρίσκονται ήδη σε χρήση πάνω από 300,000 Arduino σε παγκόσμια κλίμακα.
- 2012: Ανακοινώνονται τα Arduino Leonardo, Arduino Due και Micro.
- 2013: Κάνει την εμφάνισή του το Arduino Robot, το πρώτο επίσημο Arduino με ρόδες αλλά και το Arduino Yun, το πρώτο προϊόν wifi που συνδύαζε το Arduino με το Linux.

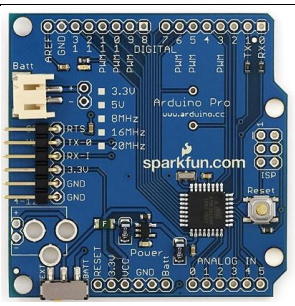
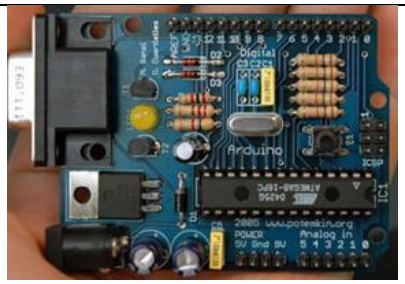
2.2 Διαθέσιμα μοντέλα

Υφίστανται διάφορα μοντέλα της πλατφόρμας, με τη διαδικασία κατασκευής και τεχνολογικής αναβάθμισης να είναι να δυναμική και συνεχής. Στην παρούσα ενότητα επιχειρείται μια συνοπτική καταγραφή των υφιστάμενων μοντέλων και των κύριων χαρακτηριστικών τους, με τη λίστα να βρίσκεται ασφαλώς υπό συνεχή ανανέωση. Στον πίνακα που ακολουθεί γίνεται μια σύντομη παρουσίαση των πρόσφατων πλακετών (με μικρή αναφορά σε χαρακτηριστικά παλαιότερα μοντέλα όπως το Robot). Σημειώνεται επίσης η διάθεση των πλακετών Genuino εκτός ΗΠΑ.

Μοντέλο	*Εικόνα*	*Περιγραφή - Στοιχεία*
Uno		Αποτελεί το μοντέλο αναφοράς. Διατίθεται στις εκδοχές (Revisions) R2 και R3
101		Η καινοτομία έγκειται στην ενσωμάτωση νέων τεχνολογιών όπως την αναγνώριση χειρονομιών, την ενσωμάτωση επιταχυνσιομέτρου και γυροσκοπίου, ενώ διαθέτει και συνδεσιμότητα Bluetooth.

Leonardo		Διαθέτει ενσωματωμένη θύρα usb, ενώ μπορεί να χρησιμοποιηθεί ως πληκτρολόγιο και ποντίκι.
Due		Αποτέλεσε την πρώτη πλατφόρμα με ARM πυρήνα 32-bit μικροελεγκτή, ενώ τροφοδοτείται μόνο με 3.3V.
Yun		Συνδυάζει τα τεχνικά χαρακτηριστικά του Leonardo με την καταλληλότητα για διασύνδεση με συστήματα Linux
Micro		Κατασκευάζεται σε συνδυασμό με την Adafruit, ενώ παρουσιάζει παρόμοιες λειτουργίες και χαρακτηριστικά με το Leonardo.
Robot		Διαθέτει ενσωματωμένα ροδάκια και μπορεί να κινηθεί στο χώρο αποτελώντας το πρώτο robot της σειράς
Mega		Πρόκειται με την πλατφόρμα με τη μεγαλύτερη υπολογιστική ισχύ.

Ethernet		Διαθέτει ενσωματωμένη θύρα Ethernet καθώς και αναγνώστη καρτών micro-SD
Mini		Αποτελεί τη μικρότερη πλατφόρμα της σειράς, με βασικό χαρακτηριστικό το κουμπί reset το οποίο βρίσκεται πάνω στην πλακέτα
Lilypad		Αποτελεί την πλατφόρμα που προτιμάται σε wearable εφαρμογές (ενσωματωμένη σε ρούχα – κατασκευές)
Arduino Gemma		Χρησιμοποιείται επίσης σε wearable εφαρμογές με χαρακτηριστικό της το πολύ μικρό μέγεθος.
Nano		Μοιάζει με το μοντέλο Duemilanove, με βασικές διαφορές το μικρότερο μέγεθος και την απώλεια εξωτερικής τροφοδοσίας.

Pro		Προτιμάται σε σταθερές εφαρμογές και για αυτό το λόγο οι υποδοχές του δεν είναι συγκολλημένες.
Serial Arduino		Χαρακτηριστικό του είναι η δυνατότητα σειριακής σύνδεσης (DE-9)
Arduino Extreme		Διαθέτει USB interface για προγραμματισμό

Πίνακας 1: Συνοπτική παρουσίαση διαφορετικών μοντέλων Arduino (http://www.arduino.cc; Χαρουπιάς, 2013; Μαυραειδόπουλος, 2015)

Μια κατηγοριοποίηση που σχετίζεται με το επίπεδο πολυπλοκότητας και το πεδίο εφαρμογών των διαθέσιμων μοντέλων παρουσιάζεται στο ακόλουθο διάγραμμα.

Διάγραμμα 1: Κατηγοριοποίηση διαθέσιμων μοντέλων (https://www.arduino.cc/en/Main/Products)

2.3 Περιγραφή λειτουργίας

Όπως προαναφέρθηκε κατά την αρχική περιγραφή της πλατφόρμας, το Arduino είναι μία «πρωτοτυποποίηση» ηλεκτρονικών κυκλωμάτων βασισμένη σε ευέλικτο και εύκολο στη χρήση «hardware» και «software» που προορίζεται για οποιονδήποτε έχει λίγη προγραμματιστική εμπειρία, στοιχειώδεις γνώσεις ηλεκτρονικών και ενδιαφέρεται να δημιουργήσει διαδραστικά αντικείμενα ή περιβάλλοντα, διευκολύνοντάς τον κατά πολύ αφού δεν τον υποχρεώνει να προγραμματίζει άμεσα σε γλώσσα μηχανής (Delta Hacker, 2009; Χαρουπιάς, 2013; Μαυραειδόπουλος 2015).

2.3.1 Υλικό

Τα βασικά στοιχεία της πλατφόρμας συνοψίζονται στα εξής (αναλυτικότερες πληροφορίες σχετικά με την κάθε δομική μονάδα του εξοπλισμού θα δοθούν σχετικά με την πλατφόρμα που χρησιμοποιήθηκε στην εφαρμογή της παρούσας εργασίας):

- Το μικροεπεξεργαστή
- Τους ακροδέκτες εισόδου εξόδου
- Τη θύρα επικοινωνίας
- Τις μνήμες
- Τους ακροδέκτες τροφοδοσίας

Η επικοινωνία της πλατφόρμας με το περιβάλλον γίνεται προσλαμβάνοντας δεδομένα από μεγάλη ποικιλία αισθητήρων, ενώ η αλληλεπίδραση ολοκληρώνεται με τη δυνατότητα άσκησης ελέγχου σε φώτα, μηχανές και άλλες ηλεκτρικές και ηλεκτρονικές διατάξεις (Delta Hacker, 2009; Μαυραειδόπουλος, 2015). Η ψηφιακή σχεδίαση του υλικού μέρους του Arduino είναι ανοιχτή και προσβάσιμη από όλους μια και είναι δημοσιευμένη υπό την άδεια Creative Commons Attribution Share-Alike. Επίσης, το περιβάλλον ανάπτυξης (IDE) του Arduino είναι ελεύθερο λογισμικό και είναι δημοσιευμένο υπό την άδεια GNU General Public License. (Φελλόπουλος και Σπύρου, 2012).

2.3.2 Λογισμικό

Τα βασικά μέρη του λογισμικού που περιλαμβάνει η πλατφόρμα συνοψίζονται στα εξής:

- ένα ολοκληρωμένο περιβάλλον ανάπτυξης της πλατφόρμας για τη συγγραφή των προγραμμάτων με συντακτική χρωματική σήμανση (με το κάθε πρόγραμμα να ονομάζεται sketch),
- έτοιμα παραδείγματα,
- έτοιμες βιβλιοθήκες για προέκταση της γλώσσας και για το χειρισμό των εξαρτημάτων
- τον compiler για την μεταγλώττιση των sketches
- ένα serial monitor που παρακολουθεί τις επικοινωνίες της σειριακής (USB) και αναλαμβάνει να στείλει αλφαριθμητικά την εκάστοτε επιλογή στο Arduino, ενώ παράλληλα μέσω αυτού γίνεται γνωστό κάθε φορά το αποτέλεσμα του debugging των sketches και

- την επιλογή για uploading των μεταγλωττισμένων sketches στο Arduino (Deltahacker, 2009; Μαυραειδόπουλος, 2015).

Το περιβάλλον ανάπτυξης του Arduino (IDE) έχει συγγραφεί με την γλώσσα προγραμματισμού Java, γεγονός που εξασφαλίζει συμβατότητα με τα περισσότερα λειτουργικά συστήματα. Η γλώσσα προγραμματισμού που χρησιμοποιείται για την συγγραφή προγραμμάτων στο Arduino είναι η Wiring (C, C++). Το IDE του Arduino χρησιμοποιεί εργαλεία GNU toolchain και AVR Libc για να παρέχει την μεταγλώττιση προγραμμάτων από C, C++ σε κατάλληλες AVR εντολές γλώσσας μηχανής, καθώς και το εργαλείο avrudude για την αποστολή του εκτελέσιμου προγράμματος στην Flash memory του Arduino. (Φελλόπουλος και Σπύρου, 2012).

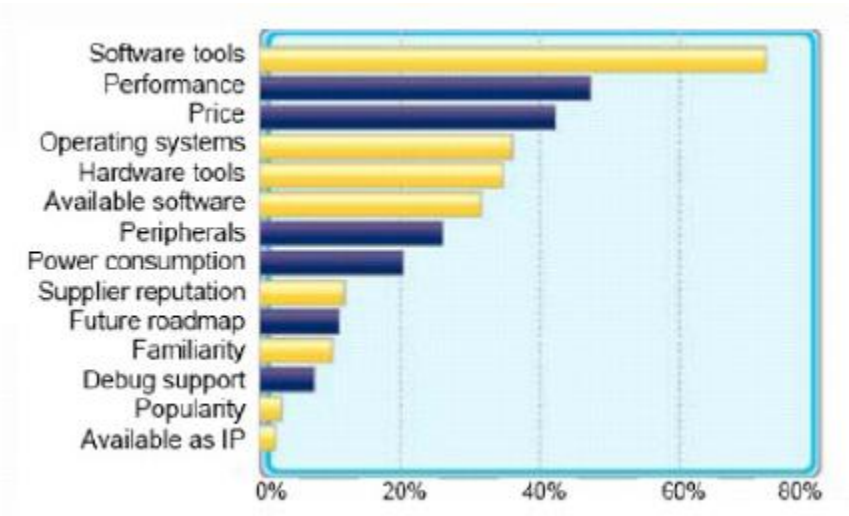
2.4 Πλεονεκτήματα της πλατφόρμας

Ένα εύλογο ερώτημα είναι γιατί να προτιμηθεί η συγκεκριμένη πλατφόρμα όσον αφορά στην υλοποίηση συγκεκριμένων εφαρμογών. Εκτός από τα γενικότερα πλεονεκτήματα των ενσωματωμένων συστημάτων, η συγκεκριμένη πλατφόρμα παρουσιάζει:

- Εξαιρετικά ευρεία ποικιλία εφαρμογών.
- Ευκολία χρήσης και ευελιξία λογισμικού.
- Συμβατότητα με διαφορετικά λειτουργικά περιβάλλοντα (Mac, Windows, and Linux).
- Χαμηλό κόστος.
- Μπορεί να αποβεί ένας παράγοντας – κλειδί προκειμένου να υποστηριχθεί η διαδικασία απόκτησης νέας γνώσης.
- Ανοικτό λογισμικό.
- Επεκτασιμότητα (χρήση πρόσθετων shields για διάφορες εφαρμογές)
- Με την πάροδο του χρόνου δημιουργείται μια προσβάσιμη βάση γνώσης πολλών και διαφορετικής κλίμακας - πολυπλοκότητας διατάξεων, ως συνεισφορά όσων ασχολούνται καθημερινά με τη συγκεκριμένη πλατφόρμα (μαθητές, χομπίστες, καλλιτέχνες, προγραμματιστές και επαγγελματίες).

Προκειμένου να εξηγηθεί η αυξημένη προτίμηση που παρουσιάζει η συγκεκριμένη πλατφόρμα, ιδιαίτερο ενδιαφέρον παρουσιάζει το ακόλουθο διάγραμμα, στο οποίο παρουσιάζονται οι λόγοι προτίμησης συγκεκριμένης ενσωματωμένης αρχιτεκτονικής σύμφωνα με αντίστοιχη έρευνα (Information Day 07/05/2007, Βρυξέλλες). Όπως μπορεί να παρατηρηθεί τα εργαλεία λογισμικού, η απόδοση και η τιμή βρίσκονται στις ψηλότερες θέσεις

της λίστας προτίμησης, κριτήρια τα οποία πληρούνται από τη συγκεκριμένη πλατφόρμα, επεξηγώντας κατά ένα ποσοστό την αυξημένη προτίμηση που της επιδεικνύεται.



Διάγραμμα 2: Λόγοι προτίμησης συγκεκριμένης ενσωματωμένης αρχιτεκτονικής (http://slideplayer.gr/slide/2953946/)

ΚΕΦΑΛΑΙΟ 3: Τα βασικά μέρη της πλατφόρμας ARDUINO UNO

Στη συνέχεια περιγράφονται τα βασικά συστατικά στοιχεία της πλατφόρμα που χρησιμοποιήθηκε στα πλαίσια υλοποίησης της εφαρμογής της παρούσας εργασίας (Shilling, 2012; Χαρουπιάς, 2013; Μαυραειδόπουλος 2015).

Θύρες (pins) εισόδου – εξόδου

Υπάρχουν 4 διαφορετικές κατηγορίες θυρών στην πλατφόρμα, σύμφωνα με τον πίνακα που ακολουθεί ως προς τα χαρακτηριστικά τους μεγέθη.

Τύπος θύρας	*Επιτρεπτές τιμές*	*Εύρος τάσης*
Ψηφιακή έξοδος	0,1	0,5
Ψηφιακή είσοδος	0,1	0,5
Αναλογική έξοδος (PWM)	0-255	0-5V
Αναλογική είσοδος	0-1024	0 -5V

Πίνακας 2: Κατηγορίες θυρών της πλατφόρμας Arduino Uno

Ψηφιακές θύρες

Η πλατφόρμα διαθέτει συνολικά 14 ψηφιακές θύρες εισόδου και εξόδου («I/O») οι οποίες μπορούν χρησιμοποιηθούν τόσο ως είσοδοι όσο και ως έξοδοι ψηφιακών σημάτων (ανάλογα με τις επιλογές του προγράμματος). Οι θύρες 3, 5, 6, 9, 10 και 11 μπορούν να λειτουργήσουν και ως ψευδοαναλογικές θύρες εξόδου μέσω της τεχνικής PWM (Pulse Width Modulation). Ο χαρακτηρισμός ψευδοαναλογική οφείλεται στο ότι δεν περιλαμβάνει ακριβώς αναλογική λειτουργία (για παράδειγμα αν στην είσοδο τροφοδοτηθεί το μισό της κλίμακας τιμών δηλαδή το 127, τότε στην έξοδο δεν θα προκύψει το μισό του εύρους τάσης δηλαδή 2,5 αλλά η κυματομορφή εξόδου θα διαμορφωθεί έτσι ως προς τη συχνότητα ώστε η μέση τιμή να ισούται τελικά με 2,5V). Για την αποστολή και τη λήψη σειριακών δεδομένων μπορούν να χρησιμοποιηθούν οι θύρες 0 και 1 και ενεργοποιούνται όταν επιλεχθεί το σειριακό «interface» της πλατφόρμας. Τα σειριακά δεδομένα προωθούνται στη θύρα USB μέσω του ελεγκτή «Serial-Over-Usb» όπως επίσης και στο pin 0 για να τα διαβάσει ενδεχομένως μία άλλη συσκευή (π.χ. ένα δεύτερο Arduino στη δικιά του θύρα 1). Οι θύρες 2 και 3 μπορούν να λειτουργήσουν ως εξωτερικά interrupts δηλαδή ως είσοδοι διακοπής (0 και 1 αντίστοιχα), στις οποίες όταν τροφοδοτηθεί κατάλληλη τάση μπορεί το τρέχον πρόγραμμα να διακοπεί και να υφίσταται παραπομπή σε μια άλλη λειτουργία.

Αναλογικές θύρες

Υφίστανται 6 αναλογικές θύρες εισόδου αριθμημένες από το 0 έως το 5, οι οποίες λειτουργούν με βάση την τεχνική ADC (Analog to Digital Converter). Έτσι η αναλογική τάση με την οποία τροφοδοτείται η θύρα μετασχηματίζεται σε ένα ακέραιο αριθμό χωρητικότητας 10-bit, από το 0 μέχρι το 1023 (για 0 και 5V αντίστοιχα).

Θύρες τροφοδοσίας

Πρόκειται για μια συστοιχία 6 διαφορετικών pins, μα το καθένα να επιτελεί μια διαφορετική λειτουργία. Αναλυτικότερα, το πρώτο, με την ένδειξη «RESET», όταν γειωθεί (με οποιοδήποτε από τα 3 pin με την ένδειξη «GND» που υπάρχουν στο Arduino) επανεκκινεί την πλατφόρμα. Το δεύτερο, με την ένδειξη 3.3V, μπορεί να τροφοδοτήσει διατάξεις, συσκευές ή αισθητήρες με τάση 3.3V και ρεύμα μέγιστης έντασης 50 mA, η οποία προέρχεται από τον ελεγκτή «Serial-over-Usb».Το τρίτο, με την ένδειξη «5V», ομοίως με το προηγούμενο αλλά με τάση των 5V η οποία προέρχεται είτε από τη θύρα USB είτε από την εξωτερική τροφοδοσία με ρύθμιση. Το τέταρτο και το πέμπτο pin με την ένδειξη «GND» είναι οι γειώσεις. Το έκτο και τελευταίο pin, με την ένδειξη «Vin» μπορεί να λειτουργήσει ως μέθοδος εξωτερικής τροφοδοσίας είτε του Arduino είτε άλλων εξαρτημάτων και συσκευών με εύρος παρεχόμενης τάσης τα 7-12V.

Διακόπτης micro-switch RESET

Χρησιμοποιείται για τη χειροκίνητη επανεκκίνηση της πλατφόρμας.

LEDs

- LED POWER: Σηματοδοτεί την ύπαρξη τροφοδοσίας.
- LEDs TX και RX: Σηματοδοτούν τη λειτουργία του σειριακού interface, όταν δηλαδή αποστέλλονται ή λαμβάνονται δεδομένα μέσω της Usb, όχι όμως όταν χρησιμοποιούνται οι ψηφιακές θύρες 0 και 1.
- LED L: Είναι συνδεδεμένο με τη θύρα 13 και χρησιμοποιείται για δοκιμαστικούς σκοπούς.

Θύρα εξωτερικής τροφοδοσίας

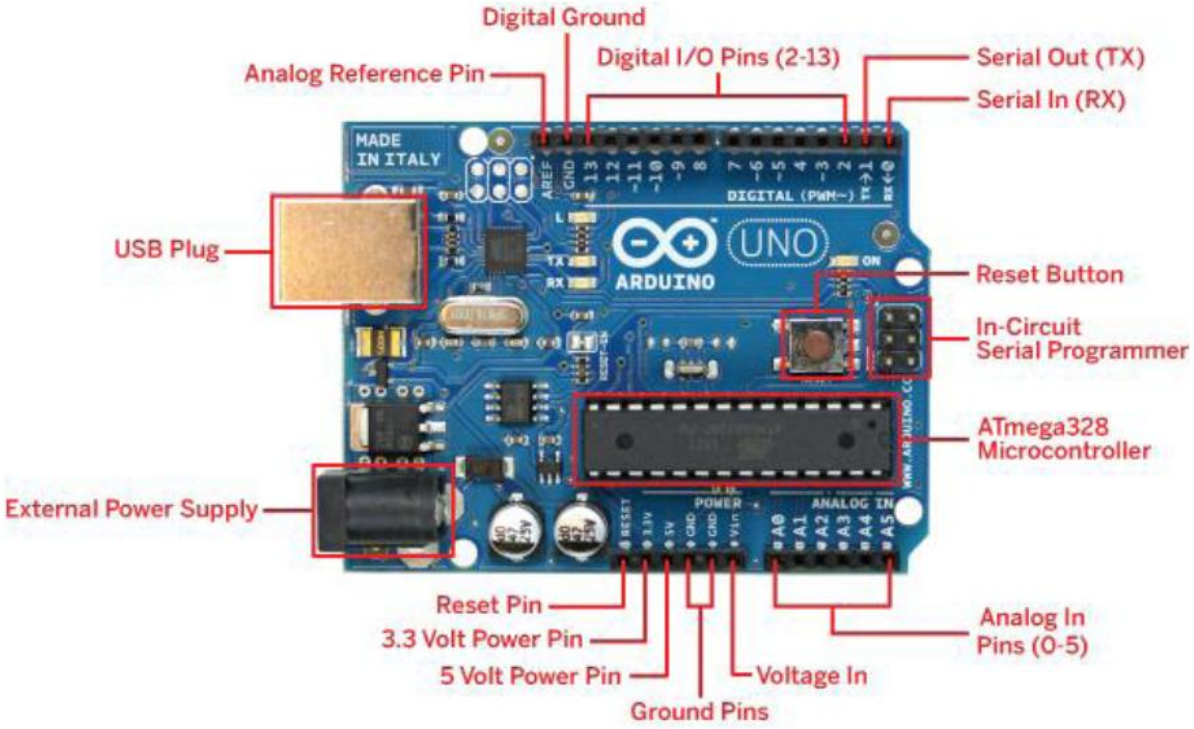
Η πλατφόρμα εκτός από την τροφοδοσία της από τη θύρα Usb ενός υπολογιστή, μπορεί να τροφοδοτηθεί και από μία εξωτερική πηγή τροφοδοσίας (9 volt) συνεχούς ρεύματος με

βύσμα 2.1mm «barrel tip», με το θετικό πόλο στην εσωτερική και τον αρνητικό στην εξωτερική πλευρά του βύσματος.

Μνήμες

- Flash memory: έχει χωρητικότητα 32Kb (με τα 2Kb να έχουν δεσμευτεί από τον κατασκευαστή για την εγκατάσταση του bootloader του firmware δηλαδή που επιτρέπει την εγκατάσταση των προγραμμάτων στον μικροελεγκτή μέσω της θύρας USB) στην οποία αποθηκεύονται τα προγράμματα που καλούνται να «τρέξουν» στην πλατφόρμα αφού πρώτα μεταγλωττιστούν στον υπολογιστή. Διατηρεί τα περιεχόμενά της σε περίπτωση απώλειας της τροφοδοσίας ή επανεκκίνησης της πλατφόρμας. Μπορεί επίσης τυχόν διαθέσιμος χώρος να χρησιμοποιηθεί για την αποθήκευση sketches.
- SRAM memory: Η μνήμη SRAM (static random access memory) των 2048 bytes λειτουργεί κατ' αναλογία με τον υπολογιστή και χρησιμοποιείται για την αποθήκευση μεταβλητών, πινάκων κ.λπ. ενώ το πρόγραμμα «τρέχει», χάνει δε τα περιεχόμενά της σε περίπτωση απώλειας της τροφοδοσίας ή επανεκκίνησης της πλατφόρμας.
- EEPROM memory: Η μνήμη EEPROM των 1024 bytes χρησιμοποιείται μόνο για ανάγνωση (read-only), ενώ έχει πεπερασμένη διάρκεια ζωής (δε μπορεί να επαναπρογραμματιστεί για περισσότερες από 100.000 φορές). Η χρήση της γίνεται μόνο αφού φορτωθεί ειδική σχετική βιβλιοθήκη.

Το σύνολο των προαναφερόμενων μερών της πλατφόρμα παρουσιάζεται στην εικόνα που ακολουθεί.



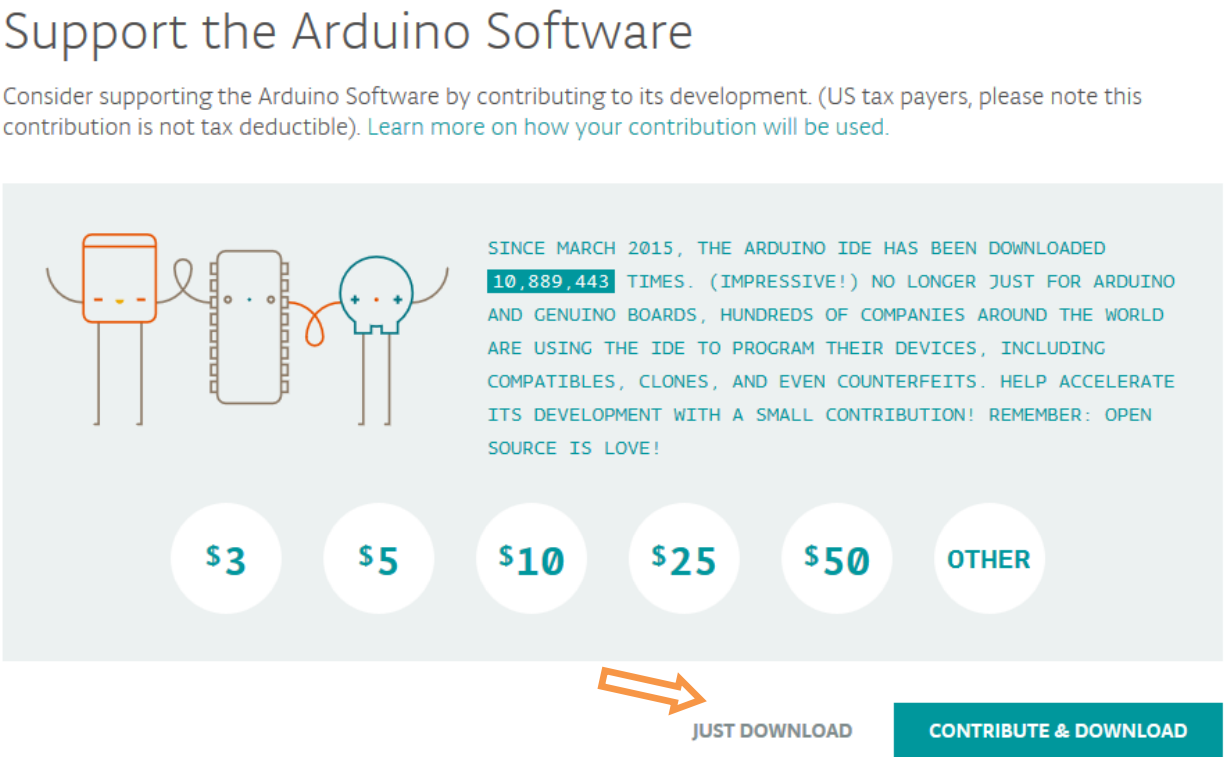
Εικόνα 3: Τα βασικά συστατικά μέρη της πλατφόρμας ARDUINO UNO (Φελλόπουλος και Σπύρου, 2012)

ΚΕΦΑΛΑΙΟ 4: ΠΑΡΟΥΣΙΑΣΗ ΤΗΣ ΕΦΑΡΜΟΓΗΣ

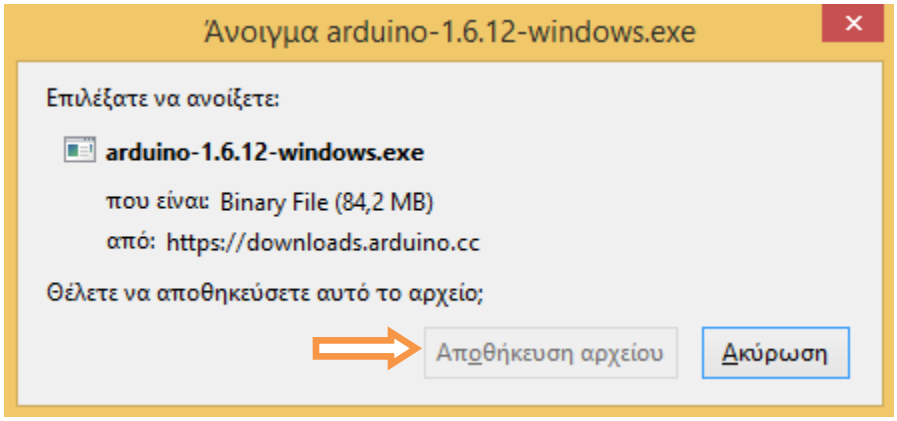
4.1 Εγκατάσταση του περιβάλλοντος ανάπτυξης

Προκειμένου να υλοποιηθεί μια εφαρμογή με τη χρήση της πλατφόρμας Arduino, θα πρέπει να εγκατασταθεί στον υπολογιστή στον οποία θα «τρέξει» η συγκεκριμένη εφαρμογή το περιβάλλον ανάπτυξής της, η βάση δηλαδή που θα «υποδεχθεί» τον κώδικα του προγράμματος και θα ελέγξει το συνολικό υλικό της εφαρμογής (κατά αντιστοιχία για παράδειγμα με τα Windows και τα Office). Στις εικόνες που ακολουθούν παρατίθενται αναλυτικά τα βήματα της διαδικασίας εγκατάστασης με το κίτρινο βέλος να επισημαίνει κάθε φορά το κουμπί επιλογής προκειμένου να ολοκληρωθεί η συγκεκριμένη εγκατάσταση.

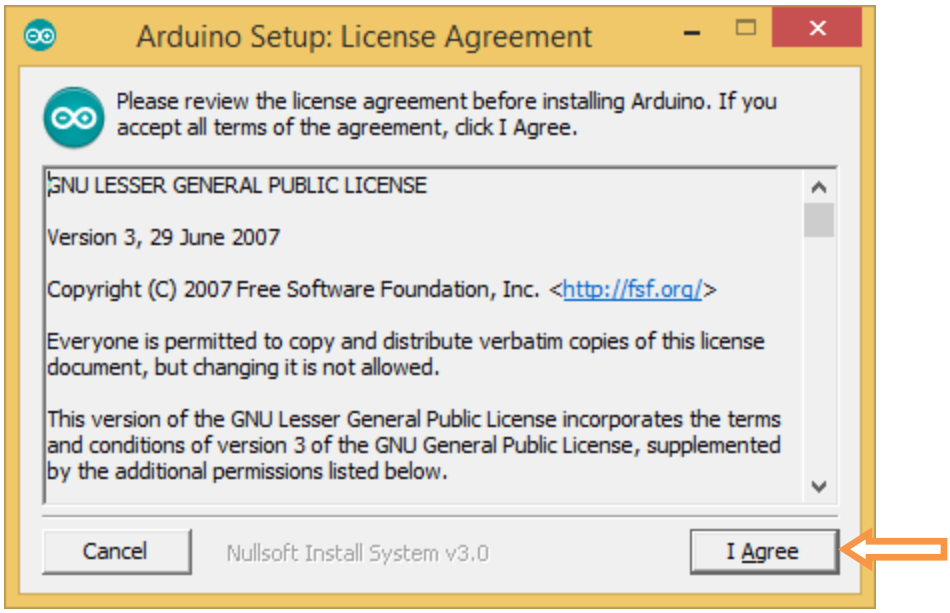
Εικόνα 4: Ιστότοπος για τη λήψη του αρχείου εγκατάστασης του περιβάλλοντος ανάπτυξης του Arduino (https://www.arduino.cc/en/Main/Software)



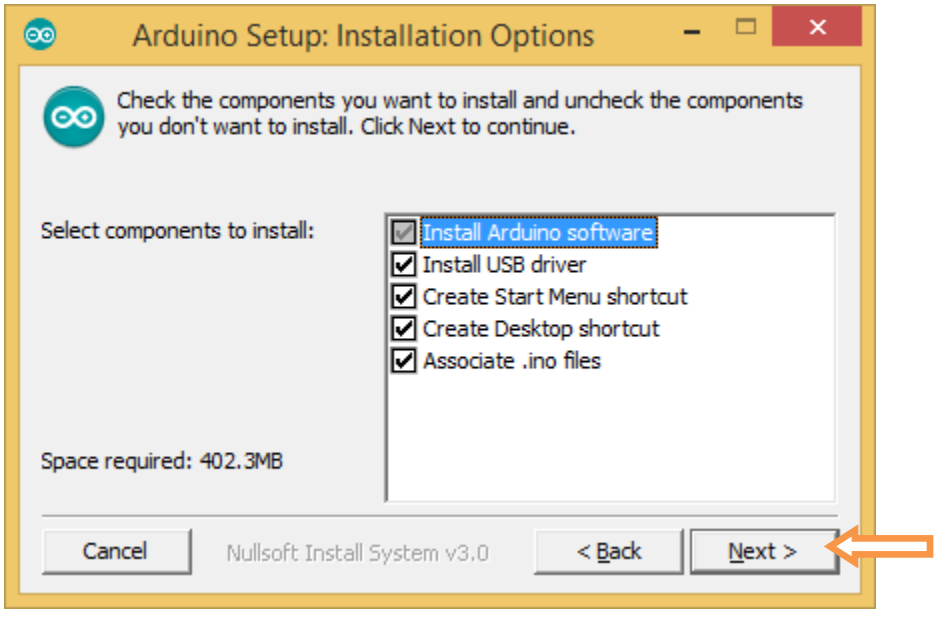
Εικόνα 5: Θέση επιλογής για τη λήψη του αρχείου εγκατάστασης του περιβάλλοντος ανάπτυξης του Arduino



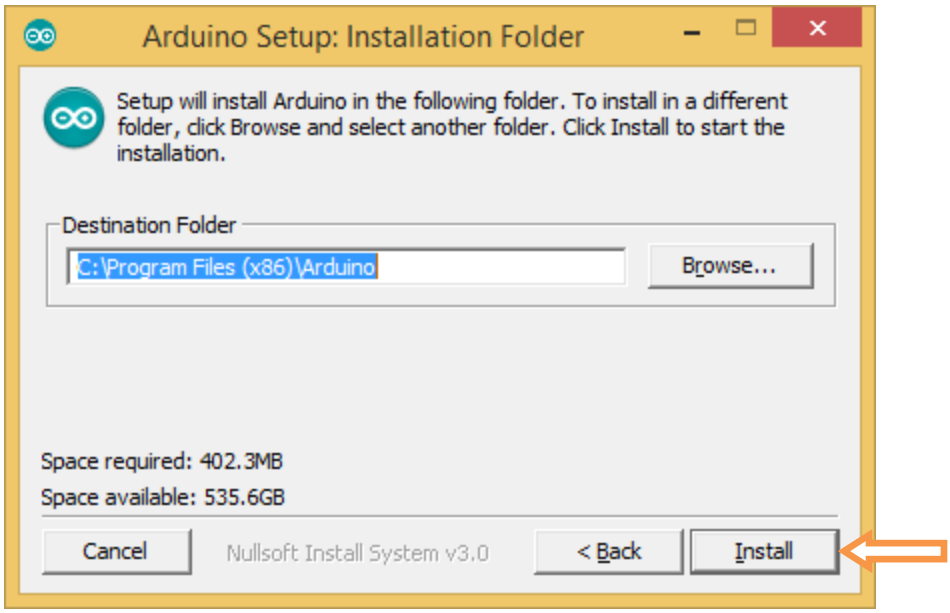
Εικόνα 6: Παράθυρο για την αποθήκευση του προς λήψη αρχείου εγκατάστασης του περιβάλλοντος ανάπτυξης του Arduino



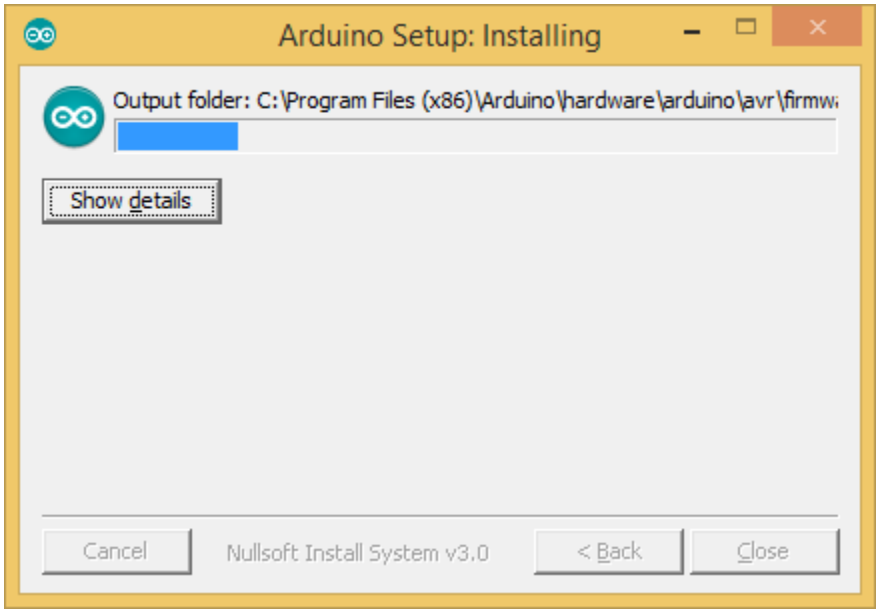
Εικόνα 7: Δήλωση αποδοχής άδειας λογισμικού για τη λήψη του αρχείου εγκατάστασης του περιβάλλοντος ανάπτυξης του Arduino



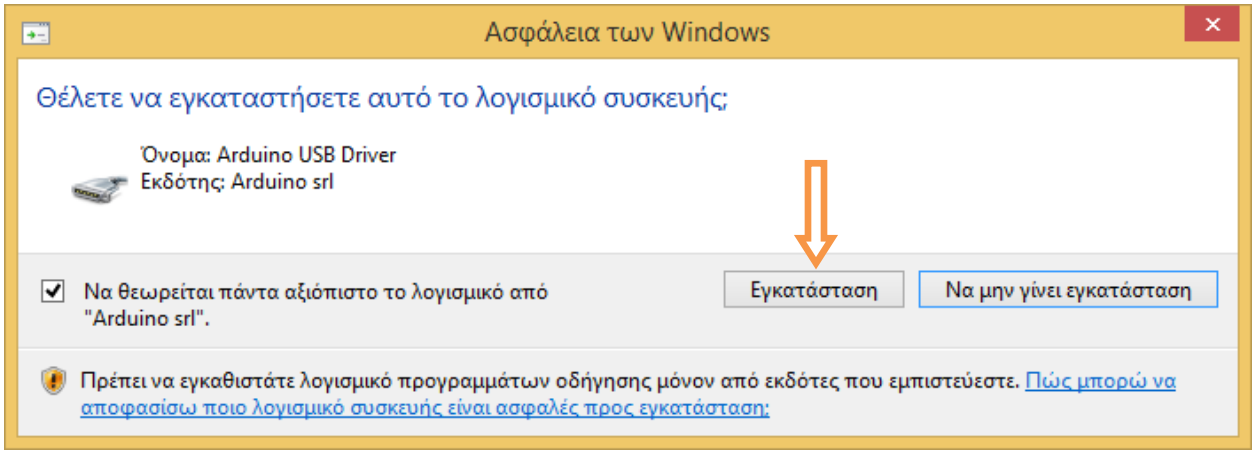
Εικόνα 8: Παράθυρο επιλογής των προς εγκατάσταση στοιχείων του περιβάλλοντος ανάπτυξης του Arduino



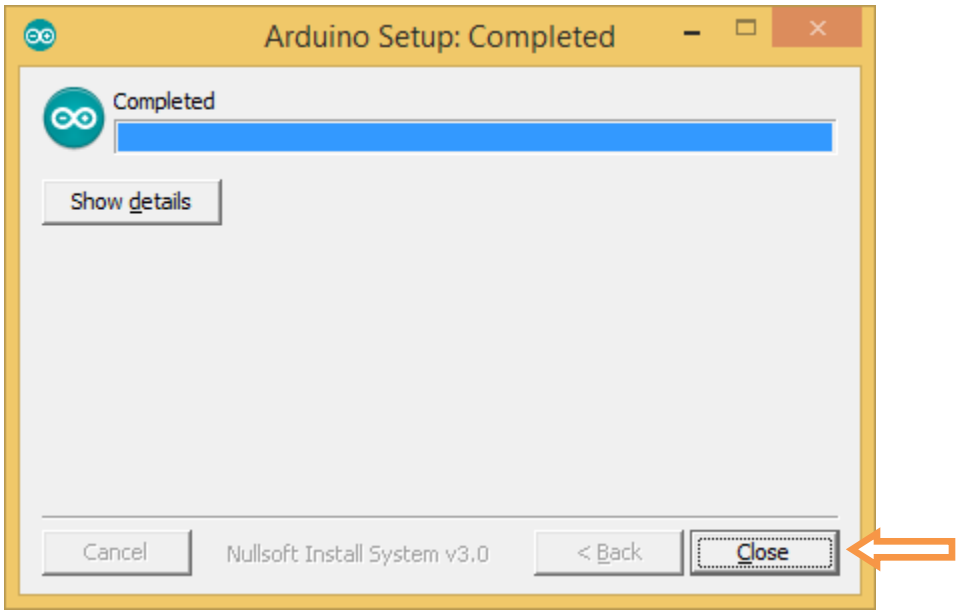
Εικόνα 9: Παράθυρο επιλογής της θέσης εγκατάστασης των στοιχείων του περιβάλλοντος ανάπτυξης του Arduino



Εικόνα 10: Παράθυρο εξέλιξης της εγκατάστασης των στοιχείων του περιβάλλοντος ανάπτυξης του Arduino



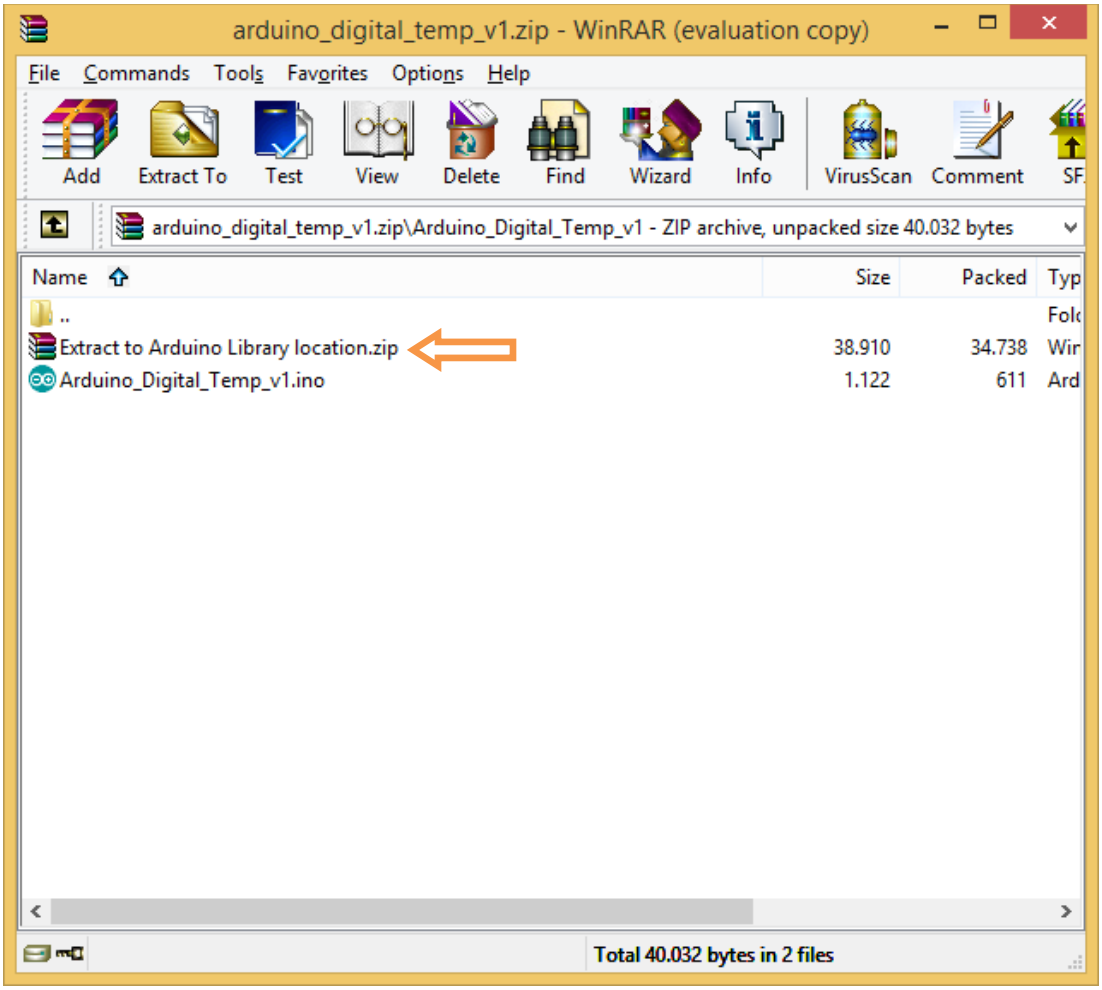
Εικόνα 11: Παράθυρο επιλογής του προγράμματος οδήγησης για τη σύνδεση του Arduino με τη θύρα του USB



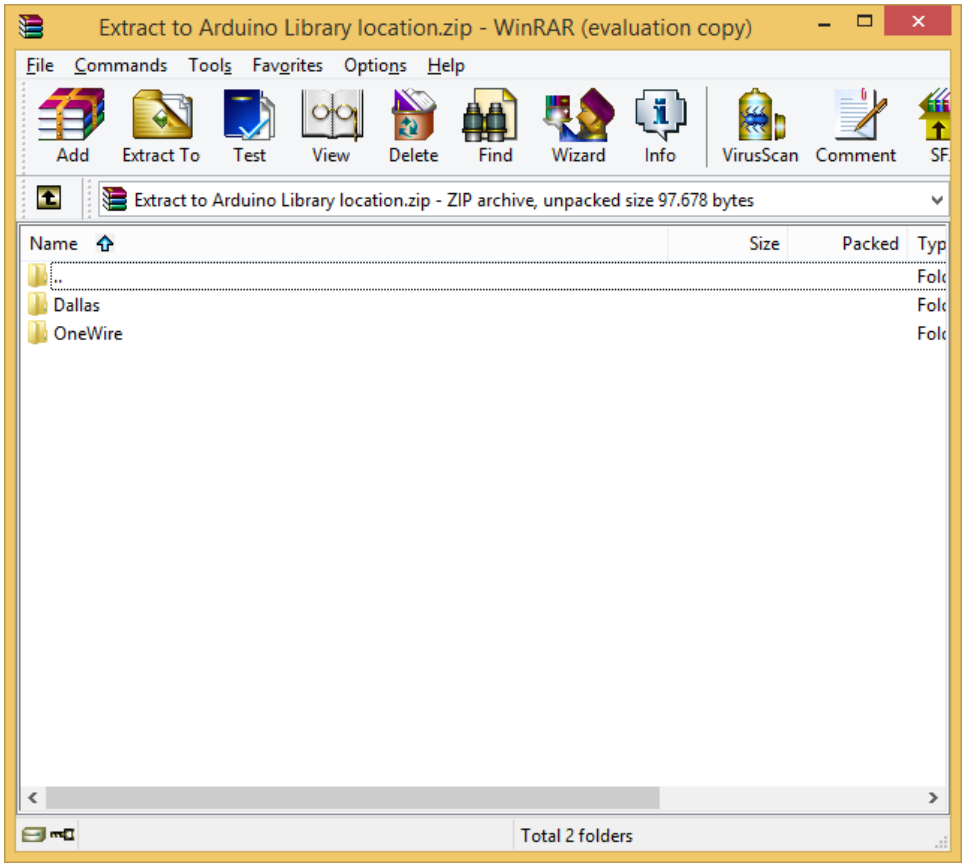
Εικόνα 12: Παράθυρο ολοκλήρωσης της εγκατάστασης των στοιχείων του περιβάλλοντος ανάπτυξης του Arduino

4.2 Εγκατάσταση των βιβλιοθηκών

Η «φόρτωση» του κώδικα στην πλατφόρμα και το «τρέξιμο» του προγράμματος προϋποθέτει τη χρήση ορισμένων βιβλιοθηκών ανάλογα με το project που υλοποιείται κάθε φορά. Στη συγκεκριμένη εφαρμογή, οι βιβλιοθήκες με τον απαραίτητο κώδικα είναι διαθέσιμες στην ηλεκτρονική διεύθυνση http://www.dwrean.net/2015/01/27-arduino.html, αξιοποιώντας ουσιαστικά το πλεονέκτημα του ανοικτού λογισμικού που χαρακτηρίζει τη χρήση της πλατφόρμας σε ευρύτερο επίπεδο. Η λήψη των βιβλιοθηκών και του κώδικα απεικονίζεται στην εικόνα που ακολουθεί.

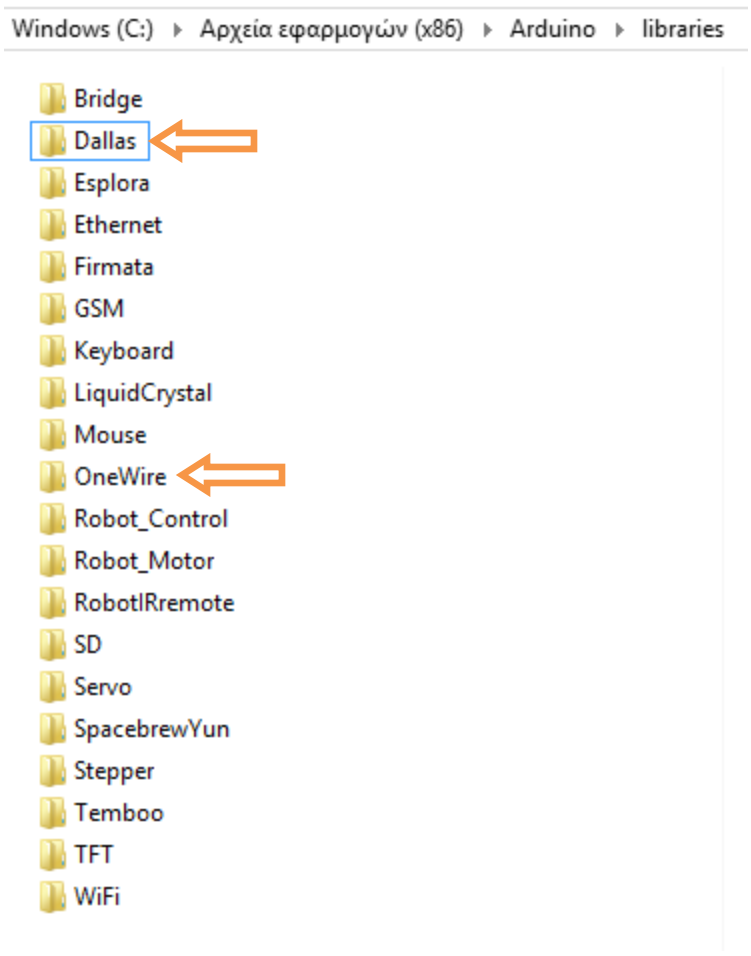


Εικόνα 13: Λήψη των βιβλιοθηκών και του κώδικα του προγράμματος για την υλοποίηση της εφαρμογής



Εικόνα 14: Οι απαιτούμενες για την υλοποίηση της εφαρμογής βιβλιοθήκες

Θα πρέπει όμως αυτές οι βιβλιοθήκες να αντιγραφούν στο φάκελο Libraries του αρχείου εγκατάστασης της πλατφόρμας, όπως φαίνεται στην παρακάτω εικόνα με το βέλος να επισημαίνει τα στοιχεία που μετακινήθηκαν.



Εικόνα 15: Μετακίνηση βιβλιοθηκών στο φάκελο Libraries του αρχείου εγκατάστασης της πλατφόρμας

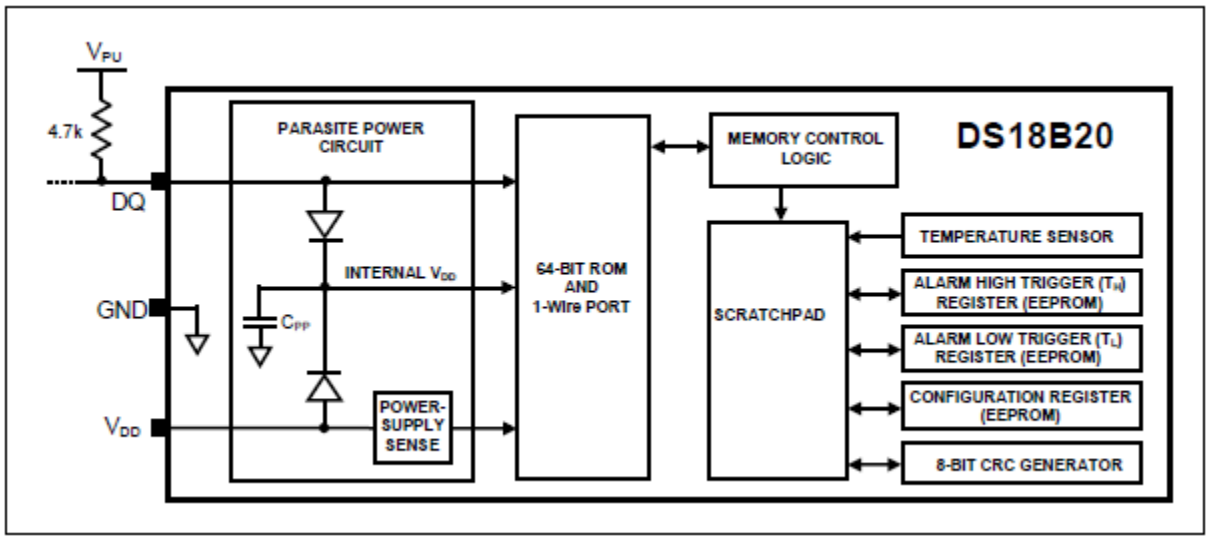
4.3 Περιγραφή και υλοποίηση της εφαρμογής

4.3.1 Στόχος και στοιχεία της εφαρμογής

Η συγκεκριμένη εφαρμογή έχει ως στόχο να αναδείξει τη χρησιμότητα της πλατφόρμας Arduino, όσον αφορά στο πεδίο καταγραφής - ελέγχου της θερμοκρασίας. Μέσω της διάταξης που υλοποιείται και παρουσιάζεται στην παρούσα εργασία είναι δυνατή η καταγραφή της θερμοκρασίας ενός χώρου σε πραγματικό χρόνο. Ως αισθητήριο χρησιμοποιείται ο Temperature Sensor DS18B20, με μερικά από τα ενδεικτικά του χαρακτηριστικά να έχουν ως εξής:

- μέτρηση θερμοκρασίας σε ◦C με ακρίβεια 9-12 bits
- εύρος μέτρησης -55◦C έως 125◦C (+/-0.5C) (–67°F to +257°F)
- δυνατότητα επικοινωνίας με τη χρήση ενός μόνο pin
- μη απαίτηση για χρήση περαιτέρω εξωτερικών εξαρτημάτων
- τάση τροφοδοσίας 3 - 5.5V
- μετατροπή της θερμοκρασίας σε λέξη των 12-bits σε μέγιστο χρόνο 750ms

Το δομικό διάγραμμα του χρησιμοποιούμενου αισθητήρα παρουσιάζεται στο διάγραμμα που ακολουθεί



Διάγραμμα 3: Δομικό διάγραμμα του αισθητήρα θερμοκρασίας DS18B20 (Maxim Integrated, 2008)

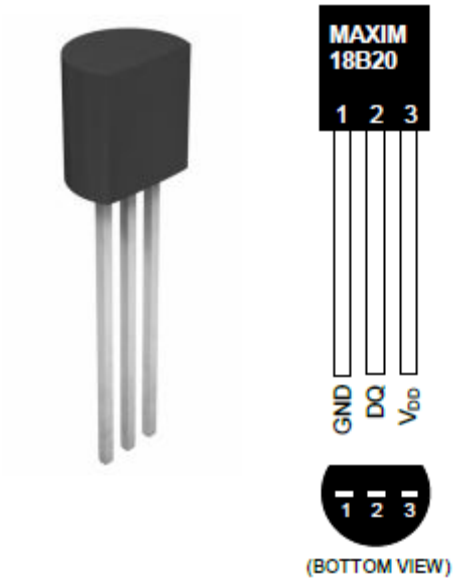
Μέσω του αισθητηρίου γίνεται η καταγραφή της θερμοκρασίας, με την τιμή να απεικονίζεται στη σειριακή οθόνη του Arduino (με την εμφάνισή της να λαμβάνει χώρα ανά 10sec, όπως αυτή απεικονίζεται στην οθόνη του υπολογιστή με τον οποίο συνδέεται η πλατφόρμα.
Από την άλλη πλευρά, μέσω της χρήσης της πλατφόρμας Arduino, αφενός αξιοποιείται μέσω μιας εύκολης διαδικασίας ο προαναφερόμενος αισθητήρας και αφετέρου μέσω της σειριακής εκτύπωσης, μπορεί η μέτρηση να γνωστοποιηθεί προς περαιτέρω έλεγχό της.
Είναι σημαντικό να γίνει αναφορά και στην επεκτασιμότητα της εφαρμογής, με τη δυνατότητα αποθήκευσης ή μεταφοράς του αρχείου της θερμοκρασίας στην επιθυμητή κάθε φορά θέση. Ενδεικτικά μπορεί για αυτό το σκοπό να χρησιμοποιηθεί το Arduino Ethernet Shield, μέσω του οποίου μπορεί να γίνει δυνατή η διαδικτυακή μεταφορά των δεδομένων (ένα σημείο που χρήζει περαιτέρω διερεύνησης).

Εικόνα 16: Arduino Ethernet Shield για την επέκταση της εφαρμογής (https://www.arduino.cc/en/Main/Products)

Η συγκεκριμένη εφαρμογή και παραλλαγές της μπορούν να αξιοποιηθούν στο βιομηχανικό τομέα, σε κάθε απαίτηση μέτρησης θερμοκρασίας συγκεκριμένου χώρου (π.χ. ψυγεία, κλιματιζόμενοι χώροι, smart homes κτλ.).

4.3.2 Συνδέσεις

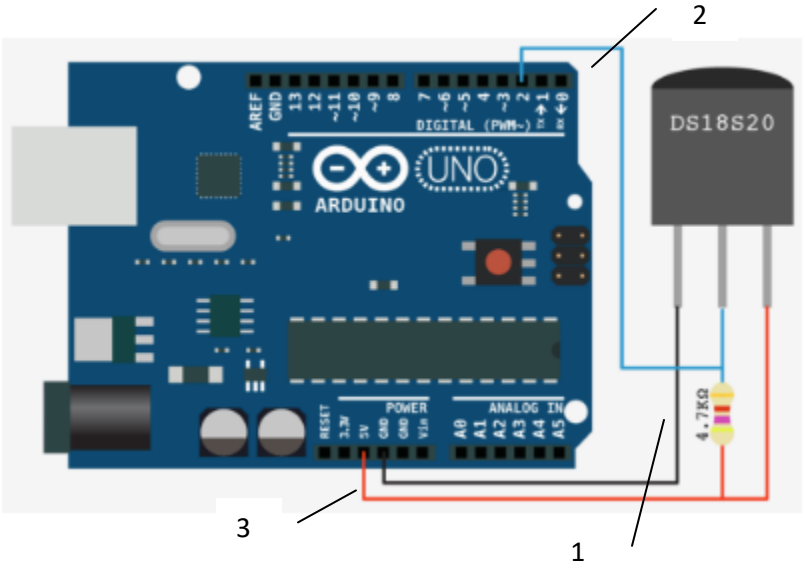
Ακολουθεί η περιγραφή της συνδεσμολογία της διάταξης για την υλοποίηση της εφαρμογής . Στην εικόνα που ακολουθεί παρουσιάζεται η απαιτούμενη σύνδεση για τη λειτουργία του αισθητήρα θερμοκρασίας DS18B20.



Εικόνα 17: Μορφή και συνδεσμολογία του αισθητήρα θερμοκρασίας DS18B20 (Maxim Integrated, 2008)

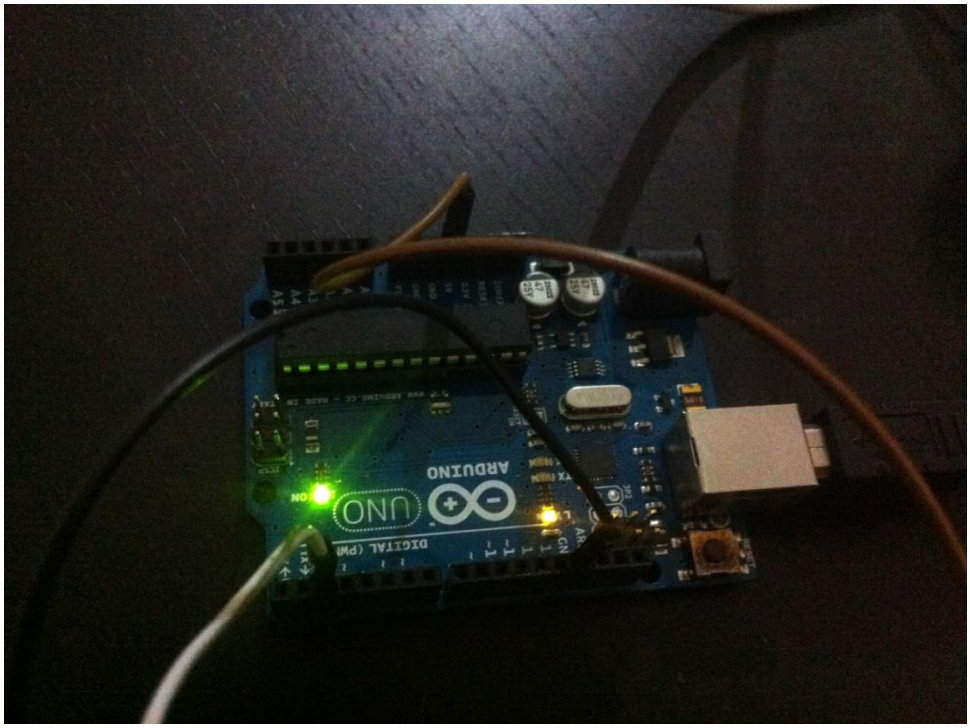
Τα «βήματα» για την υλοποίηση του κυκλώματος απεικονίζονται στην παρακάτω εικόνα και έχουν αναλυτικότερα ως εξής:

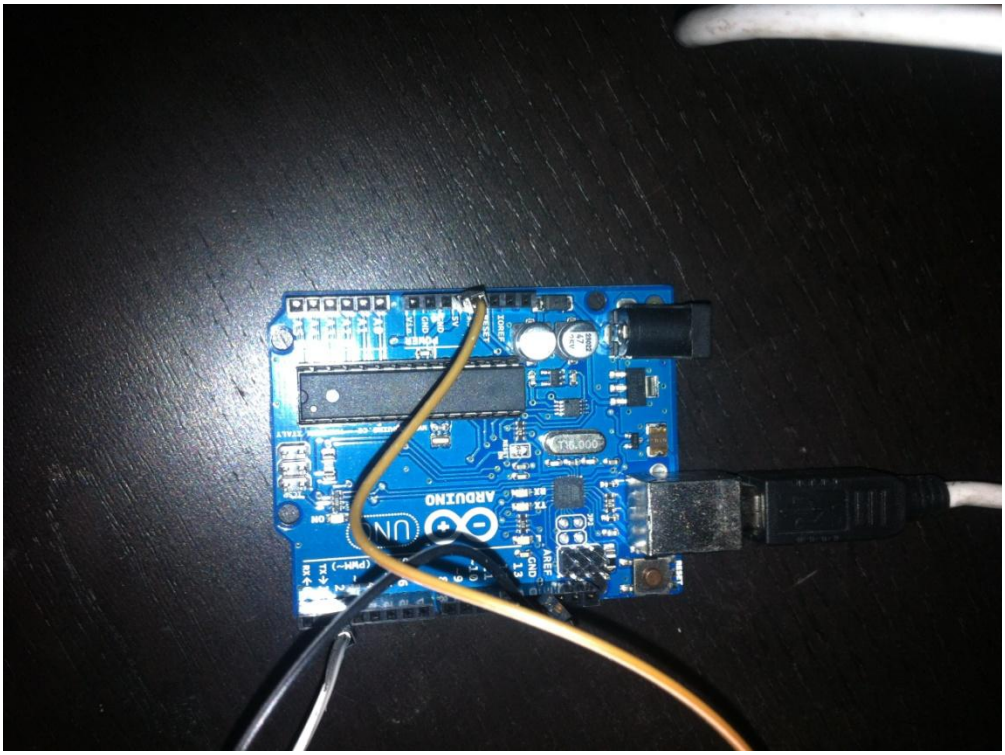
1. Γίνεται ένωση του GND του Arduino με το Vdd pin του DS18B20
2. Γίνεται ένωση του Pin 2 του DS18B20 με το αντίστοιχο Pin2 του Arduino
3. Μια αντίσταση των 4.7 KOhm συνδέεται στο ένα άκρο της με το pin 5V του Arduino και στο άλλο άκρο της με το Pin2 του DS18B20

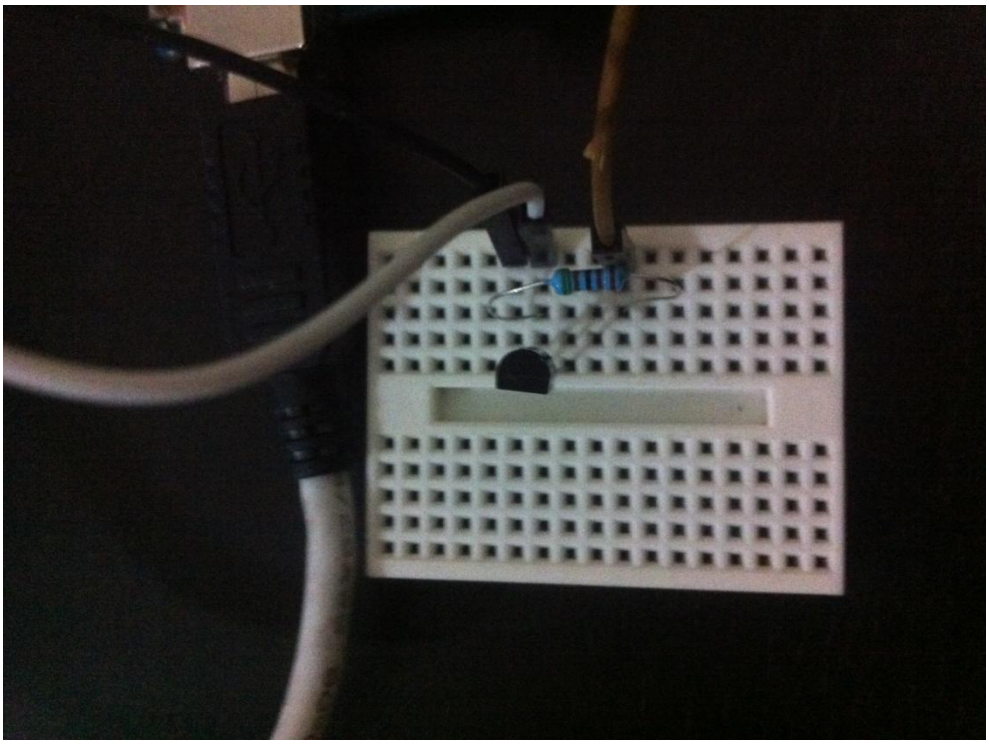


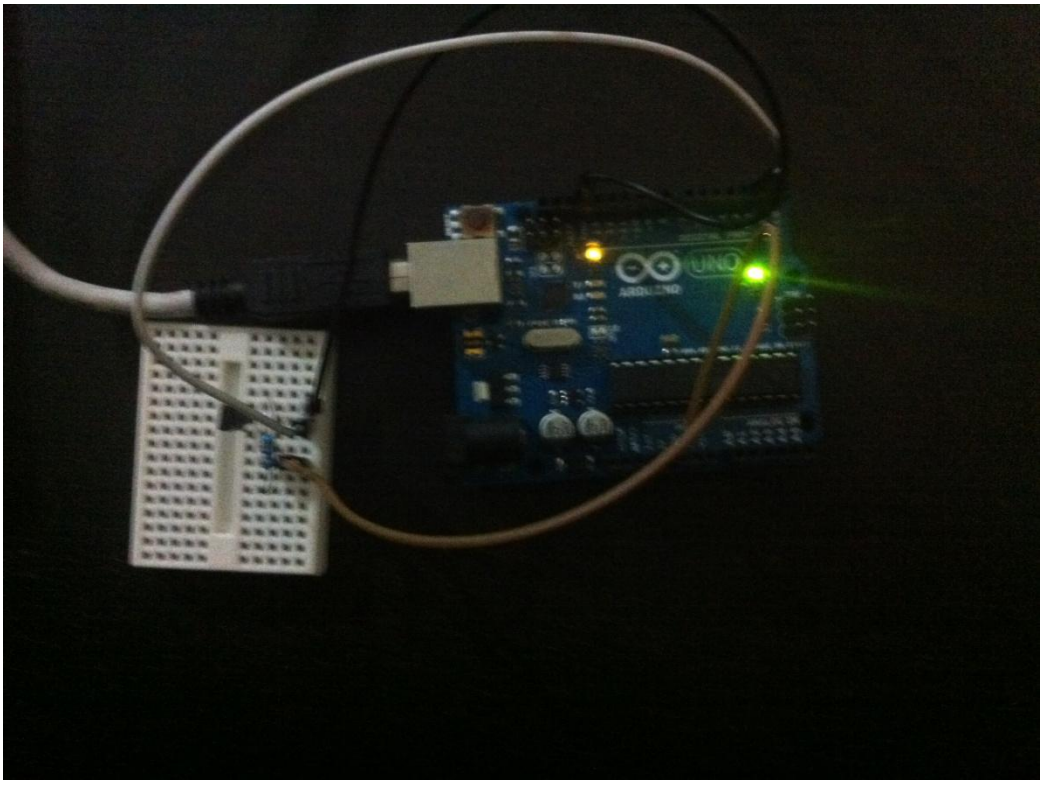
Εικόνα 18: Συνδεσμολογία του υλικού της εφαρμογής (http://learning.grobotronics.com)

Ακολουθούν στη συνέχεια οι εικόνες που παρουσιάζουν τις συνδέσεις του υλικού για την υλοποίηση της εφαρμογής.




ARDUINO
UNO
16.000

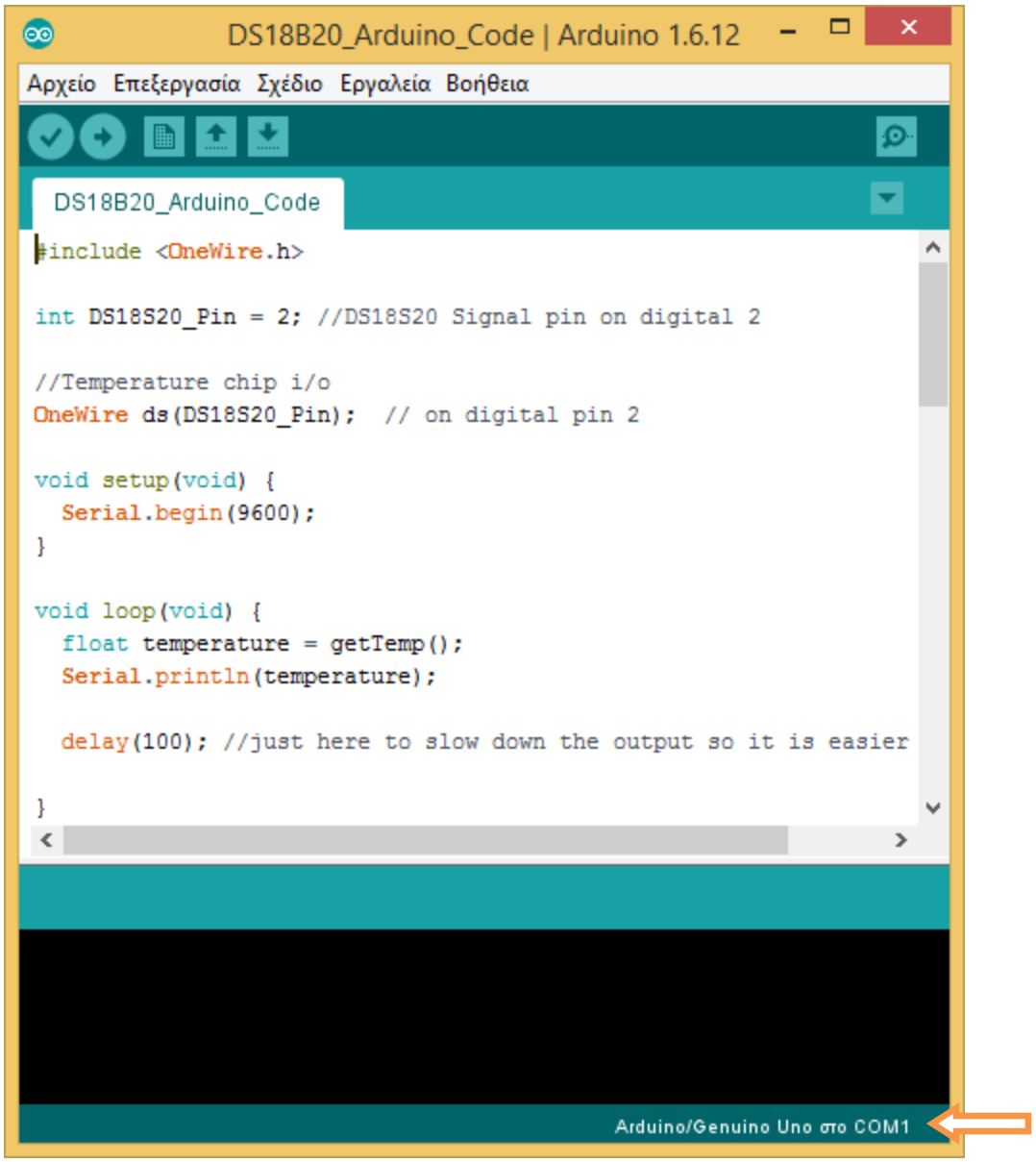




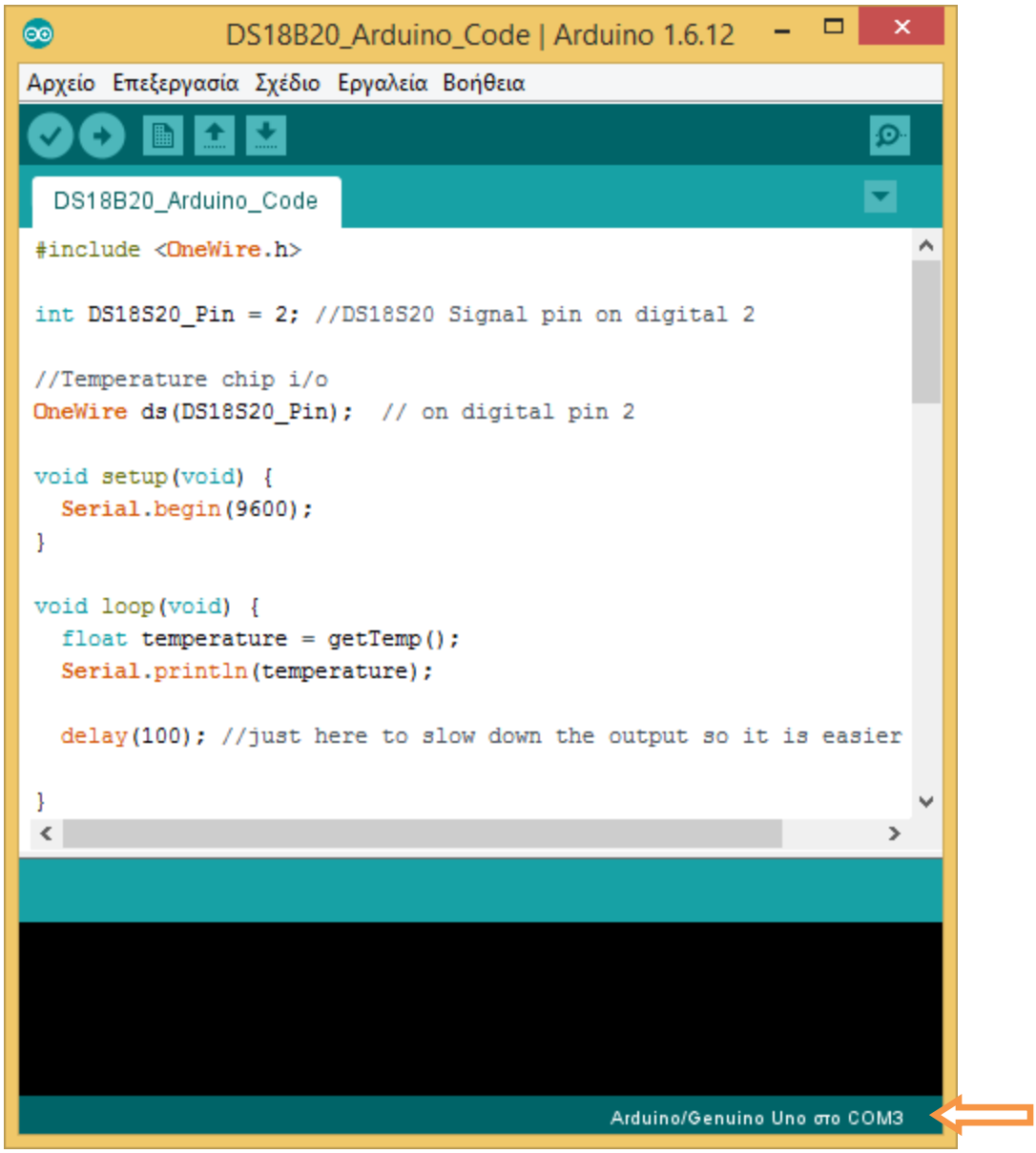
Εικόνα 19: Συνδεσμολογία του υλικού για την υλοποίηση της εφαρμογής

4.3.3 «Φόρτωση» του κώδικα

Αρχικά συνδέουμε την πλατφόρμα στον Η/Υ μέσω της θύρας USB. Για να επιτευχθεί η επικοινωνία θα πρέπει να οριστεί η θύρα σύνδεσης του προγράμματος (COM3) όπως φαίνεται χαρακτηριστικά στις ακόλουθες εικόνες, όπου σε πρώτη φάση φαίνεται η θύρα COM1 και στη συνέχεια μέσω της επιλογής Εργαλεία/ Θύρα / COM3, ως θύρα επικοινωνίας έχει οριστεί η COM3.



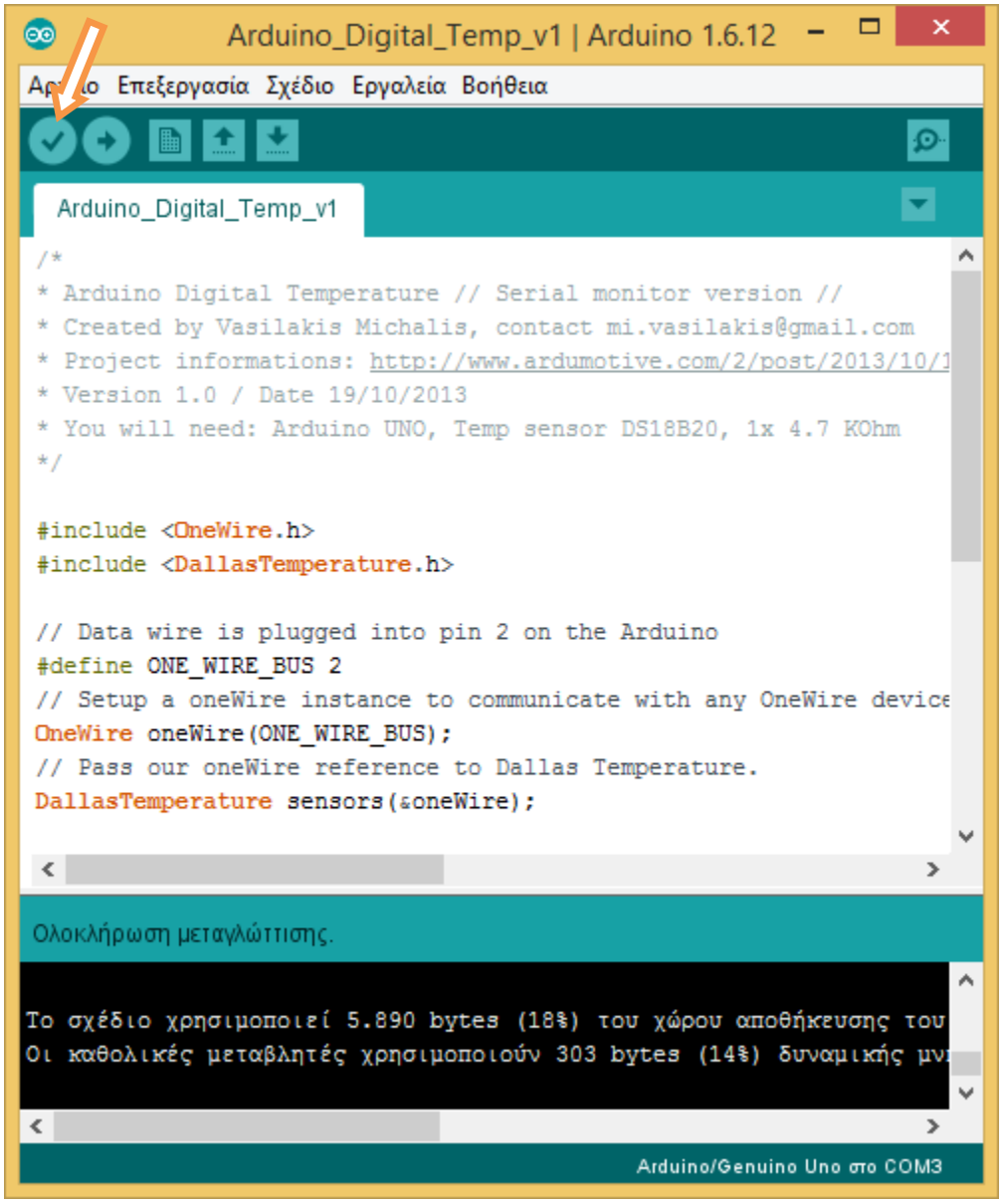
Εικόνα 20: Παράθυρο αρχικής σύνδεσης της πλατφόρμας



Εικόνα 21: Αλλαγή θύρας σύνδεσης – επικοινωνίας με την πλατφόρμα

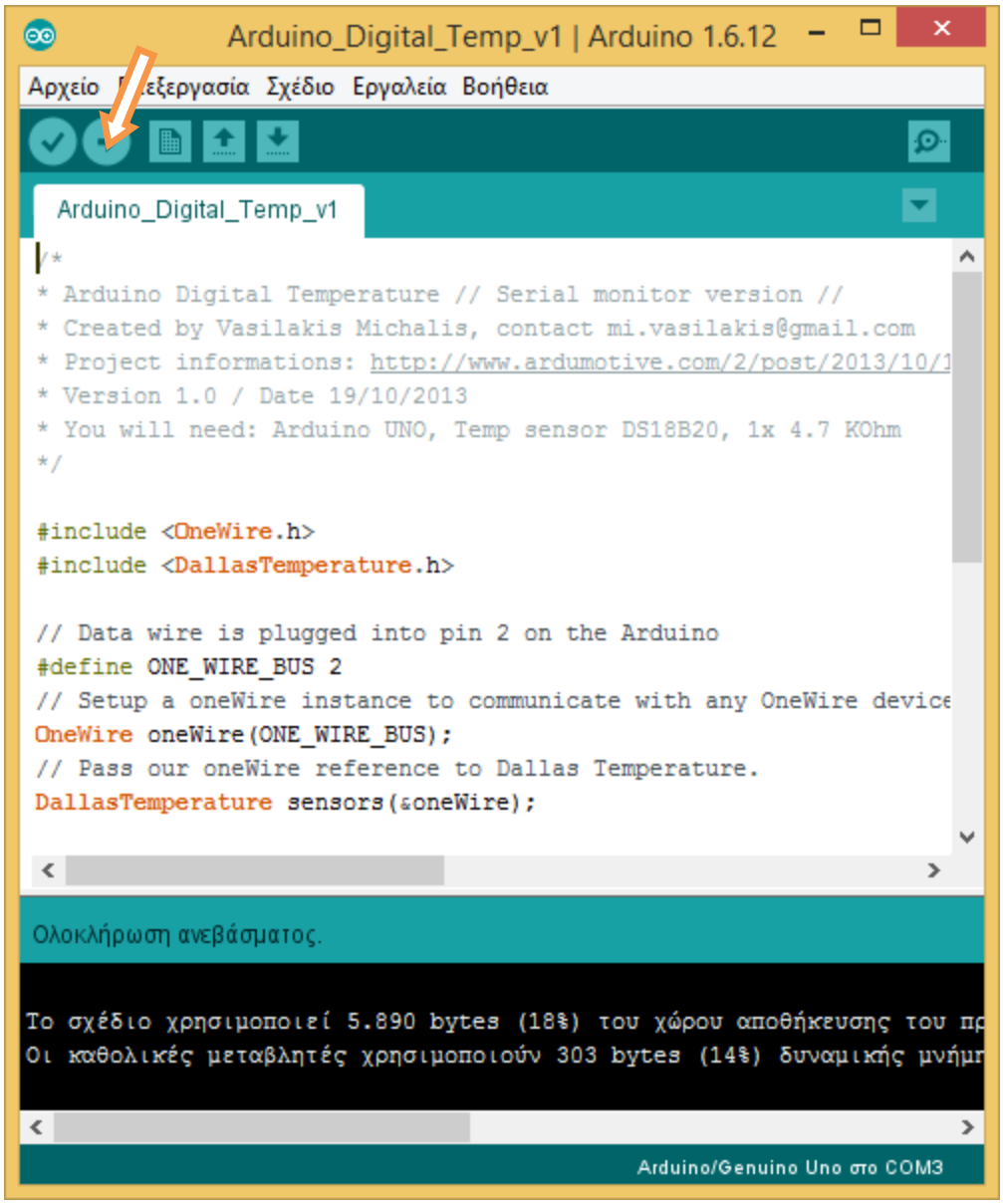
Μέσω της επιλογής Αρχείο/Άνοιγμα επιλέγουμε και ανοίγουμε το αρχείο Arduino_Digital_Temp_v1.ino το οποίο περιέχει τον κώδικα για την υλοποίηση της εφαρμογής.

Για να «φορτωθεί» ο κώδικας στην πλατφόρμα αρχικά τον ελέγχουμε για τυχόν λάθη μέσω της δυνατότητας μεταγλώττισης της πλατφόρμας (Compiling). Ο έλεγχος γίνεται με την επιλογή του button Επικύρωση (validation) όπως φαίνεται στην ακόλουθη εικόνα με το κίτρινο βέλος.



Εικόνα 22: Παράθυρο μεταγλώττισης του προγράμματος

Για να «φορτωθεί» στη συνέχεια ο κώδικας που έχει ανοιχθεί επιλέγουμε το button «Ανέβασμα» (Uploading)



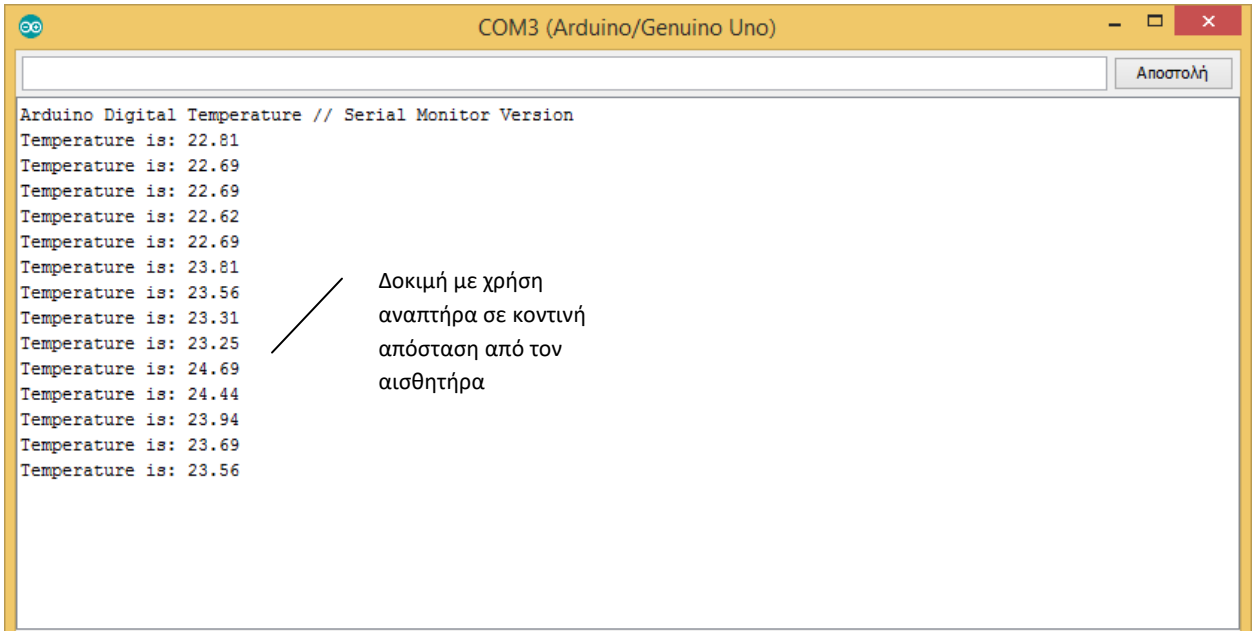
Εικόνα 23: Παράθυρο «ανεβάσματος» του κώδικα στην πλατφόρμα

Ο κώδικας που χρησιμοποιήθηκε είναι διαθέσιμος στην ηλεκτρονική διεύθυνση http://www.dwrean.net/2015/01/27-arduino.html (Βασιλάκης, 2015) και παρουσιάζεται ακολούθως

```
/*
* Arduino Digital Temperature // Serial monitor version //
* Created by Vasilakis Michalis, contact mi.vasilakis@gmail.com
* Project informations: http://www.ardumotive.com/2/post/2013/10/1.html
* Version 1.0 / Date 19/10/2013
* You will need: Arduino UNO, Temp sensor DS18B20, 1x 4.7 KOhm
*/

#include <OneWire.h>
#include <DallasTemperature.h>

// Data wire is plugged into pin 2 on the Arduino
#define ONE_WIRE_BUS 2
// Setup a oneWire instance to communicate with any OneWire devices (not just Maxim/Dallas temperature ICs)
OneWire oneWire(ONE_WIRE_BUS);
// Pass our oneWire reference to Dallas Temperature.
DallasTemperature sensors(&oneWire);

void setup(void)
{
 Serial.begin(9600);
 Serial.println("Arduino Digital Temperature // Serial Monitor Version");
 sensors.begin();
}
```

```
void loop(void)
{
  // Send the command to get temperatures
  sensors.requestTemperatures();
  Serial.print("Temperature is: ");
  Serial.println(sensors.getTempCByIndex(0)); // Why "byIndex"? You can have more than one IC on the same bus. 0 refers to the first IC on the wire
  //Update value every 10 sec.
  delay(10000);
}
```

Στη συνέχεια και προκειμένου να απεικονιστεί η καταγραφή της θερμοκρασίας επιλέγεται η διαδρομή Εργαλεία/Παρακολούθηση Σειριακής, ανοίγοντας το ακόλουθο παράθυρο. Οι μετρήσεις της θερμοκρασίας εμφανίζονται η μία μετά την άλλη με καθυστέρηση 10 δευτερολέπτων. Η απεικονιζόμενη θερμοκρασία είναι σε θερμοκρασία περιβάλλοντος, με ενδιαφέρον σημείο εκείνο κατά την οποία παρακολουθείται μικρή άνοδό της, η οποία οφείλεται σε πειραματισμό για τη λειτουργία του αισθητήρα (χρήση αναπτήρα σε κοντινή απόσταση).



Εικόνα 24: Σειριακή οθόνη απεικόνισης μετρήσεων

ΚΕΦΑΛΑΙΟ 5: ΣΥΜΠΕΡΑΣΜΑΤΑ - ΠΡΟΤΑΣΕΙΣ

Παρουσιάζεται αυξημένη προτίμηση σε εφαρμογές αυτού του είδους , η οποία έχει άμεση σχέση με τα πλεονεκτήματα της πλατφόρμας, όπως η αυξημένη ευκολία, το χαμηλό κόστος, το ευρύ φάσμα δυνατοτήτων. η δυνατότητα εύκολης επέμβασης στη λειτουργία για την πραγματοποίηση διορθωτικών κινήσεων και η βελτιστοποίηση της απόδοσης του εκάστοτε συστήματος (ευκολία η οποία προκύπτει τόσο από την απλότητα στον προγραμματισμό όσο και από το γεγονός ότι πρόκειται για πλατφόρμα ανοιχτού – ελεύθερου κώδικα.

Θα πρέπει επίσης να σημειωθεί η ραγδαία ανάπτυξη της πλατφόρμας, με ενδεικτικό το γεγονός των πολλών διαφορετικών διατιθέμενων προϊόντων ανάλογα με την εφαρμογή για τη οποία προορίζονται κάθε φορά αλλά και την αυξημένη συμμετοχή στην κοινότητα του ελεύθερου κώδικα.

Κατά την υλοποίηση της εφαρμογής παρατηρήθηκε πως πρόκειται για μια βατή διαδικασία, αρκεί να γίνει με προσεκτικά και αναλυτικά βήματα.

Ως σημείο που χρήζει περαιτέρω διερεύνησης και με βάση το στοιχείο της επεκτασιμότητας που χαρακτηρίζει την πλατφόρμα, προτείνεται η αξιοποίηση της εφαρμογής, με αποστολή και χρήση των δεδομένων μέσω διαδικτύου, έτσι ώστε να είναι δυνατή η απομακρυσμένη καταγραφή και έλεγχος της θερμοκρασίας του χώρου στον οποίο γίνεται η μέτρηση.

Βιβλιογραφία

[1] Παρουσίαση «Μικροελεγκτές και Ενσωματωμένα Συστήματα (Microcontrollers and Embedded Systems)» διαθέσιμη στην ηλεκτρονική διεύθυνση http://slideplayer.gr/slide/2953946/ [Ανακτήθηκε 22/03/16]

[2] Wikipedia, Embedded system, available at http://en.wikipedia.org/wiki/Embedded_systems [Retrieved on 22/03/16]

[3] Electronic Voting Machine Information Sheet Accupoll AVS 1000

[4] Διομήδης Π., Μέγα, Α. (2015). «Εφαρμογή Arduino μετρήσεων και καταγραφής μετεωρολογικών μεγεθών, ΑΤΕΙ Θεσσαλίας, Σχολή Τεχνολογικών Εφαρμογών, Τμήμα Μηχανικών Πληροφορικής ΤΕ, Λάρισα

[5] Ελευθεριάδης, Μ. (2012). «Εφαρμογές ασύρματης τηλεμετρίας στην αναπτυξιακή πλατφόρμα Arduino, Σχολή Τεχνολογικών Εφαρμογών Τμήμα Εφαρμοσμένης Πληροφορικής & Πολυμέσων, Τεχνολογικό Εκπαιδευτικό Ίδρυμα Κρήτης

[6] Σολωμονίδης, Ι.Χ. (2011). «ΠΑΝΖΕΥΚΤΗΣ»: Ολοκληρωμένη Πλατφόρμα Υλικού Ανάπτυξης Εφαρμογών Διάχυτης Ευφυίας, Ασύρματης Εποπτείας και Ευφυούς Ελέγχου Οικίας (domotics) με Προηγμένες Υπολογιστικές Δυνατότητες, Εθνικό Μετσόβιο Πολυτεχνείο, Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, Τομέας Επικοινωνιών, Ηλεκτρονικής και Συστημάτων Πληροφορικής

[7] Βουρδουρίδης Θ. (2012), Κατασκευή ηλεκτροκαρδιογραφήματος με τη βοήθεια μικροελεγκτή ADUC 7026, Τμήμα Ηλεκτρολόγων Μηχανικών και Τεχνολογίας Υπολογιστών της Πολυτεχνικής Σχολής Πανεπιστημίου Πατρών

[8] https://www.arduino.cc/ [Πρόσβαση 28/03/16]

[9] https://www.arduino.cc/en/Main/Products [Πρόσβαση 28/03/16]

[10] Δασυγένης, Μ. (2013). Παρουσίαση «Ενσωματωμένα Συστήματα», Ενότητα 1: Εισαγωγικές έννοιες στα ενσωματωμένα συστήματα. Ορισμός. Χαρακτηριστικά. Εφαρμογές», Εργαστήριο Ψηφιακών Συστημάτων και Αρχιτεκτονικής Υπολογιστών, Τμήμα Μηχανικών Πληροφορικής & Τηλεπικοινωνιών, Πανεπιστήμιο Δυτικής Μακεδονίας

[11] Παπαναστασίου Α. (2009), Διπλωματική εργασία με τίτλο, Κατασκευή Συστήματος Καταγραφής Ψηφιακών Δεδομένων (Data Logger) με χρήση της πλατφόρμας Arduino - Εφαρμογή με Μετεωρολογικά Δεδομένα, Εθνικό Μετσόβιο Πολυτεχνείο, Τομέας Τοπογράφων

[12] Lahart, J. (2009), «Taking an Open-Source Approach to Hardware». *The Wall Street Journal*

[13] Kushner D. (2011), "The Making of Arduino" IEEE Spectrum. http://spectrum.ieee.org/geek-life/hands-on/the-making-of-arduino [Ανακτήθηκε 23/03/16]

[14] https://en.wikipedia.org/wiki/Arduin_of_Ivrea [Πρόσβαση 28/03/16]

[15] https://el.wikipedia.org/wiki/Arduino [Πρόσβαση 28/03/16]

[16] Μαυραειδόπουλος Αθ. (2015). «Ανάπτυξη και χρήση πρωτοκόλλου modbus για την κάρτα arduino, ΤΕΙ Λαμίας Σχολή Τεχνολογικών Εφαρμογών, Τμήμα Πληροφορικής και Τεχνολογίας Υπολογιστών»

[17] Φελλόπουλος, Α και Σπύρου, Μ. (2012). Υλοποίηση ενός τρίτροχου κινούμενου ρομπότ χρησιμοποιώντας πλατφόρμα Arduino, Τεχνολογικό Εκπαιδευτικό Ίδρυμα Σερρών Σχολή Τεχνολογικών Εφαρμογών Τμήμα Πληροφορικής και Επικοινωνιών

[18] Περιοδικό DeltaHacker (01/08/2009), Άρθρο με τίτλο Εισαγωγή στο Arduino - Το απόλυτο geek toy, διαθέσιμο στην ηλεκτρονική διεύθυνση http://deltahacker.gr/arduino-intro/ [Πρόσβαση 28/03/16]

[19] Shilling T. W. (March 2012), Shilling Systems Arduino Modbus Slave V.9.0

[20] Χαρουπιάς Α.Α. (2013), Πτυχιακή εργασία με τίτλο «Επέκταση των δυνατοτήτων του εκπαιδευτικού πακέτου «LEGO-MINDSTORMS», με την χρήση του μικροελεγκτή Arduino, για μηχατρονικές εφαρμογές», ΤΕΙ Κρήτης, Ηράκλειο

[21] Maxim Integrated (2008), DS18B20 Programmable Resolution 1-Wire Digital Thermometer, REV: 042208

[22] Βασιλάκης Μ. (04/01/15), Arduino και αισθητήρας θερμοκρασίας http://www.dwrean.net/2015/01/27-arduino.html [Πρόσβαση 02/04/16]

[23] Measuring temperature with the DS18B20 sensor http://learning.grobotronics.com/2013/07/measuring-temperature-with-the-ds18b20-sensor [Πρόσβαση 30/03/16]

[24] Temperature Sensor DS18B20 http://grobotronics.com/temperature-sensor-ds18b20-el.html [Πρόσβαση 30/03/16]

Παράρτημα Ι: Βασικά στοιχεία προγραμματισμού Arduino

Όρισμα	*Είδος*	*Τύπος*	*Παράμετροι*	*Περιγραφή*
LOW	Σταθερά	int	-	Έχει την τιμή 0 και είναι αντίστοιχη του λογικού false.
HIGH	Σταθερά	int	-	Έχει την τιμή 1 και είναι αντίστοιχη του λογικού true.
INPUT	Σταθερά	int	-	Έχει την τιμή 0 και είναι αντίστοιχη του λογικού false.
OUTPUT	Σταθερά	int	-	Έχει την τιμή 1 και είναι αντίστοιχη του λογικού true.
pinMode	Εντολή	-	(*pin*, *mode*)	Καθορίζει αν το συγκεκριμένο ψηφιακό *pin* θα είναι pin εισόδου ή pin εξόδου ανάλογα με την τιμή που δίνεται στην παράμετρο *mode* (INPUT ή OUTPUT αντίστοιχα).
digitalWrite	Εντολή	-	(*pin*, *pinstatus*)	Θέτει την κατάσταση *pinstatus* (HIGH ή LOW) στο συγκεκριμένο ψηφιακό *pin*.
digitalRead	Συνάρτηση	int	(*pin*)	Επιστρέφει την κατάσταση του συγκεκριμένου ψηφιακού *pin* (0 για LOW και 1 για HIGH) εφόσον αυτό είναι pin εισόδου.
analogReference	Εντολή	-	(*type*)	Δέχεται τις τιμές DEFAULT, INTERNAL ή EXTERNAL στην παράμετρο *type* για να καθορίσει την τάση αναφοράς (V_{ref}) των αναλογικών εισόδων (5V, 1.1V ή η εξωτερική τάση με την οποία τροφοδοτείται το pin AREF αντίστοιχα)
analogRead	Συνάρτηση	int	(*pin*)	Επιστρέφει έναν ακέραιο από 0 εώς 1023, ανάλογα με την τάση που τροφοδοτείται το συγκεκριμένο *pin* αναλογικής εισόδου στην κλίμακα 0 ως V_{ref}.
analogWrite	Εντολή	-	(*pin*, *value*)	Θέτει το συγκεκριμένο ψηφιακό *pin* σε κατάσταση ψευδοαναλογικής εξόδου (PWM). Η παράμετρος *value* καθορίζει το πλάτος του παλμού σε σχέση με την περίοδο του παραγόμενου σήματος στην κλίμακα από 0 ως 255 (π.χ. με *value* 127, το πλάτος του παλμού είναι ίσο με μισή περίοδο).
millis	Συνάρτηση	unsigned long	()	Μετρητής που επιστρέφει το χρονικό διάστημα σε ms από την στιγμή που άρχισε η εκτέλεση του προγράμματος. Λάβετε υπόψη ότι λόγω του τύπου μεταβλητής (unsigned long δηλ. 32bit) θα γίνει overflow σε 2^32ms δηλαδή περίπου σε 50 μέρες, οπότε ο μετρητής θα ξεκινήσει πάλι από το μηδέν.
delay	Εντολή	-	(*time*)	Σταματά προσωρινά την ροή του

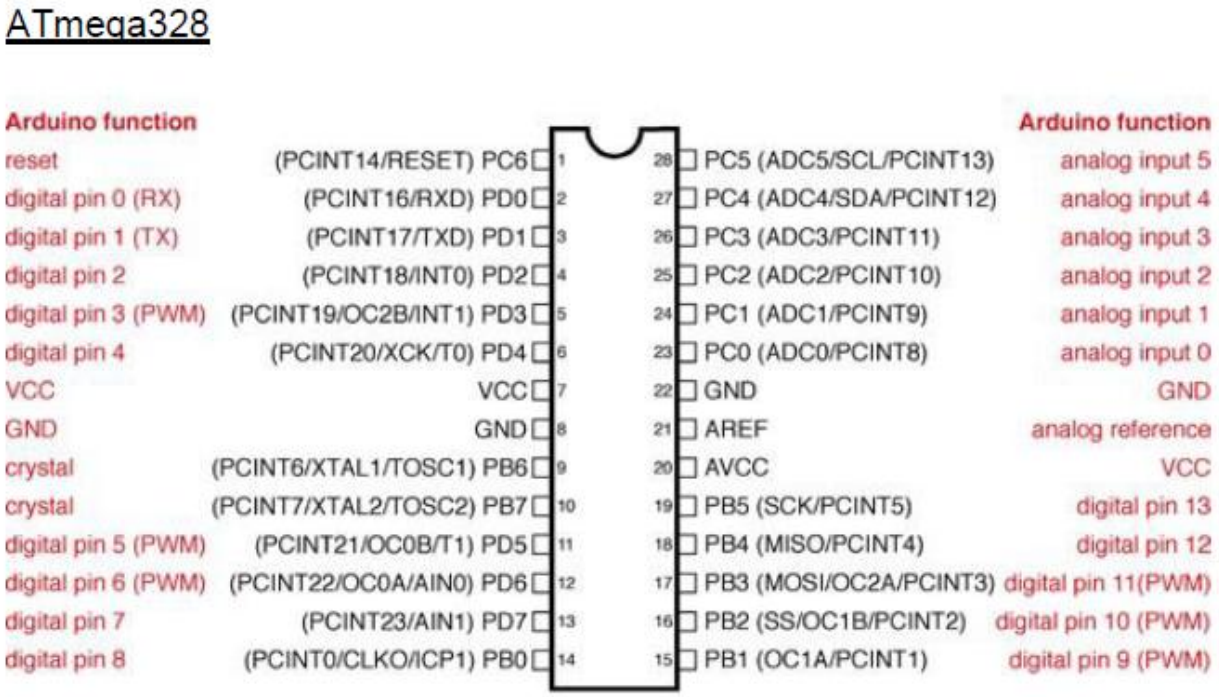
				προγράμματος για *time* ms. Η παράμετρος *time* είναι unsigned long (από 0 ως 2^32). Σημειώστε ότι παρά την προσωρινή παύση, συναρτήσεις των οποίων η εκτέλεση ενεργοποιείται από interrupt θα εκτελεστούν κανονικά κατά την διάρκεια μιας delay.
attachInterrupt	Εντολή	-	(*interrupt*, *function*, *triggermode*)	Θέτει σε λειτουργία το συγκεκριμένο *interrupt*, ώστε να ενεργοποιεί την συνάρτηση *function*, κάθε φορά που ικανοποιείται η συνθήκη που ορίζεται από την παράμετρο *triggermode*: • LOW (ενεργοποίηση όταν η κατάσταση του pin που αντιστοιχεί στο συγκεκριμένο interrupt γίνει LOW) • RISING (όταν από LOW γίνει HIGH) • FALLING (όταν από HIGH γίνει LOW) • CHANGE (όταν αλλάξει κατάσταση γενικά)
detachInterrupt	Εντολή	-	(*interrupt*)	Απενεργοποιεί το συγκεκριμένο *interrupt*.
noInterrupts	Εντολή	-	()	Σταματά προσωρινά την λειτουργία όλων των interrupt
interrupts	Εντολή	-	()	Επαναφέρει την λειτουργία των interrupt που διακόπηκε προσωρινά από μια εντολή noInterrupts.
Serial.begin	Μέθοδος κλάσης	-	(*datarate*)	Θέτει τον ρυθμό μεταφοράς δεδομένων του σειριακού interface (σε baud)
Serial.println	Μέθοδος κλάσης	-	(*data*)	Διοχετεύει τα δεδομένα *data* για αποστολή μέσω του σειριακού interface. Η παράμετρος *data* μπορεί να είναι είτε αριθμός είτε αλφαριθμητικό.

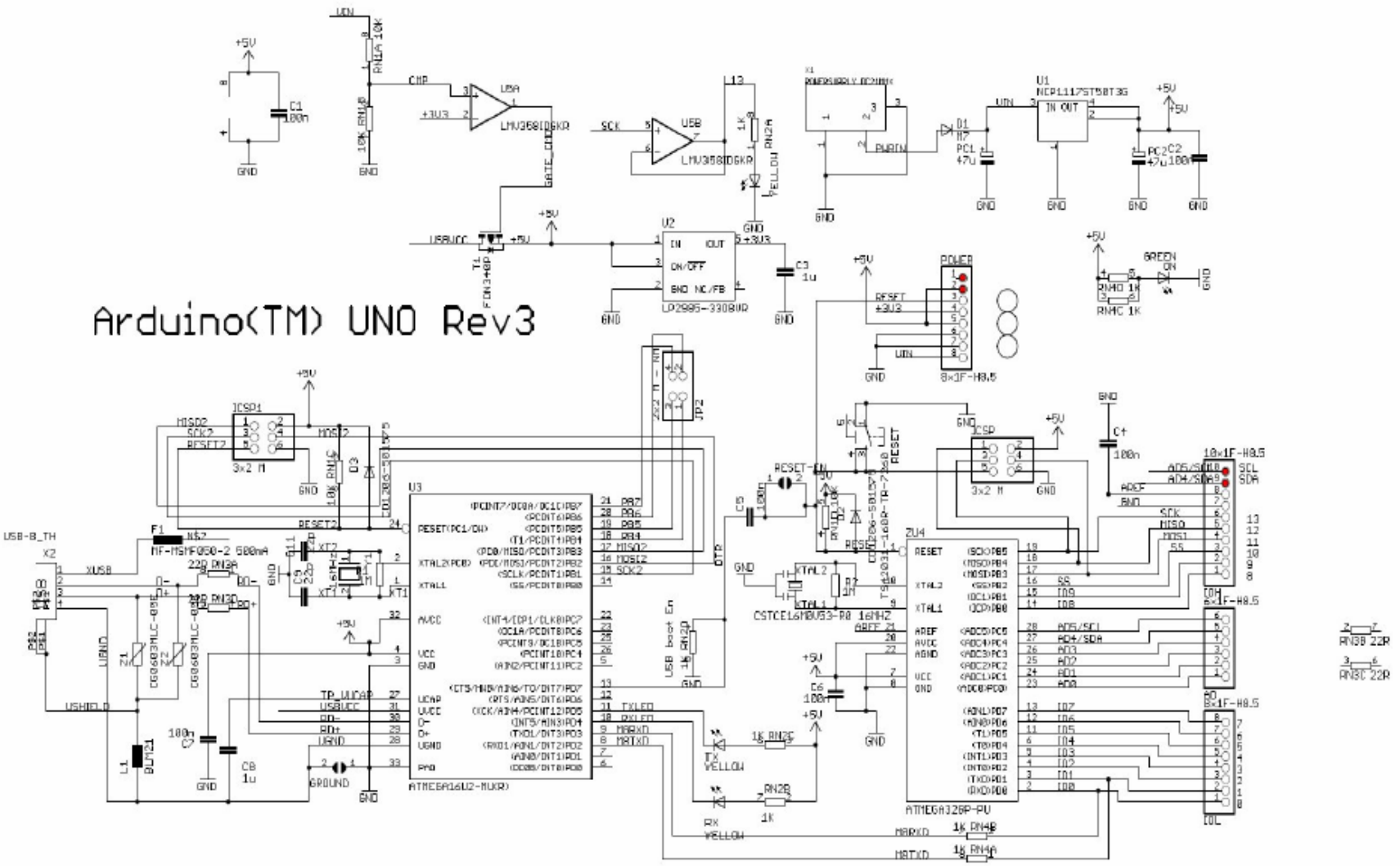
Πίνακας 3: Βασικά στοιχεία προγραμματισμού Arduino (DeltaHacker 2009)

Παράρτημα ΙΙ: Πίνακας φυσικών χαρακτηριστικών της πλατφόρμας Arduino Uno

Μικροεπεξεργαστής	ATmega328
Τάση Λειτουργίας	5V
Συνιστώμενη Τάση Εισαγωγής	7-12V
Τάση Εισαγωγή ς(όρια)	6-20V
Ψηφιακές I/O Είσοδοι	14
Αναλογικές Είσοδοι	6
Συνεχές Ρεύμα(ανά είσοδο)	40 mA
Συνεχές Ρεύμα Εισόδου 3.3V	50 mA
Μνήμη	32 KB
SRAM	2 KB
EEPROM	1 KB
Ταχύτητα Ρολογιών	16 MHz

Παράρτημα ΙΙΙ: Σχηματικά διαγράμματα μικροελεγκτή και πλατφόρμας Arduino Uno




Arduino(TM) UNO Rev3

Παράρτημα IV: Βιβλιοθήκες που χρησιμοποιήθηκαν στον προγραμματισμό

Βιβλιοθήκη OneWire.h

```
#ifndef OneWire_h
#define OneWire_h

#include <inttypes.h>

#if ARDUINO >= 100
#include "Arduino.h"       // for delayMicroseconds,
digitalPinToBitMask, etc
#else
#include "WProgram.h"      // for delayMicroseconds
#include "pins_arduino.h"  // for digitalPinToBitMask, etc
#endif

// You can exclude certain features from OneWire.  In theory, this
// might save some space.  In practice, the compiler automatically
// removes unused code (technically, the linker, using -fdata-sections
// and -ffunction-sections when compiling, and Wl,--gc-sections
// when linking), so most of these will not result in any code size
// reduction.  Well, unless you try to use the missing features
// and redesign your program to not need them!  ONEWIRE_CRC8_TABLE
// is the exception, because it selects a fast but large algorithm
// or a small but slow algorithm.

// you can exclude onewire_search by defining that to 0
#ifndef ONEWIRE_SEARCH
#define ONEWIRE_SEARCH 1
#endif

// You can exclude CRC checks altogether by defining this to 0
#ifndef ONEWIRE_CRC
#define ONEWIRE_CRC 1
#endif

// Select the table-lookup method of computing the 8-bit CRC
// by setting this to 1.  The lookup table enlarges code size by
// about 250 bytes.  It does NOT consume RAM (but did in very
// old versions of OneWire).  If you disable this, a slower
// but very compact algorithm is used.
#ifndef ONEWIRE_CRC8_TABLE
#define ONEWIRE_CRC8_TABLE 1
#endif
```

```
// You can allow 16-bit CRC checks by defining this to 1
// (Note that ONEWIRE_CRC must also be 1.)
#ifndef ONEWIRE_CRC16
#define ONEWIRE_CRC16 1
#endif

#define FALSE 0
#define TRUE  1

// Platform specific I/O definitions

#if defined(__AVR__)
#define PIN_TO_BASEREG(pin)
(portInputRegister(digitalPinToPort(pin)))
#define PIN_TO_BITMASK(pin)             (digitalPinToBitMask(pin))
#define IO_REG_TYPE uint8_t
#define IO_REG_ASM asm("r30")
#define DIRECT_READ(base, mask)         (((*(base)) & (mask)) ? 1 : 0)
#define DIRECT_MODE_INPUT(base, mask)   ((*((base)+1)) &= ~(mask))
#define DIRECT_MODE_OUTPUT(base, mask)  ((*((base)+1)) |= (mask))
#define DIRECT_WRITE_LOW(base, mask)    ((*((base)+2)) &= ~(mask))
#define DIRECT_WRITE_HIGH(base, mask)   ((*((base)+2)) |= (mask))

#elif defined(__MK20DX128__)
#define PIN_TO_BASEREG(pin)             (portOutputRegister(pin))
#define PIN_TO_BITMASK(pin)             (1)
#define IO_REG_TYPE uint8_t
#define IO_REG_ASM
#define DIRECT_READ(base, mask)         (*((base)+512))
#define DIRECT_MODE_INPUT(base, mask)   (*((base)+640) = 0)
#define DIRECT_MODE_OUTPUT(base, mask)  (*((base)+640) = 1)
#define DIRECT_WRITE_LOW(base, mask)    (*((base)+256) = 1)
#define DIRECT_WRITE_HIGH(base, mask)   (*((base)+128) = 1)

#elif defined(__SAM3X8E__)
// Arduino 1.5.1 may have a bug in delayMicroseconds() on Arduino Due.
//
http://arduino.cc/forum/index.php/topic,141030.msg1076268.html#msg1076
268
// If you have trouble with OneWire on Arduino Due, please check the
// status of delayMicroseconds() before reporting a bug in OneWire!
#define PIN_TO_BASEREG(pin)             (&(digitalPinToPort(pin)-
>PIO_PER))
#define PIN_TO_BITMASK(pin)             (digitalPinToBitMask(pin))
#define IO_REG_TYPE uint32_t
#define IO_REG_ASM
#define DIRECT_READ(base, mask)         (((*((base)+15)) & (mask)) ? 1
: 0)
#define DIRECT_MODE_INPUT(base, mask)   ((*((base)+5)) = (mask))
#define DIRECT_MODE_OUTPUT(base, mask)  ((*((base)+4)) = (mask))
#define DIRECT_WRITE_LOW(base, mask)    ((*((base)+13)) = (mask))
```

```
#define DIRECT_WRITE_HIGH(base, mask)    ((*((base)+12)) = (mask))
#ifndef PROGMEM
#define PROGMEM
#endif
#ifndef pgm_read_byte
#define pgm_read_byte(addr) (*(const uint8_t *)(addr))
#endif

#elif defined(__PIC32MX__)
#define PIN_TO_BASEREG(pin)
(portModeRegister(digitalPinToPort(pin)))
#define PIN_TO_BITMASK(pin)              (digitalPinToBitMask(pin))
#define IO_REG_TYPE uint32_t
#define IO_REG_ASM
#define DIRECT_READ(base, mask)          (((*(base+4)) & (mask)) ? 1 :
0)  //PORTX + 0x10
#define DIRECT_MODE_INPUT(base, mask)    ((*(base+2)) = (mask))
//TRISXSET + 0x08
#define DIRECT_MODE_OUTPUT(base, mask)   ((*(base+1)) = (mask))
//TRISXCLR + 0x04
#define DIRECT_WRITE_LOW(base, mask)     ((*(base+8+1)) = (mask))
//LATXCLR  + 0x24
#define DIRECT_WRITE_HIGH(base, mask)    ((*(base+8+2)) = (mask))
//LATXSET + 0x28

#else
#error "Please define I/O register types here"
#endif


class OneWire
{
  private:
    IO_REG_TYPE bitmask;
    volatile IO_REG_TYPE *baseReg;

#if ONEWIRE_SEARCH
    // global search state
    unsigned char ROM_NO[8];
    uint8_t LastDiscrepancy;
    uint8_t LastFamilyDiscrepancy;
    uint8_t LastDeviceFlag;
#endif

  public:
    OneWire( uint8_t pin);

    // Perform a 1-Wire reset cycle. Returns 1 if a device responds
    // with a presence pulse.  Returns 0 if there is no device or the
    // bus is shorted or otherwise held low for more than 250uS
    uint8_t reset(void);
```

```
    // Issue a 1-Wire rom select command, you do the reset first.
    void select(const uint8_t rom[8]);

    // Issue a 1-Wire rom skip command, to address all on bus.
    void skip(void);

    // Write a byte. If 'power' is one then the wire is held high at
    // the end for parasitically powered devices. You are responsible
    // for eventually depowering it by calling depower() or doing
    // another read or write.
    void write(uint8_t v, uint8_t power = 0);

    void write_bytes(const uint8_t *buf, uint16_t count, bool power =
0);

    // Read a byte.
    uint8_t read(void);

    void read_bytes(uint8_t *buf, uint16_t count);

    // Write a bit. The bus is always left powered at the end, see
    // note in write() about that.
    void write_bit(uint8_t v);

    // Read a bit.
    uint8_t read_bit(void);

    // Stop forcing power onto the bus. You only need to do this if
    // you used the 'power' flag to write() or used a write_bit() call
    // and aren't about to do another read or write. You would rather
    // not leave this powered if you don't have to, just in case
    // someone shorts your bus.
    void depower(void);

#if ONEWIRE_SEARCH
    // Clear the search state so that if will start from the beginning
again.
    void reset_search();

    // Setup the search to find the device type 'family_code' on the
next call
    // to search(*newAddr) if it is present.
    void target_search(uint8_t family_code);

    // Look for the next device. Returns 1 if a new address has been
    // returned. A zero might mean that the bus is shorted, there are
    // no devices, or you have already retrieved all of them.  It
    // might be a good idea to check the CRC to make sure you didn't
    // get garbage.  The order is deterministic. You will always get
    // the same devices in the same order.
    uint8_t search(uint8_t *newAddr);
#endif
```

```
#if ONEWIRE_CRC
    // Compute a Dallas Semiconductor 8 bit CRC, these are used in the
    // ROM and scratchpad registers.
    static uint8_t crc8(const uint8_t *addr, uint8_t len);

#if ONEWIRE_CRC16
    // Compute the 1-Wire CRC16 and compare it against the received
CRC.
    // Example usage (reading a DS2408):
    //    // Put everything in a buffer so we can compute the CRC
easily.
    //    uint8_t buf[13];
    //    buf[0] = 0xF0;    // Read PIO Registers
    //    buf[1] = 0x88;    // LSB address
    //    buf[2] = 0x00;    // MSB address
    //    WriteBytes(net, buf, 3);    // Write 3 cmd bytes
    //    ReadBytes(net, buf+3, 10);  // Read 6 data bytes, 2 0xFF, 2
CRC16
    //    if (!CheckCRC16(buf, 11, &buf[11])) {
    //        // Handle error.
    //    }
    //
    // @param input - Array of bytes to checksum.
    // @param len - How many bytes to use.
    // @param inverted_crc - The two CRC16 bytes in the received data.
    //                       This should just point into the received
data,
    //                       *not* at a 16-bit integer.
    // @param crc - The crc starting value (optional)
    // @return True, iff the CRC matches.
    static bool check_crc16(const uint8_t* input, uint16_t len, const
uint8_t* inverted_crc, uint16_t crc = 0);

    // Compute a Dallas Semiconductor 16 bit CRC.  This is required to
check
    // the integrity of data received from many 1-Wire devices.  Note
that the
    // CRC computed here is *not* what you'll get from the 1-Wire
network,
    // for two reasons:
    //   1) The CRC is transmitted bitwise inverted.
    //   2) Depending on the endian-ness of your processor, the binary
    //      representation of the two-byte return value may have a
different
    //      byte order than the two bytes you get from 1-Wire.
    // @param input - Array of bytes to checksum.
    // @param len - How many bytes to use.
    // @param crc - The crc starting value (optional)
    // @return The CRC16, as defined by Dallas Semiconductor.
    static uint16_t crc16(const uint8_t* input, uint16_t len, uint16_t
crc = 0);
```

```
#endif
#endif
};

#endif
```

Βιβλιοθήκη DallasTemperature.h

```
#ifndef DallasTemperature_h
#define DallasTemperature_h

#define DALLASTEMPLIBVERSION "3.7.2"

// This library is free software; you can redistribute it and/or
// modify it under the terms of the GNU Lesser General Public
// License as published by the Free Software Foundation; either
// version 2.1 of the License, or (at your option) any later version.

// set to true to include code for new and delete operators
#ifndef REQUIRESNEW
#define REQUIRESNEW false
#endif

// set to true to include code implementing alarm search functions
#ifndef REQUIRESALARMS
#define REQUIRESALARMS true
#endif

#include <inttypes.h>
#include <OneWire.h>

// Model IDs
#define DS18S20MODEL 0x10
#define DS18B20MODEL 0x28
#define DS1822MODEL  0x22

// OneWire commands
#define STARTCONVO      0x44  // Tells device to take a temperature
reading and put it on the scratchpad
#define COPYSCRATCH     0x48  // Copy EEPROM
#define READSCRATCH     0xBE  // Read EEPROM
#define WRITESCRATCH    0x4E  // Write to EEPROM
#define RECALLSCRATCH   0xB8  // Reload from last known
#define READPOWERSUPPLY 0xB4  // Determine if device needs parasite
power
#define ALARMSEARCH     0xEC  // Query bus for devices with an alarm
condition
```

```
// Scratchpad locations
#define TEMP_LSB        0
#define TEMP_MSB        1
#define HIGH_ALARM_TEMP 2
#define LOW_ALARM_TEMP  3
#define CONFIGURATION   4
#define INTERNAL_BYTE   5
#define COUNT_REMAIN    6
#define COUNT_PER_C     7
#define SCRATCHPAD_CRC  8

// Device resolution
#define TEMP_9_BIT  0x1F //  9 bit
#define TEMP_10_BIT 0x3F // 10 bit
#define TEMP_11_BIT 0x5F // 11 bit
#define TEMP_12_BIT 0x7F // 12 bit

// Error Codes
#define DEVICE_DISCONNECTED -127

typedef uint8_t DeviceAddress[8];

class DallasTemperature
{
  public:

  DallasTemperature(OneWire*);

  // initalise bus
  void begin(void);

  // returns the number of devices found on the bus
  uint8_t getDeviceCount(void);

  // Is a conversion complete on the wire?
  bool isConversionComplete(void);

  // returns true if address is valid
  bool validAddress(uint8_t*);

  // finds an address at a given index on the bus
  bool getAddress(uint8_t*, const uint8_t);

  // attempt to determine if the device at the given address is
connected to the bus
  bool isConnected(uint8_t*);

  // attempt to determine if the device at the given address is
connected to the bus
  // also allows for updating the read scratchpad
  bool isConnected(uint8_t*, uint8_t*);
```

```
  // read device's scratchpad
  void readScratchPad(uint8_t*, uint8_t*);

  // write device's scratchpad
  void writeScratchPad(uint8_t*, const uint8_t*);

  // read device's power requirements
  bool readPowerSupply(uint8_t*);

  // get global resolution
  uint8_t getResolution();

  // set global resolution to 9, 10, 11, or 12 bits
  void setResolution(uint8_t);

  // returns the device resolution, 9-12
  uint8_t getResolution(uint8_t*);

  // set resolution of a device to 9, 10, 11, or 12 bits
  bool setResolution(uint8_t*, uint8_t);

  // sets/gets the waitForConversion flag
  void setWaitForConversion(bool);
  bool getWaitForConversion(void);

  // sets/gets the checkForConversion flag
  void setCheckForConversion(bool);
  bool getCheckForConversion(void);

  // sends command for all devices on the bus to perform a temperature
conversion
  void requestTemperatures(void);

  // sends command for one device to perform a temperature conversion
by address
  bool requestTemperaturesByAddress(uint8_t*);

  // sends command for one device to perform a temperature conversion
by index
  bool requestTemperaturesByIndex(uint8_t);

  // returns temperature in degrees C
  float getTempC(uint8_t*);

  // returns temperature in degrees F
  float getTempF(uint8_t*);

  // Get temperature for device index (slow)
  float getTempCByIndex(uint8_t);

  // Get temperature for device index (slow)
  float getTempFByIndex(uint8_t);
```

```
  // returns true if the bus requires parasite power
  bool isParasitePowerMode(void);

  bool isConversionAvailable(uint8_t*);

  #if REQUIRESALARMS

  typedef void AlarmHandler(uint8_t*);

  // sets the high alarm temperature for a device
  // accepts a char.  valid range is -55C - 125C
  void setHighAlarmTemp(uint8_t*, const char);

  // sets the low alarm temperature for a device
  // accepts a char.  valid range is -55C - 125C
  void setLowAlarmTemp(uint8_t*, const char);

  // returns a signed char with the current high alarm temperature for
a device
  // in the range -55C - 125C
  char getHighAlarmTemp(uint8_t*);

  // returns a signed char with the current low alarm temperature for
a device
  // in the range -55C - 125C
  char getLowAlarmTemp(uint8_t*);

  // resets internal variables used for the alarm search
  void resetAlarmSearch(void);

  // search the wire for devices with active alarms
  bool alarmSearch(uint8_t*);

  // returns true if ia specific device has an alarm
  bool hasAlarm(uint8_t*);

  // returns true if any device is reporting an alarm on the bus
  bool hasAlarm(void);

  // runs the alarm handler for all devices returned by alarmSearch()
  void processAlarms(void);

  // sets the alarm handler
  void setAlarmHandler(AlarmHandler *);

  // The default alarm handler
  static void defaultAlarmHandler(uint8_t*);

  #endif

  // convert from celcius to farenheit
```

```
  static float toFahrenheit(const float);

  // convert from farenheit to celsius
  static float toCelsius(const float);

  #if REQUIRESNEW

  // initalize memory area
  void* operator new (unsigned int);

  // delete memory reference
  void operator delete(void*);

  #endif

  private:
  typedef uint8_t ScratchPad[9];

  // parasite power on or off
  bool parasite;

  // used to determine the delay amount needed to allow for the
  // temperature conversion to take place
  uint8_t bitResolution;

  // used to requestTemperature with or without delay
  bool waitForConversion;

  // used to requestTemperature to dynamically check if a conversion
is complete
  bool checkForConversion;

  // count of devices on the bus
  uint8_t devices;

  // Take a pointer to one wire instance
  OneWire* _wire;

  // reads scratchpad and returns the temperature in degrees C
  float calculateTemperature(uint8_t*, uint8_t*);

  void     blockTillConversionComplete(uint8_t*,uint8_t*);

  #if REQUIRESALARMS

  // required for alarmSearch
  uint8_t alarmSearchAddress[8];
  char alarmSearchJunction;
  uint8_t alarmSearchExhausted;

  // the alarm handler function pointer
  AlarmHandler *_AlarmHandler;
```

```
  #endif

};
#endif
```