Π.Μ.Σ. "ΕΦΑΡΜΟΣΜΕΝΑ ΠΛΗΡΟΦΟΡΙΑΚΑ ΣΥΣΤΗΜΑΤΑ"

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΑΝΑΠΤΥΞΗ ΕΦΑΡΜΟΓΩΝ ΜΕ ARDUINO

Ελευθέριος Κετσεμενίδης (του Ιορδανη και της Μαριας)

Εισηγήτρια: Αναστασία Ν. Βελώνη

ΑΘΗΝΑ
ΔΕΚΕΜΒΡΙΟΣ 2018

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΑΝΑΠΤΥΞΗ ΕΦΑΡΜΟΓΩΝ ΜΕ ARDUINO

Ελευθέριος Κετσεμενίδης Α.Μ. ais-0137

Εισηγήτρια: Αναστασία Ν. Βελώνη

Εξεταστική Επιτροπή:

Βελώνη Αναστασία
Έλληνας Ιωάννης
Κουκουλέτσος Κων/νος

Ημερομηνία εξέτασης: .....

ΕΥΧΑΡΙΣΤΙΕΣ

Η παρούσα διπλωματική εργασία ολοκληρώθηκε σε ένα ενδιαφέρον γνωστικό αντικείμενο της σύγχρονης τεχνολογίας, όπως αυτό της δημιουργίας αυτοματισμών με χρήση του Arduino. Την προσπάθειά αυτή υποστήριξε θερμά η καθηγήτρια μου κυρία Βελώνη Αναστασία την οποία θα ήθελα να ευχαριστήσω.

Επιθυμώ επίσης να εκφράσω τις ευχαριστίες μου σε όλους τους καθηγητές του μεταπτυχιακού για το ζήλο τους και την εξαιρετική διδασκαλία τους.

Ακόμα θα ήθελα να ευχαριστήσω την οικογένειά μου για την κατανόηση που έδειξαν καθ' όλη τη διάρκεια του μεταπτυχιακού και των σπουδών μου συνολικά.

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΠΤΥΧΙΑΚΗΣ ΕΡΓΑΣΙΑΣ

Ο κάτωθι υπογεγραμμένος **ΚΕΤΣΕΜΕΝΙΔΗΣ ΕΛΕΥΘΕΡΙΟΣ**, του **Ιορδανη** , με αριθμό μητρώου **AIS-0137** φοιτητής του Τμήματος Μηχανικών Η/Υ Συστημάτων Τ.Ε. του Α.Ε.Ι. Πειραιά Τ.Τ. πριν αναλάβω την εκπόνηση της Πτυχιακής Εργασίας μου, δηλώνω ότι ενημερώθηκα για τα παρακάτω:

«Η Πτυχιακή Εργασία (Π.Ε.) αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο του συγγραφέα, όσο και του Ιδρύματος και θα πρέπει να έχει μοναδικό χαρακτήρα και πρωτότυπο περιεχόμενο.

Απαγορεύεται αυστηρά οποιοδήποτε κομμάτι κειμένου της να εμφανίζεται αυτούσιο ή μεταφρασμένο από κάποια άλλη δημοσιευμένη πηγή. Κάθε τέτοια πράξη αποτελεί προϊόν λογοκλοπής και εγείρει θέμα Ηθικής Τάξης για τα πνευματικά δικαιώματα του άλλου συγγραφέα. Αποκλειστικός υπεύθυνος είναι ο συγγραφέας της Π.Ε., ο οποίος φέρει και την ευθύνη των συνεπειών, ποινικών και άλλων, αυτής της πράξης.

Πέραν των όποιων ποινικών ευθυνών του συγγραφέα σε περίπτωση που το Ίδρυμα του έχει απονείμει Πτυχίο, αυτό ανακαλείται με απόφαση της Συνέλευσης του Τμήματος. Η Συνέλευση του Τμήματος με νέα απόφασης της, μετά από αίτηση του ενδιαφερόμενου, του αναθέτει εκ νέου την εκπόνηση της Π.Ε. με άλλο θέμα και διαφορετικό επιβλέποντα καθηγητή. Η εκπόνηση της εν λόγω Π.Ε. πρέπει να ολοκληρωθεί εντός τουλάχιστον ενός ημερολογιακού 6μήνου από την ημερομηνία ανάθεσης της. Κατά τα λοιπά εφαρμόζονται τα προβλεπόμενα στο άρθρο 18, παρ. 5 του ισχύοντος Εσωτερικού Κανονισμού.»

ΠΕΡΙΛΗΨΗ

Όπως το περιγράφει ο δημιουργός του, το Arduino είναι μια «ανοικτού κώδικα» πλατφόρμα «πρωτοτυποποίησης» ηλεκτρονικών 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.

Το Arduino βέβαια, δεν είναι ούτε ο μοναδικός, ούτε και ο καλύτερος δυνατός τρόπος για την δημιουργία μιας οποιασδήποτε διαδραστικής ηλεκτρονικής συσκευής. Όμως το κύριο πλεονέκτημά του είναι η τεράστια κοινότητα που το υποστηρίζει και η οποία έχει δημιουργήσει, συντηρεί και επεκτείνει μια ανάλογου μεγέθους online γνωσιακή βάση. Έτσι, παρότι ένας έμπειρος ηλεκτρονικός μπορεί να προτιμήσει διαφορετική πλατφόρμα ή εξαρτήματα ανάλογα με την εφαρμογή που έχει στον νου του, το Arduino, με το εκτενές documentation, καταφέρνει να κερδίσει όλους όσους ασχολούνται με κατασκευή αυτοματισμών ιδιαίτερα για εκπαιδευτικούς σκοπούς.

ABSTRACT

Arduino is an "open source" electronic platform based on flexible and easy-to-use hardware and software designed for anyone with little programming experience, elementary electronic knowledge and interested in creating interactive objects or environments .

Arduino, of course, is neither the single nor the best possible way to create any interactive electronic device. But its main advantage is the huge community that supports it and has created, maintained and expanded a similar sized online knowledge base. Thus, although an experienced electronics person may prefer a different platform or components depending on the application he has in mind, Arduino, with extensive documentation, manages to win all those involved in building automation, especially for educational purposes.

ΕΠΙΣΤΗΜΟΝΙΚΗ ΠΕΡΙΟΧΗ: Αυτοματισμός και Μέτρηση μεγεθών
SCIENTIFIC AREA: Automation and Mesurement
ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: Αυτοματισμός, Arduino
KEYWORDS: Automation, Arduino

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΕΦΑΛΑΙΟ 1 - ARDUINO

1.1 ΕΙΣΑΓΩΓΗ

Οι **πλατφόρμες Arduino** κατασκευάζονται κυρίως από την εταιρία Smart Project. Ωστόσο, το Arduino ξεκίνησε ως έργο προς ανάπτυξη το 2005 στην Ιταλία, στο Ινστιτούτο Αλληλεπίδρασης Σχεδίασης Ivrea ώστε οι φοιτητές του Ινστιτούτου να αναπτύσσουν ενσωματωμένα συστήματα οικονομικά και αποδοτικά αξιοποιώντας τις δυνατότητες και τις ευκαιρίες που μπορεί να προσφέρει το ελεύθερο λογισμικό.

Γενικότερα, το Arduino θα λέγαμε ότι είναι ένα εργαλείο που μπορούμε να κατασκευάσουμε ένα υπολογιστικό σύστημα με την έννοια ότι αυτό θα ελέγχει συσκευές του φυσικού κόσμου, σε αντίθεση με τον κοινό Ηλεκτρονικό Υπολογιστή. Βασίζεται σε ευέλικτο, εύκολο στη χρήση υλικό και λογισμικό, σε μια αναπτυξιακή πλακέτα που ενσωματώνει επάνω έναν μικροελεγκτή και συνδέεται με τον Η/Υ για να προγραμματιστεί μέσα από ένα απλό περιβάλλον ανάπτυξης. Με το Arduino δημιουργούνται συσκευές οι οποίες εξυπηρετούν διάφορους σκοπούς έχοντας την δυνατότητα να δέχονται ερεθίσματα από το περιβάλλον τους (μέσω των αισθητήρων) και να αντιδρούν ανάλογα με το πως έχουν προγραμματιστεί.

Τα παραπάνω δεν ακούγονται πρωτότυπα. Υπάρχουν και άλλες πλατφόρμες και υλοποιήσεις που μπορούν να κάνουν τα ίδια πράγματα. Ποια είναι η ειδοποιός διαφορά; Το Arduino βασίζεται σε τεχνολογίες ανοιχτού κώδικα. Μπορεί να κατασκευαστεί από τον καθένα, μπορεί να ενσωματωθεί σε συσκευές ακόμα και για εμπορικούς σκοπούς και το σημαντικότερο είναι ότι υπάρχει μια ολόκληρη κοινότητα που χρησιμοποιεί το Arduino σε κατασκευές άρα υπάρχει μεγάλος όγκος ελεύθερης

πληροφορίας. Γενικά, 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). Το Arduino χρησιμοποιεί τώρα ένα ειδικά προγραμματιζόμενο Atmega382 αντί του chip FTDI ώστε αυτό να επιτρέπει τόσο την πιο γρήγορη ταχύτητα μεταφοράς όσο και τη γρήγορη σειριακή επικοινωνία.

Ο μικροεπεξεργαστής ενός Arduino συνήθως προγραμματίζεται εκ των προτέρων ώστε να παρέχει κάποιο φορτωτή εκκίνησης (BootLoader). Ο φορτωτής εκκίνησης υπάρχει ώστε να απλοποιεί την διαδικασία της αποθήκευσης των προγραμμάτων στην Flash Memory του Arduino μέσω σειριακής USB θύρας.

Επιπλέον, η γλώσσα προγραμματισμού, οι διάφορες βιβλιοθήκες και το ολοκληρωμένο περιβάλλον ανάπτυξης που υπάρχουν για τον προγραμματισμό της πλατφόρμας Arduino αποτελούν ανοιχτό λογισμικό προσφέροντας έτσι ανεκτίμητη γνώση σε όλους.

1.2 Βασικά Πλεονεκτήματα πλατφόρμας Arduino:

Οικονομική: Η πλατφόρμα Arduino αποτελεί οικονομική λύση διότι είναι φθηνότερη. Επιπλέον, είναι αρχιτεκτονικά ανοιχτή και μπορεί ο οποιοσδήποτε να την αναπτύξει από μόνος του.

Μεταφέρσιμη: Σε σχέση με τις υπάρχουσες πλατφόρμας στο εμπόριο η πλατφόρμα Arduino παρέχει πλήρη μεταφερσιμότητα με αποτέλεσμα να μπορεί να προγραμματιστεί στα περισσότερα λειτουργικά συστήματα.

Επεκτάσιμη: Το υλικό και το λογισμικό της πλατφόρμας Arduino είναι ανοιχτά και ελεύθερα για όλους. Καθημερινά, χιλιάδες υποστηρικτές του ελεύθερου λογισμικού αναπτύσσουν διάφορες βιβλιοθήκες για την υποστήριξη της πλατφόρμας. Παράλληλα, τόσο η αρχιτεκτονική όσο και το υλικό της πλατφόρμας εξελίσσονται συνεχώς.

Παρακάτω ακολουθούν μερικές από τις πλατφόρμες Arduino που έχουν αναπτυχθεί και όπου η κάθε μία είτε αποτελεί εξέλιξη κάποιας άλλης, είτε έχει αναπτυχθεί για κάποιο συγκεκριμένο σκοπό :

- Arduino Uno
- Arduino Diecimila
- Arduino Duemilanove
- Arduino Mega1280
- Arduino Mega2560
- Arduino Mini
- Arduino Nano
- Arduino USB
- Arduino Stamp
- Arduino Fio
- Arduino NG
- Arduino NG+
- Arduino Extreme
- Arduino Bluetooth
- LilyPad Arduino
- Serial Arduino

Ακολουθεί ένας Πίνακας όπου περιέχει για τις πιο τυπικές πλατφόρμες Arduino τα βασικά χαρακτηριστικά όσο αφορά το υλικό τους μέρος.

Πλατφόρμα Arduino	Μικροελεγκτής Atmel AVR	Flash KiB	EEPROM KiB	SRAM KiB	Ψηφιακές Επαφές E / E	PWM	Αναλογικές Επαφές Εισόδου
Diecimila	ATmega168	16	0.5	1	14	6	6
Duemilanove	ATmega168/328	16	0.5	1	14	6	6
Uno	**ATmega328**	**32**	**1**	**2**	**14**	**6**	**6**
Mega	ATmega1280	128	4	8	54	14	16
Fio	ATmega328P	32	1	2	14	6	8
Mega 2560	ATmega2560	256	4	8	54	14	16

1.3 ARDUINO - Περιγραφή

Η καρδιά του Arduino Uno είναι φυσικά ένας μικροεπεξεργαστής. Αυτός είναι το «μυαλό» του Arduino και είναι προγραμματιζόμενος ώστε να ελέγχει τα 14 ψηφιακά input/output pins και τα 6 αναλογικά που υπάρχουν πάνω στην πλακέτα ανάπτυξης. Δια μέσου αυτών των 20 pins γίνονται όλες οι διασυνδέσεις με εξωτερικά στοιχεία (κινητήρες, LEDs, LCD οθόνες κλπ) και αισθητήρες (Ultrasonic, θερμόμετρα, accelometers κ.α).

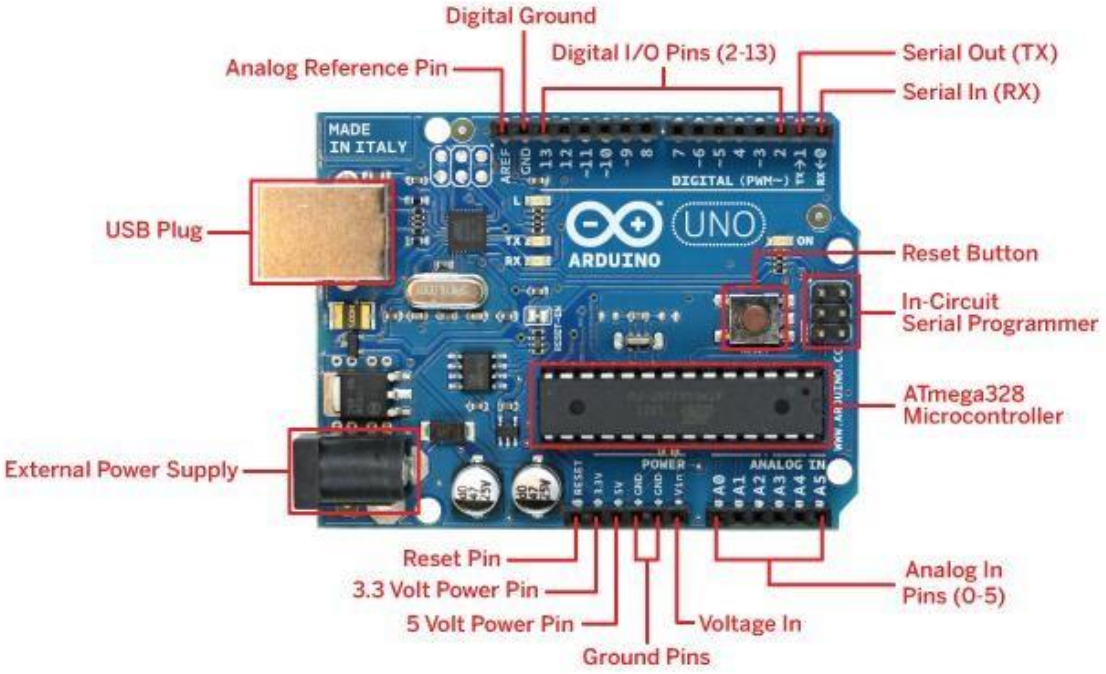
Στην πλακέτα ανάπτυξης υπάρχει μια θύρα USB. Μέσω αυτής γίνεται η μεταφορά δεδομένων από αυτήν προς κάποια άλλη συσκευή, συνήθως έναν υπολογιστή, και το αντίστροφο. Ωστόσο, η κύρια χρήση στα αρχικά στάδια εκμάθησης είναι η μεταφορά του προγράμματος από τον υπολογιστή στον μικροεπεξεργαστή αλλά και

η οπτικοποίηση των δεδομένων που απορρέουν από την λειτουργία της συσκευής μετά από το προγραμματισμό.

Το Arduino έχει 14 ψηφιακούς ακροδέκτες Εισόδου/Εξόδου οι οποίοι μπορούν να τεθούν ως είσοδοι ή ως έξοδοι με τις εντολές-συναρτήσεις pinMode(), digitalWrite(), και digitalRead() που θα αναλυθούν παρακάτω. Λειτουργούν στα 5 Volts και έχουν την δυνατότητα να παρέχουν ή να καταβυθίζουν ένταση της τάξεως των 40mA. Σε κάθε pin υπάρχει εσωτερικά ένας Pull-up αντιστάτης στα 20-50ΚΩ. Επιπλέον, έχει 6 αναλογικούς ακροδέκτες Εισόδου. Αυτοί μπορούν να διαβάσουν αναλογικές τιμές όπως η τάση μιας μπαταρίας κτλ και να τις μετατρέψουν σε έναν αριθμό από 0-1023. Η μέτρηση της τάσης γίνεται από προκαθορισμένα από 0 έως 5 volts. Εκτός αυτού, 6 εκ των 14 ψηφιακών ακροδεκτών οι P3, P5, P6, P9, P10 και P11 έχουν την δυνατότητα να προγραμματιστούν ώστε να λειτουργούν ως αναλογικές Έξοδοι.

Κάποιοι ακροδέκτες έχουν συγκεκριμένες λειτουργίες.

- **Σειριακή Λειτουργία**: 0 (RX) και 1 (TX). Χρησιμοποιούνται για λήψη (RX) και εκπομπή (TX) TTL σειριακών δεδομένων.
- **Εξωτερικές Διακοπές**: 2 και 3. Αυτοί οι ακροδέκτες μπορούν να ενεργοποιούν διακοπές αν ανιχνευθεί παλμός χαμηλής τάσης. Με την συνάρτηση attachInterrupt(). Ο σκανδαλισμός των διακοπών μπορεί να γίνεται στο λογικό 0,1.
- **PWM**: 3, 5, 6, 9, 10, και 11 pins. Παρέχουν Έξοδο 8-bit PWM με την συνάρτηση analogWrite().
- **SPI**: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). Αυτοί οι ακροδέκτες επιτρέπουν επικοινωνία SPI, η οποία αν και παρέχεται από το hardware δεν είναι ακόμα διαθέσιμη στην γλώσσα προγραμματισμού του Arduino.
- **LED**: 13. Στον ακροδέκτη 13 υπάρχει ένα ενσωματωμένο LED. Όταν ο ακροδέκτης έχει τιμή HIGH, το LED ανάβει ενώ όταν το pin είναι LOW δεν ανάβει.



1.4 Σειριακή Θύρα

Χρησιμοποιείται για επικοινωνία μεταξύ της πλατφόρμας Arduino και ενός υπολογιστή ή με άλλες συσκευές. Επομένως, όλες οι πλακέτες έχουν τουλάχιστον μια σειριακή θύρα. Επικοινωνεί με τις ψηφιακές ακίδες 0 (RX) και 1 (TX), καθώς και με τον υπολογιστή μέσω USB. Έτσι, εάν χρησιμοποιείται αυτή η λειτουργία(USB), δεν μπορούν ταυτόχρονα να χρησιμοποιηθούν οι ακίδες 0 και 1 για ψηφιακή είσοδο ή έξοδο.

Αξίζει να αναφερθεί, η ενσωματωμένη σειριακή οθόνη στο περιβάλλον του Arduino μπορεί να χρησιμοποιηθεί για να επικοινωνεί με την πλακέτα Arduino. Κάνοντας κλικ στο κουμπί Serial Monitor στην γραμμή εργαλείων και επιλέγοντας την ίδια ταχύτητα που χρησιμοποιείται στην κλήση της Serial.begin().

Οι βασικές συναρτήσεις της σειριακής θύρας είναι:

- begin() (αρχικοποίηση της σειριακής)
- end() (κλείσιμο της σειριακής)
- available() (έλεγχος αν υπάρχουν δεδομένα να διαβαστούν)
- read() (ανάγνωση των εισερχόμενων σειριακών δεδομένων)

- peek() (επιστρέφει το επόμενο byte από την σειριακή)
- flush() (άδειασμα του buffer της σειριακής από δεδομένα που έχει)
- print() (γράψιμο δεδομένων στη σειριακή)
- println() (το ίδιο με την Print(), αλλά με αλλαγή γραμμής στο τέλος)
- write() (γράφει δυαδικά δεδομένα στη σειριακή)

1.5 Χαρακτηριστικά του Arduino

- Microcontroller: ATmega328
- Τάση λειτουργίας: 5V
- Τάση εισόδου: 7-12V
- Τάση εισόδου (όριο): 6-20V
- Digital I/O Pins: 14 (εκ των οποίων 6 περιέχουν PWM εξόδους)
- Analog Input Pins: 6
- DC ρεύματος I/O Pin: 40 mA ▪ DC τρέχουσα για 3.3V Pin: 50 mA
- Flash Memory: 32 KB εκ των οποίων 0,5 KB που χρησιμοποιούνται από τον bootloader
- SRAM: 2 KB
- EEPROM: 1 KB
- Clock Speed: 16 MHz

1.5.1 Βασικές μνήμες

Οι πλατφόρμες Arduino διαθέτουν τρείς βασικές μνήμες:

- **Flash memory** (32 Kbytes) στην οποία τοποθετείται κάθε φορά το πρόγραμμα που πρόκειται να εκτελεστεί καθώς και ο φορτωτής εκκίνησης που διευκολύνει την διαδικασία του προγραμματισμού της πλατφόρμας.
- **SRAM memory** (στατική μνήμη τυχαίας προσπέλασης των 2 Kbytes) η οποία χρησιμοποιείται για την προσωρινή αποθήκευση των στατικών και των μεταβλητών δεδομένων του προγράμματος που εκτελείται.

- **EEPROM memory** (1 Kbytes) στην οποία αποθηκεύονται οι τιμές των μεταβλητών όταν η πλατφόρμα σβήσει(OFF). Χρησιμοποιείται για την αποθήκευση ρυθμίσεων και άλλων παραμέτρων ανάμεσα στα Reset του Arduino.

Πρέπει να προστεθεί, η μνήμη Flash και η μνήμη EEPROM είναι σταθερές (οι πληροφορίες παραμένουν μετά την απενεργοποίησης του ρεύματος). Η μνήμη SRAM είναι ασταθής και οι πληροφορίες χάνονται όταν εναλλάσσεται το ρεύμα.

Επειδή δεν υπάρχει πολύ διαθέσιμη SRAM, αν τελειώσει, το πρόγραμμα μπορεί να αποτύχει με απροσδόκητους τρόπους. Μπορεί να φαίνεται ότι φορτώνει με επιτυχία, αλλά δεν τρέχει, ή τρέχει παράξενα. Για να ελεγχθεί εάν αυτό συμβαίνει, μπορούν να μειωθούν τα σχόλια ή οι σειρές ή άλλες δομές δεδομένων στο sketch (χωρίς να αλλάξει ο κώδικας). Εάν λειτουργεί με επιτυχία στη συνέχεια, κατά πάσα πιθανότητα έχει εξαντληθεί η SRAM. Ένας τρόπος για να αντιμετωπιστεί αυτό το πρόβλημα είναι αν υπάρχουν πίνακες αναζήτησης ή άλλοι μεγάλοι πίνακες, τότε μπορεί να χρησιμοποιηθεί ο μικρότερος τύπος δεδομένων που είναι αναγκαίος για να αποθηκευτούν οι τιμές που χρειάζονται.

1.5.2 Τροφοδοσία

Το Arduino Uno τροφοδοτείται είτε από εξωτερική τροφοδοσία που παρέχεται είτε μέσω μιας υποδοχής των 2.1mm (θετικός πόλος στο κέντρο) που βρίσκεται στην κάτω αριστερή γωνία του Arduino είτε απευθείας από την θύρα USB του υπολογιστή. Η επιλογή της πηγής γίνεται αυτόματα από το αναπτυξιακό. Ως εξωτερική τροφοδοσία ορίζεται είτε μια μπαταρία, είτε μετασχηματιστής των 9Volt από 220V. Η μπαταρία μπορεί να συνδεθεί στις υποδοχές του Arduino Vin και GND όπου τοποθετούνται ο θετικός πόλος και ο αρνητικός αντίστοιχα. Από την άλλη αν τροφοδοτηθεί με μετασχηματιστή απλά πρέπει να τοποθετηθεί το βύσμα στην υποδοχή που υπάρχει θετικό πόλο στο κέντρο.

Η πλακέτα μπορεί να λειτουργήσει με εξωτερική πηγή από 6 έως 20 Volts. Αν ωστόσο τροφοδοτηθεί με λιγότερα από 7 Volt τα pin εξόδου 5Volt δεν θα καταφέρουν να εξάγουν τάση 5 Volts. Αντίθετα, αν δώσουμε πάνω από 12 Volts θα

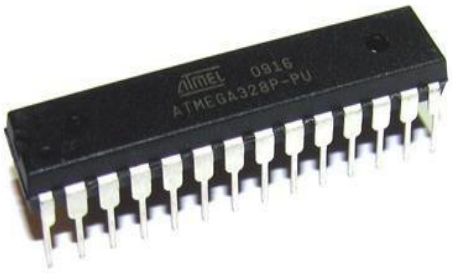
υπερθερμανθεί ο σταθεροποιητής τάσης στην πλακέτα και ενδεχομένως να καταστραφεί. Συνεπώς, μια ιδανική τάση είναι τα 9 Volts.

Οι ακροδέκτες τροφοδοσίας είναι οι εξής:

- **VIN**. Ακροδέκτης για μη σταθεροποιημένη τάση. Συνήθως εδώ συνδέεται μια εξωτερική πηγή τροφοδοσίας.
- **5V**. Ακροδέκτης σταθεροποιημένης τάσης 5Volt. Η ρυθμιζόμενη παροχή ηλεκτρικού ρεύματος που χρησιμοποιείται για την τροφοδοσία του μικροελεγκτή ή άλλων ηλεκτρονικών στοιχείων της πλακέτας. Αυτό μπορεί να προέρχεται είτε από Vin με ενσωματωμένο ρυθμιστή, ή να παρέχεται από USB ή άλλη ρυθμιζόμενη παροχή 5V
- **3V3**. Μέγιστη κατανάλωση ρεύματος είναι 50mA.
- **GND**. Γειωμένες ακίδες

1.5.3 Επικοινωνία

Το Arduino Uno έχει την δυνατότητα να επικοινωνεί με τον Ηλεκτρονικό Υπολογιστή, έναν άλλον Arduino ή άλλους μικροελεγκτές. Το ολοκληρωμένο ATMega382 παρέχει σειριακή επικοινωνία TTL 5 Volt UART, η οποία είναι διαθέσιμη από τους ακροδέκτες (λήψη RX) 0 και (εκπομπή TX) 1 του ολοκληρωμένου.



Επιπλέον, η αναπτυξιακή πλακέτα του Arduino παρέχει σειριακή επικοινωνία με τον Ηλεκτρονικό Υπολογιστή για προγραμματισμό με την βοήθεια ενός ειδικά προγραμματιζόμενου ενσωματωμένου ολοκληρωμένου ATmega382 αντί του chip FTDI. Ωστόσο, αυτό επιτρέπει την πιο γρήγορη ταχύτητα μεταφοράς και γρήγορης σειριακής επικοινωνίας. Με την σύνδεση του Arduino μέσω της θύρας USB αυτό εμφανίζεται ως εικονική σειριακή θύρα COM στο λογισμικό Εικόνα 2.2 Μικροελεγκτής ATmega328 του υπολογιστή. Το firmware ATmega382 χρησιμοποιεί τα προγράμματα οδήγησης USB COM και δεν χρειάζεται να υπάρχει εξωτερικός παράγοντας. Επομένως, στα Windows απαιτείται μόνο ένα αρχείο .inf .

Ένα Arduino περιλαμβάνει ένα τμηματικό όργανο ελέγχου το οποίο επιτρέπει την απλή μορφή κειμένου δεδομένων που αποστέλλονται προς και από τη πλακέτα Arduino. Οι RX και TX λυχνίες LED στην πλακέτα θα αναβοσβήνουν όταν γίνεται μετάδοση δεδομένων μέσω του USB-to-chip σειριακή και USB σύνδεση με τον υπολογιστή (αλλά όχι για σειριακή επικοινωνία στις ακίδες 0 και 1).

1.6 Γλώσσα Προγραμματισμού

Το Arduino Uno μπορεί να προγραμματιστεί με τη γλώσσα Wiring (ουσιαστικά πρόκειται για τη C++ με κάποιες μετατροπές). Έρχεται με ένα φορτωτή εκκίνησης που μας επιτρέπει να ανεβάζουμε νέο κώδικα χωρίς τη χρήση εξωτερικού υλικού προγραμματιστή. Επικοινωνεί χρησιμοποιώντας το αρχικό πρωτόκολλο αναπτυξιακής κάρτας STK500.

Το περιβάλλον ανάπτυξης του λογισμικού βασίζεται στην γλώσσα προγραμματισμού Processing και την γλώσσα προγραμματισμού Wiring, οι οποίες είναι ανοιχτού κώδικα (open source). Το περιβάλλον ανάπτυξης μπορεί κάποιος να το "κατεβάσει δωρεάν".

Για να γίνει σωστή εγκατάσταση του προγράμματος, πρέπει να ακολουθηθεί μια σειρά από βήματα, ανάλογα με το λειτουργικό σύστημα που διαθέτει.

- **Πλακέτα Arduino και καλώδιο USB**

. Θα χρειαστούμε ένα καλώδιο USB για να συνδεθούν πλακέτα και υπολογιστής.

Περιβάλλον Arduino

Μεταφορτώνουμε δωρεάν την τελευταία έκδοση Arduino-1.0 από την ιστοσελίδα http://arduino.cc/en/Main/Software.

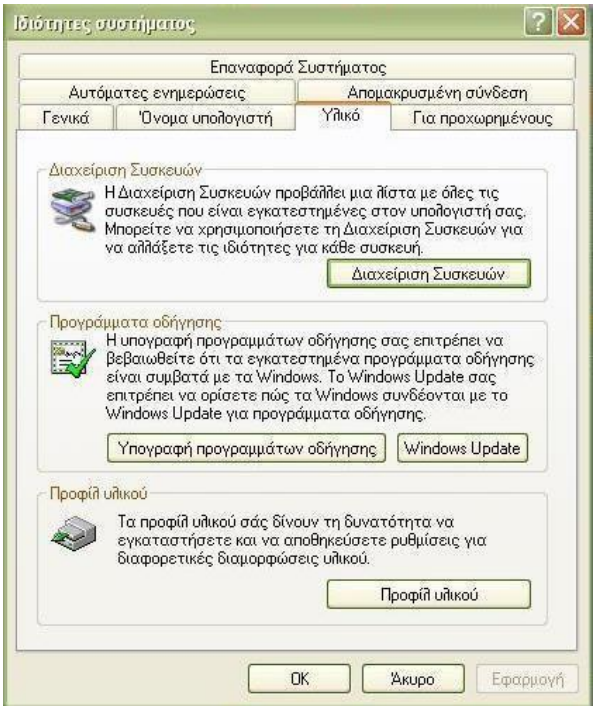
Σύνδεση της πλακέτας στον υπολογιστή

Συνδέουμε την πλακέτα Arduino Uno στον υπολογιστή χρησιμοποιώντας το καλώδιο USB. Παρατηρούμε ότι το LED της πλακέτας ανάβει.

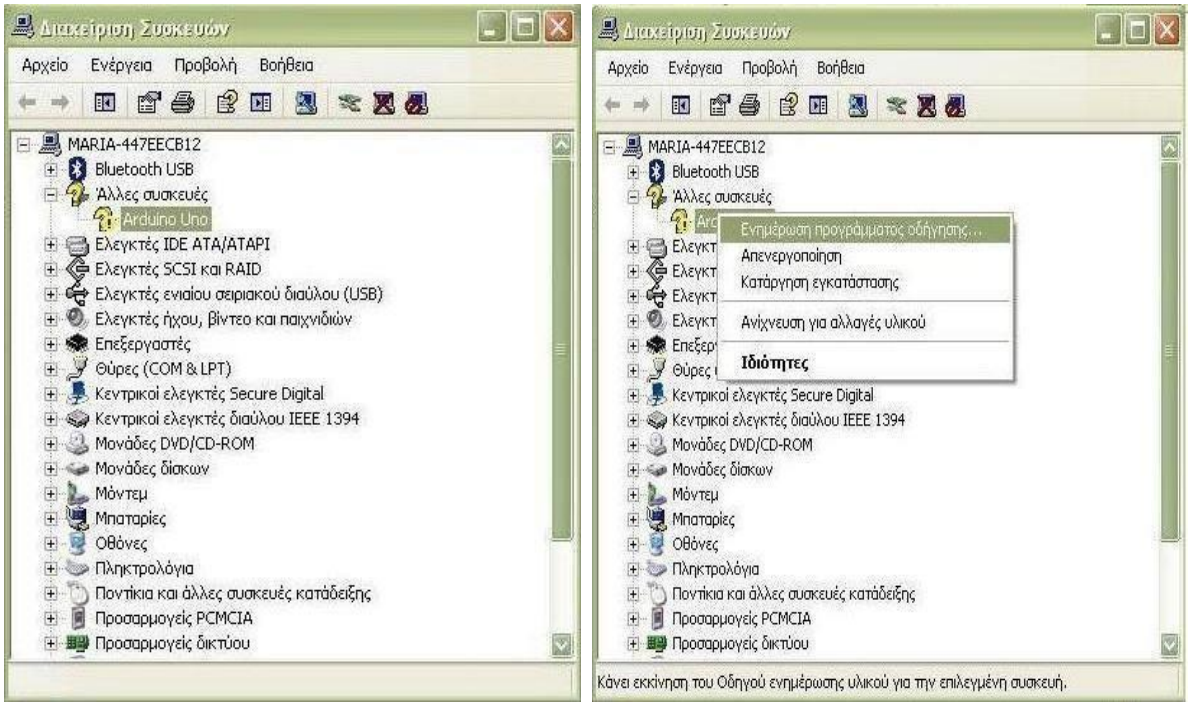
Εγκατασταση του προγράμματος

+ Κάνουμε κλικ στο μενού Έναρξη, και ανοίγουμε τον Πίνακα Ελέγχου.

+ Από τον Πίνακα Ελέγχου, μεταβαίνουμε στο «Σύστημα» και ακολούθως «Υλικό» και ανοίγουμε τη διαχείριση συσκευών.



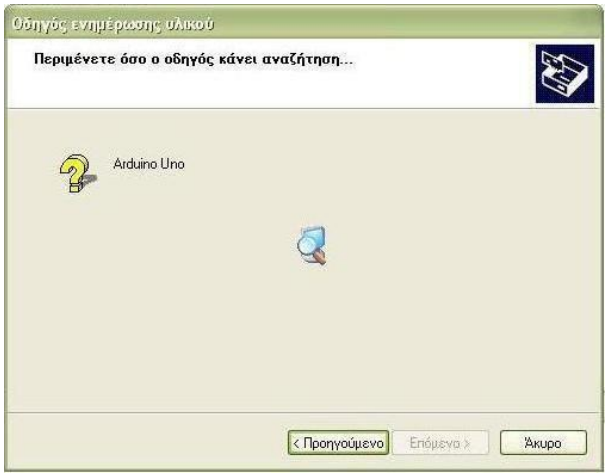
+ Βλέπουμε στις συσκευές το όνομα Arduino Uno. Κάνουμε δεξί κλικ και επιλέγουμε το «Ενημέρωση προγράμματος οδήγησης».



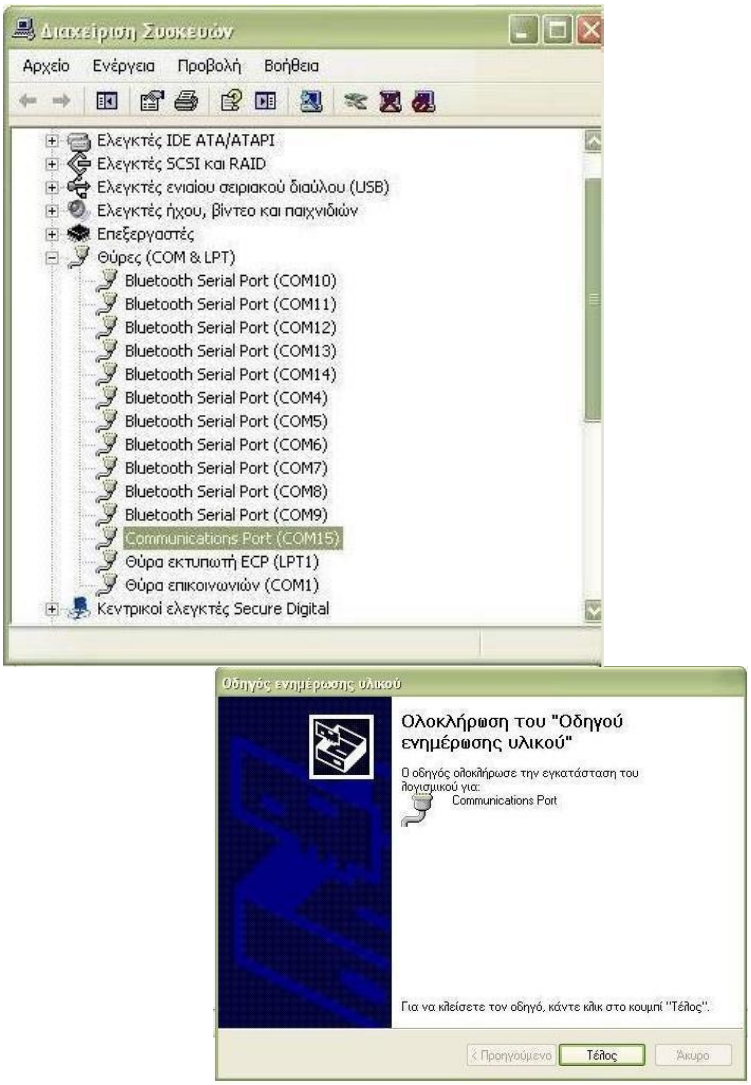
+ Ξεκινάει εγκατάσταση λογισμικού για το Arduino.

+ Κάνουμε εγκατάσταση τα drivers στον υπολογιστή μας.

+ Περιμένουμε μέχρι να τελειώσει η εγκατάσταση λογισμικού για το Arduino.



+ Τελειώνοντας παρατηρούμε ότι στις Θύρες (COM & LPT) εμφανίστηκε το Serial Port COM 15 για το Arduino που θα χρησιμοποιήσουμε. Οπότε το Arduino έχει προγραμματιστεί στη σειριακή θύρα 15.

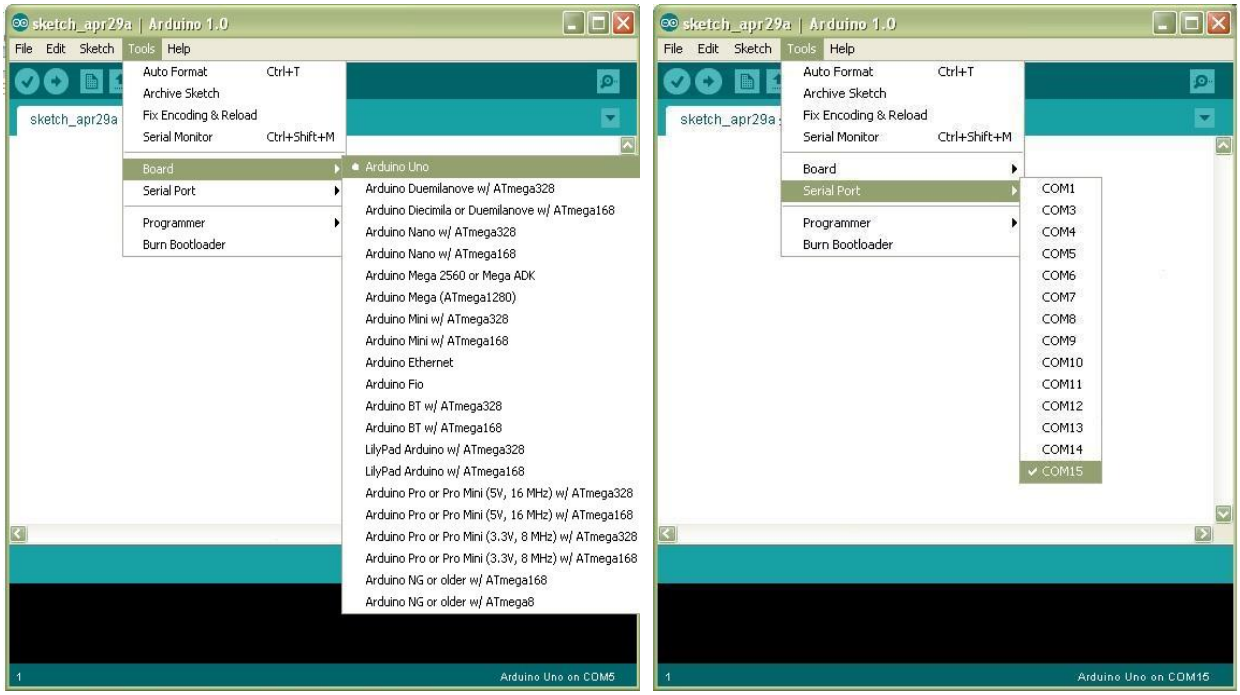


Έναρξη της εφαρμογής Arduino

Κάνουμε διπλό κλικ στην εφαρμογή Arduino.exe

Επιλογή Board και Σειριακής θύρας

Από το περιβάλλον ανάπτυξης του, από το μενού Tools επιλέγουμε για Board το Arduino Uno και για Σειριακή θύρα το COM15.



1.7 Ολοκληρωμένο Περιβάλλον Ανάπτυξης του Arduino

Το περιβάλλον ανάπτυξης Arduino περιέχει μια περιοχή επεξεργασίας κειμένου για τη συγγραφή κώδικα, μια περιοχή μηνυμάτων, ένα μενού, μια γραμμή εργαλείων με κουμπιά για κοινές λειτουργίες, καθώς και μια σειρά από μενού. Συνδέεται με το υλικό Arduino για τη φόρτωση προγραμμάτων και για να επικοινωνούν μεταξύ τους.

Ένα ολοκληρωμένο πρόγραμμα συνήθως ονομάζεται **sketch**. Αυτό το sketch είναι γραμμένο με το πρόγραμμα επεξεργασίας κειμένου. Έχει δυνατότητες για την αντιγραφή/επικόλληση και για την αναζήτηση/αντικατάσταση κειμένου. Η κονσόλα απεικονίζει την έξοδο του κειμένου από το περιβάλλον Arduino συμπεριλαμβάνοντας πλήρη μηνύματα λάθους και άλλες πληροφορίες. Τα κουμπιά της γραμμής εργαλείων επιτρέπουν τον έλεγχο και το ανέβασμα των προγραμμάτων, τη δημιουργία νέου sketch, το άνοιγμα και την αποθήκευση των sketch και άνοιγμα της σειριακής οθόνης.

1.8 Processing

Η Processing είναι μια ανοιχτού κώδικα γλώσσα προγραμματισμού και παρέχει περιβάλλον ανάπτυξης για άτομα που θέλουν να δημιουργήσουν εικόνες, κινούμενα σχέδια και διάφορες αλληλεπιδράσεις.

Αρχικά, αναπτύχθηκε ως ένα σχεδιαστικό πρόγραμμα για να διδάξει βασικές αρχές προγραμματισμού μέσα σε ένα οπτικό πλαίσιο, όμως στη συνέχεια εξελίχθηκε σε ένα εργαλείο δημιουργίας ολοκληρωμένων επαγγελματικών εργασιών. Αυτή τη στιγμή υπάρχουν δεκάδες χιλιάδες σπουδαστές, καλλιτέχνες, σχεδιαστές, ερευνητές και χομπιστές που χρησιμοποιούν την Processing για διδασκαλία, προτυποποίηση και παραγωγή.

1.8.1 Χαρακτηριστικά της Processing

- ✓ Είναι ελεύθερο/ανοικτό λογισμικό με άδεια χρήσης GPL/LGPL.
- ✓ Είναι πολυπλατφορμική, μπορεί να τρέξει σε λειτουργικά συστήματα GNU/Linux, Mac OS X και Windows.
- ✓ Δημιουργεί διαδραστικά προγράμματα, χρησιμοποιώντας δισδιάστατα(2D) ή τρισδιάστατα (3D) γραφικά.
- ✓ Ενσωμάτωση της OpelGL για επιτάχυνση 3D.
- ✓ Δημιουργία stand-alone desktop εφαρμογών και Web-based εφαρμογών
- ✓ (applets).
- ✓ Υπάρχουν αρκετές βιβλιοθήκες επέκτασης της γλώσσας για εφαρμογές ήχου, βίντεο, τεχνητής όρασης, κ.α.

Η Processing βασίστηκε στις δυνατότητες γραφικών της γλώσσας προγραμματισμού Java, απλοποιώντας τη χρήση και δημιουργώντας νέα χαρακτηριστικά.

1.8.2 Έργα βασισμένα στην Processing

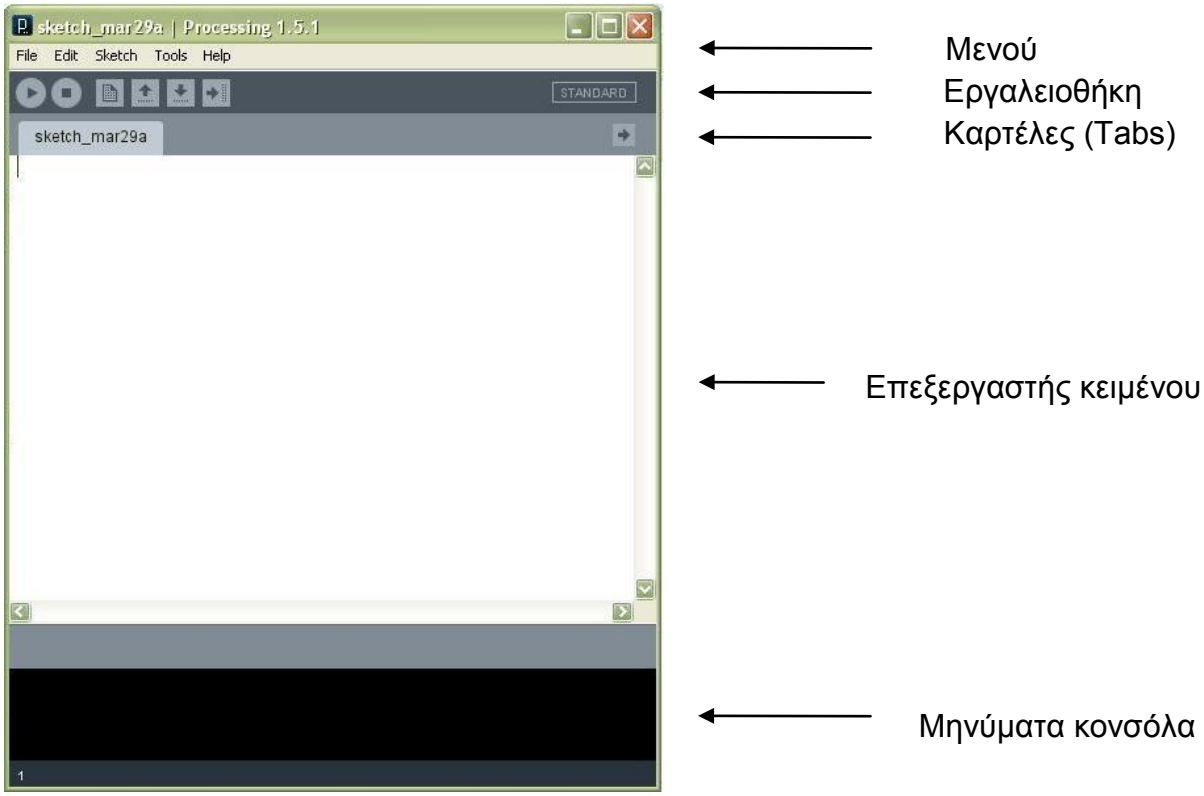
Η Processing είναι πηγή έμπνευσης για αρκετά έργα ανοιχτού κώδικα. Δύο από τα σημαντικότερα έργα που χρησιμοποιούν τη γλώσσα και το περιβάλλον της είναι η Wiring και το Arduino, οι οποίες στοχεύουν στον απλό και άμεσο προγραμματισμό hardware. Σε αντίθεση με τη Wiring, το Arduino δημιούργησε πλακέτες απλές, οικονομικές και ευκολότερες στη χρήση, ενώ κατ' επέκταση χρησιμοποιεί μια παραλλαγή της γλώσσας Processing για τον προγραμματισμό μικροελεγκτών AVR της εταιρείας Atmel. Άλλα έργα που εμπνεύστηκαν από την Processing είναι τα:

Design By Numbers, Fritzing, Mobile Processing, Processing.js, Spde, Processing in Clojure και Processing Monsters.

1.8.3 Η δομή του προγράμματος

Ένα τυπικό πρόγραμμα Processing έχει την παρακάτω δομή:

```
//δήλωση μεταβλητών
void setup()
  {
    //αρχικοποιήσεις
  }
  void draw()
  {
    //Κώδικας
  }
```



1.9 ΑΠΛΕΣ ΕΦΑΡΜΟΓΕΣ

ΕΦΑΡΜΟΓΗ 1: Συνεχές αναβόσβημα ενός Led με περίοδο 1sec, με αλλαγή της στάθμης στην ψηφιακή θύρα I/O του μικροελεγκτή. (Οι γραμμές με πορτοκαλί είναι κώδικας-εντολές όλα τα άλλα είναι σχόλια)

Το Arduino αποτελείται από δεκατρία ψηφιακά pin, τα οποία μπορούμε να τα χρησιμοποιήσουμε το κάθε ένα ξεχωριστά, είτε για είσοδο είτε για έξοδο. Μπορούμε να τα προγραμματίσουμε να συμπεριφέρονται όπως εμείς θέλουμε, αρκεί να κάνουμε τις σωστές δηλώσεις στο κώδικα που θα φορτώσουμε στη πλακέτα.

Η έξοδος του κάθε pin μπορεί να προγραμματιστεί να δίνει τιμές HIGH ή LOW. Λέγοντας HIGH ενώνουμε το δυαδικό '1' και έχουμε τάση εξόδου 5V DC, ενώ το LOW είναι το δυαδικό '0' και έχει τάση εξόδου 0V DC (ground).

Παρακάτω θα δούμε ένα παράδειγμα για να μπορέσουμε να κατανοήσουμε τον προγραμματισμό των ψηφιακών pin του Arduino. Θα ενώσουμε ένα led στη board και θα το προγραμματίσουμε να ανάβει και να σβήνει σε χρονικά διαστήματα ενός δευτερολέπτου

Blink

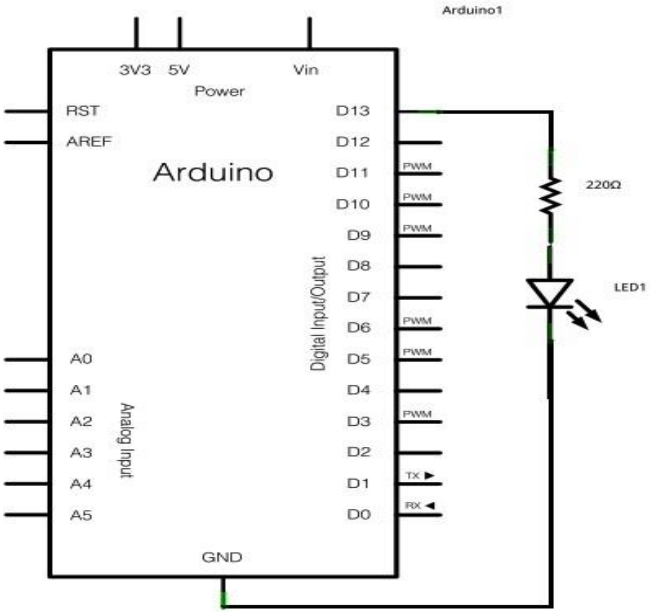
Αυτό το παράδειγμα παρουσιάζει το απλούστερο πράγμα που μπορείτε να κάνετε με ένα Arduino για να δείτε τη φυσική έξοδο: αναβοσβήνει ένα LED.

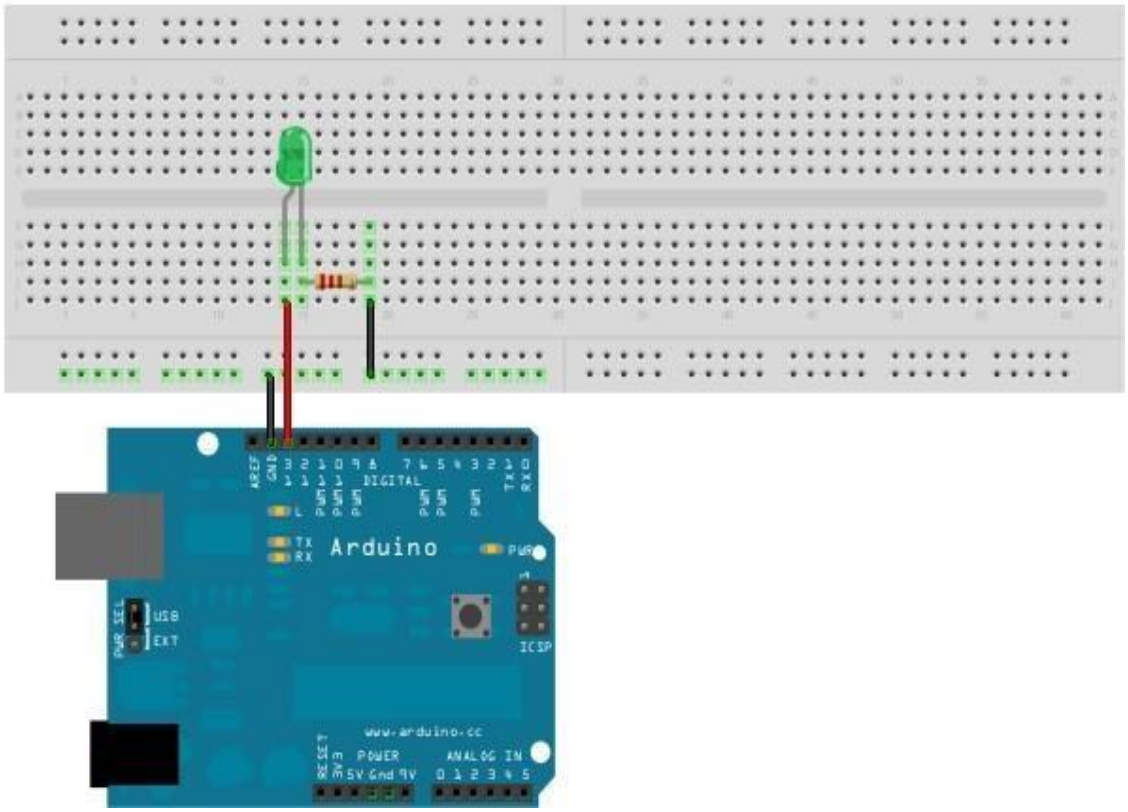
Απαραίτητα Υλικά

- Arduino Board
- breadboard
- LED, αντίσταση 220Ω,

Κύκλωμα: Για να υλοποιήσετε το κύκλωμα, συνδέστε μια αντίσταση 220Ω με το pin 13. Κατόπιν συνδέστε τον μακρύτερο ακροδέκτη (άνοδο) με την αντίσταση. Συνδέστε το κοντό ακροδέκτη (κάθοδο) με τη γείωση. Κατόπιν συνδέστε το Arduino με τον υπολογιστή σας, ξεκινήστε το πρόγραμμα Arduino, και πληκτρολογείστε τον παρακάτω κώδικα. Τα περισσότερα Arduino ήδη έχουν ένα LED στο pin 13. Εάν

τρέχετε αυτό το παράδειγμα χωρίς εξωτερικό LED και αντισταση, πρέπει να δείτε το ενσωματωμένο LED να αναβοσβήνει.



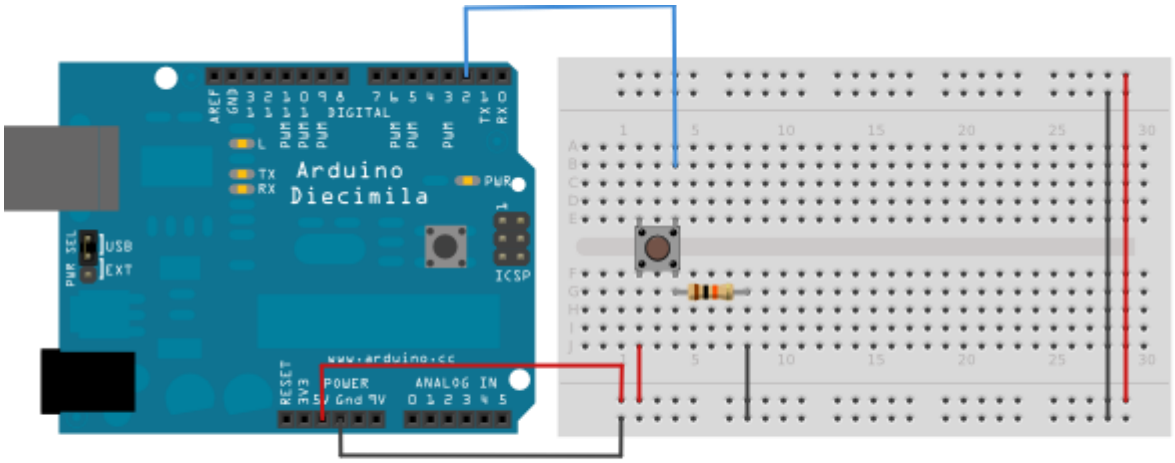


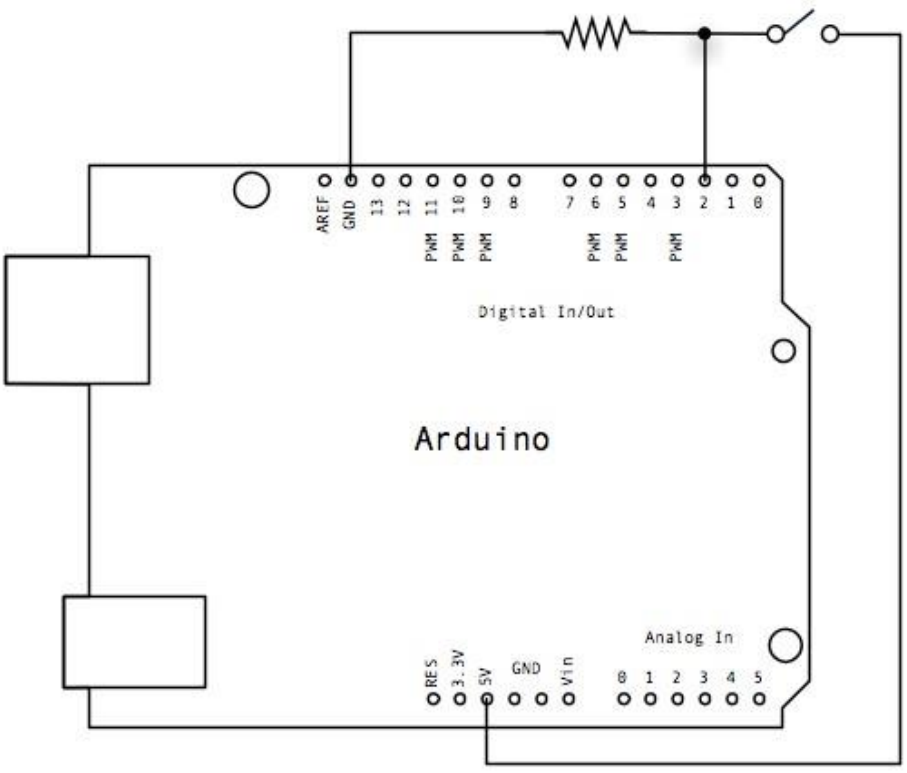
Για την δημιουργία της εικόνας χρησιμοποιήθηκε το πρόγραμμα **Fritzing**. Για περισσότερα παραδείγματα κυκλωμάτων, δείτε**: Fritzing project page**

Κώδικας

```
// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
int led = 13;   //  Ορίζουμε μία μεταβλητή τύπου  Integer (Ακεραίου) με όνομα  led και της δίνουμε την τιμή 13  με σκοπό να την χρησιμοποιήσουμε παρακάτω στον κώδικα και να δηλώσουμε που θα συνδέσουμε το led.

// the setup routine runs once when you press reset:

void setup() {    //  Η συνάρτηση αρχικών ρυθμίσεων του μικροελεγκτή
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);     // Ορίζουμε χρησιμοποιώντας την μεταβλητή led, ότι το pin 13 θα λειτουργεί σαν ψηφιακή έξοδος 0-5V
}

// the loop routine runs over and over again forever:

void loop() { // Η κύρια ρουτίνα του προγράμματος που εκτελείται συνεχώς
  digitalWrite(led, HIGH);   // turn the LED on (HIGH is the voltage level)
                             //  Βγάλε λογικό 1 (5V) στο pin 13, δηλαδή άναψε το led
  delay(1000);               // wait for a second
                             //  Συνάρτηση καθυστέρησης χρόνου 1000ms
  digitalWrite(led, LOW);    // turn the LED off by making the voltage LOW
                             //Βγάλε λογικό 0 (0V) στο pin 13, δηλαδή σβήσε το led
  delay(1000);               // wait for a second
                             //  Συνάρτηση καθυστέρησης χρόνου 1000ms
}  // Επιστρέφει στην γραμμή digitalWrite(led, HIGH);
```

Εφαρμογή 2: Άναμμα και σβήσιμο Led ελεγχόμενο από εξωτερικό button με την διαδικασία polling





Κώδικας

```
// constants won't change. They're used here to // set pin numbers:
const int buttonPin = 2;     // the number of the
pushbutton pin
//  Ορίζουμε μία σταθερά τύπου Integer (Ακεραίου) με όνομα buttonPin και της δίνουμε την τιμή 2 με σκοπό να την χρησιμοποιήσουμε παρακάτω στον κώδικα και να δηλώσουμε που θα συνδέσουμε το button.
const int ledPin =  13;      // the number of the LED pin
//  Ορίζουμε μία σταθερά τύπου Integer (Ακεραίου) με όνομα ledPin και της δίνουμε την τιμή 13 με σκοπό να την χρησιμοποιήσουμε παρακάτω στον κώδικα και να δηλώσουμε που θα συνδέσουμε το led.
// variables will change:
int buttonState = 0;         // variable for reading the
pushbutton status
// Ορίζουμε μία μεταβλητή τύπου ? Integer (Ακεραίου) με όνομα buttonState και της δίνουμε αρχικά την τιμή 0 με σκοπό να την χρησιμοποιήσουμε παρακάτω στον κώδικα για να αποθηκεύουμε την κατάσταση λειτουργίας του button (on-off)
void setup() { //  Η συνάρτηση αρχικών
ρυθμίσεων του μικροελεγκτή // initialize the
LED pin as an output:
pinMode(ledPin, OUTPUT);     // Ορίζουμε χρησιμοποιώντας την μεταβλητή ledPin, ότι το pin 13 θα λειτουργεί σαν ψηφιακή έξοδος 0-5V
// initialize the pushbutton pin as an input:
pinMode(buttonPin, INPUT);     // Ορίζουμε χρησιμοποιώντας την μεταβλητή
buttonPin, ότι το pin 2
θα λειτουργεί σαν
ψηφιακή είσοδος 0-
5V }
void loop(){  // Η κύρια ρουτίνα του προγράμματος
που εκτελείται συνεχώς // read the state of the
pushbutton value:
buttonState = digitalRead(buttonPin); // Διαβάζουμε την ψηφιακή τιμή  που έχει το Pin 2 (εκεί που συνδέεται το button) και την αποθηκεύουμε στην μεταβλητή buttonState // check if the pushbutton is pressed.
// if it is, the buttonState is HIGH:
if (buttonState == HIGH) {   //Ελέγχουμε αν το button είναι
πατημένο, αν ναι τότε: // turn LED on:
digitalWrite(ledPin, HIGH);  //  Βγάλε λογικό 1 (5V) στο pin
13, δηλαδή άναψε το led } else { //αλλιώς:
// turn LED off:
digitalWrite(ledPin, LOW);  //Βγάλε λογικό 0 (0V) στο pin 13,
δηλαδή σβήσε το led }
}
```

ΚΕΦΑΛΑΙΟ 2 – ΕΛΕΓΧΟΣ ΘΕΡΜΟΚΡΑΣΙΑΣ ΜΕ ΧΡΗΣΗ Arduino ΚΑΙ Labview

2.1 ΠΕΡΙΛΗΨΗ ΕΦΑΡΜΟΓΗΣ

Αυτή η εφαρμογή αφορά τον έλεγχο της θερμοκρασίας χρησιμοποιώντας ένα Arduino και το πρόγραμμα LabVIEW, στο οποίο ο χρήστης μπορεί να ρυθμίσει την ελάχιστη και μέγιστη θερμοκρασία, να βλέπει την υπάρχουσα θερμοκρασία στο χώρο με την χρήση ενός αισθητήρα μέτρησης θερμοκρασίας LM35, καθως και να γνωρίζει τις καταστάσεις «HOT», «COLD» και «MODERATE» με την κατάλληλη ένδειξη από το Front Panel, σε επικοινωνία με το RGB LED.

Εάν η θερμοκρασία υπερβεί το ανώτατο καθορισμένο επίπεδο που έχει τεθεί από τον χρήστη ανάβει η ένδειξη «HOT». Τότε, το Arduino ενεργοποιεί μια λειτουργία, στην περίπτωσή μας ένας ανεμιστήρας που έχει ως σκοπό να μειώσει την θερμοκρασία. Αν η θερμοκρασία “πέσει” κάτω από το ελάχιστο όριο που έχει θέσει ο χρήστης τότε ανάβει η ένδειξη «COLD». Τέλος, αν η θερμοκρασία είναι ανάμεσα στα επιτρεπτά όρια που έχουν τεθεί, ανάβει η ένδειξη «MODERATE». Κάθε μεταβολή της θερμοκρασίας παρουσιάζεται συνεχώς σε ένα διάγραμμα.

Στην παρούσα εργασία χρησιμοποιούμε το **Arduino UNO R3**.

2.2 LabView

Το LabView είναι ένα πρόγραμμα προσομοίωσης, συλλογής και ανάλυσης δεδομένων. Στηρίζεται στον γραφικό προγραμματισμό μέσω αντικειμένων και αποτελεί ένα καλό παράδειγμα του αντικειμενοστραφή προγραμματισμού (object oriented programming), εν αντιθέσει με τον προγραμματισμό διαδικασιών, όπου ο προγραμματιστής γράφει κώδικα εντολών που εκτελούνται με γραμμική διαδοχή.

Ένα εικονικό όργανο μπορεί να προσομοιώνει απλώς μια λειτουργία και να την παρουσιάζει στην οθόνη του υπολογιστή. Όμως, είναι δυνατό να συνδέεται με τις θύρες εισόδου/εξόδου του υπολογιστή ή με επιπρόσθετες κάρτες επέκτασης,

προκειμένου να κάνει πραγματική εισαγωγή ή εξαγωγή δεδομένων. Στην περίπτωση αυτή ο υπολογιστής με τη βοήθεια των εισόδων και των εξόδων μετατρέπεται σε ένα ισχυρό εργαλείο μετρήσεων, με πολλές δυνατότητες επεξεργασίας δεδομένων. Την δυνατότητα αυτή θα εκμεταλλευτούμε και εμείς ώστε να το χρησιμοποιήσουμε για την διαχείριση των δεδομένων θερμοκρασίας.

Το LabView έχει τα εξής τρία βασικά μέρη:

- Το εμπρόσθιο πλαίσιο(Front panel)
- Το δομικό διάγραμμα(Block Diagram)
- Τις παλέτες εργαλείων και ελέγχου/λειτουργιών.

Το εμπρόσθιο πλαίσιο μοιάζει με το μπροστινό μέρος ενός οργάνου. Μπορεί να περιέχει κουμπιά, διακόπτες, οθόνες γραφικών κ.ά. Τα στοιχεία του εμπρόσθιου πλαισίου παίρνουν τιμές με τη βοήθεια του ποντικιού ή του πληκτρολογίου. Για παράδειγμα, με το ποντίκι μπορούμε να πατήσουμε έναν διακόπτη και με το πληκτρολόγιο να ορίσουμε τιμή για μια τάση ή για μια θερμοκρασία. Το εμπρόσθιο πλαίσιο είναι αυτό που κυρίως χειρίζεται ο χρήστης.

Το δομικό διάγραμμα υπάρχει πάντα μαζί με το εμπρόσθιο πλαίσιο και ισοδυναμεί με τον κώδικα προγράμματος στη γραφική γλώσσα G. Κάθε στοιχείο του δομικού διαγράμματος παριστάνεται από ένα εικονίδιο. Ένα κουμπί μπορεί να είναι μια μεταβλητή που να παίρνει τιμές True/False ή μια μεταβλητή διπλής ακρίβειας για εισαγωγή δεδομένων, οπότε φαίνεται στο δομικό διάγραμμα από το αντίστοιχο εικονίδιο που συμβολίζει τη μεταβλητή. Μια συνάρτηση έχει το δικό της εικονίδιο, το ίδιο και κάθε ολοκληρωμένη λειτουργία.

Όταν μεταβαίνουμε από το παράθυρο του εμπρόσθιου πλαισίου στο παράθυρο του δομικού διαγράμματος, η παλέτα ελέγχου (controls) μετατρέπεται αυτόματα στην παλέτα Λειτουργιών (Functions). Εκεί μπορούμε να βρούμε εικονίδια για απλές αριθμητικές λειτουργίες, όπως πρόσθεση αφαίρεση κλπ., εικονίδια μεταβλητών, πινάκων, συγκρίσεων, μετατροπών, εικονίδια για τη δημιουργία αρχείων, εικονίδια συναρτήσεων (ημίτονο, μετασχηματισμούς Fourier κλπ.) εικονίδια προγραμματιστικών δομών, όπως FOR Loop, While Loop, Sequence κλπ. Και τέλος,

εικονίδια για χρήση διαφόρων πρωτοκόλλων επικοινωνίας ή συγκεκριμένων έτοιμων οργάνων. Πρόκειται για τη βιβλιοθήκη προγραμματισμού της γλώσσας G.

2.3 LINX LABVIEW MAKERHUB

Το **LINX** είναι ένα project ανοιχτού κώδικα από την Digilent και έχει σχεδιαστεί ώστε να είναι εύκολο να αναπτυχθούν εφαρμογές που χρησιμοποιούν LabVIEW. Το LINX περιλαμβάνει VIs για πάνω από 30 από τους πιο κοινούς αισθητήρες για την πρόσβαση σε περιφερειακές συσκευές, όπως ψηφιακές I/O,αναλογικά I/O,PWM,I2C,SPI και UART.Με την ανάπτυξη VIs που τρέχουν σε BeagleBone, Raspberry Pi 2/3 και Arduino, LINX και LabVIEW καθιστούν εύκολη την οπτικοποίηση των δεδομένων, τον εντοπισμό σφαλμάτων στον κώδικα, και την δημιουργία προηγμένων εφαρμογών γρηγορότερα από ποτέ.

LINX

2.4 ΥΛΙΚΑ ΚΑΤΑΣΚΕΥΗΣ

BOM(Bill of Materials)

1. Arduino Uno R3 x1
2. Breadboard x1
3. Arduino jumper wires

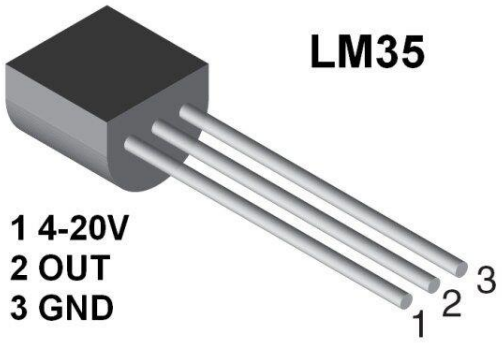
4. Αισθητήρας LM35 x1

5. DC Fan x1

6. Common Cathode RGB LED x1

2.5 Ανάλυση Υλικών

2.5.1 Αισθητήρας θερμοκρασίας LM35

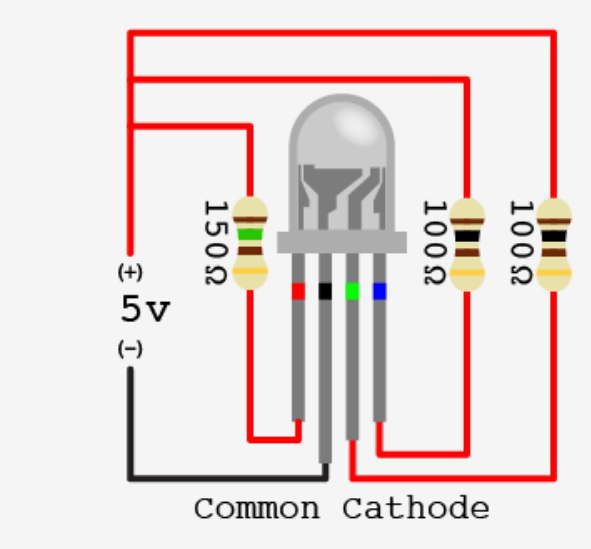


Το αισθητήριο θερμοκρασίας που επιλέχθηκε είναι το LM35. Είναι διαδεδομένο για την ακρίβειά του καθώς και το χαμηλό κόστος του. Μπορεί να μετρήσει θερμοκρασίες από −55°C έως 150°C. Διαθέτει τρία pins: Vcc, Output και Ground, τα οποία συνδέονται στην 5V τροφοδοσία του Arduino, στο pin A0 και στο GND αντίστοιχα. Ο αισθητήρας αυτός μας δίνει την έξοδο του, όταν μετρήσουμε την τάση μεταξύ του Vout pin(2) & GND pin(3). Κάθε 10mV διαφοράς στην τάση αντιστοιχούν σε 1 βαθμό Κελσίου. Για παράδειγμα, αν μετρήσουμε τάση 0,325 V τότε η θερμοκρασία είναι 32,5 βαθμοί Κελσίου, για τάση 0.335 έχουμε 33,5 βαθμούς κ.ο.κ.

Παρατίθεται το datasheet του, το οποίο μας δίνει πληροφορίες για τα τεχνικά χαρακτηριστικά του εν λόγω αισθητήρα**: http://www.ti.com/lit/ds/symlink/lm35.pdf**

2.5.2 RGB LED Common Cathode

Αυτό το LED αποτελείται από 4 pins το μεγαλύτερο pin είναι η κάθοδος. Τα υπόλοιπα pins αντιστοιχούν και στα ανάλογα χρώματα όπως φαίνεται και στην παρακάτω εικόνα. Το pin της κοινής καθόδου συνδέεται στο GND.Τα υπόλοιπα 3 pins στις τροφοδοσίες του Arduino Uno.



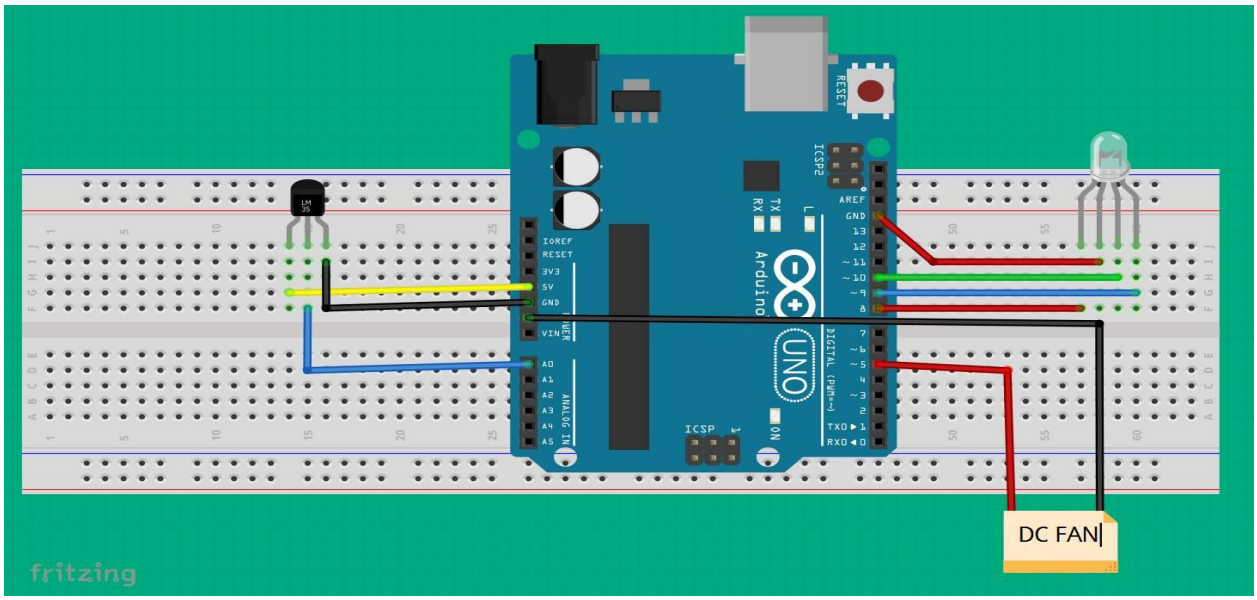
Όπως παρατηρούμε, έχουμε συνδέσει στις 3 ανόδους του RGB LED 3 αντιστάτες 100Ω. Αυτό είναι απαραίτητο ώστε να περιορίσουμε το ρεύμα που παρέχεται στο RGB LED, ώστε αυτό να μην καεί. Γενικότερα, σε οποιαδήποτε εφαρμογή χρησιμοποιούμε οποιουδήποτε τύπου led, πρέπει να συνοδεύεται από μια αντίσταση στην άνοδο του.

2.5.3 DC Fan

Αποτελείται από 2 ακροδέκτες, ο ένας συνδέεται σε κάποια από τις τροφοδοσίες του Arduino Uno ελάχιστη τροφοδοσία που μπορεί να λειτουργήσει είναι τα 5V (Ο συγκεκριμένος λειτουργεί στα 12V μπορεί όμως να λειτουργήσει και στα 5Vμε λιγότερα RPM) και ο άλλος ακροδέκτης στο GND.

2.6 ΥΛΟΠΟΙΗΣΗ ΚΑΤΑΣΚΕΥΗΣ

2.6.1 Σχηματικό Διάγραμμα Σύνδεσης



Η υλοποίηση της κατασκευής μας είναι αρκετά απλή. Συνδέουμε τα εξαρτήματα σύμφωνα με το breadboard, δίνοντας προσοχή να μην ξεχάσομε τους αντιστάτες στα LED. Έπειτα προχωράμε στον προγραμματισμό της διάταξης.

2.6.2 ΚΑΤΑΣΚΕΥΗ VI ΓΙΑ ΤΗΝ ΕΠΙΚΟΙΝΩΝΙΑ ΜΕ ΤΟ LABVIEW

Βήματα δημιουργίας

File > New Vi

Από το Front Panel

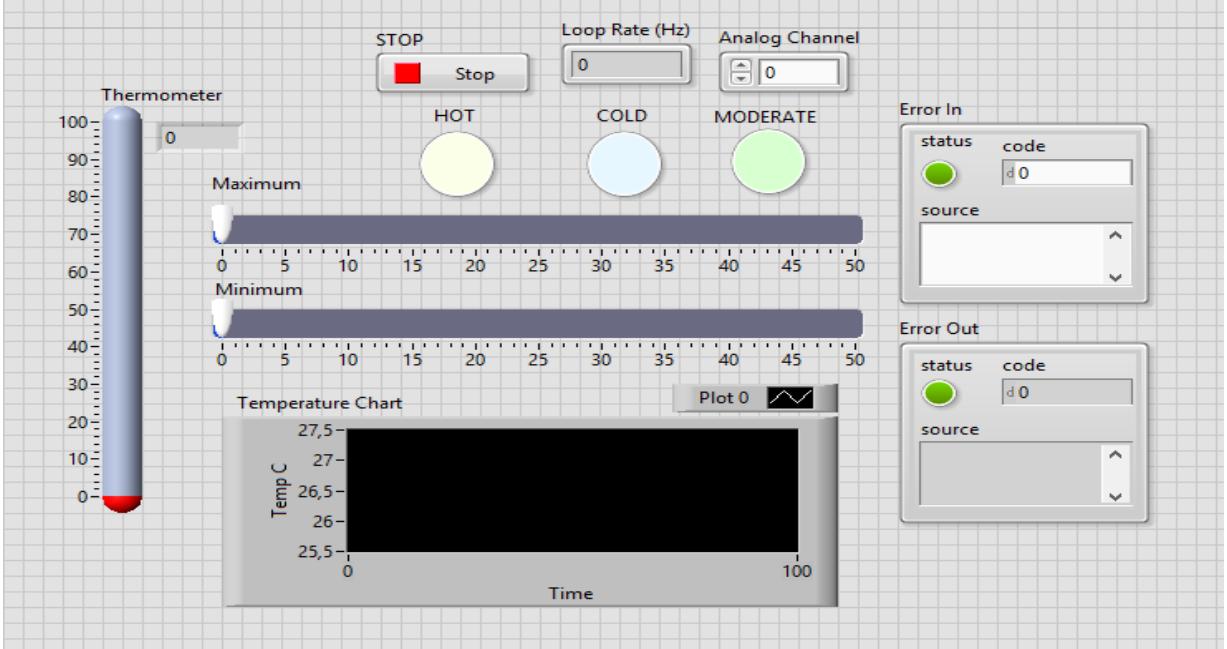
- Palette > Express > numeric indicator > thermometer
- Palette > Express > graph indicator > chart
- Palette > Modern > numeric > horizontal pointer slide x2
- Palette > Silver > Boolean > Led x3

Από το Block Diagram

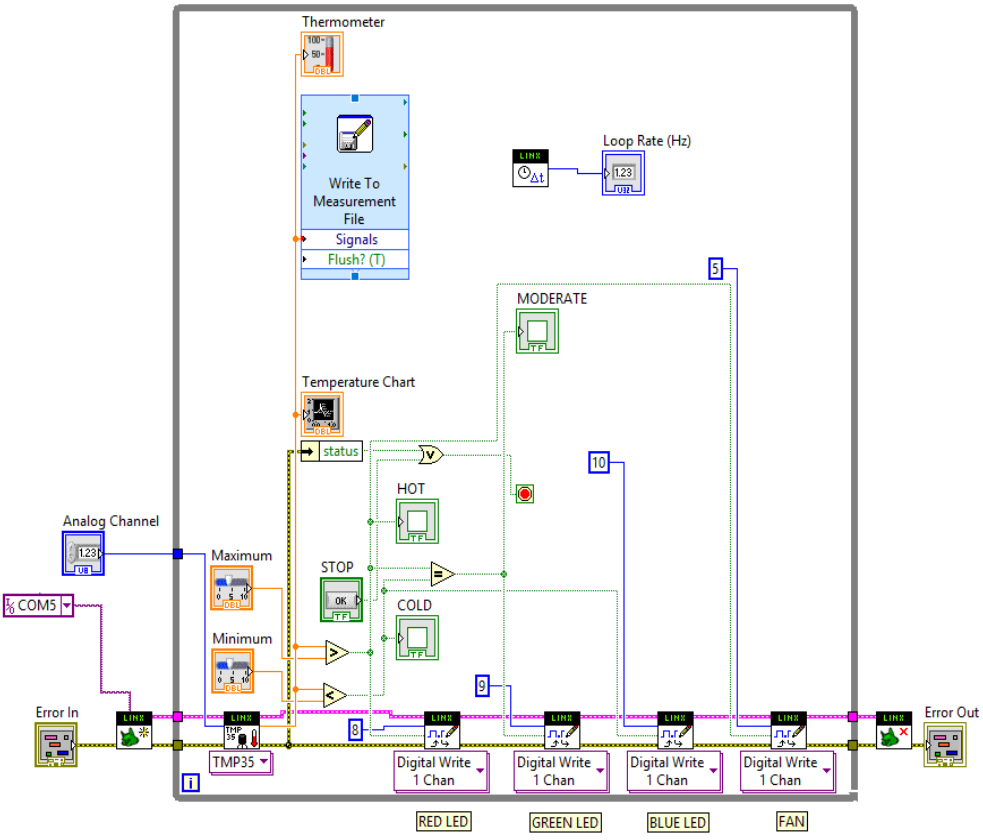
- Palette >MakerHub> LINX > Open
  - Serial Port>Control
  - Error In> Control
- Palette > Structures > while loop
- Palette >MakerHub> LINX > Sensors > Temp> Temp3x
  - analog channel> control
  - temp C >chart, thermometer
- Palette>MakerHub>LINX>Peripherals>Digital>Writex4
  - Output Value > Comparison
  - DO channel > Constant
  - Error Out > LINX close
  - LINX Resource >LINX close
- Palette > Programming > Numeric > Or
- Palette > Programming> Comparison > Less, Greater, Equal
- Palette >MakerHub> LINX > Close
  - Error Out > Control
- Palette > Modern >Silver >Boolean > Stop Button
- Palette >MakerHub> LINX > Utilities > Loop Frequency
  - Loop rate > Control
- Palette > Programming > File I/O > Write To Measurement File

Πραγματοποιώντας τα παραπάνω βήματα έχουμε το κάτωθι αποτέλεσμα:

Front Panel:



Block Diagram:



2.7 ΛΕΙΤΟΥΡΓΙΑ ΚΥΚΛΩΜΑΤΟΣ

Αρχικά το LINX open VI ανοίγει τη σύνδεση με την απομακρυσμένη συσκευή LINX και δεσμεύει τυχόν τοπικούς πόρους I/O. Δέχεται ως είσοδο το ComPort που αντιστοιχεί στο Arduino Uno, και το Error In που περιγράφει τις συνθήκες σφάλματος που εμφανίζονται πριν τελειώσει αυτός ο κόμβος. Στη συνέχεια, όλα τα υπόλοιπα Vis εκτελούνται σε ένα While Loop. Το Linx Temp35 VI αντιστοιχεί στο μοντέλο του αισθητήρα που έχουμε, και δέχεται είσοδο το Analog Channel στο οποίο έχουμε εισάγει τον αντίστοιχο αριθμό του pin που είναι συνδεδεμένος στο Arduino Uno ο αισθητήρας. Η έξοδος tempC συνδέεται με το Thermometer VI ώστε να δείχνει την θερμοκρασία του χώρου, με το Temperature Chart VI ώστε να απεικονίζει τις διακυμάνσεις της θερμοκρασίας ως πρός το χρόνο σε κάθε millisecond και τέλος με το Write to Measurement file VI ώστε να αποθηκεύει τις θερμοκρασίες σε ένα excel αρχείο.

Εν συνεχεία η θερμοκρασία συγκρίνεται μέσω τελεστών σύγκρισης. Ο τελεστής “>” δέχεται είσοδο το Maximum VI το οποίο κρατάει την τιμή της θερμοκρασίας σε μια μεταβλητή “y” που έχουμε ορίσει, και την θερμοκρασία από το Linx temp 35 που είναι η μεταβλητή “x”. Αν στην έξοδο του τελεστή έχουμε TRUE η έξοδος συνδέεται στην είσοδο output value και ενεργοποιεί το RED LED το οποίο απεικονίζεται με το digital write 1 Chan vi, στην είσοδο DO channel έχει την αντίστοιχη θύρα από το Arduino Uno που έχουμε συνδέσει το pin, την DO[8]. Το fan απεικονίζεται επίσης με το digital write 1 Chan vi το οποίο έχει στην είσοδο DO channel την αντίστοιχη θύρα από το Arduino Uno, την D0[5], και την κατάσταση HOT.

Το ίδιο ισχύει και με τον τελεστή σύγκρισης “<” αν βγει η έξοδος του τελεστή TRUE συνδέεται στην είσοδο output value και ενεργοποιεί το BLUE LED με το digital write 1 Chan vi,στην είσοδο DO channel έχει την αντίστοιχη θύρα από το Arduino Uno την DO[10] και την κατάσταση COLD, όταν η θερμοκρασία είναι μικρότερη από αυτή που έχει οριστεί.

Αν κανείς από τους τελεστές σύγκρισης δεν έχει έξοδο TRUE αλλα και οι 2 FALSE τότε εκτελείται ο συντελεστής σύγκρισης “=” γιατί η μεταβλητή “x” & “y” είναι FALSE και η έξοδος του τελεστή συνδέεται στην είσοδο output value και ενεργοποιεί το

GREEN LED με το digital write 1 Chan vi το οποίο στην είσοδο DO channel έχει την αντίστοιχη θύρα από το Arduino Uno που έχουμε συνδέσει το pin την DO[9].

Τέλος εκτός του Loop While το LINX close vi για να διακοπεί η σύνδεση με την απομακρυσμένη συσκευή και αποδεσμεύει τους πόρους. Στην έξοδο το Error Out περιέχει πληροφορίες τυχόν σφαλμάτων. Η διαδικασία Loop While σταματάει να εκτελείται αφού γίνει λογική πράξη μέσω της λογικής πύλης OR η οποία δέχεται σαν είσοδο “x” το STOP BUTTON & “y” error το οποίο έχει προκύψει. Σε κάθε Loop While που εκτελείτε όλες αυτές οι καταστάσεις που αναφέραμε παραπάνω εξετάζονται ταυτόχρονα και οποία είναι αληθής εκτελείται και η κατάλληλη ενέργεια.

ΚΕΦΑΛΑΙΟ 3 – Μέτρηση θερμοκρασίας και αποστολή δεδομένων με Bluetooth στο πρόγραμμα LabView

3.1 ΠΕΡΙΛΗΨΗ ΕΦΑΡΜΟΓΗΣ

Στην εφαρμογή αυτή θα ασχοληθούμε με την κατασκευή ενός κυκλώματος, το οποίο έχει ως σκοπό την μέτρηση της εξωτερικής θερμοκρασίας, και την αποστολή των μετρήσεων σε έναν υπολογιστή. Για την υλοποίηση του κυκλώματος αυτού θα χρησιμοποιήσουμε μια πλακέτα Arduino Uno, στην οποία συνδέουμε έναν αισθητήρα θερμοκρασίας LM35, καθώς και ένα Bluetooth module. Οι τιμές της θερμοκρασίας που εισάγονται αναλογικά στο Arduino μέσω του αισθητήρα θερμοκρασίας, αποστέλλονται μέσω του Bluetooth module στον υπολογιστή, και απεικονίζονται μέσω του προγράμματος LabView.

3.2 ΥΛΙΚΑ ΚΑΤΑΣΚΕΥΗΣ

BOM(Bill of Materials)

1. Arduino Uno R3 x1
2. Διάτρητη πλακέτα x1
3. Αρσενικά pins για το Arduino
4. Μπαταρία 9V x1
4. Bluetooth Module HC-06 x1
5. Πυκνωτής 104 nF x1

3.3 Ανάλυση Υλικών

3.3.1 Αισθητήρας θερμοκρασίας LM35

Το αισθητήριο θερμοκρασίας που επιλέχθηκε είναι το LM35. Είναι διαδεδομένο για την ακρίβειά του καθώς και το χαμηλό κόστος του. Διαθέτει τρία pins: Vcc, Output και Ground. Ο αισθητήρας αυτός μας δίνει την έξοδο του, όταν μετρήσουμε την τάση μεταξύ του Vout pin(2) & GND pin(3). Κάθε 10mV διαφοράς στην τάση αντιστοιχούν σε 1 βαθμό Κελσίου. Για παράδειγμα, αν μετρήσουμε τάση 0,325 V τότε η θερμοκρασία είναι 32,5 βαθμοί Κελσίου, για τάση 0.335 έχουμε 33,5 βαθμούς κ.ο.κ.

Παρατίθεται το datasheet του, το οποίο μας δίνει πληροφορίες για τα τεχνικά χαρακτηριστικά του εν λόγω αισθητήρα: **http://www.ti.com/lit/ds/symlink/lm35.pdf**

3.3.2 Πυκνωτής ανάμεσα στο GND και στο Vcc

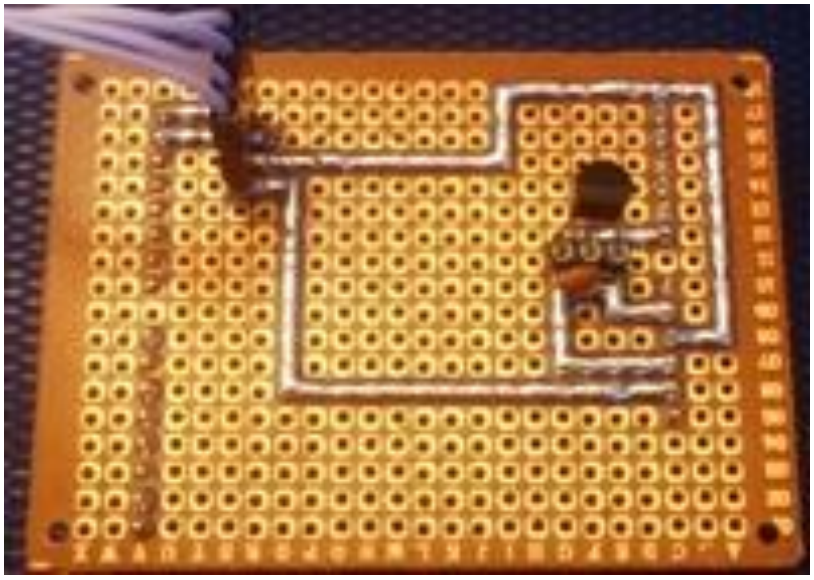
Μεταξύ των pin της γείωσης και της τροφοδοσίας, τοποθετούμε έναν πυκνωτή 104nF, ο οποίος μας χρησιμεύει για να απαλείψουμε τυχόν φαινόμενα θορύβου . Θεωρητικά στην παρούσα άσκηση θα μπορούσε και να μην υπάρχει, μιας και οι πιθανότητες να υπεισέρθει θόρυβος στο κύκλωμα είναι ελάχιστες. Σε επίπεδο παραγωγής όμως, είναι επιτακτική ανάγκη να τοποθετηθεί πυκνωτής.

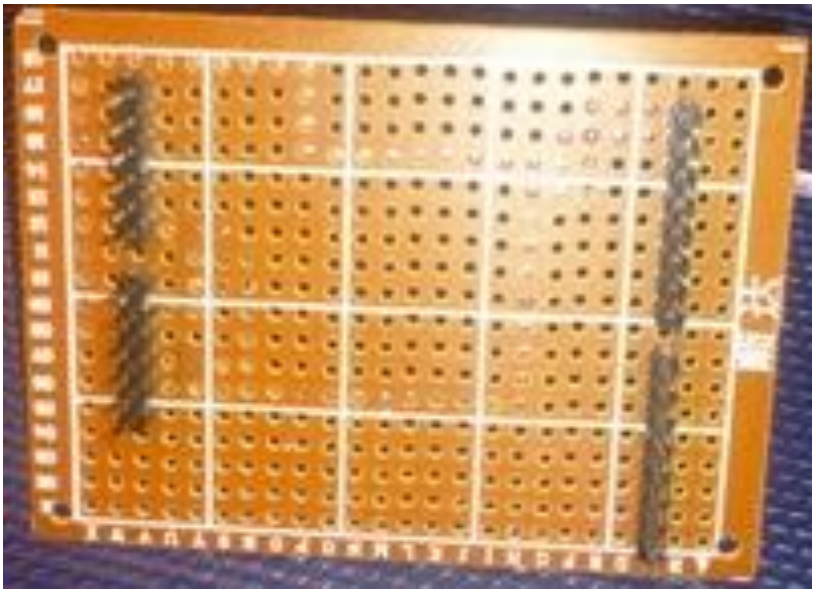
3.4 ΥΛΟΠΟΙΗΣΗ ΚΑΤΑΣΚΕΥΗΣ

3.4.1 Τοποθέτηση shields

Ένα πολύ σημαντικό στοιχείο του Arduino είναι πως δέχεται τα λεγόμενα shields. Τα shields είναι πλακέτες που εφαρμόζουν στην κορυφή του. Αφού συγκεκριμενοποιήσουμε το τί εφαρμογή θέλουμε να πραγματοποιήσουμε, μελετάμε καλά τις δυνατότητες της εκάστοτε πλακέτας Arduino, μετράμε τα θηλυκά pins, ώστε στο shield που θα εφαρμόσουμε να έχουμε τα αντίστοιχα αρσενικά pins, καθώς και τα στοιχεία που επιθυμούμε. Στην περίπτωσή μας έχουμε ένα αισθητήριο θερμοκρασίας LM35, έναν πυκνωτή 104nF και ένα Bluetooth Module.

Παρακάτω απεικονίζονται: Η πάνω όψη του Arduino , η κάτω όψη του Arduino , η πάνω όψη του Shield και η κάτω όψη του Shield.





3.5 ΚΩΔΙΚΑΣ ΚΑΤΑΣΚΕΥΗΣ

3.5.1 Προγραμματισμός Arduino

Για την λειτουργία της κατασκευής μας, είναι απαραίτητο να δημιουργήσουμε το κατάλληλο πρόγραμμα. Ο συλλογισμός είναι ο εξής: Ορίζουμε μια ακέραια μεταβλητή στην οποία αποθηκεύονται οι τιμές της θερμοκρασίας και αρχικοποιούμε το αντίστοιχο pin(έστω το 0).Έπειτα πρέπει να ανοίξουμε μια πόρτα για να γίνει η

σειριακή επικοινωνία. Τσε μια συνθήκη επανάληψης διαβάζουμε την τιμή του αισθητήρα την οποία μετατρέπουμε από αναλογική σε ψηφιακή και ορίζοντας ως καθυστέρηση ένα δευτερόλεπτο λαμβάνουμε ξανά τιμή.

3.5.2 Υλοποίηση

Προαπαιτούμενο για την υλοποίηση του προγράμματος μας, είναι να έχουμε εγκατεστημένο το περιβάλλον προγραμματισμού του Arduino, το Arduino IDE. Στο tab Εργαλεία, επιλέγουμε το Serial Monitor, με σκοπό να φαίνονται τα στοιχεία από τη σειριακή επικοινωνία. Τέλος πληκτρολογούμε τον κάτωθι κώδικα:

```
int temperature; //Αρχικοποίηση της μεταβλητής της θερμοκρασίας
int tempPin = 0; //Αρχικοποίηση του pin που διαβάζει τη θερμοκρασία
void setup()  // Μέθοδος setup, εκτελείται μία φορά κατά την εκκίνηση
{
  pinMode(tempPin,INPUT); //Ορισμός του tempPin ως είσοδος
  Serial.begin(9600); //Αρχικοποίηση σειριακής επικοινωνίας
}
void loop() //Μέθοδος που τρέχει επαναλαμβανόμενα
{
  temperature = analogRead(tempPin); //Διάβασμα αναλογικής τιμής του LM35 και καταχώρηση
  temperature = (5.0 * temperature * 100.0)/1024.0; //Χρήση εξίσωσης για την θερμοκρασία σε oC
    Serial.println((byte)temperature); // Αποτύπωση της θερμοκρασίας στο Serial Monitor
  delay(1000);   //Καθυστέρηση 1000ms μέχρι την επόμενη μέτρηση
  }
```

3.6 ΚΑΤΑΣΚΕΥΗ VI ΓΙΑ ΤΗΝ ΕΠΙΚΟΙΝΩΝΙΑ ΜΕ ΤΟ LABVIEW

3.6.1 Βήματα δημιουργίας

1. Εισάγουμε το Instrument I/O Assist στο Block Diagram από την Functions Palette →Express→Input→Instrument Assist.

Είναι αναγκαίο να δούμε ότι είμαστε στη σωστή θύρα COM. Το timeout(ms) που αναφέρει είναι ο χρόνος καθυστέρησης μέχρι να διαβάσει την επόμενη τιμή. Αν περάσει ο χρόνος και δεν έχει διαβάσει, εμφανίζει μήνυμα λάθους.

2. Εισάγουμε το Decimal String από την Functions Palette→ Programming→ String→ String/Number conversion→ Decimal String

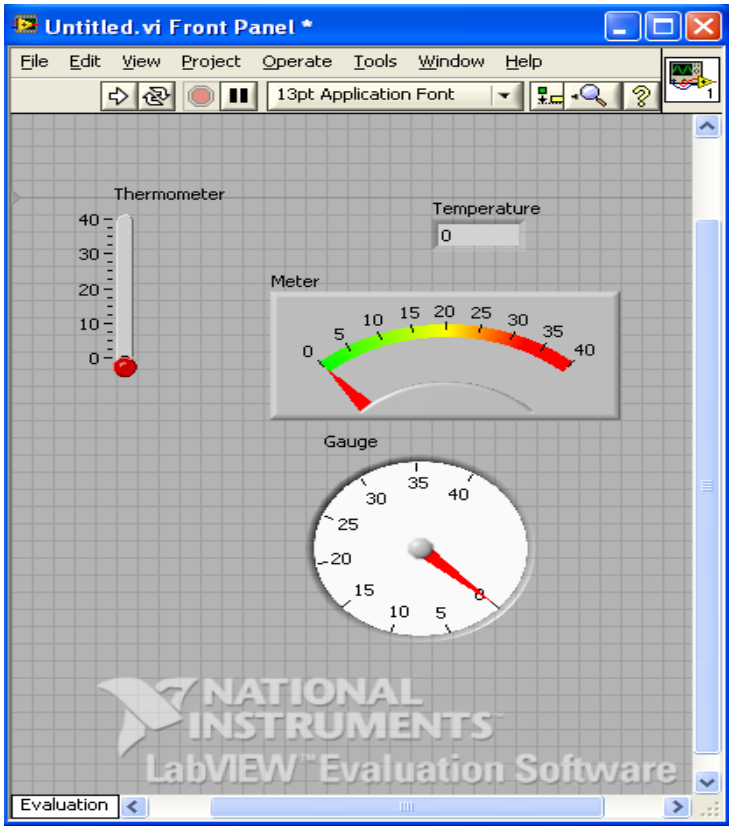
Αυτό το βήμα είναι απαραίτητο για να μετατρέπεται το String σε ακέραια τιμή.

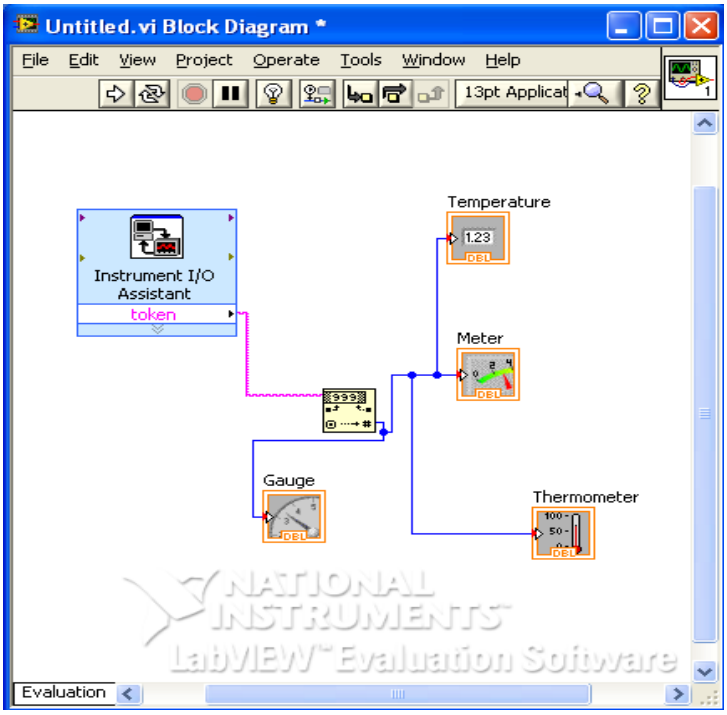
3. Εισάγουμε το Meter στο Front Panel από την Controls Palette→ Modern→ Numeric→ Meter.

4. Εισάγουμε το Thermometer στο Front Panel από την Controls Palette → Modern→ Numeric→ Thermometer.

5. Εισάγουμε το Gauge στο Front Panel από την Controls Palette → Modern→ Numeric→ Gauge.

Πραγματοποιώντας τα παραπάνω βήματα έχουμε το κάτωθι αποτέλεσμα για το front panel και το block diagram αντίστοιχα:


Untitled.vi Front Panel *
File Edit View Project Operate Tools Window Help
13pt Application Font
Thermometer
Temperature
Meter
Gauge
NATIONAL INSTRUMENTS
LabVIEW Evaluation Software
Evaluation


Untitled.vi Block Diagram *
File Edit View Project Operate Tools Window Help
Instrument I/O Assistant
token
Temperature
Meter
Gauge
Thermometer
NATIONAL INSTRUMENTS
LabVIEW Evaluation Software
Evaluation

ΚΕΦΑΛΑΙΟ 4 – Έξυπνο σπίτι με Arduino

4.1 ΠΕΡΙΛΗΨΗ ΕΦΑΡΜΟΓΗΣ

Στην εφαρμογή αυτή θα ασχοληθούμε με την μελέτη και κατασκευή μιας μακέτας ενός σπιτιού, το οποίο θα προσπαθήσουμε να μετατρέψουμε σε «Έξυπνο Σπίτι» με τη χρήση Arduino. Θα επιχειρήσουμε να κατανοήσουμε τόσο την έννοια και τον τρόπο λειτουργίας του «Έξυπνου Σπιτιού», όσο και την χρήση και τη λειτουργία του Arduino ως PID Controller.

4.2 Smart Home

Το έξυπνο σπίτι είναι ένα σύνολο αυτοματισμών, μέσω των οποίων ομαδοποιούνται και αυτοματοποιούνται οι λειτουργίες μιας κατοικίας ανάλογα με της ανάγκες του εκάστοτε ιδιοκτήτη. Τα συστήματα που χρησιμοποιούνται μπορούν να αφορούν και παραπάνω από μια χρήσεις. Το βασικό πλεονέκτημα αυτής της τεχνολογίας, είναι ότι δίνει τη δυνατότητα διαχείρισης και παρακολούθησης όλων των χώρων και εγκαταστάσεων μιας κατοικίας με οποιοδήποτε τρόπο επικοινωνίας. Το κόστος μετάβασης στο έξυπνο σπίτι δεν απαιτεί μεγάλο κόστος, αφού τα ηλεκτρονικά εξαρτήματα είναι φθηνά, ενώ δίνει τη δυνατότητα στο χρήστη να κατασκευάσει μόνος του τις λειτουργίες που χρειάζεται και να ρυθμίσει το κόστος ανάλογα με αυτές. Κάποιες από τις λειτουργίες αυτές είναι:

- Έλεγχος φωτισμού.
- Σύστημα συναγερμού
- Έλεγχος μέσω κινητού
- Σύστημα ποτίσματος

Το έξυπνο σπίτι μας προσφέρει αρκετά πλεονεκτήματα, έχει όμως και κάποια μειονεκτήματα:

- Ευάλωτο σε πιθανές δικτυακές επιθέσεις.
- Πιθανή δυσλειτουργία
- Πλήρη εξάρτηση από το ηλεκτρικό ρεύμα
- Απευθύνεται κυρίως σε εξοικειωμένο με την τεχνολογία κοινό

4.3 ΥΛΙΚΑ ΚΑΤΑΣΚΕΥΗΣ

4.3.1 Υλικά Μακέτας

1. Neofoam Χαρτόνι Μακέτας 70x100/5m
2. Πιστόλι Σιλικόνης
3. Ράβδοι σιλικόνης
4. Κοπίδι ακριβείας

4.3.2 BOM(Bill of Materials)

- **Κύκλωμα συναγερμού**

Mini breadboard x1

Pir Sensor module x1

Speaker x1

Red LED x1

Jumper Wires

- **Κύκλωμα ελέγχου κίνησης**

Mini breadboard x1

Pir Sensor module x1

White LED x1

Jumper Wires

- **Κύκλωμα ελέγχου φωτεινότητας**

Mini breadboard x1

Photo resistor x1

220Ohm resistor x1

White LED x1

Potentiometer x1

Jumper wires

4.3.3 Ανάλυση Υλικών

- **Pir Sensor module**

Το αισθητηριο PIR που χρησιμοποιειται στην εφαρμογη, δινει τη δυνατοτητα αισθησης της κινησης, και χρησιμοποιειται συνηθως για να εντοπισει αν ενας ανθρωπος εχει κινηθει εντος η εκτος του ευρους του αισθητηρα. Οι αισθητηρες αυτοι είναι μικροι σε μεγεθος, χαμηλου κοστους και καταναλωσης, ενώ ταυτοχρονα είναι ευκολοι στη χρηση και δεν φθειρονται. Για το λογο αυτό τους συναντάμε συχνα σε εφαρμογες και gadgets, ειτε για οικιακη χρηση αλλα ακομη και σε επιχειρησεις.

Το αισθητηριο αυτό είναι κατασκευασμενο από έναν αισθητηρα ο οποιος είναι ικανος να εντοπιζει την υπερυθρη ακτινοβολια. Τα παντα γυρω μας εκπεμπουν καποια επιπεδα ακτινοβολιας, και οσο περισσοτερη θερμοτητα εκπεμπει κατι, τοσο περισσοτερη ακτινοβολια εκπεμπεται. Ο αισθητηρας περιεχει λοιπον μεσα του 2 μερη. Αν το ένα από τα 2 εντοπισει περισσοτερη η λιγοτερη ακτινοβολια από το άλλο, θα αλλαξει και η εξοδος του αισθητηρα.

Το υπολοιπο κυκλωμα φροντιζει να μετατρεψει το σημα που περνει από τον αισθητηρα σε μια ψηφιακη εξοδο με σκοπο την επικοινωνια με τον οποιονδηποτε μικροελεγκτη. Στον αισθητηρα αυτό συνδεουμε στο Vcc pin τροφοδοσια 5-12V, 5 εν προκειμενω από το Arduino, στο GND pin συνδεουμε το GND του Arduino, και το out pin συνδεεται σε καποιο από τα pins του Arduino.

Στην εξοδο του εχουμε λογικο 1 (HIGH) όταν ανιχνευεται κινηση, και λογικο 0(LOW) όταν δεν ανιχνευεται κινηση. Η κινηση ανιχνευεται εως 6 μετρα μακρια, και σε γωνιες 110ο x 70ο .

Τελος, ο αισθητηρας αυτος περιλαμβανει 2 ρυθμιστες έναν για την ευαισθησια του εντοπισμου κινησης, και εναω για την ευαισθησια της χρονικης αποστασης μεταξυ των μετρησεων.

- **Photoresistor**

Το εξάρτημα αυτό, είναι ένας αντιστάτης ο οποίος μεταβάλει την τιμή της αντίστασης του ανάλογα με το φως το οποίο προσπίπτει στον αισθητήρα του. Όσο περισσότερο φως πέφτει πάνω του, τόσο μικραίνει η αντίσταση του, αφού παρουσιάζει αυτό που ονομάζουμε φωτοαγωγιμότητα. Είναι φτιαγμένος από έναν ημιαγωγό υψηλής αντίστασης, αφού στο σκοτάδι μπορεί να έχει αντίσταση έως και μερικά MOhms ενώ στο φως, φτάνει μέχρι και μερικές εκατοντάδες Ohms.

4.4 ΥΛΟΠΟΙΗΣΗ ΚΑΤΑΣΚΕΥΗΣ

4.4.1 Μακέτα Έξυπνου σπιτιού

Οι λειτουργίες που υλοποιήθηκαν για την μακέτα της εργασίας μας είναι:

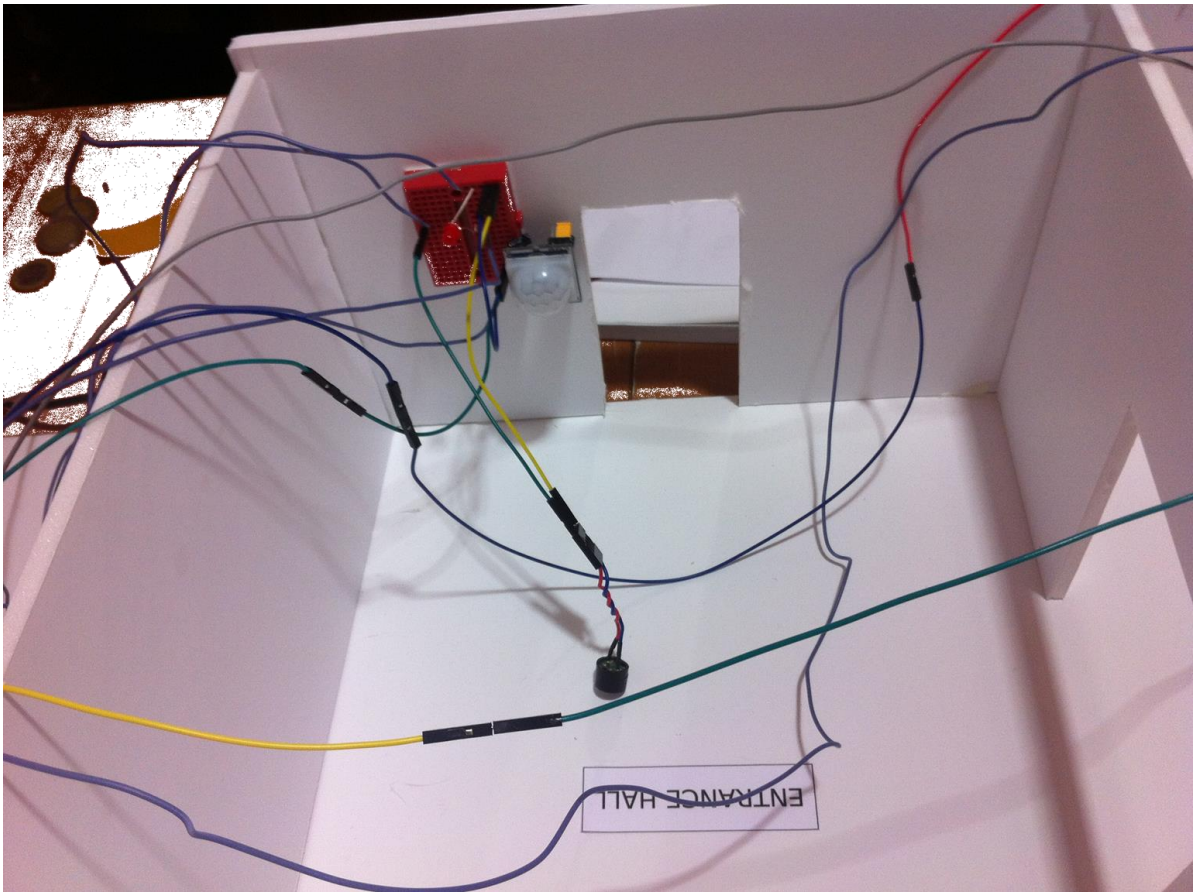
1. Έλεγχος Κίνησης – Συναγερμός
2. Έλεγχος Κίνησης – Φωτισμός
3. Έλεγχος Φωτισμού
4. Έλεγχος Θερμοκρασίας

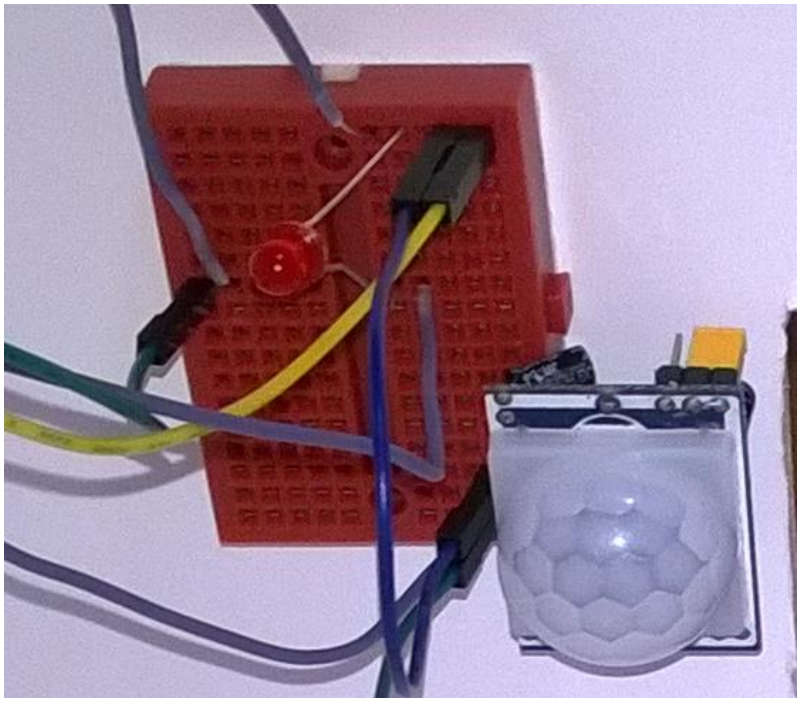
- **Κύκλωμα Συναγερμού**

Ο συναγερμός υλοποιήθηκε στην είσοδο του σπιτιού (Entrance Hall). Στο χώρο αυτό, μόλις ο αισθητήρας κίνησης αντιληφθεί κίνηση, ενεργοποιείται τόσο το σχετικό LED, οσο και το αντίστοιχο ηχείο. Τα υλικά που χρησιμοποιήθηκαν είναι:

- Mini Breadboard
- Pir Sensor Module
- Speaker
- Κόκκινο Led 5mm
- Jumper Wires


SPEAKER


ENTRANCE HALL

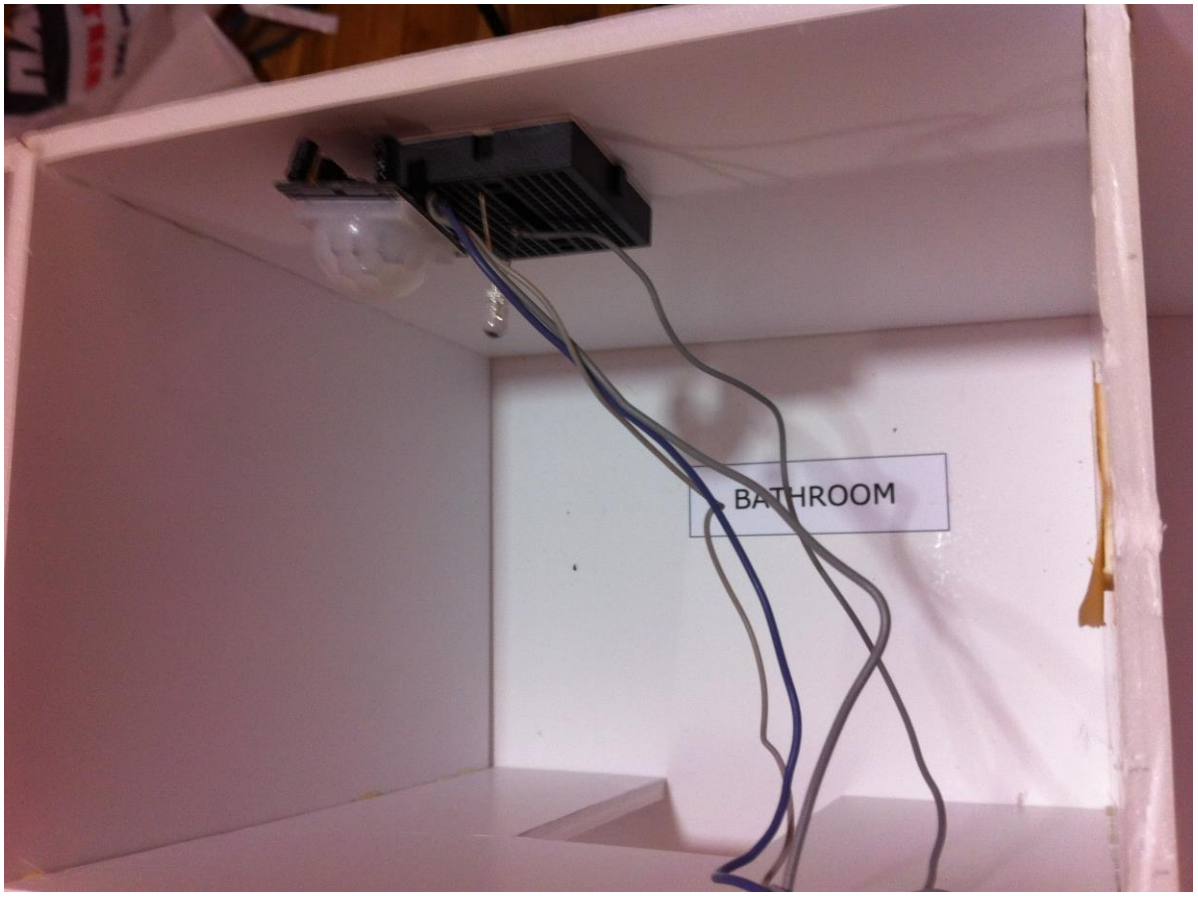


- **Κύκλωμα ελέγχου Κίνησης**

Η λειτουργία αυτή υλοποιήθηκε στο μπάνιο της μακέτας. Με την είσοδο κάποιου στον συγκεκριμένο χώρο, ενεργοποιείται ο φωτισμός. Όπως θα δούμε και παρακάτω, στον κώδικα της εργασίας, για να παραμείνει ο ανιχνευτής ενεργός για περισσότερο χρονικό διάστημα θα πρέπει να τροποποιήσουμε το delay του κώδικα. Για τις ανάγκες της μακέτας μας, η ενεργοποίηση του φωτός θα διαρκεί μικρό χρονικό διάστημα. Τα υλικά που χρησιμοποιήθηκαν είναι:

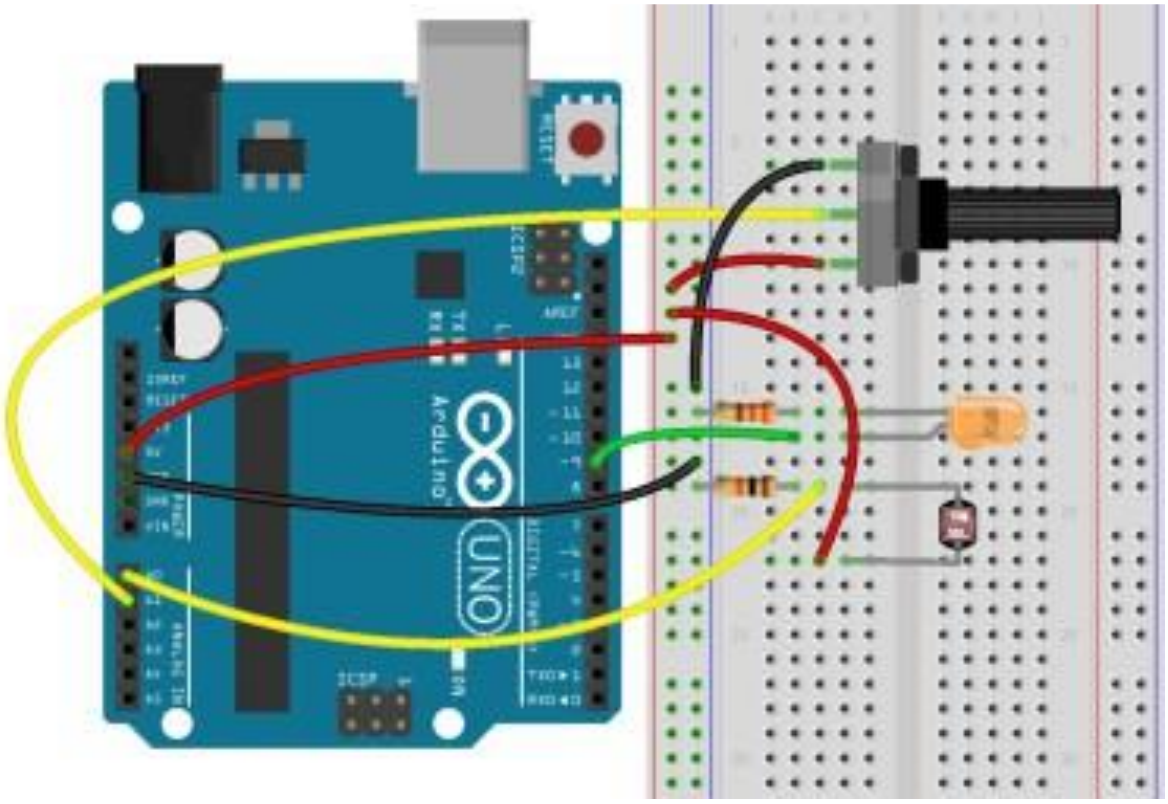
- Mini Breadboard
- Pir Sensor Module
- Λευκό Led 5mm
- Jumper Wires


W


BATHROOM

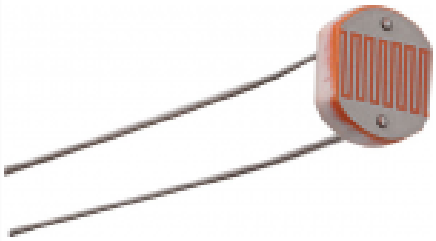
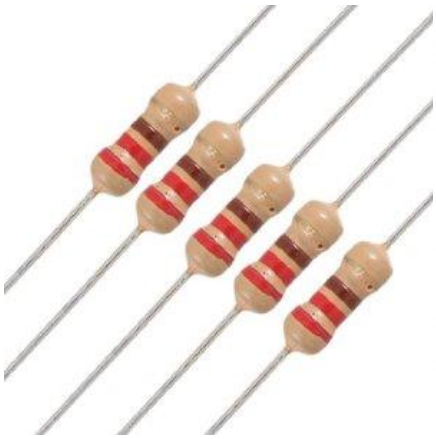
- **Κύκλωμα ελέγχου φωτεινότητας**

Η συγκεκριμένη λειτουργία έχει εφαρμοστεί στο υπνοδωμάτιο της μακέτας(Bedroom). Ο φωτοαισθητήρας μας δίνει τη δυνατότητα με το που σκοτεινιάσει να ενεργοποιούνται τα φώτα. Μόλις συμβεί αυτό, μπορούμε να διαχειριστούμε την ένταση του φωτός με τη χρήση ενός ποτενσιόμετρου.

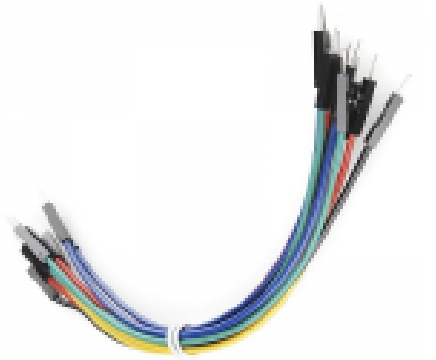


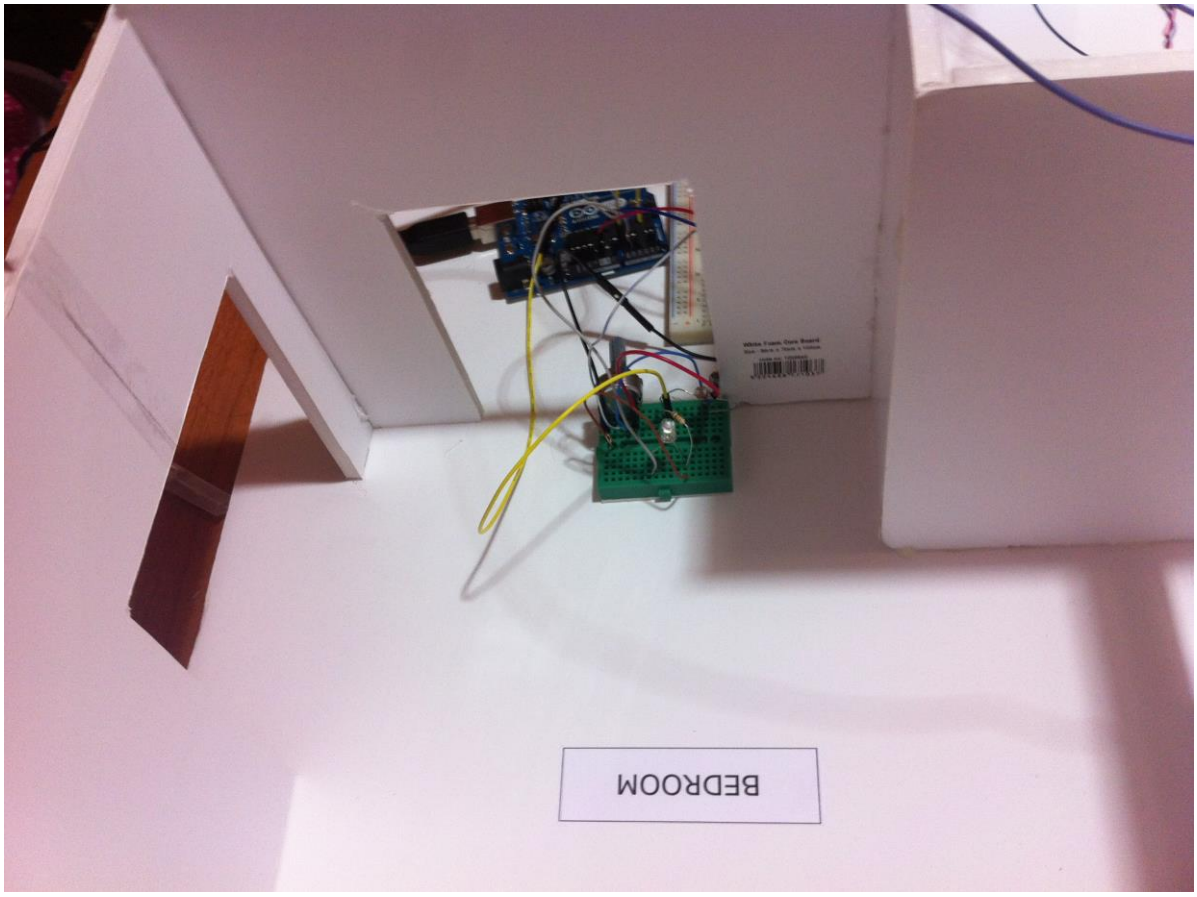
Τα υλικά που χρησιμοποιήθηκαν είναι:

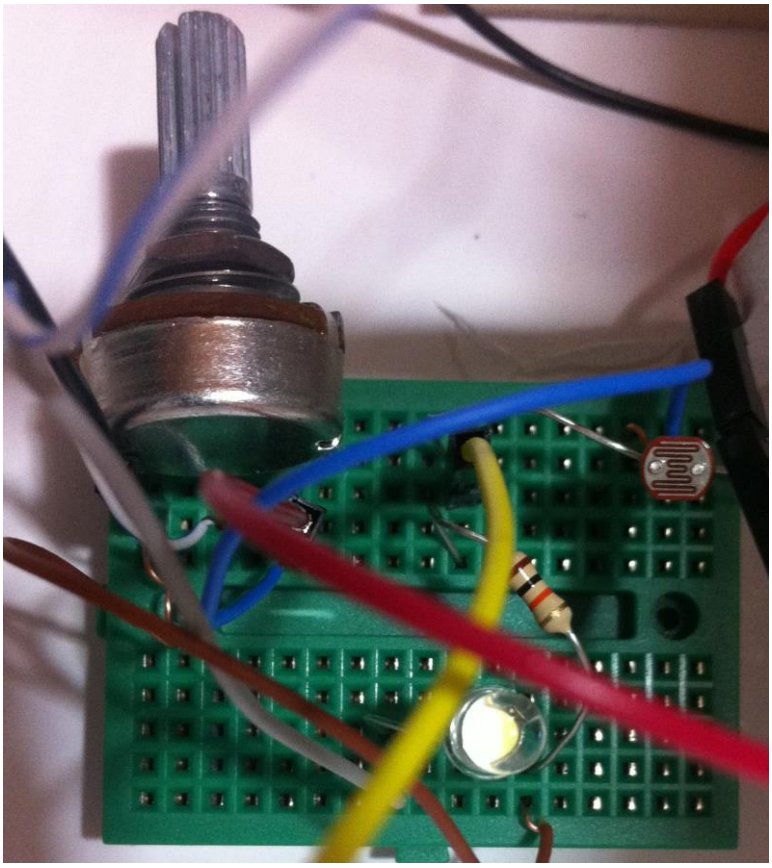
- Mini Breadboard
- Photo Resistor
- Resistors 220Ω
- Λευκό Led 5mm
- Ποτενσιόμετρο
- Jumper Wires





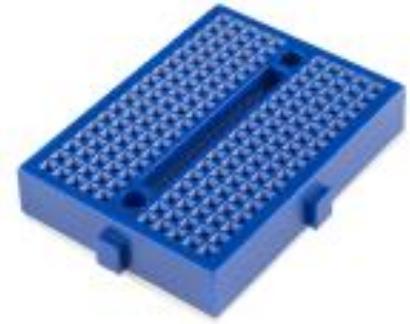

BEDROOM

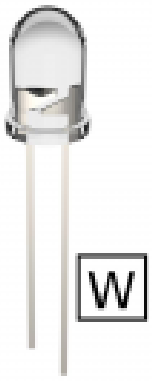


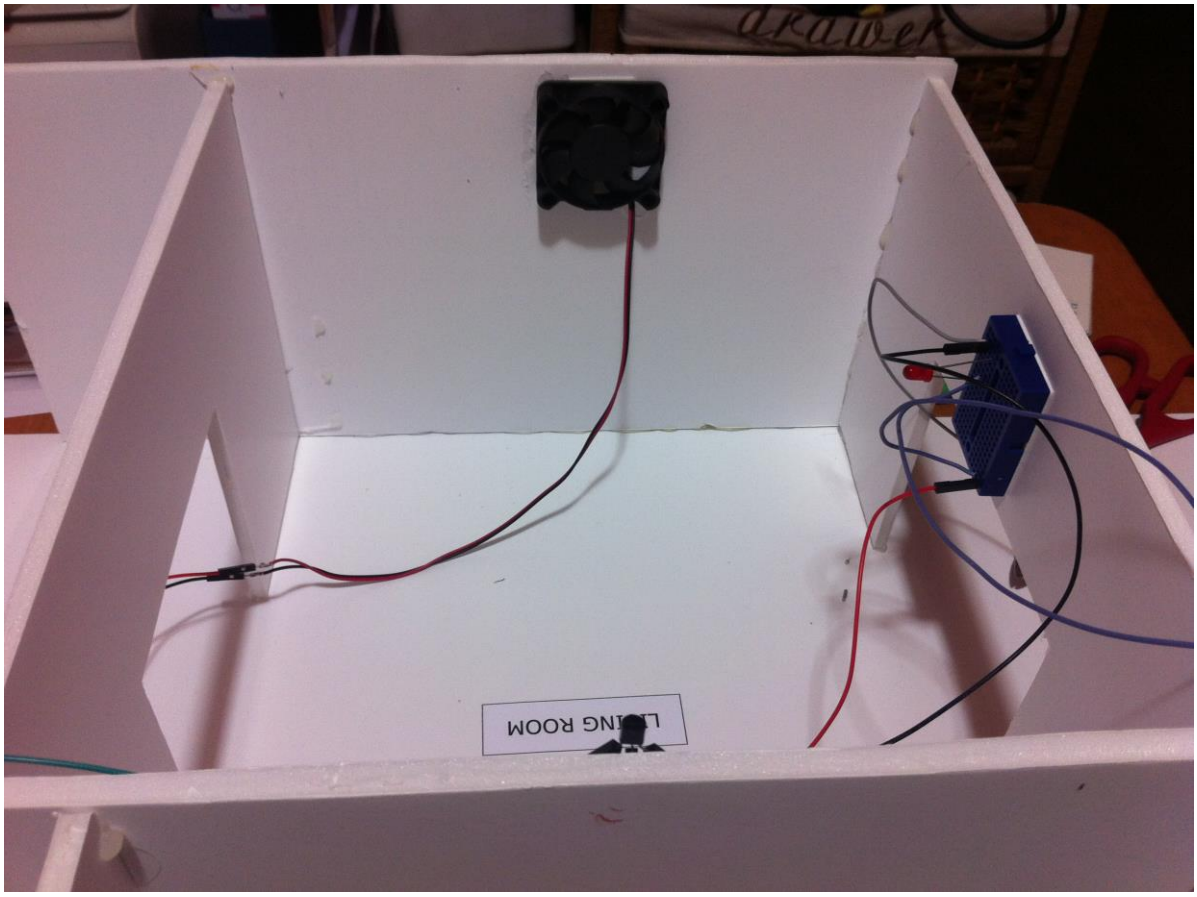
- **Κύκλωμα ελέγχου θερμοκρασίας**

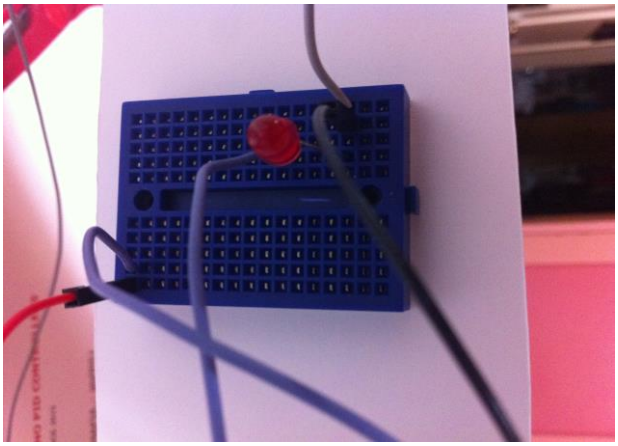
Η συγκεκριμένη λειτουργία υλοποιήθηκε στο σαλόνι της μακέτας (Living Room).Στον κώδικα μας θέτουμε ως μέγιστη τιμή θερμοκρασίας τους 30oC. Μόλις ο αισθητήρας φτάσει την επιθυμητή θερμοκρασία, ενεργοποιείται ο ανεμιστήρας(σύστημα ψύξης). Μόλις η θερμοκρασία στο χώρο πέσει ξανά, ο ανεμιστήρας απενεργοποιείται. Τα υλικά που χρησιμοποιήθηκαν είναι:

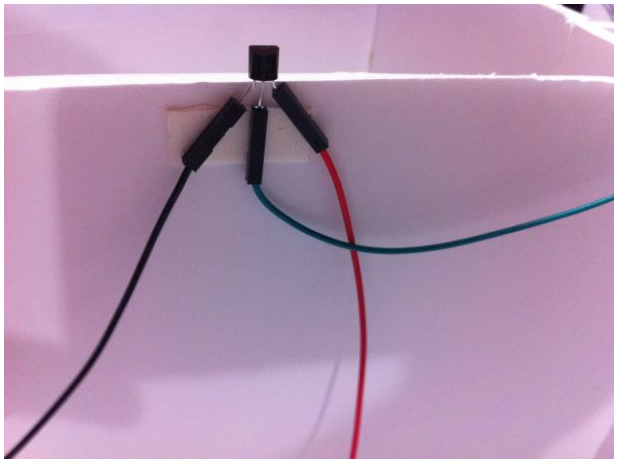
- Mini Breadboard
- Αισθητήρας Θερμοκρασίας LM35
- Ανεμιστήρας
- Λευκό Led 5mm
- Jumper Wires











4.5 ΚΩΔΙΚΑΣ ΚΑΤΑΣΚΕΥΗΣ

```
#include <PID_v1.h>

/*LedController Variables*/

const int photores = A0;                  // LDR input pin
const int pot = A1;                       // Potentiometer input pin
const int bedLed = 9;                     // LED output pin
double lightLevel;                        // Indirectly store the light level

/*Tuning parameters*/

float Kp = 0;                             // Proportional gain
float Ki = 10;                            // Integral gain
float Kd = 0;                             // Differential gain

/*Record the set point as well as the controller input and output*/

double Setpoint, Input, Output;

/*Create a controller that is linked to the specified Input, Ouput and Setpoint*/

PID myPID(&Input, &Output, &Setpoint, Kp, Ki, Kd, DIRECT);
const int sampleRate = 1;                 // Time interval of the PID control
const long serialPing = 500;              // How often data is recieved from the Arduino

unsigned long now = 0;                    // Store the time that has elapsed
unsigned long lastMessage = 0;            // The last time that data was recieved


/*Temperature Variables*/

float tempC;                              // create variable to store the temperature in.
```

```
const int tempPin = A3;            // Attach vout to analog pin 3.
int led3 = 12;                     // attach led to pin 12.
int fan = 5;                       // attach base of transistor to digital pin 5.
unsigned long previousMillis = 0;   // will store last time temperature was updated
const long interval = 3000;        // interval at which to measure temperature
(milliseconds)

/*Alarm Variables*/
int ledPin = 13;                   // choose the pin for the LED
int inputPin = 2;                  // choose the input pin (for PIR sensor)
int pirState = LOW;                // we start, assuming no motion detected
int val = 0;                       // variable for reading the pin status
int pinSpeaker = 11;               //Set up a speaker on a PWM pin (digital 9, 10, or 11)


/*Led Variables*/

int led2 = 7;                      // choose the pin for the LED
int inPin = 8;                     // choose the input pin (for PIR sensor)
int pir2State = LOW;               // we start, assuming no motion detected
int val2 = 0;                      // variable for reading the pin status

void setup()                       // Will execute once at the start of the code.
{
  Serial.begin(9600);              // opens serial port, sets data rate to 9600 bps

/*Alarm Setup*/
  pinMode(ledPin, OUTPUT);         // declare LED as output
  pinMode(inputPin, INPUT);        // declare sensor as input
  pinMode(pinSpeaker, OUTPUT);     // declare speaker as output

/*Led Setup*/
  pinMode(led2, OUTPUT);           // declare LED as output
  pinMode(inPin, INPUT);           // declare sensor as input

/*Temperature Setup*/
  pinMode (led3, OUTPUT);          // sets the led pin 12 up as an output.
  pinMode (fan, OUTPUT);           // sets the fan1 pin 5 up as an output.

/*LedController Setup*/
  lightLevel = analogRead(photores);          // Read the set point
  /*Arduino has an analogueRead() resolution of 0-1023 and an analogueWrite()
resolution of 0-255*/
   Input = map(lightLevel, 0, 1023, 0, 255);          // Scale the input
   Setpoint = map(analogRead(pot), 0, 1023, 0, 255);  // Scale the set point
   myPID.SetMode(AUTOMATIC);                          // Turn on the PID control
   myPID.SetSampleTime(sampleRate);      // Assign the sample rate of the control
   //Serial.println("Begin");           // Let the user know that the set up s complete
```

```
  lastMessage = millis();              // Serial data will be recieved relative to this first
point
}

void loop()                            // code here will continue to replay nutil powered off.
{
        functionAlarm();
        functionLed();
        functionTemperature();
        functionLedController();
}

void functionTemperature(){
 /*Temperature Loop*/
 tempC = analogRead(tempPin);          // read the analog value from the lm35
sensor.
 tempC = (5.0 * tempC * 100.0)/1024.0; // convert the analog input to temperature
in centigrade.
 unsigned long currentMillis = millis();
 //Serial.println((byte)tempC);        // send the data to the computer.
 if (currentMillis - previousMillis >= interval) {
   previousMillis = currentMillis;

   if (tempC > 30)                     // creates bool expression for analyzation. if it
evaluates to true,
   {                                   // the body of the if statement will execute.
     digitalWrite (led3, HIGH);        // turns on led.
     digitalWrite (fan, HIGH);         // turns on fan1.
   }
   else                                // if the if equation evaluates to false the else statement
will execute.
   {
     digitalWrite (led3, LOW);         // turns off led.
     digitalWrite (fan, LOW);          // turns off fan1.
   }
 }
}

void functionAlarm(){
 /*Alarm Loop*/
 val = digitalRead(inputPin);          // read input value
 if (val == HIGH) {                    // check if the input is HIGH
   digitalWrite(ledPin, HIGH);         // turn LED ON
   playTone(300, 160);
   //delay(150);
   if (pirState == LOW) {              // we have just turned on
     //Serial.println("Motion detected!");
     pirState = HIGH;                  // We only want to print on the output change, not
state
   }
```

```
  } else {
      digitalWrite(ledPin, LOW);         // turn LED OFF
      playTone(0, 0);
     // delay(300);
      if (pirState == HIGH){            // we have just turned off
        //Serial.println("Motion ended!");
        pirState = LOW;                 // We only want to print on the output change, not
state
      }
    }
}

void playTone(long duration, int freq) { //Duration in mSecs, frequency in hertz
    duration *= 1000;
    int period = (1.0 / freq) * 1000000;
    long elapsed_time = 0;
    while (elapsed_time < duration) {
        digitalWrite(pinSpeaker,HIGH);
        delayMicroseconds(period / 2);
        digitalWrite(pinSpeaker, LOW);
        delayMicroseconds(period / 2);
        elapsed_time += (period);
    }
}

void functionLed(){
  /*Led Loop*/
    val2 = digitalRead(inPin);          // read input value
    if (val2 == HIGH) {                 // check if the input is HIGH
    digitalWrite(led2, HIGH);           // turn LED ON
    if (pir2State == LOW) {             // we have just turned on
      pir2State = HIGH;                 // We only want to print on the output change, not
state
    }
  } else {
      digitalWrite(led2, LOW);          // turn LED OFF
      if (pir2State == HIGH){           // we have just turned off
        pir2State = LOW;                // We only want to print on the output change, not
state
      }
    }
}

void functionLedController(){
  /*Photoresistor Loop*/
  Setpoint = map(analogRead(pot), 0, 1023, 0, 255);  // Continue to read and scale
the set point
  lightLevel = analogRead(photores);                 // Read the light level
  Input = map(lightLevel, 0, 900, 0, 255);           // Scale the input to the PID
```

```
  myPID.Compute();                       // Calculates the PID output at a specified
sample time
  analogWrite(bedLed, Output);           // Power the LED
  now = millis();                        // Keep track of the elapsed time
  if(now - lastMessage > serialPing)     // If enough time has passed send
data
  {
    Serial.print("Setpoint = ");
    Serial.print(Setpoint);
    Serial.print(" Input = ");
    Serial.print(Input);
    Serial.print(" Output = ");
    Serial.print(Output);
    Serial.print("\n");
/*The tuning parameters can be retrieved by the Arduino from the serial monitor:
0,0.5,0 set Ki to 0.5. - Commas are ignored by the Serial.parseFloat() command*/
    if (Serial.available() > 0)
    {
      for (int x = 0; x < 4; x++)
      {
        switch(x)
        {
          case 0:
            Kp = Serial.parseFloat();
          break;
          case 1:
            Ki = Serial.parseFloat();
          break;
          case 2:
            Kd = Serial.parseFloat();
          break;
          case 3:
          for (int y = Serial.available(); y == 0; y--)
          {
            Serial.read();
          }
          break;
        }
      }
      Serial.print(" Kp,Ki,Kd = ");     // Display the new parameters
      Serial.print(Kp);
      Serial.print(" ,");
      Serial.print(Ki);
      Serial.print(" ,");
      Serial.print(Kd);
      myPID.SetTunings(Kp, Ki, Kd);     // Set the tuning of the PID loop
    }
    lastMessage = now;                  // Reference the next serial communication to this
point
  }
```

ΚΕΦΑΛΑΙΟ 5 – Κατασκευή αυτόματου ποτιστικού συστήματος

5.1 ΠΕΡΙΛΗΨΗ ΕΦΑΡΜΟΓΗΣ

Στην εφαρμογή αυτή θα ασχοληθούμε με την κατασκευή ενός αυτόνομου ποτιστικού συστήματος χρησιμοποιώντας τον μικροελεγκτή Arduino ως PLC. Το σύστημα αυτό θα περιλαμβάνει 3 αυτόνομα δοχεία με χώμα, τις υποτιθέμενες γλάστρες. Κάθε δοχείο θα περιέχει ένα αισθητήρα υγρασίας ο οποίος θα παρακολουθεί την υγρασία μέσα στην εκάστοτε γλάστρα. Η κάθε γλάστρα θα διαθέτει επίσης και την δική της βαλβίδα ποτίσματος, ένα ειδικό ρελέ γι' αυτή τη δουλειά(solenoid valve). Παρακάτω θα κάνουμε μια εκτενή αναφορά στο τι είναι, και πως δουλεύει ένα Arduino και ένα PLC, αλλά και στο πως θα προγραμματίσουμε το Arduino ως τέτοιο.

5.2 PLC (Programmable Logic Controller)

Το PLC είναι ένας προγραμματιζόμενος λογικός ελεγκτής (Programmable Logic Controllers). Είναι δομικό στοιχείο των αυτοματισμών στην σημερινή βιομηχανία και

προσφέρει μεγάλη ευκολία προγραμματισμού κα αποσφαλμάτωσης καθώς και έναν εξαιρετικά υψηλό βαθμό αξιοπιστίας. Είναι στην ουσία ένα σύστημα ελέγχου το οποίο παρακολουθεί συνεχώς τα δεδομένα των εισόδων του και λαμβάνει αποφάσεις για τις εξόδους του με βάση ένα εξειδικευμένο για την εκάστοτε εργασία πρόγραμμα. Τα PLC αποτελούνται από:

- Εισόδους – Εξόδους
- Μνήμη
- Επεξεργαστή

Ο επεξεργαστής είναι η “καρδία” του PLC με την έννοια ότι είναι ο λόγος που μπορούμε να προγραμματίσουμε την συσκευή. Διαβάζει τις εισόδους και στη συνέχεια θέτει τις ανάλογες τιμές στις εξόδους ανάλογα με τις εντολές που διαβάζει από την μνήμη.

Υπάρχουν 4 βασικά βήματα για την λειτουργία των PLC:

1) Έλεγχος εισόδων: Το PLC διαβάζει την κατάσταση όλων των συσκευών που είναι συνδεδεμένες στις εισόδους του.

2) Εκτέλεση: Εκτελείται το πρόγραμμα που έχει ορίσει ο χρήστης.

3) Ενεργοποίηση ή απενεργοποίηση των συσκευών εξόδου ανάλογα με το πρόγραμμα.

4) Επικοινωνία με τερματικά η διαγνωστικά κα.

Τα βήματα αυτά βρίσκονται σε ένα βρόχο και επαναλαμβάνονται συνεχώς.

Ο ίδιος ο προγραμματισμός των PLC γίνεται με συμβολική γλώσσα σε αντίθεση με τις κλασσικές γλώσσες προγραμματισμού (π.χ C, C++, java κτλ.). Η γλώσσα αυτή αποτελείται από διαγράμματα ή απλές εντολές. Το πρόγραμμα αποτελείται από μια

σειρά εντολών που λύνουν έναν αλγόριθμο με σκοπό μα υλοποιηθεί ένα σύστημα αυτοματισμού.

Η γλώσσα προγραμματισμού εξαρτάται από την εσωτερική αρχιτεκτονική του μικροεπεξεργαστή. Άρα κάθε PLC που έχει διαφορετικό επεξεργαστή θα απαιτούσε από τον προγραμματιστή να έχει βαθιά γνώση ως προς την αρχιτεκτονική για να δουλέψει πάνω σε αυτό. Αυτό είναι πρακτικά και λειτουργικά αδύνατο διότι ο προγραμματισμός των PLC θα πρέπει να γίνεται με έναν τρόπο βατό για όλους τους μικροεπεξεργαστές.

Έτσι καταλήξαμε σε 3 κατηγορίες γλωσσών.

1. Ladder Diagram(LAD)
2. Control System Flowchart (C.S.F)
3. Statement list(STL)

Το **Ladder διάγραμμα** χρησιμοποιεί τα αμερικάνικα σύμβολα των επαφών. Το C.S.F χρησιμοποιεί τα λογικά σύμβολα των πυλών και της άλγεβρας Boole και το STL μοιάζει με την γλώσσα BASIC.

Η διαδικασία του προγραμματισμού ενός PLC απαιτεί κάποια βήματα πριν την υλοποίηση του κώδικα. Αυτά είναι:

1. Διατύπωση του προβλήματος
2. Καθορισμός στοιχείων εισόδου – εξόδου
3. Κατασκευή πίνακα αληθείας.
4. Εξαγωγή λογικών εξισώσεων
5. Απλοποίηση με Karnaugh
6. Σχεδίαση διαγράμματος Ladder
7. Κατασκευή προγράμματος και εισαγωγή στην μνήμη του PLC

Εσωτερικά το PLC διαβάζει την κάθε εντολή του προγράμματος μας και εξάγει τα αντίστοιχα αποτελέσματα. Η δομή των εντολών είναι πολύ απλή. Ουσιαστικά αποτελείται από λογικές πράξεις(OR, XOR, AND κτλ.). Όταν τελειώσει η σειρά των

εντολών το PLC ξαναγυρνάει στην πρώτη εντολή του προγράμματος και εκτελείται πάλι σειριακά. Αυτό που διαφοροποιεί την πορεία του προγράμματος είναι η τιμή των εισόδων ή και των εξόδων. Αυτό επιτυγχάνεται μέσω του καταχωρητή RR(Result Register) οπού αποθηκεύονται οι καταστάσεις των εισόδων και των εξόδων. Όλες οι λογικές πράξεις γίνονται ανάμεσα στον RR και σε κάποια είσοδο ή έξοδο. Το αποτέλεσμα αποθηκεύεται στον RR ή σε κάποια έξοδο.

Στην δική μας εργασία, θα προγραμματίσουμε τον μικροελεγκτή Arduino με την χρήση της γλώσσας προγραμματισμού του Arduino, αλλά με χρήση βρόγχων if, δηλαδή με την λογική ενός διαγράμματος ladder.

5.2.1 Ladder Διάγραμμα

Όπως αναφέραμε και παραπάνω το ladder είναι μια γραφική γλώσσα, με την οποία μπορούμε να μετατρέψουμε ένα ηλεκτρολογικό σχέδιο σε γλώσσα κατανοητή από τα PLC. Χρησιμοποιούμε γραφικά εργαλεία και έτσι δομείται ένα πρόγραμμα που ακολουθεί το ηλεκτρολογικό σχέδιο που θέλουμε να υλοποιήσουμε.

Ένα πρόγραμμα Ladder αποτελείται από rungs, δηλαδή ένα σύνολο από γραφικές εντολές οι οποίες είναι τοποθετημένες ανάμεσα σε δύο σημεία (την τροφοδοσία και την γραμμή επιστροφής).

Οι πιθανές εντολές των rung είναι:

- Είσοδοι και έξοδοι όπως διακόπτες μπουτόν και αισθητήρια
- Λειτουργίες του PLC όπως μετρητές και λογικές πράξεις(OR, AND κτλ.), πράξεις συγκρίσεως και αριθμητικές λειτουργίες

Αυτά τα εργαλεία συνδυάζονται για οδηγηθούν στο τέλος σε μία ή περισσότερες εξόδους ή στοιχεία.

- **Κανόνες Διαγραμμάτων Ladder**

1.Διαβάζεται από αριστερά προς τα δεξιά και πάνω προς τα κάτω.

2. Οι μπάρες αντιπροσωπεύουν το δυναμικό του κυκλώματος, δηλαδή AC ή DC και είναι από 6-480V.

3. Ηλεκτρικές συσκευές σχεδιάζονται σε ηρεμία.

4. Επαφές που συνδέονται με συσκευές έχουν τον ίδιο αριθμό ή γράμμα με την συσκευή που τα ελέγχει.

5. Οι συσκευές με λειτουργία stop, συρματώνονται στη σειρά.

6. Οι συσκευές με λειτουργία start, συρματώνονται παράλληλα.

5.3 Πλεονεκτήματα PLC σε σχέση με αυτοματισμό

- Έχουν γενική χρήση
- Οι επαφές είναι στοιχεία μνήμης και όχι φυσικά στοιχεία
- Μπορούμε να αλλάξουμε τον αυτοματισμό σε όποιο στάδιο θέλουμε
- Εύκολος οπτικός έλεγχος μέσω τον LED που υπάρχουν
- Δεν είναι σημαντική η συντήρηση.
- Μπορούμε να συνδέσουμε οθόνες, εκτυπωτές κτλ.
- Δυνατότητα αντιγραφής εφαρμογής
- Καταλαμβάνουν πολύ μικρό χώρο

5.4 ΥΛΙΚΑ ΚΑΤΑΣΚΕΥΗΣ

5.4.1 Bill of Materials(BOM)

- Arduino Uno clone x1

- Soil moisture sensor modules x3

- Solenoid water valves mini x3

- Αντλία νερού DC 3-6V brushless motor submersible x1

- 9V battery x1

5.4.2 Sensors

- **Temperature thermistor sensor module KY-028**

Είναι ένα αισθητήριο θερμοκρασίας με εμβέλεια από τους -55 έως +125 βαθμούς κελσίου. Έχει δύο LED τα οποία δείχνουν ότι το αισθητήριο παίρνει τάση και άρα μετρήσεις και το άλλο ανιχνεύει αν υπάρχει μαγνητικό πεδίο.

- **Soil moisture sensor**

Πρόκειται για έναν αισθητήρα υγρασίας. Ο αισθητήρας αυτός τοποθετείται στο χώμα της εκάστοτε γλάστρας. Λειτουργεί μετρώντας την αντίσταση μεταξύ των δυο άκρων του όταν αυτά βρίσκονται μέσα στο χώμα, με λίγα λόγια, εξαρτάται από την αγωγιμότητα του χώματος.

- **Αντλία νερού**

Για την εργασία μας θα χρησιμοποιήσουμε μια αντλία νερού DC 3-6V brushless motor submersible water pump.

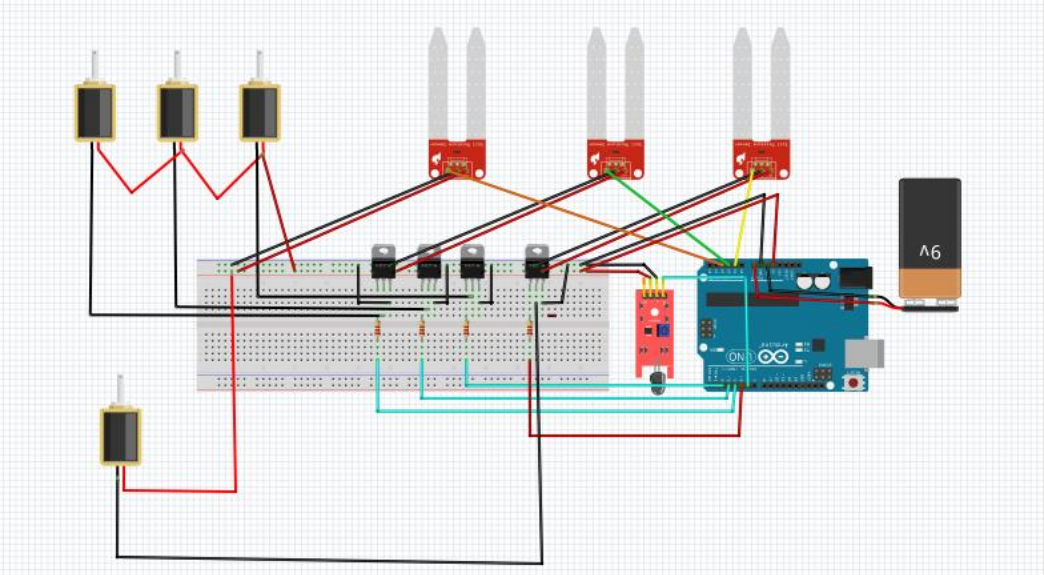
5.4.3 Υπόλοιπα Υλικά

- Κομμένο μπουκάλι νερού
- Ξύλινη επιφάνεια ως έδρα για την κατασκευή
- 3 πλαστικά ποτήρια που προσομοιώνουν τις γλάστρες
- Λάστιχο αυτόματου ποτίσματος

5.5 ΥΛΟΠΟΙΗΣΗ ΚΑΤΑΣΚΕΥΗΣ

5.5.1 ΣΥΝΔΕΣΜΟΛΟΓΙΑ FRITZING

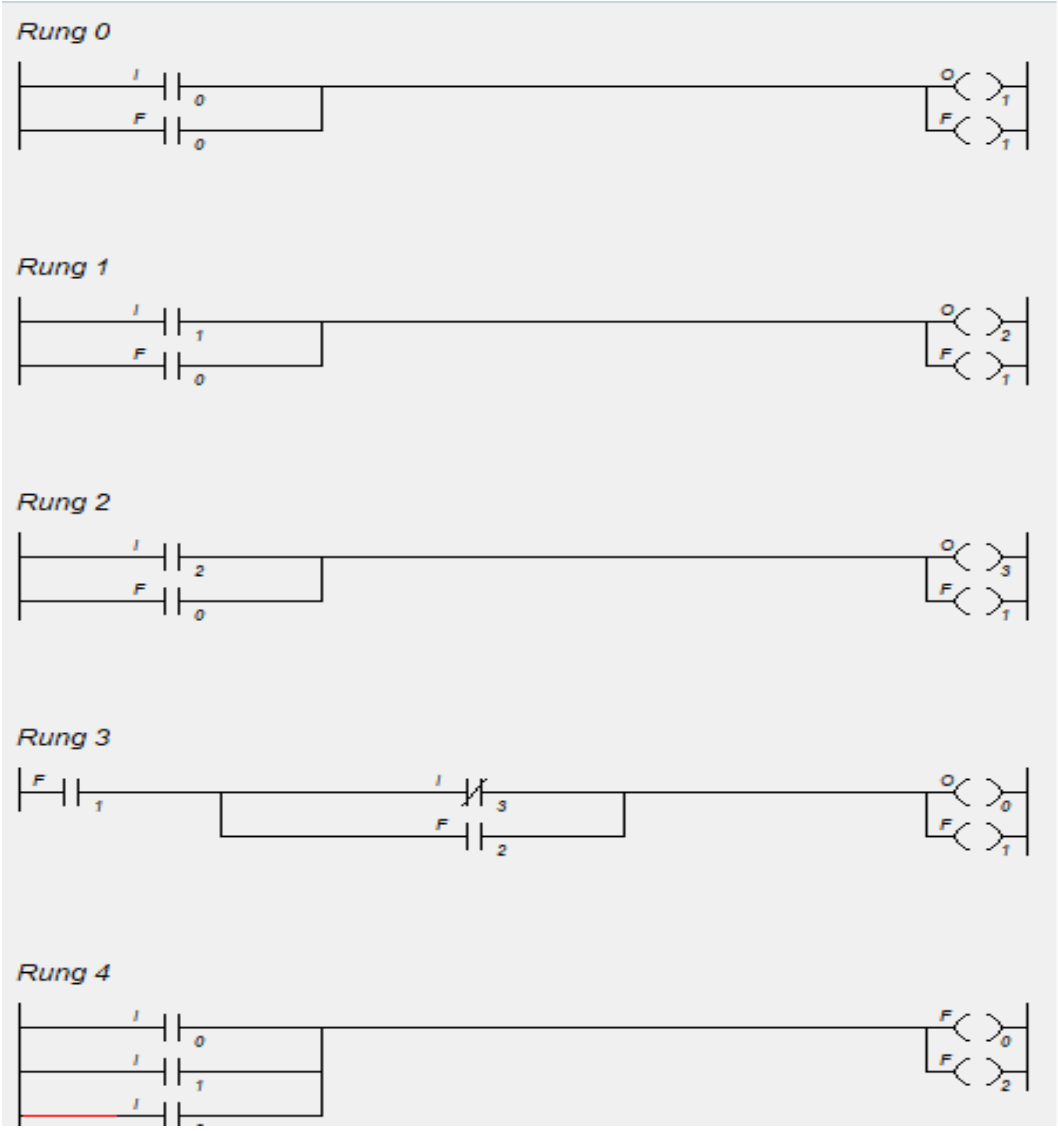
Το Fritzing είναι ένα open-source hardware initiative το οποίο προσφέρει στους χρήστες καταγραφή, διαμοιρασμό καθώς και εκμάθηση για ηλεκτρονικές κατασκευές. Για την εφαρμογή μας χρησιμοποιήσαμε το πρόγραμμα αυτό με σκοπό την κατασκευή ενός γραφικού σχηματικού διαγράμματος για την εργασία μας, και μας δόθηκε το παρακάτω αποτέλεσμα:



5.5.2 Το σύστημα

Στα πλαίσια της εργασίας θα χρησιμοποιήσουμε το Arduino σαν ένα PLC. Χρησιμοποιώντας τη γλώσσα του Arduino, δηλαδή μια παραλλαγή της γλώσσας C, και με την χρήση βρόγχων if, θα προσομοιώσουμε την Ladder λογική του προγράμματος. Το αρχικό μας βήμα είναι η κατασκευή του Ladder διαγράμματος για το ποτιστικό σύστημα.

Στο διάγραμμα αυτό παρατηρούμε τις εισόδους και τις εξόδους του συστήματος. Οι είσοδοι είναι ο αισθητήρας θερμοκρασίας, και οι αισθητήρες υγρασίας, ενώ οι έξοδοι είναι η τρόμπα νερού καθώς και οι βαλβίδες ποτίσματος.



Ας αναλύσουμε κάθε rung:

Rung 0: Η είσοδος 0 είναι ο αισθητήρας υγρασίας της 1ης γλάστρας και αντίστοιχα η έξοδος 1 είναι η βαλβίδα του ποτίσματος. Τα F1, F0 θα το αναλύσουμε παρακάτω

Rung 1: Ισχύουν τα παραπάνω για την 2η γλάστρα.

Rung 2: Ισχύουν τα παραπάνω για την 3η γλάστρα.

Rung 3: Εδώ έχουμε μια συνθήκη OR μεταξύ του αισθητήρα θερμοκρασίας και του F2 που θα αναλύσουμε παρακάτω. Η συνθήκη αυτή συνδέεται σε σειρά με το F1 οπού το αναφέραμε προηγουμένως. Ουσιαστικά το F1 έχει την τιμή ‘1’ όταν ενεργοποιείται οποιοσδήποτε αισθητήρας υγρασίας. Αποτέλεσμα είναι να ανοίγει η τρόμπα νερού (O3) και να παίρνει την τιμή ‘0’ το F1.

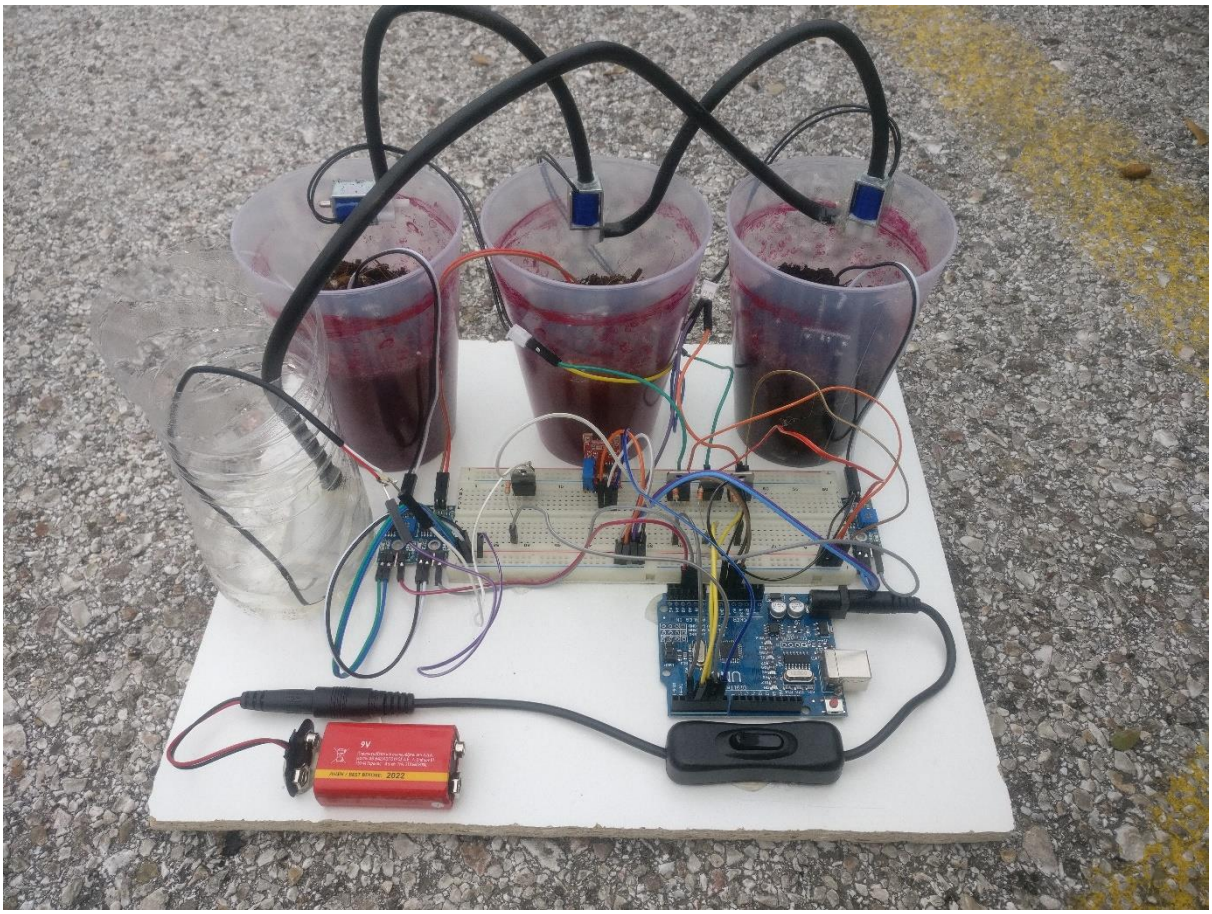
Rung 4: Για να γίνει το έκτακτο πότισμα υπάρχουν δύο συνθήκες. Είτε η θερμοκρασία να είναι πάνω από το όριο, είτε η υγρασία να έχει φτάσει στο κατώτερο όριο αντίστοιχα. Για τον λόγο αυτό εάν η υγρασία του χώματος έστω και μιας γλάστρας έχει πέσει σε επικίνδυνα επίπεδα τότε δίνουμε την τιμή ‘1’ στο F0 που σημαίνει ότι ανοίγουν και οι τρείς βαλβίδες ποτίσματος αλλά ταυτόχρονα δίνουμε και την τιμή ‘1’ και στο F2 που σημαίνει ότι ακόμα και αν ο αισθητήρας θερμοκρασίας δεν λειτουργεί, τότε οι γλάστρες θα ποτιστούν(αφού η υγρασία του χώματος είναι σε χαμηλά επίπεδα).

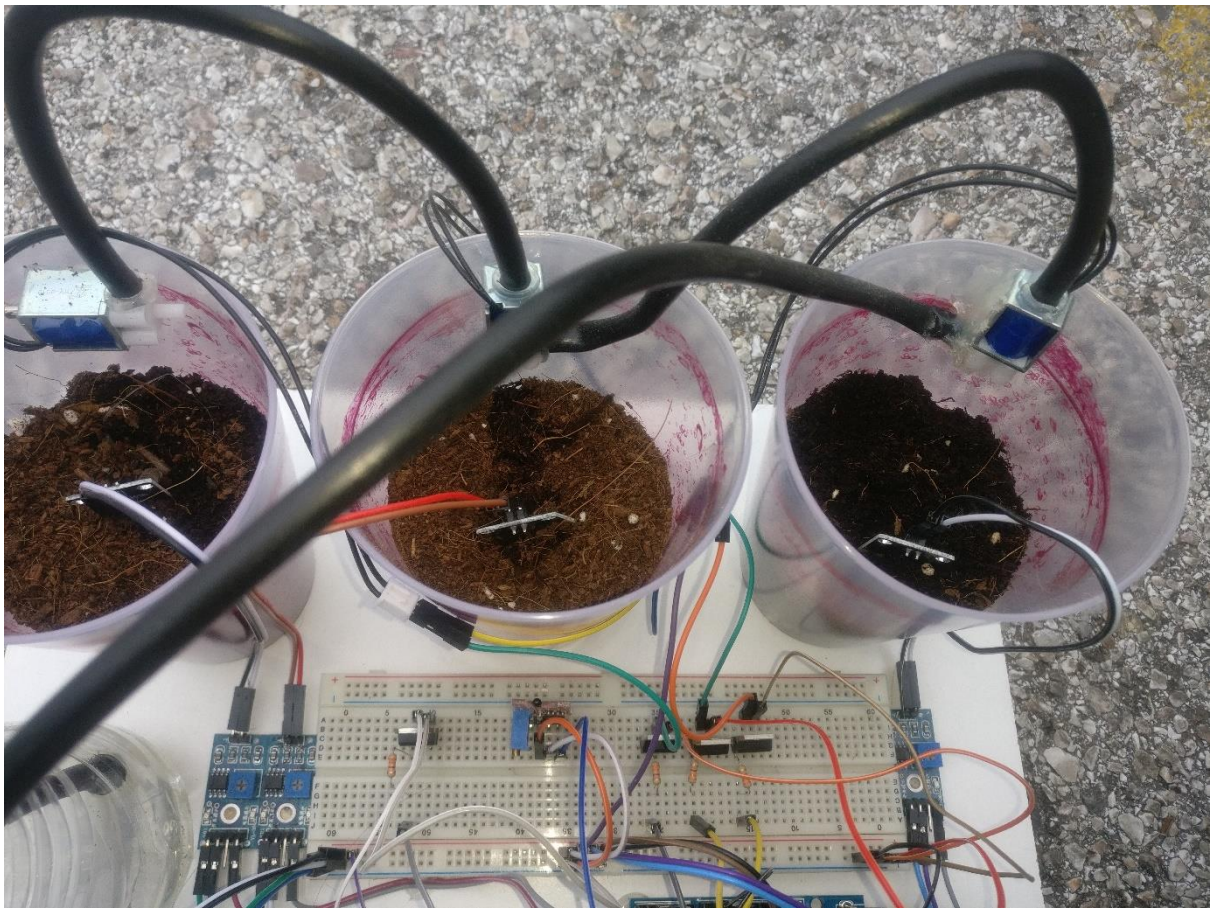
5.5.3 Η λογική λειτουργίας

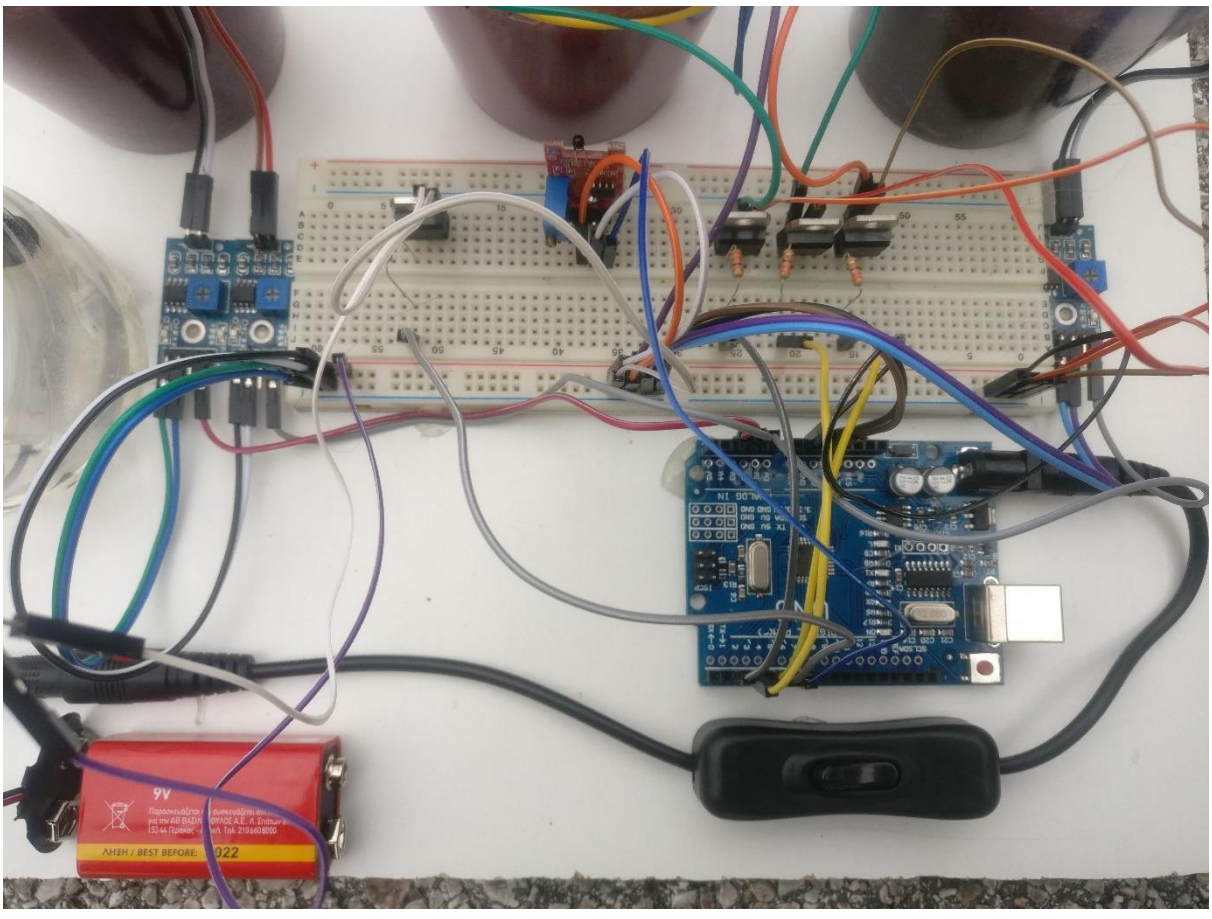
Το σύστημα μας όπως αναφέραμε και παραπάνω αποτελείται από 3 δοχεία με χώμα, τις ενδεχόμενες γλάστρες. Κάθε δοχείο έχει τον δικό του αισθητήρα υγρασίας, καθώς και την δική του βαλβίδα ποτίσματος. Επιπλέον, η κατασκευή μας περιλαμβάνει και ένα δοχείο νερού μέσα στο οποίο υπάρχει μια βυθιζόμενη τρόμπα. Τέλος το σύστημα μας περιέχει μια μπαταρία 9V για την τροφοδοσία του project και έναν αισθητήρα θερμοκρασίας.

Το Arduino χρησιμοποιώντας ως εισόδους τους αισθητήρες υγρασίας, ελέγχει την τιμή της υγρασίας του χώματος της εκάστοτε γλάστρας. Αν η τιμή αυτή πέσει κάτω από το 40%, τότε το Arduino δίνει σήμα στις εξόδους του και ενεργοποιεί την αντίστοιχη βαλβίδα ποτίσματος και έπειτα την τρόμπα νερού. Έτσι, καθεμία από τις γλάστρες λειτουργεί αυτόνομα.

Στο σημείο αυτό, έρχεται να παίξει τον ρόλο του και ο αισθητήρας θερμοκρασίας. Τους καλοκαιρινούς μήνες, το πότισμα των φυτών δεν ενδείκνυται τις μεσημεριανές ώρες οπού η θερμοκρασία πολλές φορές αγγίζει και τους 40 βαθμούς κελσίου. Ο αισθητήρας θερμοκρασίας λοιπόν μας δίνει την τιμή του στην είσοδο του Arduino. Αν η τιμή αυτή είναι πάνω από το όριο που έχουμε θέσει, τότε η βαλβίδα ποτίσματος δεν θα ανοίξει, μέχρι να πέσει η θερμοκρασία. Επειδή όμως αυτό μπορεί να καθυστερήσει αρκετά, όταν το ποσοστό της υγρασίας πέσει κάτω από το 90% περίπου, ενεργοποιείται το πότισμα έκτακτης ανάγκης που παρακάμπτει τον αισθητήρα θερμοκρασίας και ανοίγει τις βαλβίδες ποτίσματος όλων των γλαστρών.






9V

5.6 ΚΩΔΙΚΑΣ ΚΑΤΑΣΚΕΥΗΣ

Στο επόμενο βήμα, θα εφαρμόσουμε την παραπάνω λογική Ladder προγραμματίζοντας το Arduino. Ο κώδικας μας θα είναι ο ακόλουθος:

```
int valve1 = 2; // Δήλωση pin που θα χρησιμοποιήσουμε για την βαλβίδα της 1ης γλάστρας

int valve2 = 3; // Δήλωση pin που θα χρησιμοποιήσουμε για την βαλβίδα της 2ης γλάστρας

int valve3 =4; //Δήλωση pin που θα χρησιμοποιήσουμε για την βαλβίδα της 3ης γλάστρας

int pump = 5; // Δήώση pin για την τρόμπα νερού

int tempSensor = 6; // Δήλωση pin για τον αισθητήρα θερμοκρασίας

int sensor1 = A3; // Δήλωση pin για τον 1ο αισθητήρα υγρασίας

int sensor2 = A2; // Δήλωση pin για τον 1ο αισθητήρα υγρασίας

int sensor3 = A1; // Δήλωση pin για τον 1ο αισθητήρα υγρασίας

int hum1; // Αρχικοποίηση της μεταβλητής που θα χρησιμοποιηθεί για την μέτρηση υγρασίας της 1ης γλάστρας

int hum2; // Αρχικοποίηση της μεταβλητής που θα χρησιμοποιηθεί για την μέτρηση υγρασίας της 2ης γλάστρας

int hum3; // Αρχικοποίηση της μεταβλητής που θα χρησιμοποιηθεί για την μέτρηση υγρασίας της 3ης γλάστρας

int temp; // Αρχικοποίηση της μεταβλητής για την θερμοκρασία

boolean flag0; // Βοηθητικά flags

boolean flag1;
```

```
boolean flag2;
boolean flag3;

void setup() {

  pinMode(valve1,OUTPUT); // Δήλωση της βαλβίδας ως OUTPUT
  pinMode(valve2,OUTPUT);
  pinMode(valve3,OUTPUT);
  pinMode(pump,OUTPUT); // Δήλωση της τρόμπας ως OUTPUT
  pinMode(tempSensor,INPUT); // Δήλωση του αισθητήρα θερμοκρασίας ως INPUT
  pinMode(sensor1,INPUT); // Δήλωση του αισθητήρα υγρασίας ως INPUT
  pinMode(sensor1,INPUT);
  pinMode(sensor1,INPUT);
  Serial.begin(9600); // Το baud rate είναι 9600 για να εγκαθιδρυθεί η επικοινωνία με το PC

}

void loop() {

hum1 = analogRead(sensor1); // Διαβάζουμε τις τιμές των αισθητήρων υγρασίας και της αποθηκεύουμε

hum2 = analogRead(sensor2);
hum3 = analogRead(sensor3);
temp = digitalRead(tempSensor); // Διαβάζουμε την τιμή του αισθητήρα θερμοκρασίας και την αποθηκεύουμε

Serial.print("The first sensor has a measurement of: "); // Βοηθητικά μηνύματα για τον χρήστη στην σειριακή οθόνη

Serial.println(hum1);
```

```
Serial.print("The second sensor has a measurement of:  ");
Serial.println(hum2);
Serial.print("The third sensor has a measurement of: ");
Serial.println(hum3);
Serial.print("The temperature sensor has a measurement of: ");
Serial.println(temp);
Serial.println(" n/");
delay(1000);


int opthum = 400; // Δήλωση της ιδανικής υγρασίας για πότισμα
int crithum = 1000; // Δήλωση για την επικίνδυνη τιμή υγρασίας


    //Rung 0


    if(hum1 > opthum || flag0 == true) { digitalWrite(valve1,HIGH); flag1 = true;} //
Αν η υγρασία ξεπερνάει το όριο τότε ανοίγουν οι βαλβίδες της εκάστοτε γλάστρας


    //Rung 1


    if(hum2 > opthum || flag0 == true) { digitalWrite(valve2,HIGH); flag1 = true;} //
Θέτουμε την τιμή 1 στο flag1 για να χρησιμοποιηθεί αργότερα


    //Rung 2


    if(hum3 > opthum || flag0 == true){ digitalWrite(valve3,HIGH); flag1 = true;}


    //Rung 4


     if( temp == 1 && hum1 > crithum || hum2 >crithum || hum3 > crithum){ // Αν η
υγρασία είναι στο κρίσιμο σημείο και η θερμοκρασία ξεπερνάει το όριο τότε
ανοίγουν όλες οι βαλβίδες για έκτακτο πότισμα
```

```
      flag2 = true; // Θέτουμε την τιμή 1 στο flag2 για να χρησιμοποιηθεί αργότερα

      digitalWrite(valve1,HIGH); // Ανοίγουν οι βαλβίδες
      digitalWrite(valve2,HIGH);
      digitalWrite(valve3,HIGH);

      }

    //Rung 3

    if((temp != 1 || flag2 == true) && flag1 == true){ // Για να ανοίξει η τρόμπα πρέπει είτε να έχουμε έκτακτο πότισμα είτε η θερμοκρασία να είναι καλή για πότισμα. Επίσης πρέπει να είναι ανοιχτές η βαλβίδες, δηλαδή το flag1

      digitalWrite(pump,HIGH); // Άνοιγμα τρόμπας
      Serial.println("Pump is on");
      delay(5000); // Εδώ δηλώνεται πόση ώρα θα ποτίζονται οι γλάστρες. Η τιμή είναι σε ms, άρα 5 δευτερόλεπτα

      flag1=false; // Θέτουμε το flag1 σε '0' για να μην επαναληφθεί το πότισμα

      digitalWrite(pump,LOW); // Κλείσμο τρόμπας
    }

    if(flag1 == false){ // Έλεγχος της τιμής του flag1
      digitalWrite(valve1,LOW); // Κλείσμο βαλβίδων
      digitalWrite(valve2,LOW);
      digitalWrite(valve3,LOW);
      }
  }
```

ΚΕΦΑΛΑΙΟ 6 – Γεννήτρια σημάτων

6.1 ΠΕΡΙΛΗΨΗ ΕΦΑΡΜΟΓΗΣ

Στην εφαρμογή αυτή θα ασχοληθούμε με την μελέτη και κατασκευή μιας γεννήτριας σημάτων. Η κατασκευή αυτή θα έχει τη δυνατότητα να δώσει στην έξοδο της 4 διαφορετικά είδη σημάτων, τετραγωνικό, τριγωνικό, πριονωτό και ημιτονοειδή. Επιπλέον, για τον τετραγωνικό παλμό θα υπάρχει και η δυνατότητα επιλογής Duty Cycle. Ο χείριστης της κατασκευής θα έχει τη δυνατότητα να επιλέξει μεταξύ αυτών των ειδών, με την χρήση του αντίστοιχου επίλογέα, ενώ θα μπορεί να ρυθμίσει το Κέρδος(Gain) του σήματος καθώς και την συχνότητα(Frequency) μέσω των αντίστοιχων ποτενσιόμετρων. Επιπλέον, η κατασκευή διαθέτει ένα reset button, μια έξοδο mini jack, καθώς και ένα switch για τον έλεγχο της λειτουργίας του ηχείου.

6.2 Γεννήτριες σημάτων

Οι γεννήτριες σημάτων χρησιμοποιούνται για τον έλεγχο και την αποσφαλμάτωση ηλεκτρονικών κυκλωμάτων. Συχνά χρησιμοποιούνται για τον έλεγχο της απόκρισης συχνότητας εξαρτημάτων όπως ενισχυτές και αισθητήρες. Η συγκεκριμένη γεννήτρια έχει ως βάση ένα Arduino.

6.3 ΥΛΙΚΑ ΚΑΤΑΣΚΕΥΗΣ

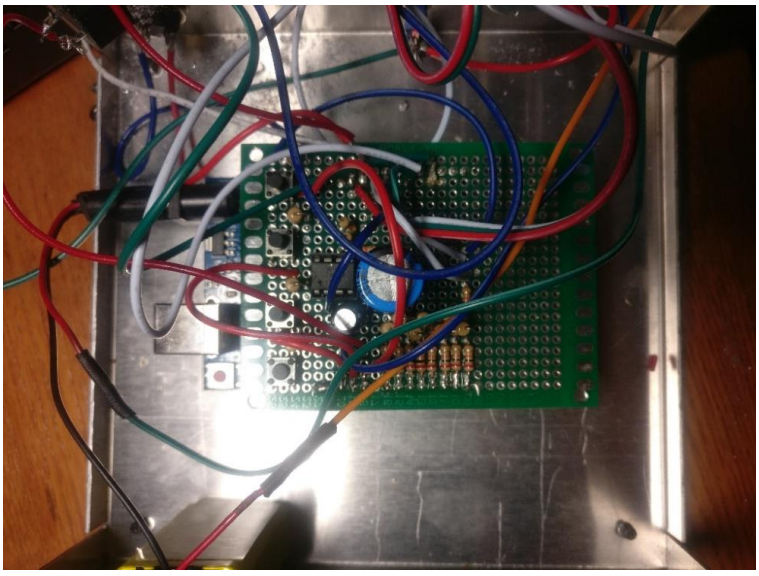
6.3.1 Bill of Materials(BOM)

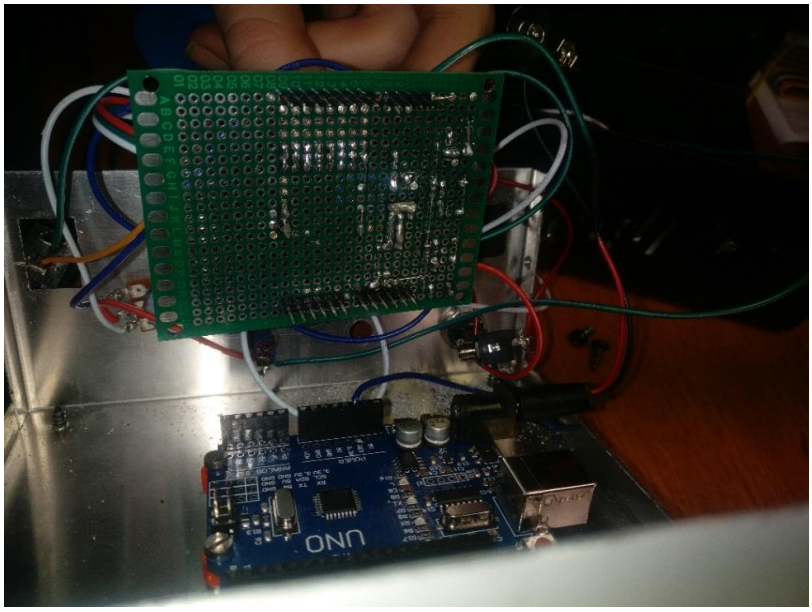
Mini Momentary Pushbutton Switch x1
10K Ohm 1/4-Watt Carbon Film Resistor x8
20K Ohm 1/4-Watt Carbon Film Resistor x9
10K-Ohm Linear-Taper Potentiometer x3
10K-Omh Audio-Taper Potentiometer x1
1/8" Stereo In-Line Audio Jack x1
4.7K Ohm 1/4-Watt Resistor x2
8 Pin Socket x1
LM386 Low Voltage Audio Power Amplifier x1
220μF 35V Electrolytic Capacitor x2
Arduino Uno REV 3 x1
Arduino Custom Shield x1
Toggle Switch x1

6.4 ΥΛΟΠΟΙΗΣΗ ΚΑΤΑΣΚΕΥΗΣ

6.4.1 Πλακέτα κατασκευής

Για την υλοποίηση της κατασκευής μας θα χρησιμοποιήσουμε μια διάτρητη πλακέτα που θα προσομοιώσει ένα Arduino proto shield. Η πλακέτα αυτή είναι έτσι σχεδιασμένη ώστε με τις κατάλληλες συνδέσεις να «κουμπώνει» πάνω από το Arduino. Έτσι λοιπόν ξεκινώντας κολλάμε τα αντίστοιχα pins του Arduino που θα χρειαστούμε επάνω στην πλακέτα.





6.4.2 DAC (Digital to Analog Converter)

Η κατασκευή μας θα ξεκινήσει με την κατασκευή ενός R2R ladder DAC, δηλαδή ενός DAC κατασκευασμένου με τη χρήση αντιστατών. Αυτό είναι απαραίτητο αφού το Arduino δεν έχει αναλογικές εξόδους, κι έτσι θα πρέπει να μετατρέψουμε ένα δυαδικό σήμα 8-bit προερχόμενο από τα pins 0-7 σε αναλογικό. Στα pins 0-7 λοιπόν έχουμε δυαδικές καταστάσεις με 0 η 5 volts. Το DAC παίρνει όλες αυτές τις τάσεις και βγάζει μια τιμή μεταξύ 0 και 5 Volts, είναι δηλαδή κάτι σαν ένας πολυεπίπεδος διαιρέτης τάσης. Ο δικός μας DAC ο οποίος είναι 8-bit μπορεί να παράξει 256(2^8) διαφορετικές τιμές τάσης. Υπάρχουν πολύ πιο περιπλοκά DAC από αυτό που θα χρησιμοποιήσουμε στην κατασκευή μας, ενσωματωμένα σε ολοκληρωμένα κυκλώματα, που δίνουν τη δυνατότητα ακόμα και για στερεοφωνικό ήχο, όμως δεν θα χρειαστούμε κάτι τέτοιο στη δική μας κατασκευή.

6.4.3 Η εντολή PORTD

Στο Arduino τα ψηφιακά pins 0-7 είναι όλα συνδεδεμένα στην port D του μικροεπεξεργαστή Atmega. Η εντολή PORTD μας δίνει ουσιαστικά τη δυνατότητα να κάνουμε set τα pins 0-7 σε HIGH η LOW σε μια γραμμή, αντί με τη χρήση 8 εντολών

digitalWrite. Επομένως εφόσον η port D έχει 8 pins μπορούμε να στείλουμε 2^8 = 256 διαφορετικούς συνδυασμούς τιμών (0 - 255). Για παράδειγμα με την εντολή PORTD = 125, σετάρουμε τα pins 7-0 με τον δυαδικό αριθμό 01111101(125) αντίστοιχα.

Για τον υπολογισμό της τάσης εξόδου από τον DAC χρησιμοποιούμε τον τύπο:

Vout = [(portdValue)/255] * 5V, οπότε για το παράδειγμα μας θα έχουμε 125/5 = 2.45V

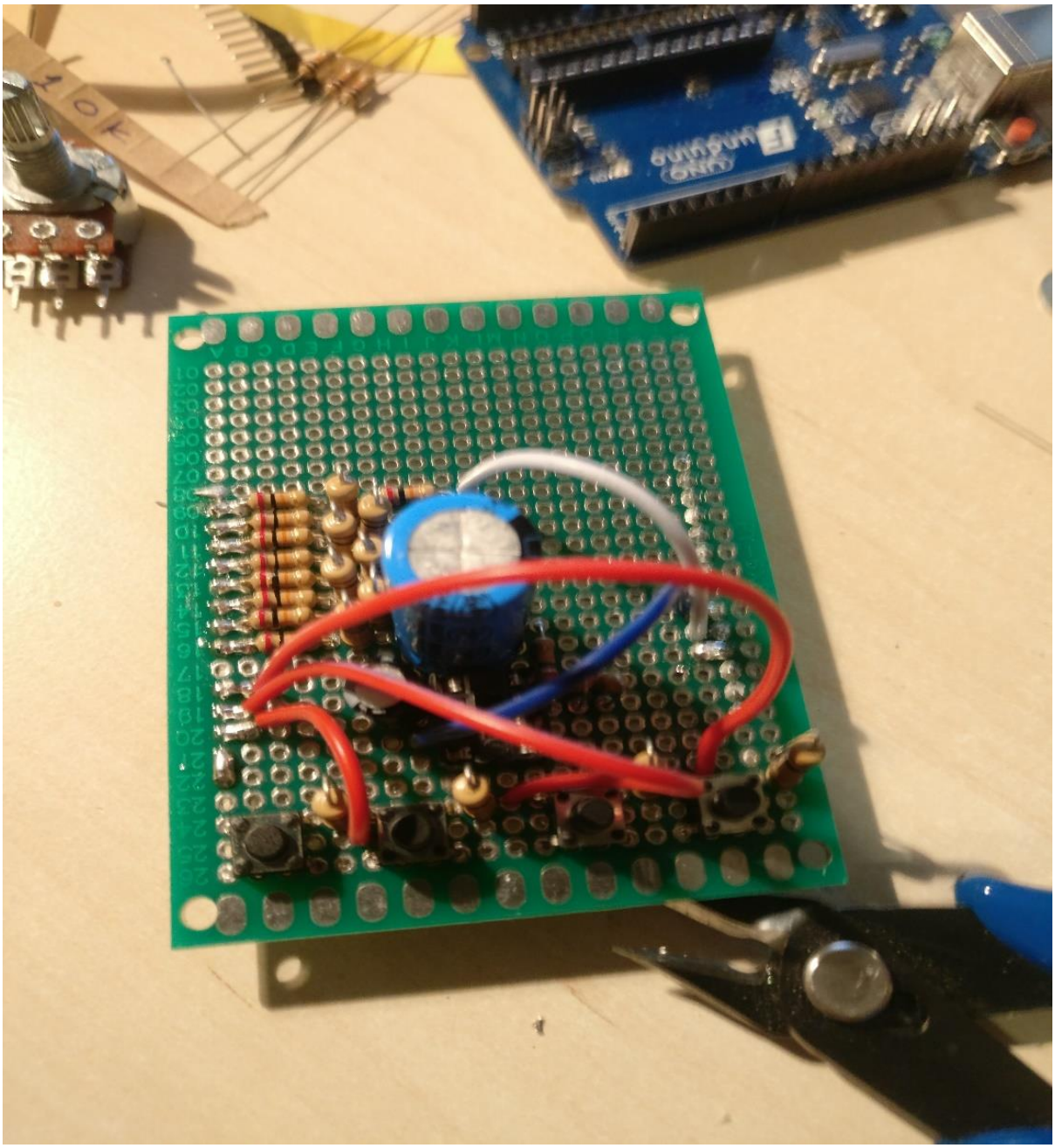
6.4.4 Οι 4 παλμοί

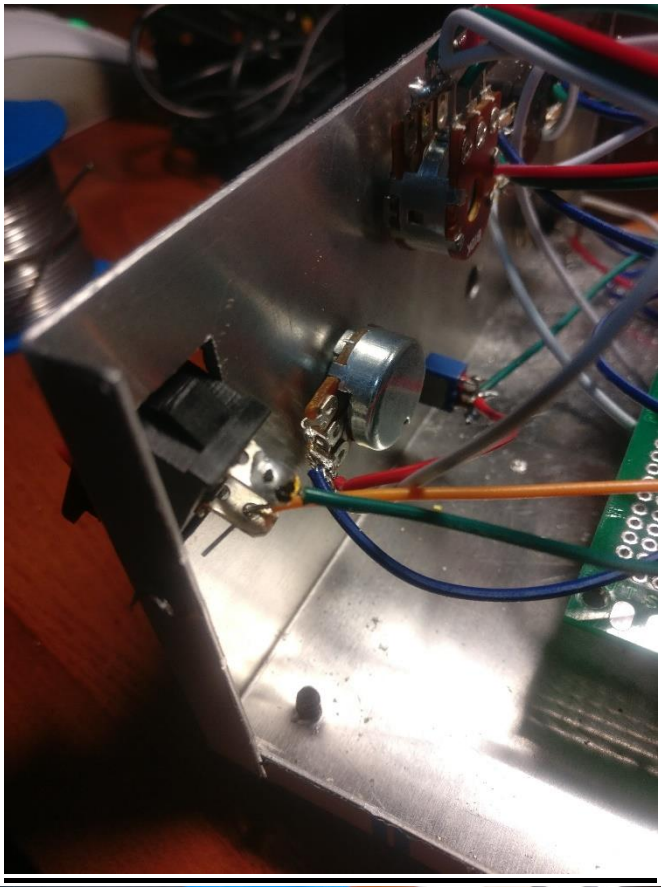
Επειδή στο ημιτονοειδές κυμα το Arduino χρειάζεται αρκετό χρόνο για να κάνει τους υπολογισμούς, αφού έχουμε πράξεις με ημίτονα και δεκαδικούς, πρέπει να βρούμε έναν άλλο τρόπο να διαχειριστούμε τις πράξεις αυτές. Για το λόγο αυτό, δηλώνουμε στον κώδικα μας ένα array το οποίο περιέχει τις τιμές για ένα ημιτονοειδές κυμα από 0 έως 255. Με αυτόν τον τρόπο, το μόνο που έχει να κάνει το Arduino είναι να ανακαλέσει από το array τις τιμές αυτές.

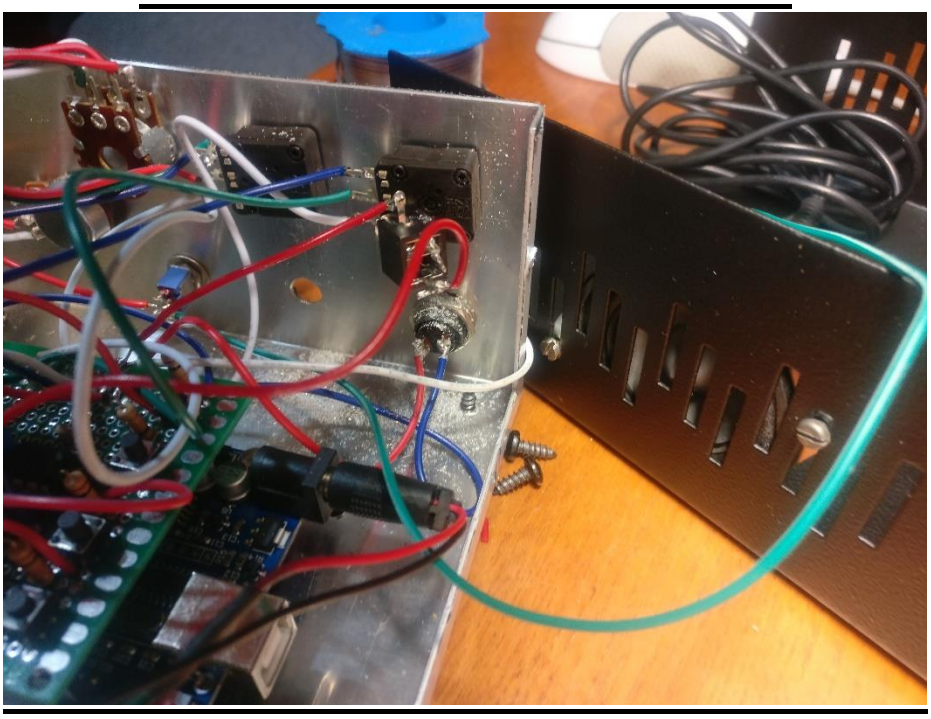
6.4.5 Ενισχυτής

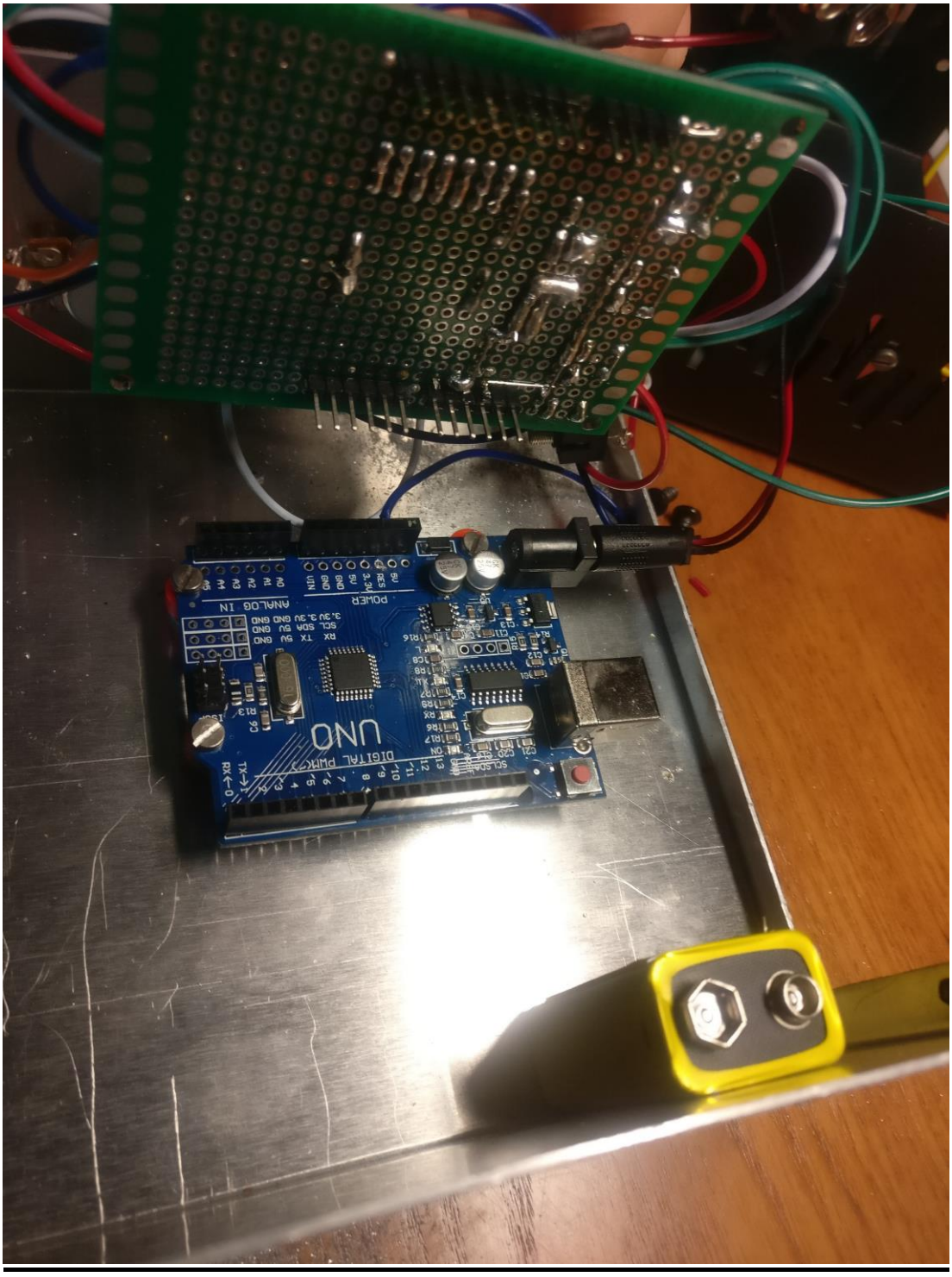
Στη συνέχεια το κύκλωμά μας περιλαμβάνει έναν τελεστικό ενισχυτή, χρησιμοποιώντας το ολοκληρωμένο LM386N. Επιχειρώντας τις απαραίτητες συνδέσεις, και με τη χρήση ενός πυκνωτή στην έξοδο του ενισχυτή, στο pin6, παίρνουμε στην έξοδο το ενισχυμένο σήμα, το οποίο τροφοδοτούμε στο ηχείο της κατασκευής.

6.5 ΠΡΑΓΜΑΤΙΚΗ ΣΥΝΔΕΣΜΟΛΟΓΙΑ









6.6 ΚΩΔΙΚΑΣ ΚΑΤΑΣΚΕΥΗΣ

```
// Code to make 16384 bytes of sinewave:
// sine = round((sin(seq(0,2*pi, length=16384))+1)*127.7)
// sink("temp.txt")
// for (a in 1:length(sine)) cat(sine[a],",")
```

```
// sink()

//Sine-wave calculated values
const byte sinewave[] PROGMEM = {128-128}; //Real parameters can be found in .ino file

const int Arefs[3][10] = { // Pot values to map 10 positions on each pot.
                    {5,43, 206, 366, 487, 617, 722, 842, 970, 1020},
                    {10,113, 251, 371, 472, 599, 725, 842, 987, 1020},
                    {10,63, 205, 342, 480, 598, 724, 861, 991, 1020}
                  };


const int musicalNotes[] = {131,147,165,175,196, 220, 247, 261, 294, 330, 350, 392, 440, 494, 523, 588, 659, 698, 784, 880, 988, 1046, 1175, 1319};//C major octave frequencies table.


unsigned long waveSize = 16384;                    // The sine wave bytes we calculated above, which is 1/4 of our sampling rate.


byte bitShifter = 2;                               // This multiplies everything by 4, for higher prescision.


byte bitShiftTRI = bitShifter + 5;                 // helps bringing the y-value for triangle into a range between 0 and 512


byte bitShiftST = bitShifter + 6;                  // helps bringing the y-value for saw tooth a range between 0 and 256


unsigned long waveSizeScaled = waveSize << bitShifter ;     // Multiplies and scales the waveSize by 4


unsigned long triSizeHalf = waveSizeScaled / 2;            // We will use this for the sawtooth wave

unsigned long squareWavePerc[] = {3276, 9830, 19661, 32768, 45875, 55706, 62259}; // This sets duty cycle percentages 5,15,30,50,70,85,95 % for square wave (Scaled)
```

```
int valA[] = {0,0,0};                          // Initialization of the three
potentiometers

int setfreq = 0;                               // Possible next frequency

unsigned long sampleRate = 65536;              // the sampling rate of the
project

unsigned long nextVal = 0;                     // The position after each cycle of
the sampling rate

unsigned int wavePos = 0;                      // Current position in the waveform.
This value is for the bitshifted calculation and will overflow to 0 when
waveSizeScaled is reached

unsigned long waveP = 0;                       // The actual position in the
sinewave array

unsigned long sampleFactor = sampleRate / waveSize;    // This reaches 4 when
16384 values are sampled with 65536 Hz

float correctionFactor = 1.016;                // This is a correction used, to
amend a lag due to the Arduino being a little slow.

unsigned long frequency = 0;                   // This is set from the
potentiometer positions.

unsigned long freqDisplay = 0;                 //For use possibly on serial
monitor, to update frequency with 5hz difference.

byte freqDigits[] = {0,0,0,0,0};               // This is for using the pots to create
the frequency with scaling, x10 x100 etc.

byte sweepDecVal = 0;                          // Current decimal value of the
sweep pot

byte powerDecVal = 0;                          // Current decimal value of the
power pot

byte modeDecVal = 0;                           //  Current decimal value of the
mode  pot

byte oldMode = 0;                              // Mode change flag,to reset some
stuff


void setup() {

  //activateFreq();

  //freqDisplay = frequency; //For use if we see freq on the serial monitor


  //Here we will setup the port/pin modes
```

```
  //TIMER INTERRUPT SETUP
  DDRD = 0xFF;               //Set all portD pins as outputs
  DDRC = 0x00;               //Set all analog pins as inputs
  DDRB = 0x00;               //Set all portB pins as  inputs
  cli();                     //disable interrupts, needed to setup the interrupt registers
  Serial.end();              // begin(9600);

  //set timer1 interrupt at 65536kHz, which is our sampling rate

  TCCR1A = 0;                // set entire TCCR1A(Timer/Counter Control register A)
register to 0
  TCCR1B = 0;                // same for TCCR1B
  TCNT1  = 0;                //initialize counter value to 0;

  // set timer count for 65536khz increments, on the OCRA(Output Compare register
A)
  OCR1A =30;                 // = (16*10^6) / (65536*8) - 1
  // turn on CTC mode
  TCCR1B |= (1 << WGM12);
  // Set CS11 bit for 8 prescaler
  TCCR1B |= (1 << CS11);
  // enable timer compare interrupt
  TIMSK1 |= (1 << OCIE1A);
  sei(); // enable interrupts
}

ISR(TIMER1_COMPA_vect){  // the timer-driven interrupt routine
  wavePos = wavePos + nextVal;
  if (modeDecVal == 0){ // sine wave
    waveP = wavePos >> bitShifter;
```

```
    PORTD = pgm_read_byte_near(sinewave + waveP);
  }
  if (modeDecVal == 1){// sawtooth wave
    PORTD = wavePos >> bitShiftST;
    }
  if (modeDecVal == 2){// triangle wave
    if (wavePos < triSizeHalf)
      PORTD = wavePos >> bitShiftTRI;
    if (wavePos > triSizeHalf)
      PORTD = (waveSizeScaled-wavePos) >> bitShiftTRI;
    }
  if (modeDecVal > 2){// square wave
    if (wavePos < squareWavePerc[modeDecVal-3])
      PORTD = 255;
    else
      PORTD = 0;
    }
  }

void activateFreq(){ // calculates the nextval-value for the current frequency
  unsigned long freq;
  freq = frequency * correctionFactor;
  nextVal = (freq << bitShifter) / sampleFactor;
  }

byte getDecVal(byte whichA){ // returns the decimal value (0-9) for the requested
analog port
  byte retval = 0;
  if (valA[whichA] >= Arefs[whichA][9])
    retval = 9;
  else
```

```
   while (Arefs[whichA][retval] + ( (Arefs[whichA][retval+1] -
Arefs[whichA][retval])/2 ) < valA[whichA] && retval < 8)
      retval ++;
  return (retval);
  }

void setCurrentFreq(){ // gets current frequency from the poti positions
  sweepDecVal = getDecVal(0);
  powerDecVal = getDecVal(1);
  modeDecVal =  getDecVal(2);
  if (modeDecVal != oldMode)
    activateFreq();
     if (powerDecVal <= 4){
       freqDigits[powerDecVal] = sweepDecVal;
       setfreq = freqDigits[4]*10000 + freqDigits[3]*1000 + freqDigits[2]*100 +
freqDigits[1]*10 + freqDigits[0];
       }
     if (powerDecVal == 5)
       setfreq = valA[0];
     if (powerDecVal == 6)
       setfreq = 1000 + (valA[0] * 8.79);
     if (powerDecVal > 6)
       setfreq = musicalNotes[(powerDecVal-7)*7 + sweepDecVal];
     if (setfreq != frequency){
       frequency = setfreq;
       activateFreq();
       }
  oldMode = modeDecVal;
  }

void loop() {
```

```
  for (byte a=0;a<3;a++) // reading the current values of the 3 pots
    valA[a] = analogRead(a);
//
//    valA0 = analogRead(A0);
//    valA1 = analogRead(A1);
    setCurrentFreq();
 }
```

Αρχικά, πριν μιλήσουμε για τον κώδικά μας, παραθέτουμε ένα κομμάτι κώδικα γραμμένο για τερματικό ο οποίος υπολογίζει τις τιμές από 0 έως 255 για το ημιτονοειδές σήμα που αποθηκεύουμε στη μνήμη του Arduino:

```
// Code to make 16384 bytes of sinewave:
// sine = round((sin(seq(0,2*pi, length=16384))+1)*127.7)
// sink("temp.txt")
// for (a in 1:length(sine)) cat(sine[a],",")
// sink()
```

Πέραν από τον σχολιασμό του κώδικά μας, θα χρειαστεί να εξηγήσουμε και κάποια σημεία περαιτέρω.

Ο λόγος που χρησιμοποιούμε το bit shifting στον κώδικά μας είναι ότι μας βοηθάει στην ακρίβεια των μετρήσεων του κύματος καθώς και στο να μην χρησιμοποιήσουμε πράξεις με δεκαδικούς, κάτι που είναι χρονοβόρο για τον επεξεργαστή του Arduino. Επιπλέον στον κώδικά μας με τη χρήση left bit shifting κατά 2 κάνουμε scale το σήμα μας πολλαπλασιάζοντας το επί 4, ώστε να πετύχουμε μεγαλύτερη ακρίβεια. Αυτό συμβαίνει στην δήλωση της μεταβλητής waveSizeScaled.

Παρακάτω, στην μεταβλητή squareWavePerc, θέτουμε τις τιμές με βάση τις οποίες θα έχουμε τα ποσοστά του duty cycle του τετραγωνικού παλμού.

Παρακάτω θέτουμε το sample factor, που φτάνει έως την τιμή 4 όταν διαβαστούν και οι 16384 τιμές του κύματος από το σχετικό array. Σε αυτό το σημείο

αρχικοποιούμε και την μεταβλητή correctionFactor η οποία θα χρησιμεύσει ώστε να συντονίσουμε ακριβώς τον επεξεργαστή του Arduino με τις κινήσεις των ποτενσιόμετρων.

Περνώντας στην συνάρτηση setup, συναντάμε την καρδιά της κατασκευής, που είναι τα interrupts στην CPU του Arduino. Ας κάνουμε μα εκτενή ανάλυση σε αυτά. Αρχικά θέτουμε με τη χρήση των Data Direction Registers τις εισόδους και τις εξόδους του Arduino. Χρησιμοποιούμε τη μεθοδολογία αυτή ώστε να χειριζόμαστε όλα τα pins ταυτόχρονα, καθώς η ξεχωριστή ανάθεση με τη χρήση της εντολής pinMode, έχει μια καθυστέρηση μεταξύ κάθε εκτέλεσης της τάξεως των λίγων ms, κάτι που σε time-intensive εφαρμογές όπως η δική μας έχει ιδιαίτερη σημασία.

Έπειτα, απενεργοποιούμε τα interrupts καθώς και την σειριακή επικοινωνία ώστε να μπορούμε να χειριστούμε τα timer registers του Atmega328p. Θέτουμε τα Timer/Counter Registers A και B στο 0, και αρχικοποιούμε την τιμή του μετρητή πάλι με 0. Με την επόμενη εντολή, ενεργοποιούμε το CTC mode. Στο κανονικό mode των interrupts, όταν προκύπτει ένα τέτοιο, ο επεξεργαστής σταματάει κάθε λειτουργία και δεσμεύεται στην ολοκλήρωση της λειτουργίας του interrupt. Εμείς με τη χρήση του CTC mode πετυχαίνουμε να εκτελεί το interrupt παράλληλα με τις υπόλοιπες διεργασίες του. Στις επόμενες εντολές, θέτουμε το bit για τον prescaler και έπειτα ενεργοποιούμε το interrupt σύγκρισης του timer καθώς και τα interrupts του microprocessor με την εντολή sei();. Παρακάτω θα αναλύσουμε και την συνάρτηση ISR, γνωστή και ως Interrupt Service Routine.

- ### ISR (Interrupt Service Routine)

Η συνάρτηση αυτή είναι η συνάρτηση που καλείται όταν γίνει ένα interrupt request. Στο δικό μας πρόγραμμα, αυτό συμβαίνει με την κίνηση των ποτενσιόμετρων. Στην περίπτωση μας δέχεται σαν όρισμα το διάνυσμα TIMER1_COMPA_vect, το οποίο περιέχει τις αντιστοιχίες των interrupts, πότε δηλαδή ένα interrupt έχει ζητηθεί, ώστε να ξεκινήσει η ISR. Στην συνάρτηση αυτή αρχικά ανανεώνουμε τη θέση του κύματος στην επόμενη τιμή του πίνακα, και έπειτα ελέγχουμε την τιμή του ποτενσιόμετρου επιλογής κυματομορφής. Για 0 έχουμε

ημίτονο, για 1 έχουμε ράμπα, για 2 έχουμε τρίγωνο και για μεγαλύτερο του 2 έχουμε τετραγωνικό παλμό, με διαφορετικά ποσοστά duty cycle.

Πέρα από τον κώδικά των interrupts, ας ρίξουμε μια μικρή ματιά και στις υπόλοιπες συναρτήσεις του προγράμματος.

- **activateFreq()**

Η συνάρτηση πολλαπλασιάζει τη συχνότητα με τον παράγοντα διόρθωσης που θέσαμε παραπάνω, και στη συνέχεια υπολογίζει την επόμενη τιμή της συχνότητας κάνοντας το scaling κι έπειτα τη διαίρεση με το sampleFactor.

- **getDecVal(pot)**

Η συνάρτηση αυτή δέχεται σαν όρισμα την αναλογική τιμή ενός ποτενσιόμετρου και στη συνέχεια, υπολογίζει με βάση τον πίνακα τιμών που έχουμε ορίσει παραπάνω, σε ποια θέση (1-9) βρίσκεται το ποτενσιόμετρο και την επιστρέφει.

- **setCurrentFreq()**

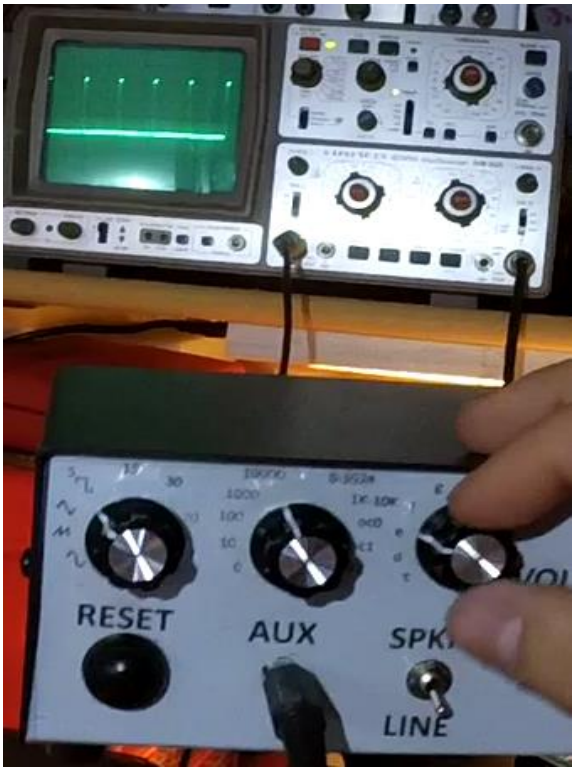
Είναι απ' τις βασικότερες συναρτήσεις του προγράμματος. Στην αρχή καλούμε την συνάρτηση getDecVal() για κάθε ένα από τα 3 ποτενσιόμετρα ξεχωριστά. Έπειτα ελέγχουμε με τη μεταβλητή oldMode αν έχει αλλάξει το mode, και αν όχι καλούμε τη συνάρτηση activateFreq(); Σε άλλη περίπτωση, διαβάζει το μεσαίο ποτενσιόμετρο, και θέτει είτε τις κλίμακες για τη συχνότητα, για τιμή από 0-4, για τιμή 5 παίρνει την αναλογική είσοδο του ποτενσιόμετρου, δηλαδή 0-1024, δίνοντας μας ακρίβεια στις χαμηλές συχνότητες, για τιμή 6 μας δίνει συχνότητα από 1kHz – 10kHz , ενώ για τις υπόλοιπες τιμές μας δίνει στην έξοδο 3 διαφορετικές οκτάβες από την Nτο μείζονα κλίμακα (έχουμε ορίσει τις συχνότητες στην αρχή του κώδικά). Τέλος ελέγχει για αλλαγές στην συχνότητα και αν υπάρχουν θέτει την τιμή που έχει πάρει από τους προηγουμένους βρόγχους if, ενώ ταυτόχρονα κρατάει στο flag oldMode το currentMode.

Τέλος, έχουμε την συνάρτηση loop οπού θα εκτελείται συνεχώς στον κώδικα και περιλαμβάνει την συνεχή ανάγνωση της τιμής των 3 ποτενσιόμετρων καθώς και την κλήση της συνάρτησης setCurrentFreq().

6.7 Σήματα εξόδου κατασκευής

Παρακάτω βλέπουμε με τη σειρά τις κυματομορφές που αποδίδει η γεννήτρια μας στην έξοδο.

Με τη σειρά: α. Ημίτονο β. Sawtooth γ. Τριγωνικός δ. Τετράγωνος (50%) ε. Τετράγωνος (5%) στ. Τετράγωνος (15%) ζ. Τετράγωνος (30%) η. Τετράγωνος (95%)


RESET
AUX
LINE


SPKR
LINE


SPKR

ΒΙΒΛΙΟΓΡΑΦΙΑ

- http://www.oomlout.com/a/products/ardx/
- http://www.arduino.cc/
- http://el.wikipedia.org/wiki
- http://creativec0d1ng.blogspot.gr/2012/09/introduction-to-arduino-part-1.html
- http://arduino.cc/en/Main/Products?from=Main.Hardware
- http://shieldlist.org/
- http://arduino.cc/en/main/software
- http://www.modk.it
- http://blog.minibloq.org/
- http://blogs.sch.gr/ioarvanit/2012/12/17/%CF%84%CE%BF-arduino-%CE%B1%CF%80%CF%8C-%CF%84%CE%B7%CE%BD-%CF%83%CE%BA%CE%BF%CF%80%CE%B9%CE%AC%CE%B5%CE%BD%CF%8C%CF%82-newbie/ Βιβλία και σημειώσεις που χρησιμοποιήθηκαν:

[A] Δρ. Βολογιαννίδης Σταύρος (2009). Ευφυής Έλεγχος, Θεωρία και Εφαρμογής.
[B] Banzi, M. (2009). *Getting Started with Arduino*. O'Reilly.

Ιστοσελίδες που χρησιμοποιήθηκαν:

Arduino

[1] http://arduino.cc/en/Guide/Environment?from=Tutorial.Bootloader
[2] http://arduino.cc/en/Guide/Windows
[3] http://arduino.cc/en/Reference/HomePage [4]
http://arduino.cc/en/Main/ArduinoBoardUno [5]
http://arduino.cc/en/Tutorial/Memory